Data Science Approaches for Electrochemical Engineers: An Introduction through Surrogate Model Development for Lithium-Ion Batteries

Neal Dawson-Elli,[1,*] Seong Beom Lee,[1,*] Manan Pathak,[1,*] Kishalay Mitra,[2] and Venkat R. Subramanian[1,3,,z]**

[1] *Department of Chemical Engineering, University of Washington, Seattle, Washington 98195, USA*
[2] *Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Medak 502 285, Telangana, India*
[3] *Pacific Northwest National Laboratory, Richland, Washington 99352, USA*

Data science, hailed as the fourth paradigm of science, is a rapidly growing field which has served to revolutionize the fields of bio-informatics and climate science and can provide significant speed improvements in the discovery of new materials, mechanisms, and simulations. Data science techniques are often used to analyze and predict experimental data, but they can also be used with simulated data to create surrogate models. Chief among the data science techniques in this application is machine learning (ML), which is an effective means for creating a predictive relationship between input and output vector pairs. Physics-based battery models, like the comprehensive pseudo-two-dimensional (P2D) model, offer increased physical insight, increased predictability, and an opportunity for optimization of battery performance which is not possible with equivalent circuit (EC) models. In this work, ML-based surrogate models are created and analyzed for accuracy and execution time. Decision trees (DTs), random forests (RFs), and gradient boosted machines (GBMs) are shown to offer trade-offs between training time, execution time, and accuracy. Their ability to predict the dynamic behavior of the physics-based model are examined and the corresponding execution times are extremely encouraging for use in time-critical applications while still maintaining very high (∼99%) accuracy.

Manuscript submitted September 21, 2017; revised manuscript received December 4, 2017. Published January 3, 2018. This was Paper 1138 presented at the New Orleans, Louisiana, Meeting of the Society, May 28–June 1, 2017.

Data science, also known as data-intensive scientific discovery, is hailed as the fourth paradigm of science.[1] A field focused on extracting knowledge or understanding from data, it includes the subdomains of machine learning, classification, data mining, databases, and data visualization. In the age of internet-scale data, these techniques are not only powerful, but also necessary to extract the signal from the noise and to have the throughput to do so in a reasonable amount of time. It has revolutionized the fields of bio-informatics, climate science, word recognition, advertising, medicine, and is finding more applications daily. In Google's Translate application, substantial improvements over previous methods were achieved using artificial neural network (ANN) structures, making 60% fewer errors than the previous state-of-the-art algorithm.[2] In climate science, where models are sophisticated and numerous, data science techniques are used to determine which of 20 models will give the best prediction on future and historical data, the accuracy of which surpasses the accuracy of the average of all models, the current benchmark.[3] As chemical engineers are increasingly tasked with the analysis of more complex data sets, these same data science tools which have revolutionized other fields become more relevant.[4]

When data sets grow, they must be managed intentionally in order to be useful. Data management, a subfield of data science, fills this role and gives the tools to be able to correct for missing data points, ensure consistency of the data, and transform the content of the data such that it is suitable for use in other aspects of data science. For missing data, several techniques exist, such as deletion of the sample, deletion of the feature column, assignment of the average value of the feature column, and creation of a predictive model to fill the gaps. Of these, the creation of a predictive model is of significant interest for electrochemical modelers, as it can result in a better performing model through the preservation of more information.[5] In addition to cleansing the data set of missing values, this step is often where exploration and correlation is done, and is an opportunity for initial feature engineering to take place. Feature engineering, discussed in more detail in the Results and discussion section, is the practice of combining input features that highlight patterns in the data with more efficacy than the individual features themselves. Feature engineering is a very manual activity and, along with data preprocessing, it will generally constitute 80% of the time spent on a data science project.[6]

In addition to analysis and organization of data, data science techniques are also important in the communication of information. As the scale of data increases, the way information is communicated becomes more important. Data visualization is the field involving the research and implementation of the most efficient ways to communicate large scale, complex data. Historically, the ability of researchers to present findings in engaging and consumable ways is outpaced by the rate of acquisition of new findings, especially in large data-mined spaces.[7] As this field improves, the barrier to entry lowers, and information can be more effectively communicated.

Machine learning is an exciting subfield of data science which allows computers to learn patterns in data without being explicitly programmed.[8] There exist a multitude of algorithm types, all of which work on fundamentally different paradigms and which excel in different problem types. ML algorithms have been used to build functional relationships between input and output variables for engineering processes in the past.[4,9,10]

Although there are countless different algorithms associated with machine learning, this paper is primarily concerned with decision trees (DTs), random forests (RFs), and gradient-boosted machines (GBMs). Decision trees create sets of rules which can be applied to new data of a similar format. RFs are ensembles of DTs which are randomized such that each has the possibility of yielding a different response for a given input. The outputs of the DTs are combined using a weighted sum, resulting in RFs outperforming DTs in a majority of cases, at the cost of larger size on disk and longer execution time. GBMs are ensembles of many small DTs where the maximum depth is heavily constrained such that the model generalizes more aggressively. Their implementation is based upon the theory that a combination of a large number of weak predictors enables the creation of a single, strong predictor. The boosted term refers to the practice of focusing on samples from the training set which are poorly predicted by the previous model structure, and this is an iterative practice that leads to significantly longer training times than RFs and DTs.

Tree-based models were selected for this work due to their favorability in large parameter spaces and relatively low CPU time,[11,12] their comparable results to ANNs,[12,13] and their ability to act as a

*Electrochemical Society Student Member.
**Electrochemical Society Member.
zE-mail: vsubram@uw.edu

sensitivity analysis using feature importance, which improves the ability to construct new features in a process called feature engineering. In this context, a feature is any single column of input variables – a list of values which are of constant type and location in the input vector. In general, RFs and DTs are extremely practical for prototyping due to their flexibility, rapid training times, and rapid execution times. To the best of our knowledge, studies reporting these techniques used for building functional relationships among inputs and outputs for battery models are very rare, though some studies using several versions of static and dynamic ANNs have surfaced in the recent past.[14,15]

The performance of lithium-ion batteries is heavily dependent upon the operating conditions present during their use in addition to the states of several internal variables. This sensitivity has driven the development of a wide range of models to simulate battery behaviors, from computationally expensive molecular dynamics simulations down to simple empirical models, which trade predictive fidelity for decreased computation time.[16,17] Between these extremes lie a variety of continuum-scale models, which include the single particle model (SPM),[18,19] and pseudo two-dimensional (P2D) model,[16,20,21] the latter of which represents a good tradeoff between physical fidelity, predictive validity across chemistries, and execution time.[22]

Empirical models are generally simple functions which have been fit to experimental data and are used to predict future battery performance, and often perform very poorly outside of their narrow window of accuracy. Polynomial, trigonometric, logarithmic, and exponential fits to experimental data are examples of empirical models which may offer some form of local accuracy. Equivalent circuit (EC) models are a class of empirical models which combine a series of linear circuit elements in order to approximate the behavior of a battery. Typically used in state of charge (SOC) estimation, ECs are the one of the most widely implemented battery models due to their simplicity and speed of calculation.[23] SOC, defined as the fraction of total capacity remaining in the battery and represented as a percentage ranging from 0% to 100%, cannot be directly measured from a battery and must be inferred or modeled.

$$\mathrm{SOC}\,(t) = \frac{Q\,(t)}{Q_n} \qquad [1]$$

When higher fidelity or higher charge rates are needed, the P2D model is typically used. Based on the principles of electrochemistry, transport phenomena, and thermodynamics, the P2D model is represented by coupled nonlinear partial differential equations (PDEs) which vary with respect to electrode thickness x, particle radius r, and time t. The predictive capabilities of the model are improved by the inclusion of internal variables, including electrolyte concentration, electrolyte potential, solid-state potential, and solid-state concentration within the porous electrodes, as well as the concentration and potential of the electrolyte within the separator. This higher fidelity model typically solves on the order of minutes. In an effort to make these models easier to implement, faster to solve, and to allow for greater flexibility of application, surrogate models will be created using ML algorithms.

In this work, RFs, DTs, and GBM based surrogate models are created and their abilities to predict the dynamic behavior of the physics-based model are examined. The most comprehensive P2D model has been utilized to create the data set for this study and the results are analyzed for accuracy and execution time. Trade-offs among training time, execution time, and accuracy for different ML algorithms are reported. Although surrogate models based on the P2D model are demonstrated in this paper, the concept is applicable for detailed 2D and 3D models for batteries, including multiscale thermal models.

The rest of the paper is organized in the following fashion – the second section discusses the formulation of P2D model, the associated parameters and values of those parameters, and how the complete parameter set can be reduced to a set of most important parameters. The third section presents the basics of the machine learning algorithms used in this work, including an overview of the techniques and details about the structures they create. Results are discussed in the fourth section, followed by a perspective of where machine learning algorithms can fit into the context of numerical modeling and surrogate modeling in the fifth section, and a summary of the findings can be found in the conclusions in the sixth section.

Choice of the Model

The model used to generate this data set is a Newman-type P2D porous electrode model, as described in other works.[16,20,21] For convenience, a summary of equations and parameters will be recounted in Table I, Table II, and Table III here.

The P2D model is favored in literature[16,24] due to its balance between accuracy and relatively low execution time, typically 80 seconds when solved using finite difference techniques with 50,20,50 node points in cathode, separator, and anode, respectively. The model describes the intercalation and deintercalation of lithium particles from spherical anode and cathode particles in combination with ionic and electronic conductivities of the anode and cathode materials and electrolyte. The model includes effects from porosity of the anode, cathode, and separator, and also includes effects related to the diffusivity of ions through the electrolyte solution. Rates of reaction are modeled using Butler-Volmer relationships and the potentials are dictated by open circuit potentials, which are electrochemical properties of the materials that constitute the anode and cathode.

The parameter count for this model is 46, including the 20 values used for a piecewise-continuous linear fit for the open circuit potential, Up. In an effort to reduce the parameter space, the shape of this curve was approximated prior to data generation by iteratively varying the values and selecting the one with the lowest absolute error. All of the values in the linear piecewise function were then scaled from 0.95 to 1.05 to allow for variance in the data set. This reduced the dimensionality to 27, while still retaining the flexibility associated with variance in the open circuit potential. Practically, the dimensionality of the model can be further reduced using sensitivities from the ML models, which will be discussed further in the fourth section.

Surrogate Models from P2D Models

In this section, the processes of building surrogate models using ML techniques and the intricacies associated with them are described in detail. A flowchart representing the methodology is shown in Figure 1. When attempting optimization and real-time control using a surrogate for a computationally expensive model, the first step is to create a data set for training, testing, and validating the surrogate model. Here, the size of the data set is 24,000 individual trials with variance across 27 parameters as dictated by a linearly-scaled Sobol sampling arrangement, implemented using the Julia package Sobol.jl.[25,26] The ranges for each parameter are shown in Table III and were chosen based on a combination of the sensitivity of the model to these parameters and value ranges taken from reasonable percentage deviations from values from literature for nickel manganese cobalt oxide (NMC) type batteries,[16] and others were estimated using conventional optimization techniques. The total time of data generation was 533 CPU-hours spread across 12 threads in an i7-6800k running at 3.8 GHz.

Practical considerations when working with RFs include limiting the number of features used, limiting the size of the data set, and limiting the number of DTs in the forest. Training a RF is extremely parallelizable, but CPU time and RAM requirements scale linearly with the size of the forest and the number of output values, and can quickly out scale typical hardware capabilities, occasionally requiring over 30 Gigabytes of RAM for a single model.

Three types of models are created, which will be henceforth referred to as constant time forward, constant time inverse, and recurrent. The forward surrogate model takes in a vector of parameters and outputs a voltage discharge curve, acting as a faster version of the physics-based model. The inverse model takes a voltage discharge curve as an input and estimates a set of parameters that were used to create that discharge curve, allowing for O(1) parameter estimation. The recurrent model takes a vector of parameters and voltages from previous times as an input and estimates the next voltage, or

Table I. Transformed Governing Equations.

Governing Equation	Boundary Conditions
Positive Electrode	
$\varepsilon_{\mathrm{p}} \frac{\partial c_p}{\partial t} = \frac{1}{l_p} \frac{\partial}{\partial X} [\frac{D_{\mathrm{eff,p}}}{l_p} \frac{\partial c_p}{\partial X}] + a_{\mathrm{p}}(1 - t_+) j_{\mathrm{p}}$	$\begin{gathered} \frac{\partial c_p}{\partial X}\vert_{X=0} = 0 \\ \frac{-D_{\mathrm{eff,p}}}{l_p} \frac{\partial c_p}{\partial X}\vert_{X=1} = \frac{-D_{\mathrm{eff,s}}}{l_s} \frac{\partial c_s}{\partial X}\vert_{X=0} \end{gathered}$
$\frac{-\sigma_{\mathrm{eff,p}}}{l_p} \left(\frac{\partial \Phi_{1,p}}{\partial X}\right) - \frac{\kappa_{\mathrm{eff,p}}}{l_p} (\frac{\partial \Phi_{2,p}}{\partial X}) + \frac{2\kappa_{\mathrm{eff,p}} RT}{F} \frac{(1-t_+)}{l_p} (\frac{\partial \ln c_p}{\partial X}) = I$	$\begin{gathered} \frac{\partial \Phi_{2,p}}{\partial X}\vert_{X=0} = 0 \\ \frac{-\kappa_{\mathrm{eff,p}}}{l_p} \frac{\partial \Phi_{2,p}}{\partial X}\vert_{X=1} = \frac{-\kappa_{\mathrm{eff,s}}}{l_s} \frac{\partial \Phi_{2,s}}{\partial X}\vert_{X=0} \end{gathered}$
$\frac{1}{l_p} \frac{\partial}{\partial X} [\frac{\sigma_{\mathrm{eff,p}}}{l_p} \frac{\partial}{\partial X} \Phi_{1,p}] = a_{\mathrm{p}} F j_{\mathrm{p}}$	$\begin{gathered} (\frac{1}{l_p} \frac{\partial \Phi_{1,p}}{\partial X})\vert_{X=0} = -\frac{I}{\sigma_{eff,p}} \\ \frac{\partial \Phi_{1,p}}{\partial X}\vert_{X=1} = 0 \end{gathered}$
$\frac{\partial c_p^s}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} [r^2 D_p^s \frac{\partial c_p^s}{\partial r}]$	$\begin{gathered} \frac{\partial c_p^s}{\partial r}\vert_{r=0} = 0 \\ -D_p^s \frac{\partial c_p^s}{\partial r}\vert_{r=R_s} = j_p \end{gathered}$
$\rho_p C_{p,p} \frac{dT_p}{dt} = \frac{1}{l_p} \frac{\partial}{\partial X} [\frac{\lambda_p}{l_p} \frac{\partial T_p}{\partial X}] + Q_{\mathrm{rxn,p}} + Q_{\mathrm{rev,p}} + Q_{\mathrm{ohm,p}}$	$-\kappa_{eff,p} \frac{\partial T_p}{\partial X}\vert_{X=0} = h_{env}(T_p\vert_{X=0} - T_{air}) - \frac{\lambda_p}{l_p} \frac{\partial T_p}{\partial X}\vert_{X=1} = -\frac{\lambda_s}{l_s} \frac{\partial T_s}{\partial X}\vert_{X=0}$
Separator	
$\varepsilon_s \frac{\partial c_s}{\partial t} = \frac{1}{l_s} \frac{\partial}{\partial X} [\frac{D_{\mathrm{eff,s}}}{l_s} \frac{\partial c_s}{\partial X}]$	$\begin{gathered} c_p\vert_{X=1} = c_s\vert_{X=0} \\ c_s\vert_{X=1} = c_n\vert_{X=0} \end{gathered}$
$-\frac{\kappa_{\mathrm{eff,s}}}{l_s} (\frac{\partial \Phi_{2,s}}{\partial X}) + \frac{2\kappa_{\mathrm{eff,s}} RT}{F} \frac{(1-t_+)}{l_s} (\frac{\partial \ln c_S}{\partial X}) = I$	$\begin{gathered} \Phi_{2,p}\vert_{X=1} = \Phi_{2,s}\vert_{X=0} \\ \Phi_{2,s}\vert_{X=1} = \Phi_{2,n}\vert_{X=0} \end{gathered}$
$\rho_s C_{p,s} \frac{dT_s}{dt} = \frac{1}{l_s} \frac{\partial}{\partial X} [\frac{\lambda_s}{l_s} \frac{\partial T_s}{\partial X}] + Q_{\mathrm{ohm,s}}$	$\begin{gathered} T_p\vert_{X=1} = T_s\vert_{X=0} \\ T_s\vert_{X=0} = T_n\vert_{X=1} \end{gathered}$
Negative Electrode	
$\varepsilon_n \frac{\partial c_n}{\partial t} = \frac{1}{l_n} \frac{\partial}{\partial X} [\frac{D_{\mathrm{eff,n}}}{l_n} \frac{\partial c_n}{\partial X}] + a_n(1 - t_+) j_n$	$\begin{gathered} \frac{\partial c_n}{\partial X}\vert_{X=1} = 0 \\ \frac{-D_{\mathrm{eff,s}}}{l_s} \frac{\partial c_s}{\partial X}\vert_{X=1} = \frac{-D_{\mathrm{eff,n}}}{l_n} \frac{\partial c_n}{\partial X}\vert_{X=0} \end{gathered}$
$\frac{-\sigma_{\mathrm{eff,n}}}{l_n} (\frac{\partial \Phi_{1,n}}{\partial X}) - \frac{\kappa_{\mathrm{eff,n}}}{l_n} (\frac{\partial \Phi_{2,n}}{\partial X}) + \frac{2\kappa_{\mathrm{eff,n}} RT}{F} \frac{(1-t_+)}{l_n} (\frac{\partial \ln c_n}{\partial X}) = I$	$\begin{gathered} \Phi_{2,n}\vert_{X=1} = 0 \\ \frac{-\kappa_{\mathrm{eff,s}}}{l_s} \frac{\partial \Phi_{2,s}}{\partial X}\vert_{X=1} = \frac{-\kappa_{\mathrm{eff,n}}}{l_n} \frac{\partial \Phi_{2,n}}{\partial X}\vert_{X=0} \end{gathered}$
$\frac{1}{l_n} \frac{\partial}{\partial X} [\frac{\sigma_{\mathrm{eff,n}}}{l_n} \frac{\partial}{\partial X} \Phi_{1,n}] = a_n F j_n$	$\begin{gathered} \frac{\partial \Phi_{1,n}}{\partial X}\vert_{X=0} = 0 \\ (\frac{1}{l_n} \frac{\partial \Phi_{1,n}}{\partial X})\vert_{X=1} = -\frac{I}{\sigma_{eff,n}} \end{gathered}$
$\frac{\partial c_n^s}{\partial t} = \frac{1}{r^2} \frac{\partial}{\partial r} [r^2 D_n^s \frac{\partial c_n^s}{\partial r}]$	$\begin{gathered} \frac{\partial c_n^s}{\partial r}\vert_{r=0} = 0 \\ -D_n^s \frac{\partial c_n^s}{\partial r}\vert_{r=R_s} = j_n \end{gathered}$
$\rho_n C_{p,n} \frac{dT_n}{dt} = \frac{1}{l_n} \frac{\partial}{\partial X} [\frac{\lambda_n}{l_n} \frac{\partial T_n}{\partial X}] + Q_{\mathrm{rxn,n}} + Q_{\mathrm{rev,n}} + Q_{\mathrm{ohm,n}}$	$-\frac{\lambda_s}{l_s} \frac{\partial T_s}{\partial X}\vert_{X=1} = -\frac{\lambda_n}{l_n} \frac{\partial T_n}{\partial X}\vert_{X=0} - \kappa_{eff,n} \frac{\partial T_n}{\partial X}\vert_{X=1} = h_{env}(T_{air} - T_n\vert_{X=1})$

number of voltages. The extremely flexible nature of problem formulation allows for a variety of applications from the same data set.

The three types of models used are based heavily on the classification and regression trees (CART) algorithm, which greedily creates binary splits in order to grow trees in a top-down fashion, meaning that it begins at the inputs and ends with the outputs.[27] This process continues until the termination condition is reached, typically a maximum depth or perfect accuracy. The accuracy and training time of the models are functions of the size of the training set, the parameter space sampled in the data set, and several hyper-parameters of the models.

Surrogate model formulation.—The process begins with a loss metric for tree f, typically the mean squared error (MSE) between the predictions of an existing tree structure and the data, which is evaluated at each terminal node $\mathbf{T}$, where $\mathbf{L_j}$ represents the total loss at node $\mathbf{j}$.[27,28] In this context, each interior node in the tree represents a set of rules used to split the data, which is optimized to increase the predictability of the tree. For example, for some portion of the data, if variance in feature 25 can separate the discharge curves with high purity, a threshold will be chosen above which the data are sorted right and below which the data are sorted left. The rules are generally not as simple as a single feature value, and typically default to either considering every feature or a random number up to log (n_{features}), depending upon the implementation. The total number of trees, or weak learners, is represented by $\mathbf{n}$, and the potential attributes which can be used to split are represented by $\mathbf{I_j}$. The loss is effectively the sum of the error accumulated across each tree, $\mathbf{n}$, at each terminal node, $\mathbf{T}$, after considering the splits associated with each feature from the data set, whose indexes are represented by $\mathbf{I_j}$.

$$L(f) = \sum_{i=1}^{n} \sum_{j=1}^{T} \sum_{i \in I_j} L(y_i, w_j) \equiv \sum_{j=1}^{T} L_j \quad [2]$$

Table II. Additional Equations.

$$Q_{\text{rxn},i} = Fa_i j_i(\Phi_{1,i} - \Phi_{2,i} - U_i),\ i = p, n$$

$$Q_{\text{rev},i} = Fa_i j_i T_i \frac{\partial U_i}{\partial T},\ i = p, n$$

$$Q_{\text{ohm},i} = \sigma_{\text{eff},i}(\frac{1}{l_i}\frac{\partial \Phi_{1,i}}{\partial X})^2 + \kappa_{\text{eff},i}(\frac{1}{l_i}\frac{\partial \Phi_{2,i}}{\partial X})^2 + \frac{2\kappa_{\text{eff},i} RT_i}{F}(1 - t_+^0)$$
$$\frac{1}{l_i^2}\frac{1}{c_i}\frac{\partial c_i}{\partial X}\frac{\partial \Phi_{2,i}}{\partial X},\ i = p, n$$

$$Q_{\text{ohm,s}} = \kappa_{\text{eff},s}(\frac{1}{l_s}\frac{\partial \Phi_{2,s}}{\partial x})^2 + \frac{2\kappa_{\text{eff},s} RT_s}{F}(1 - t_+^0)\frac{1}{c_s}\frac{1}{l_s^2}\frac{\partial c_s}{\partial X}\frac{\partial \Phi_{2,i}}{\partial X}$$

$$U_i(T_i, \theta_i) = U_{\text{i,ref}}(T_{\text{ref}}, \theta_i) + (T_i - T_{\text{ref}})[\frac{dU_i}{dT}]|_{T_{\text{ref}}},\ i = p, n$$

$$D_{i,eff}^s = D_i^s \exp(-\frac{E_a^{D_i^s}}{R}[\frac{1}{T} - \frac{1}{T_{ref}}]), i = p, n$$

$$k_{i,eff} = k_i \exp(-\frac{E_a^{k_i}}{R}[\frac{1}{T} - \frac{1}{T_{ref}}]), i = p, n$$

$$U_n = 0.7222 + 0.1387\theta_n + 0.029\theta_n^{0.5} - \frac{0.0172}{\theta_n} + \frac{0.0019}{\theta_n^{1.5}}$$
$$+0.2808\exp(0.90 - 15\theta_n) - 0.7984\exp(0.4465\theta_n - 0.4108)$$

$$\kappa_{eff,i} = \varepsilon_i^{brugg}\begin{pmatrix} 4.1253 \times 10^{-2} + 5.007 \times 10^{-4}c - 4.7212 \times 10^{-7}c^2 \\ +1.5094 \times 10^{-10}c^3 - 1.6018 \times 10^{-14}c^4 \end{pmatrix},$$
$$i = p, s, n$$

$$D_{eff,i} = D \cdot \varepsilon_i^{brugg},\ i = p, s, n$$

$$\sigma_{eff,i} = \sigma_i \cdot \left(1 - \varepsilon_i - \varepsilon_{f,i}\right),\ i = p, n$$

$$a_i = \frac{3}{R_i}(1 - \varepsilon_i - \varepsilon_{f,i}),\ i = p, n;\ \theta_p = \frac{c^s|_{r=R_p}}{c_{\text{max},p}^s};\ \theta_n = \frac{c^s|_{r=R_n}}{c_{\text{max},n}^s}$$

A proposed split at node **k**, such that **k = / = j**, is done by maximizing gain, or the difference in loss before and after the split. In this way, new splits are created without redundancy, where the gain is defined as the difference between the previous loss associated with node **k**, $\mathbf{L_k}$, and the losses associated with the new terminal nodes which are the left and right splits, $\mathbf{L_{kL}}$ and $\mathbf{L_{kR}}$, respectively. This process is demonstrated in Figure 2.

$$Gain = L\left(f_{before}\right) - L\left(f_{after}\right) = L_k - (L_{kL} + L_{kR}) \qquad [3]$$

This process establishes the structure of the DT or RF but does not instantiate any of the nodes with weights. A separate process optimizes the weights at each node, $\mathbf{w_j}$, such that the total loss is minimized.

$$w_j = \arg\min_w \sum_{i \in I_j} L(y_i, w) \qquad [4]$$

This is a typical stopping criterion for a DT, which contains only one structure. This can be further extended and used as the termination criterion for a RF by repeatedly solving for w_j for additional structures until $\mathbf{n_{estimators}}$ is reached. The structures for these models are very large and will go on to perfectly memorize the training data set unless an artificial limit is imposed. For GBMs, this process is iteratively repeated using a boosting algorithm, which greedily fits additional small (e.g. max depth < 5) structures until the $\mathbf{n_{estimators}}$ limit is achieved. This iterative nature leads to considerably longer training times than either DTs or RFs. However, the ensemble of multiple weak and varied predictors allows for better generalizability than either DTs or RFs.[27] Formally, GBMs are a solution to the below problem statement, where $\boldsymbol{\theta_m}$ is the weights for the structure $\boldsymbol{\phi_m}$ is a list of which minimizes the

Table III. Parameter Value Ranges.

Symbol	Parameter	Positive Electrode	Separator	Negative Electrode	Units
σ_i	Solid phase conductivity	7.00238–14.2888		70.0044–142.855	S/m
$\varepsilon_{f,i}$	Filler fraction	0.0174523–0.0356141		0.0229289–0.0467899	
ε_i	Porosity	0.380838–0.413231	0.3124–0.637511	0.256371–0.523131	
Brugg	Bruggeman Coefficient	1.05–2.14	1.05–2.14	1.05–2.14	
D	Electrolyte diffusivity	2.4e-11–1.5e-07	2.4e-11–1.5e-07	2.4e-11–1.5e-07	m^2/s
D_i^s	Solid Phase Diffusivity	9.62383e-16–7.37308e-12		1.38021e-15–7.20549e-12	m^2/s
k_i	Reaction Rate constant	1.35627e-11–1.34804e-09		1.03462e-11–1.03141e-09	$mol/(s\ m^2)\ /(mol/m^3)^{1+\alpha a,i}$
$c_{i,\text{max}}^s$	Maximum solid phase concentration	43996.0–47738.3		29027.3–32163.0	mol/m^3
$c_{i,0}^s$	Initial solid phase concentration	15616.7–21880.8		25563.1–32055.5	mol/m^3
c_0	Initial electrolyte concentration		1140.0–1263.14		mol/m^3
$R_{p,i}$	Particle Radius	3.97514e-06–1.58973e-05		4.98866e-06–1.99519e-05	m
l_i	Region thickness	4.10747e-05–4.36543e-05	1.282e-05–2.003e-05	4.5559e-05–4.6484e-05	m
t_+	Transference number		0.31375–0.387338		
F	Faraday's Constant		96487		C/mol
R	Gas Constant		8.314		J/(mol K)
T_{ref}	Temperature		298.15		K
P	Density	2500	1100	2500	kg/m^3
C_p	Specific Heat	700	700	700	J/(kg K)
Λ	Thermal Conductivity	2.1	0.16	1.7	J/(m K)
$E_a^{D_i^s}$	Activation Energy for Temperature Dependent Solid Phase Diffusion	5000		5000	J/mol
$E_a^{k_i}$	Activation Energy for Temperature Dependent Reaction Constant	5000		5000	J/mol

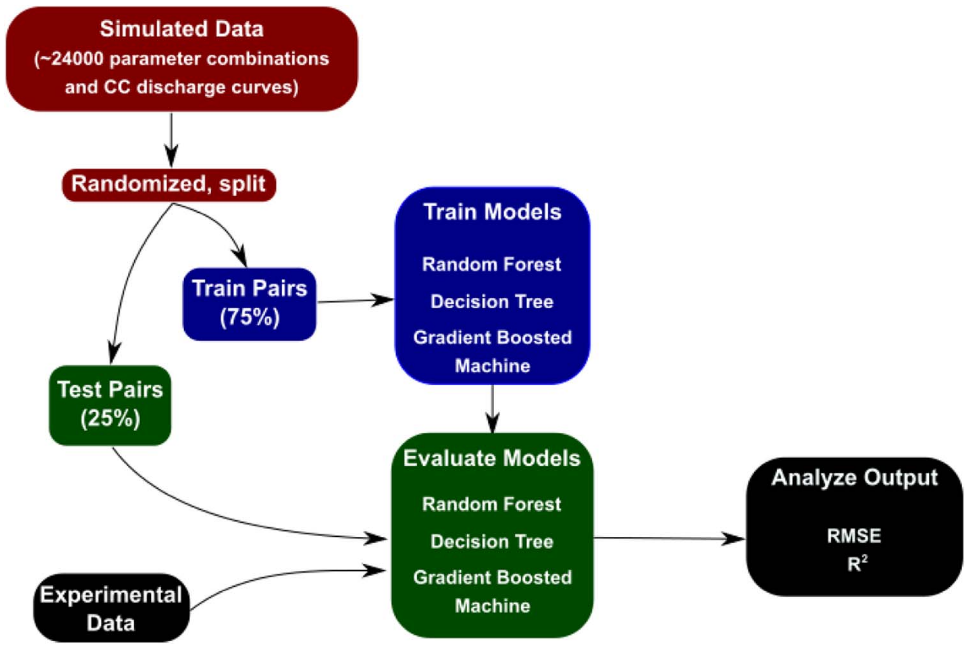


Figure 1. Process Flowchart for the Creation of Surrogate Models from Simulated Data.

loss function.

$$\{\theta_m, \phi_m\} = \arg\min_{\{\theta_m,\phi_m\}} \sum_{i=1}^{n} L\left(y_i, f^{(m-1)}(x_i) + \theta_m \phi_m(x_i)\right) \quad [5]$$

The boosting process requires a gradient calculation of the loss in function space $\mathbf{g_m(x)}$, which is calculated by fitting a regression tree model using MSE as its loss function.

$$-g_m(x) = -\left[\frac{\partial L(y, f(x))}{\partial f(x)}\right]_{f(x)=f^{(m-1)}(x)} \quad [6]$$

$$\phi_m = \arg\min_{\phi} \sum_{i=1}^{n} [(-g_m(x_i)) - \phi(x_i)]^2 \quad [7]$$

In addition to the direction of the step, a step length $\boldsymbol{\rho_m}$ must also be determined. This is typically done using a line search algorithm which inherits a learning rate from the initial GBM call.

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{n} L\left(y_i, f^{(m-1)}(x_i) + \rho\phi_m(x_i)\right) \quad [8]$$

$$f_m(x) = \eta\rho_m\phi_m(x) \quad [9]$$

Finally, these concepts are implemented in the python package Sci-Kit Learn.[23]

Building the models – constant time models.—Constructing a forward model is a multi-input, multi-output problem, which disqualified GBMs from this section, as their current implementation is limited to a single output. While an ensemble of GBMs on the basis of single output model could have been created, the training time was prohibitive. When implementing a machine learning model, it is important to format the data such that the containers are of constant size. In this case, the discharge voltage was sampled at 100 points using linear interpolation, with the time value at the end of the vector equal to 1800 seconds such that samples which overshoot the final experimental time of 1600 seconds can still be fully captured. In the case of the forward models, the inputs are the physical parameters, as shown in the second section, and the output is a time-scaled voltage discharge curve.

Construction of the inverse models was similar to that of the forward models, however, the inputs and outputs were switched. In this way, it was possible to use the inverse models for O(1) parameter estimation for the surrogate or physics-based models. Due to the reduced parameter space, it was also feasible to create per-parameter GBMs.

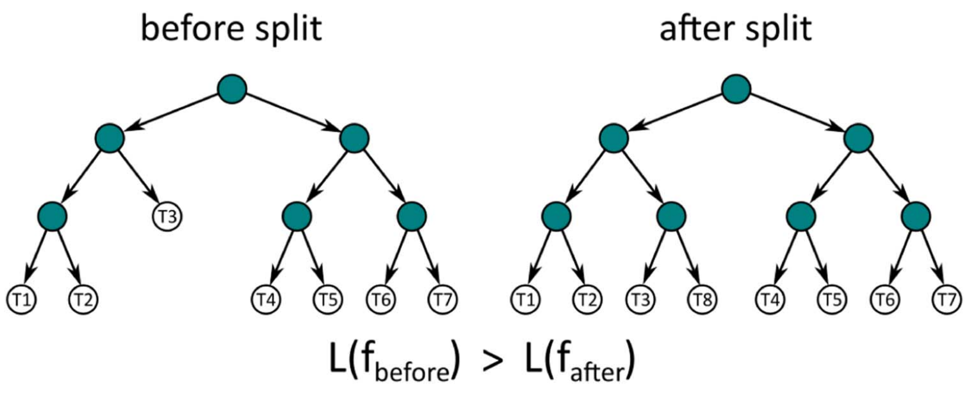
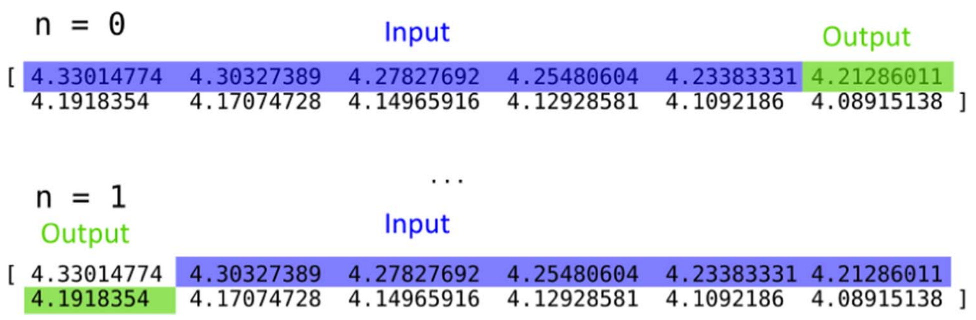


Figure 2. Visual representation of addition of splits during tree structure building. A split is created at T3 and reduces the total loss L, so it is kept.



```
[ 4.33014774  4.30327389  4.27827692  4.25480604  4.23383331 4.21286011
  4.1918354   4.17074728  4.14965916  4.12928581  4.1092186   4.08915138 ]
```

```
[ 4.33014774  4.30327389  4.27827692  4.25480604  4.23383331 4.21286011
  4.1918354   4.17074728  4.14965916  4.12928581  4.1092186   4.08915138 ]
```

Figure 3. Representation of windowed clustering for creation of recurrent data set. For each time step, n voltages are taken as the input and the next voltage is taken as the output. In this case, n = 5.

Building the models – recurrent models.—The extreme flexibility of machine learning models allows for more creative uses of a typical data set than matching or swapping input and output. In order to create a recurrent model, the training and test sets need to be manipulated in such a way that time or previous voltages are used as inputs to predict some next range of voltages. This is done by first disassembling the vectors of voltages and then reassembling the data such that the input values contain a list of parameters and $\boldsymbol{\alpha}$ voltages, and the output value is the next voltage. In Figure 3 below, and in the later models, $\boldsymbol{\alpha}$ is equal to 5. This can be modified to predict SOC by simply generating a value for the SOC at each voltage point and substituting the SOC value for the predicted voltage value. While GBMs are restricted to single value prediction, DTs and RFs can easily estimate anything related to the data set. In the equation below, new X and Y arrays are constructed using the old arrays which use a set of parameters and previous voltages to predict the next voltage.

$$X_{new}[i] = [X_{old}[i, n : n + \alpha], Y[i]],$$
$$Y_{new}[i] = X\_old[i, n + \alpha + 1] \qquad [10]$$

Results and Discussion

Effect of sample size.—When splitting the data into test and training sets, it is advantageous to randomize the trials to minimize the effects of sampling bias. In this data set, the individual trials were randomized in batches of 100, such that smaller subsamples of the data cannot contain runs from later batches. For example, when selecting only the first 1000 trials, the training and test sets are randomized, but do not contain samples from simulations 1000–24000, as they might be in a single batch randomization. In this way, it is possible to sequentially give more information to an RF or DT and examine the effects of a larger training set, as demonstrated in Figure 4.

Constant time models.—The more traditional form of surrogate models are effectively constant time models, in that they take in a set of parameters and output the entire discharge curve at once, rather than solving for the next time step iteratively, as is the case with the recurrent models. The recurrent models can be used to iteratively rebuild the constant time model result for a given set of parameters, but doing so is a slower implementation of the same constant time model, as the function must be called for each discretization of time rather than once. The constant time surrogate models perfectly learn the training data set, and any new set of parameters, like a sample from the test set, yields a curve from the training set whose parameters are similar in value. Depending upon the coarseness of the parameter sampling, this may or may not yield a good result.

DTs are known as constant output estimators, meaning that they allow for continuous inputs, but they do not interpolate between the data points, the values are connected in a way similar to a step function. In other words, although the input space is continuous, the output space can only have the same values as the discrete values in the training set. As such, an optimized result of a surrogate model using a DT will yield, at best, the closest result from the training set. While this is a good result, the act of using a DT contributes no tangible benefits over calling a look-up table of the training data, other than a significant speed increase. The closest result from the lookup table is used as a benchmark in other problem formulations, so these results are not shown here.

Since the constant time models exclusively use the parameters to estimate the output curves, their ability to predict the output voltages is inherently dependent upon their ability to identify meaningful parameters and correlate them to the changes they cause. The physics-based model is more sensitive to some parameters than others, and this sensitivity carries over to the surrogate models. RFs are able to report this relative sensitivity in a metric called feature importance, the definition of which can vary by package. In this instance, it is defined as the total number of times that feature is used in a split divided by the total number of splits. A feature is a single input from the input vector, meaning that each parameter in these constant time models is a feature. In addition to the given features, it is also possible to create artificial features that are combinations of your inputs in a practice known as feature engineering.

Feature engineering is one of the most overlooked and most important facets of machine learning.[6] It entails findings relationships between features which can better represent the trends in the data than the individual features themselves. Other works[29] have already demonstrated that the types of engineered features that can be learned by various models can differ, and helps give insight into which types of features to look for. If RFs can learn the relationship between an input x and an output x^2, for example, there is no need to explicitly create the feature x^2 in the input set, as no new information will be added. This can help reduce the dimensionality of the feature engineering problem.

Since this data is simulated, the equations themselves can be a good source of inspiration for engineered features. Upon looking in the equations, some of the below relationships became apparent.

Physically building these features requires creating a new array containing the old features and having room for the new ones, and then adding in the values which represent the combinations of features to complete the array. This process can be vectorized trivially for performance improvements, as shown in Equation 11. Figure 5 demonstrates the feature importance of the entire spectrum of features, which includes 26 of the original 27 parameters, where the single parameter representing the scale of the OCP has been expanded to the actual values, bringing the parameter total up to 46. Additional parameters are those represented in Table IV, in order of appearance. Only some of the features have a strong feature importance, demonstrating that these features have added patterns that accurately represent trends in the data. The algorithm score as a function of importance threshold is shown in Figure 6.

The best possible result would come from exhaustively creating and trying every permutation of feature combinations, but that is an intractable amount of features and is generally not computationally feasible. Even with only 27 parameters, and for a single relationship, like division, this represents 2^{27} parameter combinations. When

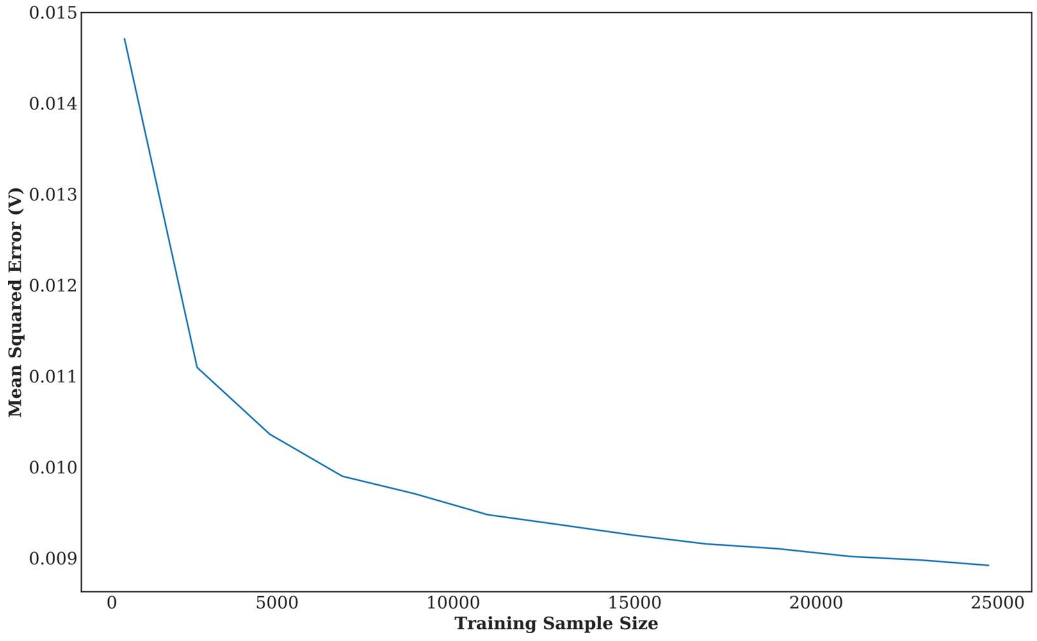


Figure 4. Effect of training size on test set error for a RF of size n = 100.

higher-order interactions are considered, manual feature engineering becomes the only option. To demonstrate this, both processes have been done – 40 manual features have been created using relationships found in the data and various dimensionless groups, and ∼250 features have been generated by randomly selecting 2 features and randomly multiplying or dividing them. This has been done in 3 batches of sizes 150, 50, and 50, in order to allow for the later trials to create more complex interactions by potentially sampling from parameter groups.

$$F_{new}\,[:] = Y_{old}\,[:, 5]\,.*\,Y_{old}\,[:, 6]\,./Y_{old}\,[:, 7] \qquad [11]$$

Once engineered features have been added, it is important to remove any features which are not contributing to the accurate prediction

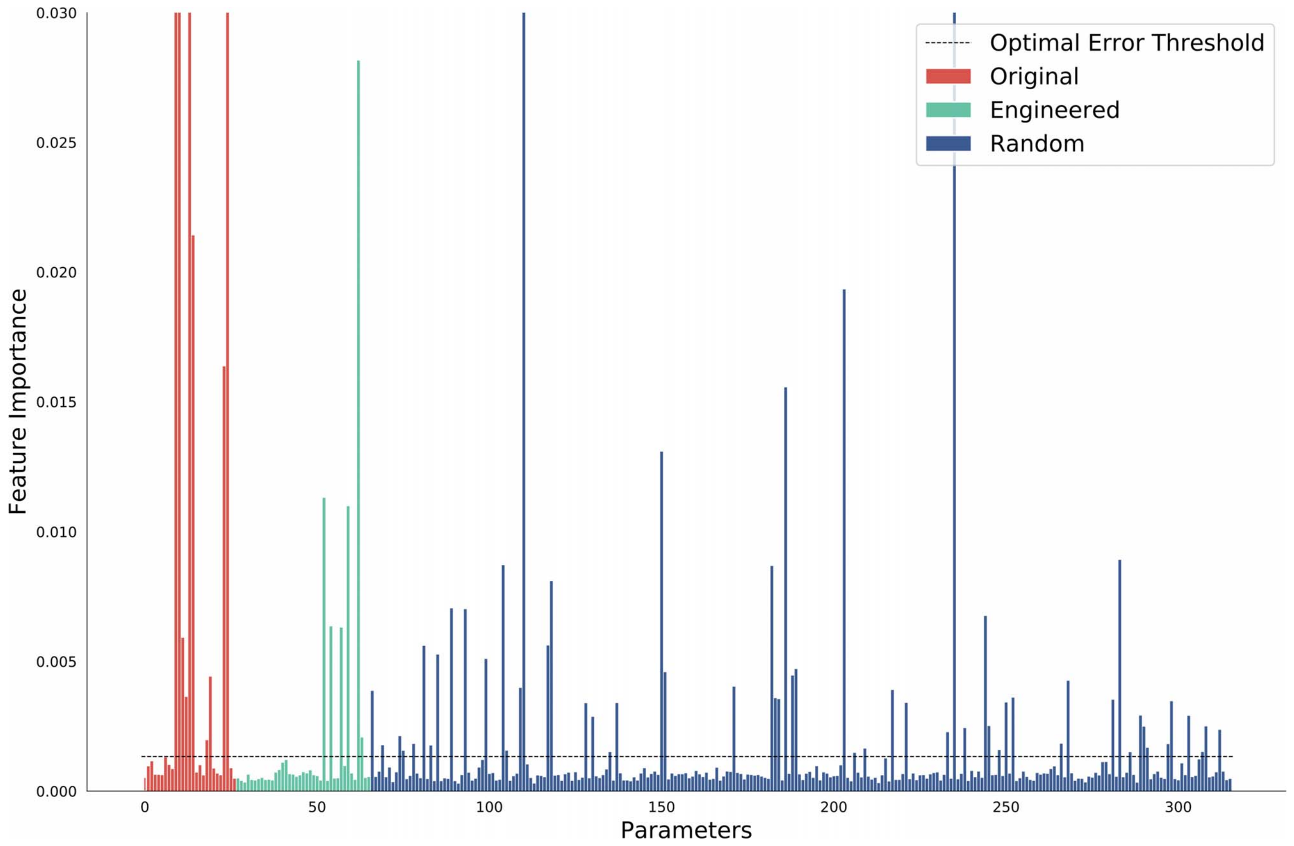


Figure 5. Per-parameter feature importance. Parameters 0–27 represent the original input parameters, 28–66 represent the engineered parameters, and 67–316 represent the randomly created parameters. Clipped at 0.030 for visibility.

Table IV. Engineered features, groups with importance > 0.0064 bolded.

Number	Relationship	Positive Electrode	Separator	Negative Electrode	Units
1,2,3	$\frac{\epsilon_i^{Brugg} D}{{l_i}^2 \epsilon_i}$	X	X	X	1/s
4,**5**	$a_i(1-0.1t_+k_i)$	X		**X**	$m^2/m^3(mol/(s\ m^2)\ /(mol/m^3)^{1+\alpha a,I})$
6,7	$a_i(1-0.1t_+k_i)\sqrt{c_0}$	X		X	$m^2/m^3(mol/(s\ m^2)\ /(mol/m^3)^{1+\alpha a,I})(mol/m^3)^{0.5}$
8,9	$a_i(1-0.1t_+k_i)\sqrt{c_0}\sqrt{c_{i,0}^s}$	X		X	$m^2/m^3(mol/(s\ m^2)\ /(mol/m^3)^{1+\alpha a,I})(mol/m^3)$
10,11,12	$\frac{\epsilon_i^{Brugg} D}{l_i}$	X	X	X	m/s
13,14	$\frac{\sigma_i}{l_i}$	X		X	s/m^2
15,17,19	$\frac{\epsilon_i^{Brugg}}{l_i}$	**X**	X	X	1/m
16,18,20	$\frac{\epsilon_i^{Brugg}(1-0.1t_+)}{l_i}$	**X**	X	X	1/m
21,22	$\frac{\sigma_i}{l_i^2}$	X		X	s/m^3
23,24	$\frac{D_{i,eff}^s}{{R_{p,i}}^2}$	X		X	1/s
25,26	$\frac{k_i}{\sqrt{c_0}}$	X		X	$mol/(s\ m^2)/(mol/m^3)^{1+\alpha a,i}/(mol/m^3)^{0.5}$
27,29	$\frac{{R_{p,i}}^2}{D_{i,eff}^s}$	**X**		X	S
28,30	$\frac{{R_{p,i}}^2}{\epsilon_i^{Brugg} D}$	**X**		X	S
31,**32**	$\frac{R_{p,i}}{2k_i c_0^{0.5}}$	X		**X**	$m/(mol/m^3\ mol/(s\ m^2)\ /(mol/m^3)^{1+\alpha a,i})$
33,**34**	$\epsilon_i c_{i,0}^s c_0$	X		**X**	$(mol/m^3)^2$
35,36	$\frac{\epsilon_i^{Brugg} l_i}{\epsilon_s^{Brugg} l_s}$	X		X	m/m
37	$\boldsymbol{\frac{\epsilon_p}{\epsilon_n}}$				1
38	$\boldsymbol{\frac{l_p}{l_n}}$				m/m
39,40	$\frac{1}{R_{p,i}}$	X		X	1/m

of new data in a process called feature pruning.[30] These features, which have no true predictive value, can only make contributions to overfitting, and cannot be relied upon to accurately predict the test data. By removing features whose importance is below a certain threshold, the accuracy of the validation set is improved. At the two pruning extremes, the model would use either all the features or one of the features, and response of the score to the pruning threshold should be a U-shaped curve, with a maximum at an intermediate value, as implied by the bias-variance tradeoff.[31] With increasing feature count, the variance is reduced, but if some features contain little predictive power, their effects may only be seen in the training set and not in the test set, leading to increased bias, also called over-training.

As demonstrated in Figure 6, there is an optimal range of threshold values where features which contribute exclusively to overfitting are discarded, but features which add valuable information are retained, increasing the accuracy on both seen and unseen data. The lines in Figure 5 and Figure 6 are drawn at the same importance threshold, which is determined by selecting the value that corresponds to the

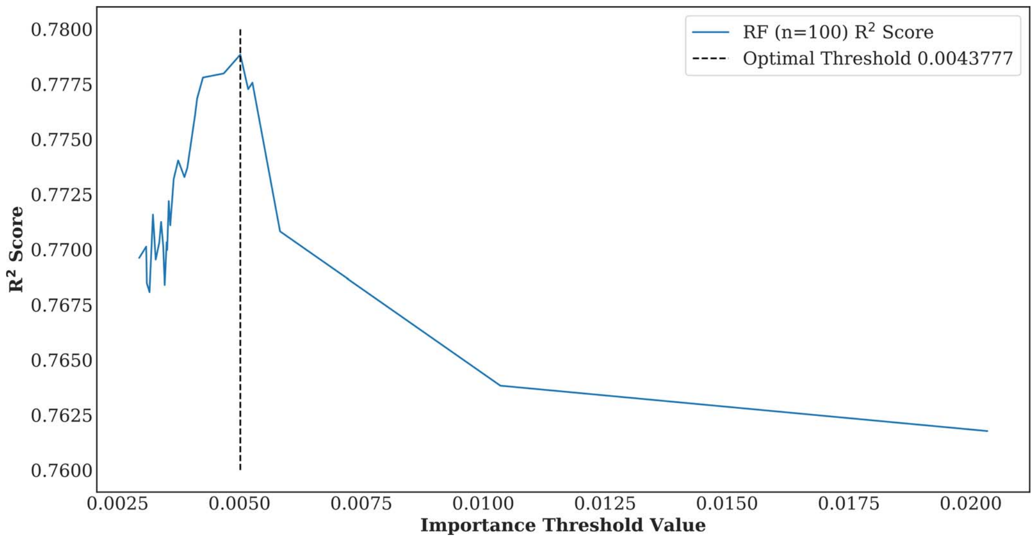


Figure 6. Random forest score with varying feature importance threshold, obtained from model trained on original and engineered features.

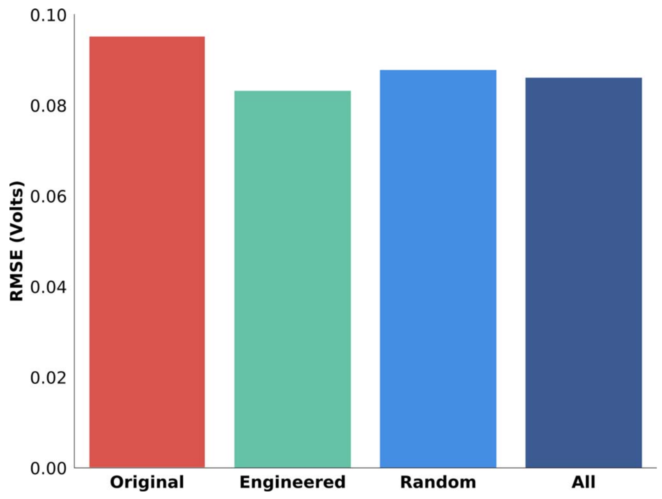


Figure 7. Test data set RMSE of identical models with original, original and engineered, original and random, or all features.

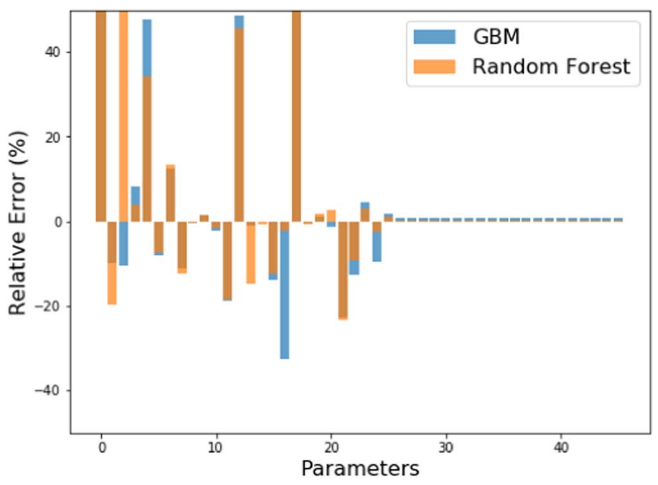


Figure 8. Relative error with respect to every parameter for inverse model predictions, cropped at ± 50% error for visibility.

highest R^2 score, where the R^2 value represents the quality of the fit of the line y = x on a plot of predicted values vs true values. In this instance, a value of 1.0 is perfect and a value of 0.0 is uncorrelated. Of the 40 hand-created features, 5 represented high scoring features, compared to 13 of the 250 randomly generated features and 7 of the initial 27 features. It is worth nothing that the feature importance is relative to the other features, and does not define an absolute relationship between any particular input with the outputs.

Initially, it seemed that feature importance was an extremely reliable method of feature pruning, and while it should certainly be considered, it is not enough for a new feature to simply have a high sensitivity – it must also describe a relationship that is not currently described in the existing features. For instance, the most important feature is socp, or initial state of charge in the positive electrode, with a feature importance of 0.206. This feature is so important that its presence in another random group, like $\frac{1}{socp}$, can have a high selectivity without necessarily generating new useful information. In this instance, the addition of the randomized features actually decreases the training accuracy as these artificially high-scoring parameter combinations can edge out features which may be less selective but which may add meaningful relationships. By pruning the original features, it is possible to achieve a test set RMSE of 0.0953 volts. Adding in hand-engineered features can reduce this to a test set RMSE of 0.0832 volts, while the addition of randomly created features, even with the hand-engineered features present, raises this again to an test set RMSE 0.0861 volts. When only the original and randomized features are used, the test set RMSE peaks at 0.0878 volts, as shown in Figure 7. Despite this, it can be seen in Figure 5 that many of the randomly created features have a very high feature importance when compared to the hand-created features. As such, feature engineering is a useful tool, but having features with higher importance does not guarantee a better result on the test data set.

In addition to engineering new features, feature scaling is also an important attribute. Due to the loss function lacking a normalization term, the data must be scaled such that error can be adequately represented in the loss function, even for small numbers. As a concrete example from this data set, the parameter $\mathbf{k_n}$, representing the rate of reaction in the negative electrode, has a value of the order 10^{-14}. Due to its very small value, even extremely large changes in the relative value of the prediction will be counted as accurate guesses by the loss function.

In addition to using the forward model to create a discharge curve from a set of model parameters, it is also possible to train a model in reverse, such that for a given discharge curve, the model outputs the parameters used to create that curve, which enables an O(1) function for parameter estimation. Since this model is also trained using a DT, it is subject to the same limitations as the forwards models above, namely that the best result achievable is the closest fit from the training data set.

Table V. Root mean square error (RMSE) and execution times of inverse-model estimated discharge curves.

Model	Error (V)	Execution Time (s)
Training Set Fit	0.02049	4.98
Gradient Boosted Machine	0.1850	5e-6
Random Forest	0.1074	5e-4

Once this inverse model is trained, it is simple to apply to experimental data, which requires scaling the X-axis to the same scaled time as the training data and ensuring that the length of the vectors is the same. Once this is done, it can be directly predicted using the algorithms. By taking the closest result from the training set as the correct values, it is possible to score the algorithms on their accuracy, the results of which are shown in Figure 8 and whose scores and execution times are documented in Table V. The resulting discharge curves are shown in Figure 9. The voltage remaining at 2.5 V after discharge is not physically meaningful, but padding the relevant discharge information with values is necessary to keep a constant vector length

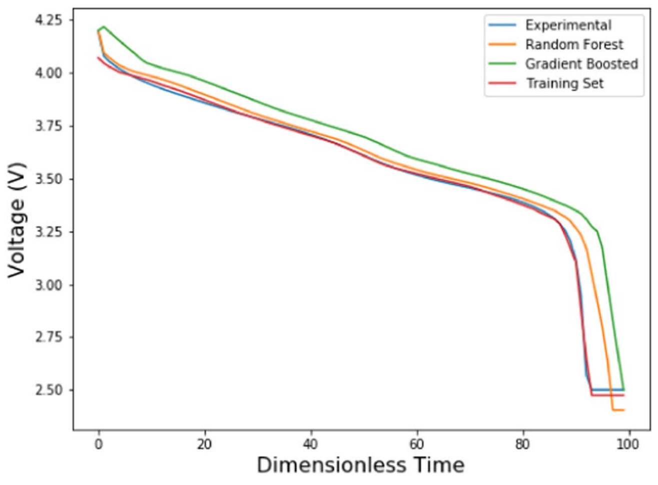


Figure 9. Simulated discharge curves associated with the estimated parameters.

Table VI. RMSE (Volts) as a function of distance from current voltage for experimental data, where nt represents the number of trees in the model.

Method	n+1	n+2	n+3	n+4	n+5
DT	0.0290	0.0333	0.0720	0.1068	0.1414
RF (nt = 100)	0.0314	0.0326	0.0609	0.0955	0.1345
GBM (nt = 500)	0.0319	0.0711	0.1058	0.1364	0.1640

for the model, so the minimum voltage from the discharge curve was used.

While this method is easy to implement, the results are not phenomenal. It is possible that with more data, or with a machine learning technique that is capable of true interpolation, that this technique could be more successful. However, it is apparent that even the closest fit from the training set is not a great fit to the experimental data. This implies that in order to have the potential for a better fit using this technique, it would be necessary to either generate more data or to reduce the sampled parameter space, which is difficult to do without knowing the values for the optimal fit. However, the speed increase is significant, even compared to the lookup table.

Recurrent voltage prediction.—One of the more interesting applications of the data set is the ability to create models for closed-loop, real-time control which are not possible using the set of equations in the physics-based model as-is. For example, it becomes possible to create models where the previous voltages are used to predict the next voltage, or the next series of voltages, and are updated at regular intervals. This allows for a fit which outperforms the training set in the upper portion of the discharge curve, but which overshoots the final voltage, as shown in Figure 10. The GBM RMSE is 0.0267 volts, and the RMSE of the closest training data is 0.0158.

By changing the number of voltage points predicted, it is possible to create a moving window of prediction where the errors can be used to quantify the confidence of each prediction. Table VI below represents the RMSE of the points as a function of distance from the voltage points used to predict. Interestingly, the GBM ensemble outperforms both the DT and RF at predicting the next voltage point of the experimental data, but does significantly worse predicting further voltages. This is demonstrated in Figure 11, where the downward choppiness of the prediction belies the tendency of the model to estimate an early termination of the discharge curve.

Due to the requirements of DTs, RFs, and GBMs for consistent dimensions of inputs, all information other than the direct model inputs are discarded. As such, the model has no record of previous discharge history beyond the previous 5 voltage values. There are two approaches to quantifying error using these methods: the error of the predicted points can be calculated sequentially using the experimental data as inputs, or the error can be calculated by iteratively calling the model using previously predicted values until the voltage has hit a certain value, and examining how that accuracy changes as a function of time. In other words, it is possible to sequentially predict only the next voltage and rebuild a single curve from this series using exclusively experimental data as input, as in Figure 11, and it is also possible to extend each individual prediction past a single point to an entire discharge curve, and to examine this series of curves, as done in the next section.

Recurrent SOC prediction.—So far, these methods have been applied exclusively to voltage prediction, but they can easily be adapted to estimation of state of charge. In the data set, rather than having voltage as a function of time, it is possible to simply coulomb count and give each time point a SOC measure. There are many different ways to structure the predictive model. Based on the control scheme, it may be desirable to have a SOC estimate for the current time, for a future time, or for both. The flexibility of machine learning allows for any formulation using any input, all generated from the same data set. In this analysis, the current SOC was predicted using [**n-4. . . n**] voltages.

Using the previously mentioned voltage discharge curve data set, another output array was created that represents the simulated battery SOC vs time. The original data were created using a constant 2C discharge, so the SOC is linearly decreasing at all points between the maximum voltage and the minimum voltage, as shown in Figure 12. The recurrent methods were created such that the present voltage and previous 4 voltage points can be used to predict the present SOC. The variance of the data set is an advantage when using previous voltages to predict the next voltage, as it allows for the model to adapt to

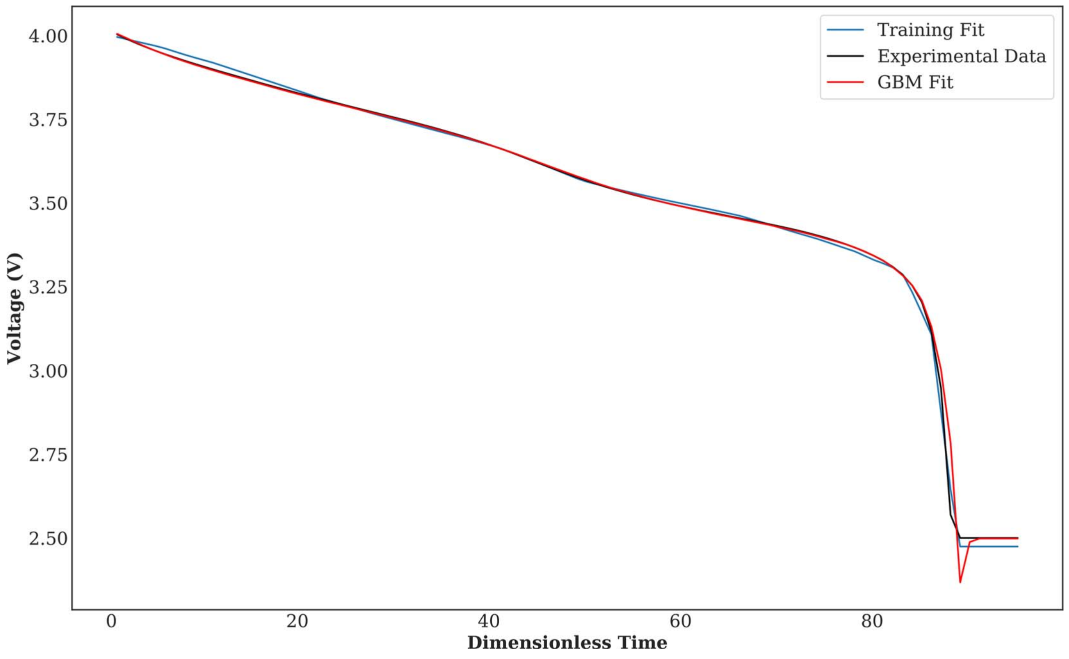


Figure 10. Gradient-boosted machine voltage estimation performance using 5 previous voltages to predict the next voltage, and the closest discharge curve from the training set.

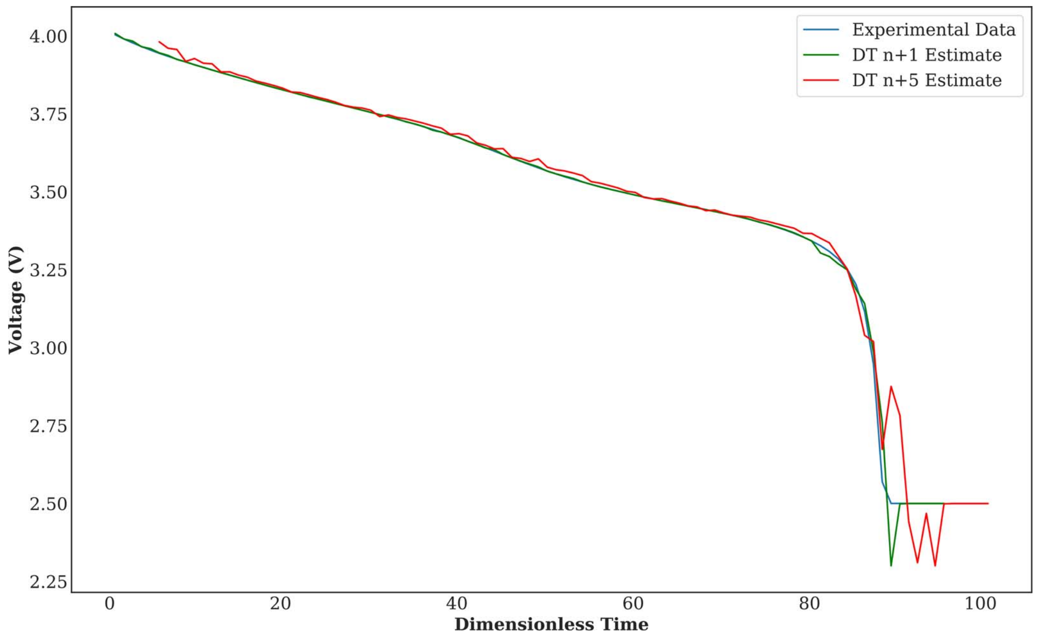


Figure 11. DT predicted voltage vs dimensionless time for n+1 and n+5. Using 5 input voltages, either the voltage at n+1 (green line) or the voltage at n+5 (red line) are predicted. This demonstrates the change in accuracy as a function of distance in time from n.

patterns that it has seen even if the entirety of the new curve is not in the training set. However, this high variance causes SOC estimation to be difficult due to the sensitivity of the calculation to the ending time of the simulated data.

To create the SOC data set, the simulated data that did not get below the typical SOC cutoff of 2.5 volts was omitted, which halved the size of the data set. The lowest voltage was taken as 0% SOC and the highest voltage was taken as 100% SOC. Since the current was constant for these simulations, the SOC decreased linearly with time. This decreased the uniqueness of the data set, simply requiring that two simulated batteries hit their lowest voltage at the same time to be considered identical.

Figure 13a gives a bit of insight into what is happening when the DT fits a given curve via a series of voltage predictions, the output of

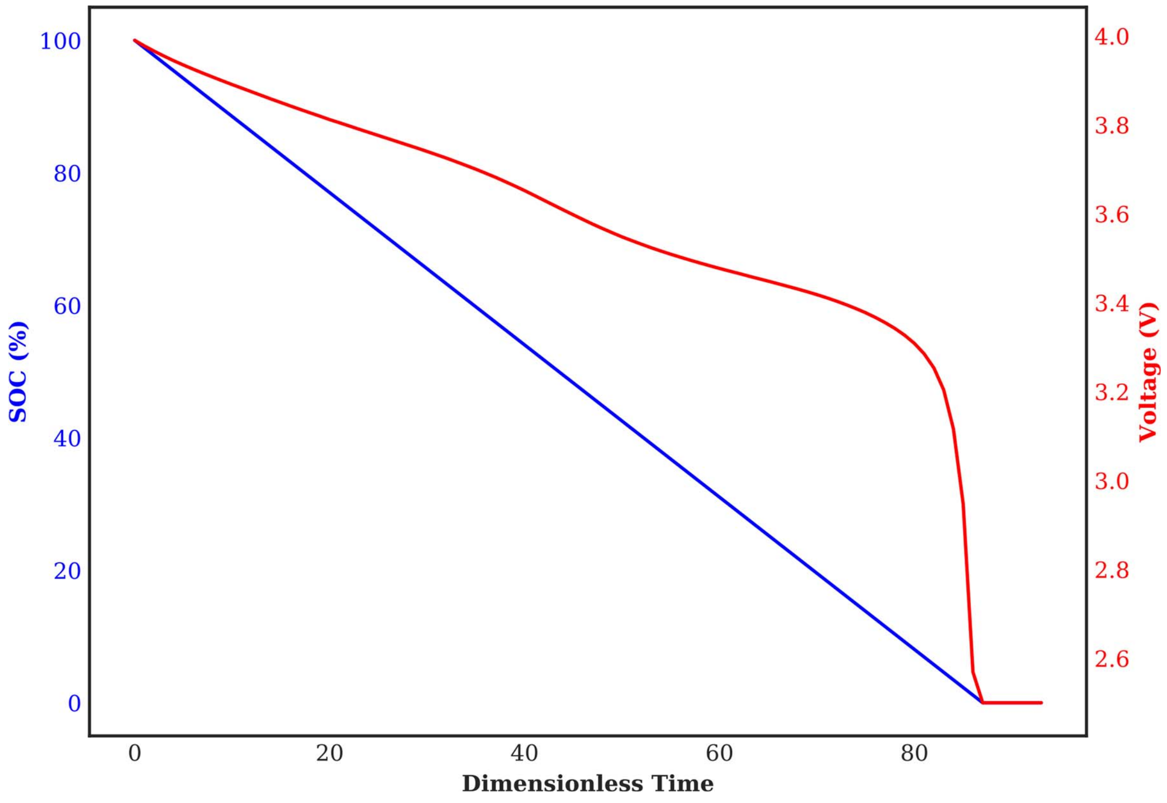


Figure 12. SOC and voltage relationship at constant current draw.

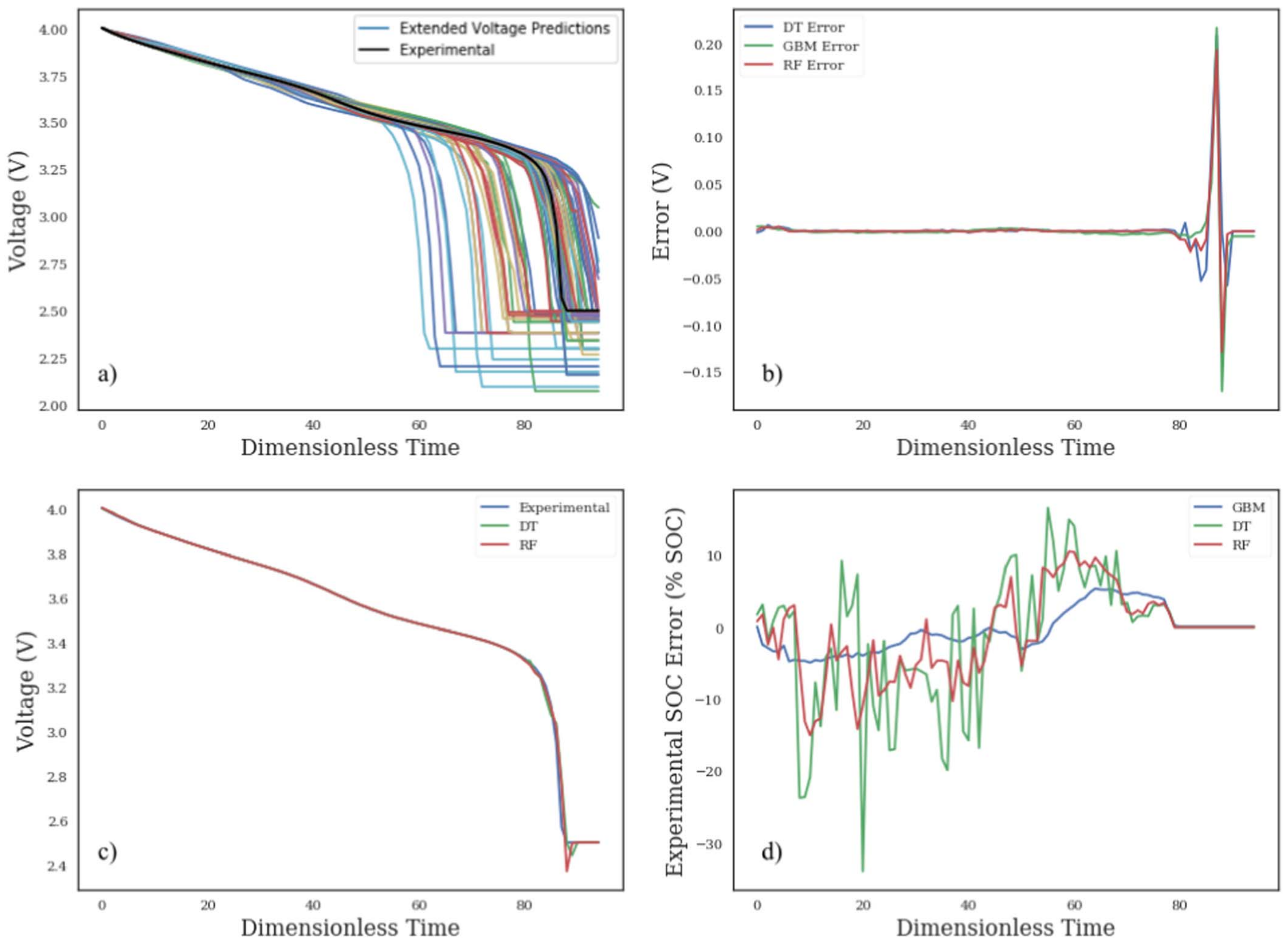


Figure 13. a) Extended sequential voltage predictions for the target data, demonstrating the long-term variance of the predictions which are accurate in the short-term. b) Error between the experimental and predicted curves for recurrent networks. c) Discharge graphs compared in b. d) Recurrent SOC error for experimental data.

which is shown in Figure 13b. Figure 13a represents the combinations of voltage forecasts that the DT makes as it iteratively fits sections of voltage data. At the beginning of the discharge curve, there are fewer predictions, and they are all very similar. By the 5th estimate, the deviations of the data from the training set can be seen, and several curves develop. While the extended voltage predictions are fairly tightly grouped during the bulk of the discharge, we can see there is a large amount of variance toward the end, in particular when the curve achieves minimum voltage. The relatively large parameter variance covers a number of batteries, and the target curve is similar to only a handful of them. While the large error in the extended voltage predictions does not hurt the short-range voltage predictions, it is very detrimental to the SOC predictions, which are heavily dependent upon the ending time of the curve. Figure 13b represents the accompanying prediction of subsequent voltages, sampled only at the next point. The errors in the predictions, in absolute volts, are shown in Figure 13c. It can be seen that the accuracy is well under 0.01 volts for the majority of the discharge, and the major source of error occurs when the experimental data stops at 2.5 volts, while the simulated data often continues. This error can be reduced by restricting the data set discharge voltages to terminate at exactly 2.5 volts. In Figure 13d, the corresponding SOC estimate errors can be seen for each of the predictive methods. The GBM performs the best, and also has the smoothest output, likely due to its ability to generalize, enabled by the creation of an ensemble of weak learners rather than completely learning each data sample independently, as with RFs and DTs.

In order to combat the errors associated with the recurrent SOC prediction, it is possible to restructure the data once again in order to give only the information present at certain points of the discharge, like the recurrent method, but to include the voltage history of the battery in order to improve the SOC prediction accuracy, like the constant time model. To do this, the constant time data is modified such that any points past the current point in time are padded with zeros. For instance, the first data point, when SOC is 100%, will contain just the first voltage from the discharge cycle followed by 99 zeros. The fifth data point will contain the first five voltages followed by 95 zeros. In this way, much more relevant information is available to the algorithm in order to predict SOC, and the results are significantly better, with a greatly reduced MSE of 1.27%, down from the previous best of 3.25%. The SOC prediction error is shown in Figure 14.

Perspective

One of the main advantages of using machine learning in a simulation application is for the creation of highly accurate, extremely fast and robust surrogate models. These models are approximations for the original model, in this case the P2D lithium-ion battery model, which is itself an approximation of reality.

There have been other instances in which researchers resort to approximations for algorithms, and must verify that the result remains meaningful. A good example from the battery community is the work by Doyle[20] using BAND(J) routines, which were restricted to a single

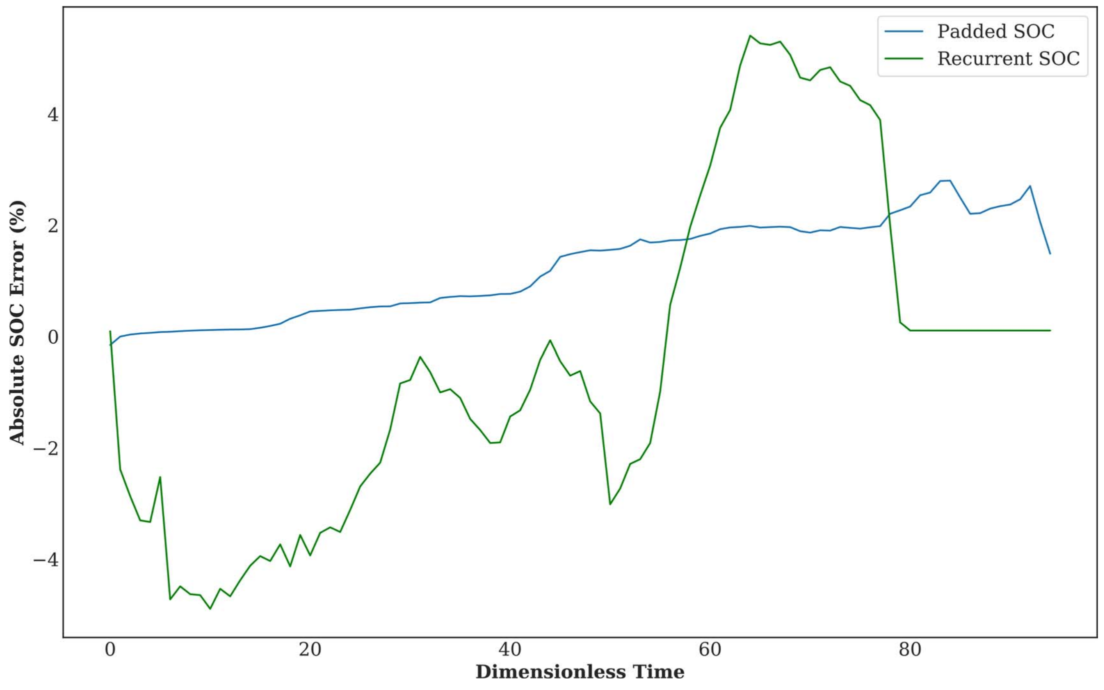


Figure 14. Effects of problem formulation on SOC estimation. The recurrent method uses only voltages [n-4…n] and the padded method uses [n = 0…n], which results in significantly lower error for the experimental data set.

dimension in x. In order to solve for the diffusion in the radial direction of the particle, rather than adding node points in the radial direction, the authors used Duhamel's superposition theorem, which is typically used to give a solution for the model when the constant current boundary position is replaced with a known profile which varies dynamically in time. In the P2D model, this time-variant profile for porewall flux is not known a priori, and the authors were able to solve the model by approximating the unknown time integral. Another example in the battery modeling literature is the paper by Paxton and Newman.[32] Since a robust code that can solve when D is a constant was already present, they provided an approach for materials where D changes with θ, the state of charge. For example, by focusing just on the single particle, the authors compared a D(θ) model to a $D(i_{app})$ model and concluded that the D(θ) can be replaced by the $D(i_{app})$ model under a range of operating conditions, although the variation in the magnitude and direction of D(θ) will determine the accuracy. This means that the Duhamel's superposition based model can be used for D(θ) cases.

Similarly, a parabolic profile[33] approximation for single particle diffusion has been derived and compared to higher[34–37] order approximations for varying applied current. Both the parabolic profile and higher order approximations have successfully been applied to the P2D model and have been used for simulation, control, design, and estimation purposes.[33,38] The effective approximate model for solid phase diffusion is different for constant diffusivity, where the Galerkin approach[36] is valid, and for varying diffusivity, where the only valid approach is orthogonal collocation.[39,40] It is often possible to switch between different approximate models depending upon operating conditions – using Faraday's law for low rates, a parabolic profile for low rates, a higher order model for moderate rates, and boundary layer type finite difference or finite element models[22,41,42] for very high rates. As evidenced by many of these effective approximations, there are always pros and cons for such approximations. Sometimes, approximation is only the way to simulate[43] and, sometimes, approximation is necessitated by the availability of software and codes. Nevertheless, the approximations had always been validated by comparing with the full-order model and/or experimental data. Similarly, surrogate models from ML, if carefully developed, are expected to move the field forward significantly.

Future work includes the creation of surrogate models which are trained only at certain scales, but which can be extended to become part of a more complex model, much like the parabolic profile or similar approximations for a particular scale. An important consideration during the creation of these models will be the ensuring the conservation of flux, mass, and charge. For example, when solid phase diffusion is approximated with any method, the average concentration inside the particle is directly proportional to the applied current or porewall flux.[32,44] Enforcing this constraint on surrogate models will result in much faster convergence.

The success or failure of a machine learning project depends heavily upon the problem formulation and the desired outputs. For example, this project began with a specific goal – to create a surrogate model which allows for the prediction of a voltage vs time discharge curve for a given set of model input parameters. After varying degrees of success with this approach, a new goal was set – estimate the SOC as well as the voltage, but using the same data set. While the approach adopted for this task was reasonable, given the constant discharge rate assumption, it would have failed for a varying discharge rate. However, the P2D model is capable of calculating the SOC by reporting the fraction of lithium present in the anode and cathode, which are a much better means of calculating SOC than using voltage. Since those data were not stored when creating the data set, they were not used for this paper, but a new surrogate model could easily be created from a data set containing these lithium concentrations, electrolyte concentrations, or other internal states, which may have much more success in predicting SOC than the current surrogate model and may prove more useful, especially in control applications.

Conclusions

Machine learning is a promising new field of research that gives the ability to create surrogate models for faster execution of higher fidelity models, as well as giving the ability to restructure the input-output structures of the model without compromising accuracy. In this work, decision trees, random forests, and gradient boosted machines have been evaluated as candidates for the creation of surrogate models for the porous electrode pseudo two-dimensional physics-based lithium-ion battery model during a 2C constant-current discharge event. In this scenario, the surrogate models perform exceptionally well at next-voltage prediction, but due to the structure of the data set, have no ability to function at currents that are not 2C and chemistries that are dissimilar to that of the simulated data set. In this version of the P2D model, the open circuit potential for the positive electrode is fit using a piecewise continuous linear curve. If a data set were created that varied this curve significantly, it may be possible to express different chemistries, or to have multiple chemistries contained in the same data set. To incorporate them, a new model could be trained using the same techniques demonstrated here.

It can be seen that while the voltage prediction using recurrent methods is exceptional, the state of charge (SOC) estimation is fairly poor due to the large variance in the terminal times of the simulated discharge curves. The same variability that allows for high-accuracy voltage prediction causes higher error in SOC estimates. By restructuring the data set, it is possible to improve the resulting estimate for SOC, which is highly dependent upon the discharge history of the battery. The ability to solve high fidelity models with low computational cost has many applications, including real-time control in BMS applications, fast optimization for battery design, and multiscale modeling. Future work will include the use of more complex ML algorithms, such as ANNs, which take longer to train, but are similarly fast to execute and offer superior interpolation performance in addition to much smaller model size on disk. Additionally, while decision trees can only select from training set data, and hence can only give physically meaningful results while the training data remains physically meaningful, this may not be the case for other models while interpolating or extrapolating, and may not be the case for random forests. In random forests, the outputs of several curves are directly averaged together, often resulting in a curve that is closer in error but qualitatively dissimilar – the curve may be blocky, with sudden drops in voltage as the combined discharge curves are averaged. Future work will also examine using multiple models whose algorithmic layouts are based on physically relevant relationships, and work will be done to ensure that the outputs remain meaningful. Although surrogate models based on the P2D model are demonstrated in this paper, the concept is applicable for detailed 2D and 3D models for batteries, including multiscale thermal models.

A saved version of the recurrent model can be found at https://github.com/nealde/P2D_Surrogate, along with instructions for downloading and executing it locally.

Acknowledgments

The authors thank the Department of Energy (DOE) for providing partial financial support for this work, through the Advanced Research Projects Agency (ARPA-E) award number DE-AR0000275. The authors would also like to thank the Clean Energy Institute (CEI) at the University of Washington (UW) and the Data Intensive Research Enabling Clean Technologies (DIRECT), a UW CEI program funded by the National Science Foundation (NSF), in addition to the Washington Research Foundation (WRF) for their monetary support of this work. The battery modeling work has been supported by the Assistant Secretary for Energy Efficiency and Renewable Energy, Office of Vehicle Technologies of the U. S. Department of Energy through the Advanced Battery Material Research (BMR) Program (Battery500 Consortium).

References

1. S. Tansley and K. M. Tolle, *The Fourth Paradigm: Data-intensive Scientific Discovery*, Microsoft Research, 2009.
2. Y. Wu et al., "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation," *ArXiv160908144 Cs*, Sep. 2016.
3. C. Monteleoni, G. A. Schmidt, S. Saroha, and E. Asplund, "Tracking climate models," *Stat. Anal. Data Min.*, **4**(4) 372 (2011).
4. D. A. C. Beck, J. M. Carothers, V. R. Subramanian, and J. Pfaendtner, "Data science: Accelerating innovation and discovery in chemical engineering," *AIChE J.*, **62**(5) 1402 (2016).
5. 2016 at 3:30am Posted by Jacob Joseph on April 11 and V. Blog, "How to Treat Missing Values in Your Data." [Online]. Available: http://www.datasciencecentral.com/profiles/blogs/how-to-treat-missing-values-in-your-data-1. [Accessed: 25-Aug-2017].
6. P. Domingos, "A Few Useful Things to Know About Machine Learning," *Commun ACM*, **55**(10) 78 (2012).
7. U. M. Fayyad, A. Wierse, and G. G. Grinstein, *Information Visualization in Data Mining and Knowledge Discovery.*, Morgan Kaufmann, 2002.
8. A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM J. Res. Dev.*, **3**(3) 210 (1959).
9. K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, **1**(1) 4 (1990).
10. K. Mitra and M. Ghivari, "Modeling of an industrial wet grinding operation using data-driven techniques," *Comput. Chem. Eng.*, **30**(3) 508 (2006).
11. A. Nasridinov, Y. Lee, and Y.-H. Park, "Decision tree construction on GPU: ubiquitous parallel computing approach," *Computing*, **96**(5) 403 (2014).
12. M. Liu, M. Wang, J. Wang, and D. Li, "Comparison of random forest, support vector machine and back propagation neural network for electronic tongue data classification: Application to the recognition of orange beverage and Chinese vinegar," *Sens. Actuators B Chem.*, **177**, 970 (2013).
13. I. K. Sethi, "Entropy nets: from decision trees to neural networks," *Proc. IEEE*, **78**(10) 1605 (1990).
14. J. Liu, A. Saxena, K. Goebel, B. Saha, and W. Wang, "An Adaptive Recurrent Neural Network for Remaining Useful Life Prediction of Lithium-ion Batteries," National Aeronautics and Space Administration Moffett Field ca ames Research Center, National Aeronautics and Space Administration Moffett Field ca Ames Research Center, Oct. 2010.
15. G. Capizzi, F. Bonanno, and G. M. Tina, "Recurrent Neural Network-Based Modeling and Simulation of Lead-Acid Batteries Charge #x2013;Discharge," *IEEE Trans. Energy Convers.*, **26**(2) 435 (2011).
16. P. W. C. Northrop, M. Pathak, D. Rife, S. De, S. Santhanagopalan, and V. R. Subramanian, "Efficient Simulation and Model Reformulation of Two-Dimensional Electrochemical Thermal Behavior of Lithium-Ion Batteries," *J. Electrochem. Soc.*, **162**(6) A940 (2015).
17. P. W. C. Northrop, B. Suthar, V. Ramadesigan, S. Santhanagopalan, R. D. Braatz, and V. R. Subramanian, "Efficient Simulation and Reformulation of Lithium-Ion Battery Models for Enabling Electric Transportation," *J. Electrochem. Soc.*, **161**(8) E3149 (2014).
18. M. B. Pinson and M. Z. Bazant, "Theory of SEI Formation in Rechargeable Batteries: Capacity Fade, Accelerated Aging and Lifetime Prediction," *J. Electrochem. Soc.*, **160**(2) A243 (2013).
19. M. Guo, G. Sikha, and R. E. White, "Single-Particle Model for a Lithium-Ion Cell: Thermal Behavior," *J. Electrochem. Soc.*, **158**(2) A122 (2011).
20. M. Doyle, T. F. Fuller, and J. Newman, "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell," *J. Electrochem. Soc.*, **140**(6) 1526 (1993).
21. J. Newman and W. Tiedemann, "Porous-electrode theory with battery applications," *AIChE J.*, **21**(1) 25 (1975).
22. V. Ramadesigan, P. W. C. Northrop, S. De, S. Santhanagopalan, R. D. Braatz, and V. R. Subramanian, "Modeling and Simulation of Lithium-Ion Batteries from a Systems Engineering Perspective," *J. Electrochem. Soc.*, **159**(3) R31 (2012).
23. W.-Y. Chang, "The State of Charge Estimating Methods for Battery: A Review," *International Scholarly Research Notices*, 2013. [Online]. Available: https://www.hindawi.com/journals/isrn/2013/953792/. [Accessed: 03-Jul-2017].
24. A. Jokar, B. Rajabloo, M. Désilets, and M. Lacroix, "Review of simplified Pseudo-two-Dimensional models of lithium-ion batteries," *J. Power Sources*, **327**, 44 (2016).
25. "The Julia Language." [Online]. Available: https://julialang.org/. [Accessed: 20-Sep-2017].
26. S. G. Johnson, *Sobol.jl: generation of Sobol low-discrepancy sequence (LDS) for the Julia language.* 2017.
27. D. Nielsen, "Tree Boosting With XGBoost - Why Does XGBoost Win 'Every' Machine Learning Competition?," 2016.
28. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, **12**, 2825−2830 (2011).
29. J. Heaton, "An Empirical Analysis of Feature Engineering for Predictive Modeling," *ArXiv170107852 Cs*, Jan. 2017.
30. K. Nakagawa, S. Suzumura, M. Karasuyama, K. Tsuda, and I. Takeuchi, "Safe Feature Pruning for Sparse High-Order Interaction Models," *ArXiv150608002 Stat*, Jun. 2015.
31. A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artif. Intell.*, **97**(1) 245 (1997).
32. B. Paxton and J. Newman, "Variable Diffusivity in Intercalation Materials A Theoretical Approach," *J. Electrochem. Soc.*, **143**(4) 1287 (1996).
33. V. R. Subramanian, V. D. Diwakar, and D. Tapriyal, "Efficient Macro-Micro Scale Coupled Modeling of Batteries," *J. Electrochem. Soc.*, **152**(10), A2002 (2005).
34. S. De, B. Suthar, D. Rife, G. Sikha, and V. R. Subramanian, "Efficient Reformulation of Solid Phase Diffusion in Electrochemical-Mechanical Coupled Models for Lithium-Ion Batteries: Effect of Intercalation Induced Stresses," *J. Electrochem. Soc.*, **160**(10) A1675, (2013).
35. S. De, P. W. C. Northrop, V. Ramadesigan, and V. R. Subramanian, "Model-based simultaneous optimization of multiple design parameters for lithium-ion batteries for maximization of energy density," *J. Power Sources*, **227**, 161 (2013).
36. V. Ramadesigan, V. Boovaragavan, J. C. Pirkle, and V. R. Subramanian, "Efficient Reformulation of Solid-Phase Diffusion in Physics-Based Lithium-Ion Battery Models," *J. Electrochem. Soc.*, **157**(7) A854 (2010).

37. J. C. Forman, S. Bashash, J. L. Stein, and H. K. Fathy, "Reduction of an Electrochemistry-Based Li-Ion Battery Model via Quasi-Linearization and Padé Approximation," *J. Electrochem. Soc.*, **158**(2) A93 (2011).
38. M. Pathak, S. Kolluri, and V. R. Subramanian, "Generic Model Control for Lithium-Ion Batteries," *J. Electrochem. Soc.*, **164**(6) A973 (2017).
39. P. W. C. Northrop, V. Ramadesigan, S. De, and V. R. Subramanian, "Coordinate Transformation, Orthogonal Collocation, Model Reformulation and Simulation of Electrochemical-Thermal Behavior of Lithium-Ion Battery Stacks," *J. Electrochem. Soc.*, **158**(12) A1461 (2011).
40. L. Cai and R. E. White, "Lithium ion cell modeling using orthogonal collocation on finite elements," *J. Power Sources*, **217**, no. Supplement C, 248 (2012).
41. K. A. Smith, C. D. Rahn, and C. Y. Wang, "Model-based electrochemical estimation of lithium-ion batteries," in 2008 *IEEE International Conference on Control Applications*, 714, 2008.
42. Y. Zeng et al., "Efficient Conservative Numerical Schemes for 1D Nonlinear Spherical Diffusion Equations with Applications in Battery Modeling," *J. Electrochem. Soc.*, **160**(9) A1565 (2013).
43. S. Y. Joshi, M. P. Harold, and V. Balakotaiah, "Low-dimensional models for real time simulations of catalytic monoliths," *AIChE J.*, **55**(7) 1771 (2009).
44. V. R. Subramanian and R. E. White, "New separation of variables method for composite electrodes with galvanostatic boundary conditions," *J. Power Sources*, **96**(2) 385 (2001).