35th International Electric Vehicle Symposium and Exhibition (EVS35)
Oslo, Norway, June 11-15, 2022

Simulating multicell battery packs using physics-based reduced-order models

Gregory L. Plett[1,2], M. Scott Trimboli[1,3]
[1]*Department of Electrical and Computer Engineering, University of Colorado Colorado Springs, Colorado Springs, CO 80918, United States,* [2]gplett@uccs.edu, [3]mtrimbol@uccs.edu

Summary

EV and HEV battery packs require cells connected both in parallel and in series, often configured into modules. The "parallel cell module" approach wires cells within a module in parallel and then wires modules in series; the "series cell module" approach wires cells within a module in series and then wires modules in parallel. Earlier work showed how to simulate these battery packs using equivalent-circuit cell models. This paper describes a simulation system developed to evaluate different scenarios using physics-based reduced-order cell models and presents some preliminary findings. The primary advantage of the physics-based approach is that it allows evaluating cell internal variables that can indicate premature aging and failure, which is not possible when using equivalent-circuit models.

1 Abstract

High-energy battery packs require cells connected in parallel (to increase capacity beyond that of a single cell) and/or in series (to increase voltage). For practical and safety reasons the packs are often comprised of cells configured into modules. Two extreme cases of how this might be done for a 288-cell battery pack are illustrated in Fig. 1. In one case, groups of three cells are wired in parallel—making a "parallel-cell module" or PCM—and then 96 of these PCMs are wired in series. In the other case, groups of 96 cells are wired in series—making a "series-cell module" or SCM—and then 3 of these SCMs are wired in parallel. Some practical and economic tradeoffs between these two approaches are discussed in [1, 2], which also shows how to simulate multi-cell battery packs using equivalent-circuit models (ECMs) of cells. Other early references on modeling multi-cell battery packs using ECMs include: [3, 4].
ECMs are sufficient to evaluate the input/output (current/voltage) response of the cells and pack. However, they cannot predict values of the electrochemical variables internal to the cells. It is exactly these internal variables that are descriptive of aging processes [5–7] and can give insight into when a pack is near a condition that will cause excessive premature aging. To predict these variables, we require physics-based models (PBMs); for this to be done efficiently, we require reduced-order models.
In this paper, we show how to simulate a battery pack in a computationally efficient way using physics-based reduced-order models (PB-ROMs, or simply ROMs) of cells and present some summary results.

2 A single-cell PB-ROM

We begin with a review of the type of single-cell ROM we will use with this paper. Steps taken to construct this ROM from the common Doyle–Fuller–Newman PDE model [8–10] include:

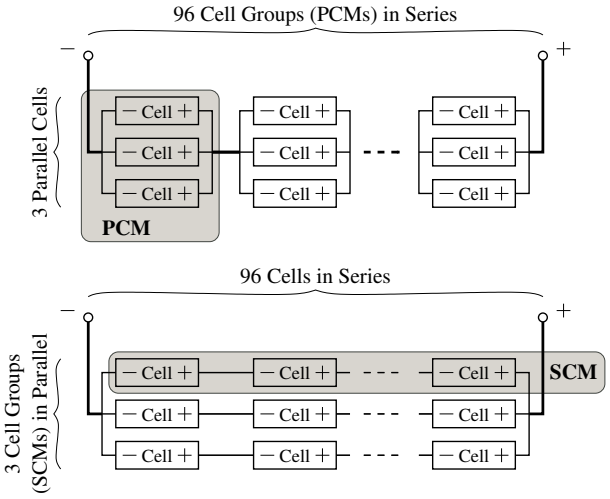

Figure 1: An example 288-cell pack comprised of 96 PCMs (top) or 3 SCMs (bottom), from [1].

- Deriving transfer functions (TFs) of all electrochemical variables of the cell with respect to input current [11], linearized around a specific SOC and temperature setpoint.
- Invoking a "realization algorithm" to convert the transfer functions into a reduced-order discrete-time state-space form [12], applicable in the neighborhood of this SOC and temperature setpoint.

The output of this procedure is a discrete-time state-space model of the form:

$$\boldsymbol{x}[k+1] = \boldsymbol{A}\boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k] \tag{1}$$

$$\boldsymbol{y}[k] = \boldsymbol{C}\boldsymbol{x}[k] + \boldsymbol{D}\boldsymbol{u}[k], \tag{2}$$

where the *state vector* of the system is represented by $\boldsymbol{x}[k] \in \mathbb{R}^{n\times 1}$, the *input vector* by $\boldsymbol{u}[k] \in \mathbb{R}^{m\times 1}$ and the *output vector* by $\boldsymbol{y}[k] \in \mathbb{R}^{q\times 1}$. When we apply this type of model to describe lithium-ion cells, $\boldsymbol{u}[k]$ corresponds to the electrical current applied to the cell (i.e., $\boldsymbol{u}[k] = i_{\text{app}}[k]$ and $m = 1$) and $\boldsymbol{y}[k]$ corresponds to the set of electrochemical variables that we would like to predict using the model. The state vector $\boldsymbol{x}[k]$ is defined numerically by the realization algorithm.
Most of the realization algorithms in [12] cannot work directly with TFs having integration dynamics. So, we take the following approach, where $\boldsymbol{G}(s)$ is a vector of the TFs we desire to implement:

- Compute the integrator residue of all TFs in $\boldsymbol{G}(s)$ as $\mathbf{res}^* = \lim_{s\to 0} s\boldsymbol{G}(s)$.
- Remove the integrators from the TFs by defining $[\boldsymbol{G}(s)]^* = \boldsymbol{G}(s) - \mathbf{res}^*/s$.
- Invoke the realization algorithm to find a discrete-time state-space model of order n for integrator-removed $[\boldsymbol{G}(s)]^*$.
- Finally, augment the identified model with the integration dynamics that were removed previously. That is, we write:

$$\mathfrak{A} = \begin{bmatrix} \boldsymbol{A} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \quad \mathfrak{B} = \begin{bmatrix} \boldsymbol{B} \\ 1 \end{bmatrix}, \quad \mathfrak{C} = \begin{bmatrix} \boldsymbol{C} & \mathbf{res}^* \end{bmatrix}, \tag{3}$$

  and $\mathfrak{D} = \boldsymbol{D}$ is unchanged. The augmented model is of order $n+1$.
- Using this augmented system, we can write:[1]

$$\underbrace{\begin{bmatrix} \boldsymbol{x}[k+1] \\ x_0[k+1] \end{bmatrix}}_{\mathfrak{x}[k+1]} = \underbrace{\begin{bmatrix} \boldsymbol{A} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\mathfrak{A}} \underbrace{\begin{bmatrix} \boldsymbol{x}[k] \\ x_0[k] \end{bmatrix}}_{\mathfrak{x}[k]} + \underbrace{\begin{bmatrix} \boldsymbol{B} \\ 1 \end{bmatrix}}_{\mathfrak{B}} u[k]$$

[1]Note that we now make a notational distinction between state vectors for models having an integrator, denoted as $\mathfrak{x}[k]$, and models without integrator, denoted as $\boldsymbol{x}[k]$. We do not make a notational distinction between outputs from these two types of systems, choosing to write both as $\boldsymbol{y}[k]$ since they both define the same variables. However, they are technically distinct since $\boldsymbol{y}[k]$ in Eq. (2) describes variables without integration dynamics and $\boldsymbol{y}[k]$ in Eq. (5) describes the same variables but with integration dynamics. The context in which $\boldsymbol{y}[k]$ is used should clarify how it should be interpreted.

$$\boldsymbol{y}[k] = \underbrace{\left[\begin{array}{cc} \boldsymbol{C} & \mathbf{res}^* \end{array}\right]}_{\mathfrak{C}} \underbrace{\left[\begin{array}{c} \boldsymbol{x}[k] \\ x_0[k] \end{array}\right]}_{\mathfrak{X}[k]} + \boldsymbol{D}u[k],$$

where $x_0[k]$ is the integration state, or,

$$\mathfrak{X}[k+1] = \mathfrak{A}\mathfrak{X}[k] + \mathfrak{B}u[k] \tag{4}$$

$$\boldsymbol{y}[k] = \mathfrak{C}\mathfrak{X}[k] + \mathfrak{D}u[k]. \tag{5}$$

3 Simulating a single cell in the time domain, near a setpoint

We have now computed a discrete-time state-space ROM that approximates the TFs of a cell populated with parameter values corresponding to a specific SOC and temperature setpoint. Our next step is to use this model to simulate cell dynamics near that setpoint.

Simulation of the basic ROM form presented in Eqs. (4)–(5) is straightforward. We initialize the state $\mathfrak{X}[0] = 0$. Then, we iterate the equations in order to compute linearized variables $\boldsymbol{y}[k]$. To compute accurate predictions of the nonlinear variables we desire to model, we will also need to apply nonlinear corrections. And, to predict cell voltage, we will need to combine some variables in a nonlinear way. In the following, we see how to do both. (For a description of the notation being used, please confer [13].)

Update cell state of charge $z\,[k]$: First, we look at how to update electrode and cell states of charge. Electrode state of charge (i.e., average solid stoichiometry) can be computed as:

$$\theta^{\mathrm{r}}_{\mathrm{s,avg}}\,[k] = \theta^{\mathrm{r}}_{\mathrm{s},0} + \tilde{\theta}^{\mathrm{res}^{\mathrm{r}}_0}_{\mathrm{ss}} x_0\,[k]\,, \tag{6}$$

where $\theta^{\mathrm{r}}_{\mathrm{s},0}$ is the initial stoichiometry and $\tilde{\theta}^{\mathrm{res}^{\mathrm{r}}_0}_{\mathrm{ss}}$ is the integrator residue for the $\widetilde{\Theta}^{\mathrm{r}}_{\mathrm{s}}(\tilde{x}, s)/I_{\mathrm{app}}(s)$ TF. Initial stoichiometry is determined from cell SOC as:

$$\theta^{\mathrm{r}}_{\mathrm{s},0} = \theta^{\mathrm{r}}_0 + z_0\,(\theta^{\mathrm{r}}_{100} - \theta^{\mathrm{r}}_0)\,. \tag{7}$$

The integrator residues are the following constants:

$$\tilde{\theta}^{\mathrm{res}^{\mathrm{n}}_0}_{\mathrm{ss}} = \frac{-\left|\theta^{\mathrm{n}}_{100} - \theta^{\mathrm{n}}_0\right|}{3600Q} \quad \text{and} \quad \tilde{\theta}^{\mathrm{res}^{\mathrm{p}}_0}_{\mathrm{ss}} = \frac{\left|\theta^{\mathrm{p}}_{100} - \theta^{\mathrm{p}}_0\right|}{3600Q}.$$

Next, cell-level SOC can now be predicted as:

$$z\,[k] = \frac{\theta^{\mathrm{n}}_{\mathrm{s,avg}}\,[k] - \theta^{\mathrm{n}}_0}{\theta^{\mathrm{n}}_{100} - \theta^{\mathrm{n}}_0} \qquad \text{or} \qquad z\,[k] = \frac{\theta^{\mathrm{p}}_{\mathrm{s,avg}}\,[k] - \theta^{\mathrm{p}}_0}{\theta^{\mathrm{p}}_{100} - \theta^{\mathrm{p}}_0}.$$

Interfacial lithium fluxes: We continue by considering how to convert ROM linear outputs to nonlinear predictions for every electrochemical variable that can be predicted by the TFs. In the case of the interfacial lithium fluxes, the linear outputs of the ROM for $\dot{n}[\tilde{x}, k]$ are unbiased predictions of the true fluxes. No corrections are needed.

Solid surface stoichiometry: The ROM can produce debiased outputs $\tilde{\theta}_{\mathrm{ss}}[\tilde{x}, k]$, from which the nonlinear prediction is made via:

$$\theta_{\mathrm{ss}}[\tilde{x}, k] = \tilde{\theta}_{\mathrm{ss}}[\tilde{x}, k] + \theta^{\mathrm{r}}_{\mathrm{s},0}. \tag{8}$$

Phase potential difference: According to the original definition, the phase-potential difference variable can be computed as:

$$\phi^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x}, k] = \tilde{\phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x}, k] + U^{\mathrm{r}}_{\mathrm{ocp}}(\theta^{\mathrm{r}}_{\mathrm{s},0}),$$

where $\tilde{\phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x}, k]$ is obtained directly from the system linear output, and the electrode OCP function is evaluated at initial local state of charge that is previously determined from Eq. (7).

However, performance can be improved by looking deeper at what is actually happening. The TF $\widetilde{\Phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}\,(\tilde{x}, s)\,/I_{\mathrm{app}}\,(s)$ has a pole at the origin, which is removed prior to using the realization algorithm to give $[\widetilde{\Phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}\,(\tilde{x}, s)]^*/I_{\mathrm{app}}\,(s)$. The integrator response could be added back manually as:

$$\tilde{\phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x}, k] = \left[\tilde{\phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x}, k]\right]^* + \tilde{\phi}^{\mathrm{res0,r}}_{\mathrm{s\text{-}e}} \times x_0[k].$$

However, recall the integrator residues of $\widetilde{\Phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}(\tilde{x}, s)/I_{\mathrm{app}}(s)$,

$$\tilde{\phi}^{\mathrm{res}^{\mathrm{n}}_0}_{\mathrm{s\text{-}e}} = \frac{-\left|\theta^{\mathrm{n}}_{100} - \theta^{\mathrm{n}}_0\right|\,[U^{\mathrm{n}}_{\mathrm{ocp}}]'}{3600Q}$$

$$\tilde{\phi}_{\text{s-e}}^{\text{res}_0^{\text{p}}} = \frac{\left|\theta_{100}^{\text{p}} - \theta_0^{\text{p}}\right| \left[U_{\text{ocp}}^{\text{p}}\right]'}{3600Q}.$$

We notice that the residuals of the two unstable TFs are linked as:

$$\tilde{\phi}_{\text{s-e}}^{\text{res}_0^{\text{r}}} = \left[U_{\text{ocp}}^{\text{r}}\right]' \times \tilde{\theta}_{\text{ss}}^{\text{res}_0^{\text{r}}}.$$

Therefore, we can write:

$$\tilde{\phi}_{\text{s-e}}^{\text{r}}[\tilde{x}, k] = [\tilde{\phi}_{\text{s-e}}^{\text{r}}[\tilde{x}, k]]^* + \left[U_{\text{ocp}}^{\text{r}}\right]' \times \tilde{\theta}_{\text{ss}}^{\text{res}_0^{\text{r}}} \times x_0[k],$$

Substituting the relationship from Eq. (6) yields:

$$\tilde{\phi}_{\text{s-e}}^{\text{r}}[\tilde{x}, k] = [\tilde{\phi}_{\text{s-e}}^{\text{r}}[\tilde{x}, k]]^* + \left[U_{\text{ocp}}^{\text{r}}\right]' \left(\theta_{\text{s,avg}}^{\text{r}}[k] - \theta_{\text{s},0}^{\text{r}}\right).$$

We recognize the rightmost terms as a linearization of $U_{\text{ocp}}^{\text{r}}[\theta_{\text{s,avg}}^{\text{r}}[k]]$. Therefore, we achieve more accurate results if we compute:

$$\phi_{\text{s-e}}^{\text{r}}[\tilde{x}, k] = [\tilde{\phi}_{\text{s-e}}^{\text{r}}[\tilde{x}, k]]^* + U_{\text{ocp}}^{\text{r}}[\theta_{\text{s,avg}}^{\text{r}}[k]]. \tag{9}$$

Potential in solid: The ROM can directly compute debiased $\tilde{\phi}_{\text{s}}^{\text{r}}[\tilde{x}, k]$. To use the nonlinear prediction, we must add back the term at the current collector:

$$\phi_{\text{s}}^{\text{n}}[\tilde{x}, k] = \tilde{\phi}_{\text{s}}^{\text{n}}[\tilde{x}, k] + \phi_{\text{s}}^{\text{n}}[0, k] = \tilde{\phi}_{\text{s}}^{\text{n}}[0, k] + 0 \tag{10}$$

$$\phi_{\text{s}}^{\text{p}}[\tilde{x}, k] = \tilde{\phi}_{\text{s}}^{\text{p}}[\tilde{x}, k] + \phi_{\text{s}}^{\text{p}}[3, k] = \tilde{\phi}_{\text{s}}^{\text{p}}[\tilde{x}, k] + v_{\text{cell}}[k], \tag{11}$$

where $v_{\text{cell}}[k]$ will be computed from Eq. (14).

Potential in electrolyte: The ROM can directly compute debiased $\tilde{\phi}_{\text{e}}^{\text{r}}[\tilde{x}, k]$. To compute the nonlinear prediction, notice:

$$\begin{aligned}\phi_{\text{e}}^{\text{r}}[\tilde{x}, k] &= \tilde{\phi}_{\text{e}}^{\text{r}}[\tilde{x}, k] + \phi_{\text{e}}^{\text{n}}[0, k] = \tilde{\phi}_{\text{e}}^{\text{r}}[\tilde{x}, k] + \underbrace{\phi_{\text{s}}^{\text{n}}[0, k]}_{0} - \phi_{\text{s-e}}^{\text{n}}[0, k] \\ &= \tilde{\phi}_{\text{e}}^{\text{r}}[\tilde{x}, k] - \phi_{\text{s-e}}^{\text{n}}[0, k],\end{aligned}$$

where $\phi_{\text{s-e}}^{\text{n}}[0, k]$ is found via Eq. (9). As a result,

$$\phi_{\text{e}}^{\text{r}}[\tilde{x}, k] = \tilde{\phi}_{\text{e}}^{\text{r}}[\tilde{x}, k] - [\tilde{\phi}_{\text{s-e}}^{\text{n}}[0, k]]^* - U_{\text{ocp}}^{\text{n}}(\theta_{\text{s,avg}}^{\text{n}}[k]). \tag{12}$$

Lithium stoichiometry in electrolyte: The ROM can directly compute debiased $\tilde{\theta}_{\text{e}}^{\text{r}}[\tilde{x}, k]$. The actual electrolyte concentration ratio prediction is found by:

$$\theta_{\text{e}}^{\text{r}}[\tilde{x}, k] = \tilde{\theta}_{\text{e}}^{\text{r}}[\tilde{x}, k] + \theta_{\text{e},0} = \tilde{\theta}_{\text{e}}^{\text{r}}[\tilde{x}, k] + 1, \tag{13}$$

where we recall that $\theta_{\text{e},0} = 1$.

Cell voltage: The voltage of the cell is computed by combining different electrochemical variables:

$$\begin{aligned}v_{\text{cell}}[k] = &\left(\eta^{\text{p}}[3, k] - \eta^{\text{n}}[0, k]\right) + \left(U_{\text{ocp}}^{\text{p}}(\theta_{\text{ss}}^{\text{p}}[3, k]) - U_{\text{ocp}}^{\text{n}}(\theta_{\text{ss}}^{\text{n}}[0, k])\right) \\ &+ \left(\phi_{\text{e}}^{\text{p}}[3, k] - \phi_{\text{e}}^{\text{n}}[0, k]\right) + F\left(\bar{R}_{\text{f}}^{\text{p}} \dot{n}^{\text{p}}[3, k] - \bar{R}_{\text{f}}^{\text{n}} \dot{n}^{\text{n}}[0, k]\right).\end{aligned}$$

If the charge-transfer coefficient $\alpha = 0.5$ and a Butler–Volmer model is assumed, the overpotentials η^{r} are computed as:

$$\begin{aligned}\eta^{\text{r}}[\tilde{x}, k] &= \frac{2RT}{F} \text{asinh}\left(\frac{\dot{n}^{\text{r}}[\tilde{x}, k]}{2\dot{n}_0^{\text{r}}}\right) \\ &= \frac{2RT}{F} \text{asinh}\left(\frac{\dot{n}^{\text{r}}[\tilde{x}, k]}{2\bar{k}_0^{\text{r}} \sqrt{\theta_{\text{e}}^{\text{r}}[\tilde{x}, k]\left(1 - \theta_{\text{ss}}^{\text{r}}[\tilde{x}, k]\right)\theta_{\text{ss}}^{\text{r}}[\tilde{x}, k]}}\right).\end{aligned}$$

The electrolyte-potential difference between the two current collectors is known to be:

$$\phi_{\text{e}}^{\text{p}}[3, k] - \phi_{\text{e}}^{\text{n}}[0, k] = \tilde{\phi}_{\text{e}}^{\text{p}}[3, k],$$

where $\tilde{\phi}_{\text{e}}^{\text{p}}[3, k]$ is a direct linear output from the ROM.
With all previous definitions, we can finally write cell voltage as:

$$\begin{aligned}v_{\text{cell}}[k] = &\left(\eta^{\text{p}}[3, k] - \eta^{\text{n}}[0, k]\right) + \left(U_{\text{ocp}}^{\text{p}}(\theta_{\text{ss}}^{\text{p}}[3, k]) - U_{\text{ocp}}^{\text{n}}(\theta_{\text{ss}}^{\text{n}}[0, k])\right) \\ &+ \tilde{\phi}_{\text{e}}^{\text{p}}[3, k] + F\left(\bar{R}_{\text{f}}^{\text{p}} \dot{n}^{\text{p}}[3, k] - \bar{R}_{\text{f}}^{\text{n}} \dot{n}^{\text{n}}[0, k]\right),\end{aligned} \tag{14}$$

where $\theta_{\text{ss}}^{\text{p}}[3, k]$ and $\theta_{\text{ss}}^{\text{n}}[0, k]$ are evaluated using Eq. (8).

A short summary: Table 1 lists the equation numbers that address nonlinear corrections for each variable. The last column addresses all variables required to compute cell voltage.

Table 1: Summary of nonlinear corrections.

Variable	Definition	Correction	Needed for $v_{\text{cell}}[k]$
$\dot{n}^{\text{r}}[\tilde{x},k]$	—	—	$\dot{n}^{\text{n}}[0,k]$, $\dot{n}^{\text{p}}[3,k]$
$\theta^{\text{r}}_{\text{ss}}[\tilde{x},t]$	$\tilde{\theta}^{\text{r}}_{\text{ss}}[\tilde{x},k]+\theta^{\text{r}}_{\text{s},0}$	Eq. (8)	$\tilde{\theta}^{\text{n}}_{\text{ss}}[0,k]$, $\tilde{\theta}^{\text{p}}_{\text{ss}}[3,k]$
$\phi^{\text{r}}_{\text{s-e}}[\tilde{x},k]$	$\tilde{\phi}^{\text{r}}_{\text{s-e}}[\tilde{x},k]+U^{\text{r}}_{\text{ocp}}(\theta^{\text{r}}_{\text{s},0})$	Eq. (9)	$[\tilde{\phi}^{\text{n}}_{\text{s-e}}[0,k]]^*$
$\phi^{\text{n}}_{\text{s}}[\tilde{x},k]$	$\tilde{\phi}^{\text{n}}_{\text{s}}[\tilde{x},k]+0$	Eq. (10)	—
$\phi^{\text{p}}_{\text{s}}[\tilde{x},k]$	$\tilde{\phi}^{\text{p}}_{\text{s}}[\tilde{x},k]+v_{\text{cell}}[k]$	Eq. (11)	—
$\phi^{\text{r}}_{\text{e}}[\tilde{x},k]$	$\tilde{\phi}^{\text{r}}_{\text{e}}[\tilde{x},k]+\phi^{\text{n}}_{\text{e}}[0,k]$	Eq. (12)	$\tilde{\phi}^{\text{p}}_{\text{e}}[3,k]$
$\theta^{\text{r}}_{\text{e}}[\tilde{x},k]$	$\tilde{\theta}^{\text{r}}_{\text{e}}[\tilde{x},k]+1$	Eq. (13)	$\tilde{\theta}^{\text{n}}_{\text{e}}[0,k]$, $\tilde{\theta}^{\text{p}}_{\text{e}}[3,k]$
$v_{\text{cell}}[k]$	$\phi^{\text{p}}_{\text{s}}[3,k]-\phi^{\text{n}}_{\text{s}}[0,k]$	Eq. (14)	—

4 Simulating a cell over a wide operating range

Until now, we have created ROMs that are specialized to predict cell internal variables and voltage in the neighborhood of a specific SOC and temperature setpoint. However, since cell dynamics vary with temperature and SOC, a single model is not sufficient for applications that must give accurate predictions over a large operating window. In [10, 14], we proposed a *model-blending* approach to be able to make predictions over a wide range of SOC and temperature. Since that time, we have discovered problems with model blending, and here we present an improved method that we call *output blending*.

As with model blending, we precompute ROMs ($\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{C}$, $\mathfrak{D}$ matrices) for multiple SOC and temperature setpoints. The time-varying SOC and temperature of the cell define a trajectory through the set of all N ROMs. At any point in time, the present SOC and temperature is surrounded by four closest models. So far, this explanation is the same as for model blending. But, with model blending we created time-varying $\mathfrak{A}[k]$, $\mathfrak{B}[k]$, $\mathfrak{C}[k]$, and $\mathfrak{D}[k]$ matrices based on bilinear interpolation among the static $\{\mathfrak{A}, \mathfrak{B}, \mathfrak{C}, \mathfrak{D}\}$ matrices of the four nearest-neighbor models at every timestep k. There is a flaw with this logic. It assumes that the $\mathfrak{X}[k]$ vector for every precomputed model has the same interpretation. That is, it assumes that the first element of $\mathfrak{X}[k]$ has the same physical meaning for all models, and so forth for all other elements. If we use a realization algorithm to generate the models, we are not able to guarantee this. The realization algorithms are numeric procedures that automatically optimize a semi-physical state for each model. We have no control over the meaning of the individual components of the state vector.

To summarize, the problem with model blending is that it interpolates quantities for which there is no guarantee of physical compatibility. Output blending overcomes this problem by blending model outputs $\boldsymbol{y}[k]$, for which we *do* have a guarantee of physical compatibility. Even if the individual ROMs have incompatible state vectors $\mathfrak{X}[k]$, their outputs are all organized in the same format to predict the same sets of linearized variables. With output blending, every model has its own unique state vector, which is updated every time step. Linear outputs $\boldsymbol{y}_{0,0}[k]$, $\boldsymbol{y}_{0,1}[k]$, $\boldsymbol{y}_{1,0}[k]$, and $\boldsymbol{y}_{1,1}[k]$ are computed only for the four nearest-neighbor models to the present operating condition (cf. Fig. 2). These four outputs are blended together to make a prediction of the cell's present linearized output. Then, we apply nonlinear corrections to this linear output. Specifically, the real-time output-blending steps are:

Step 1: Determine the present cell SOC $z[k]$ and temperature $T[k]$.

Step 2: Update state vectors of *all* pre-computed ROMs as:

$$\mathfrak{X}_j\left[k+1\right]=\mathfrak{A}_j\boldsymbol{x}_j\left[k\right]+\mathfrak{B}_j i_{\text{app}}\left[k\right].$$

where subscript j denotes a specific ROM among the N precomputed ROMs in the model space.

Step 3: Compute the linear outputs for *only the four closest* pre-computed ROMs as:

$$\boldsymbol{y}_j\left[k\right]=\mathfrak{C}_j\mathfrak{X}_j\left[k\right]+\mathfrak{D}_j i_{\text{app}}\left[k\right].$$

The structure of matrices $\mathfrak{C}$ and $\mathfrak{D}$ of a $n+1$ states q output system are denoted as:

$$\mathfrak{C}=\begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} & \text{res}^*_1 \\ c_{21} & c_{22} & \cdots & c_{2n} & \text{res}^*_2 \\ c_{31} & c_{32} & \cdots & c_{3n} & \text{res}^*_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{q1} & c_{q2} & \cdots & c_{qn} & \text{res}^*_{\text{q}} \end{bmatrix},$$

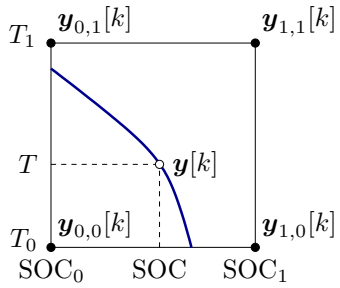

Figure 2: Bilinear interpolation to blend nearest-neighbor outputs.

$$\mathfrak{D} = \left[\lim_{s\to\infty} \boldsymbol{G}_1(s), \quad \lim_{s\to\infty} \boldsymbol{G}_2(s), \quad \cdots \quad \lim_{s\to\infty} \boldsymbol{G}_q(s)\right]^T.$$

Step 4: Blend the linear outputs for only the four models linearized at setpoints closest to the present (z, T) operating point. This is illustrated in Fig. 2, where the cell's present SOC is marked "SOC", and SOC_0 and SOC_1 are the SOC setpoints of the closest precomputed models that bracket the present SOC: $\text{SOC}_0 \leq \text{SOC} \leq \text{SOC}_1$. The cell's present temperature is marked on the figure as "T" and T_0 and T_1 are the temperature setpoints of the closest precomputed models that bracket T: $T_0 \leq T \leq T_1$. We define:

$$\lambda = \frac{\text{SOC} - \text{SOC}_0}{\text{SOC}_1 - \text{SOC}_0} \quad \text{and} \quad \tau = \frac{T - T_0}{T_1 - T_0};$$

then if $\bar{\lambda} = 1 - \lambda$ and $\bar{\tau} = 1 - \tau$,

$$\begin{aligned}\boldsymbol{y}[k] &= \bar{\tau}(\bar{\lambda}\boldsymbol{y}_{0,0}[k] + \lambda\boldsymbol{y}_{1,0}[k]) + \tau(\bar{\lambda}\boldsymbol{y}_{0,1}[k] + \lambda\boldsymbol{y}_{1,1}[k]) \\ &= \gamma_{0,0}\boldsymbol{y}_{0,0}[k] + \gamma_{0,1}\boldsymbol{y}_{0,1}[k] + \gamma_{1,0}\boldsymbol{y}_{1,0}[k] + \gamma_{1,1}\boldsymbol{y}_{1,1}[k] \qquad (15) \\ &= \sum_{j=1}^{N} \gamma_j \boldsymbol{y}_j[k]. \qquad (16)\end{aligned}$$

Note: Although we must update the state vector of all models every time step, we are not required to compute the output vector of any model that is not being used for the present calculation of $\boldsymbol{y}[k]$. That is, in the set of weights $\{\gamma_j\}$ for $1 \leq j \leq N$ in Eq. (16), only the four closest models have nonzero weights, denoted as $\gamma_{0,0}$, $\gamma_{0,1}$, $\gamma_{1,0}$, and $\gamma_{1,1}$ in Eq. (15), reducing required computation.

Step 5: Apply the nonlinear corrections summarized in Table 1 to $\boldsymbol{y}[k]$ to produce nonlinear outputs at this point in time.

Step 6: Repeat Steps 1 through 5 for every iteration k.

5 Simulating constant voltage and constant power

Until this point, we have considered that the input to the ROMs is the applied current $i_{\text{app}}[k]$ and that the ROMs produce $v_{\text{cell}}[k]$ as output (as well as predictions of the cell's internal electrochemical variables). We now modify this approach somewhat to be able to use voltage or power as input to the ROMs instead. This allows us, for example, to simulate constant-voltage or constant-power conditions.

Simulating constant voltage: In order to implement a simulation where voltage is the input (i.e., cell voltage is held constant for one or more time intervals), we take an approach similar to the one presented in [2]. In that case, we used an equivalent-circuit model, which happened to simplify the analysis since all terms in its voltage prediction that were not dependent on the present value of state were linear in the applied current. The physics-based ROMs complicate the process since their voltage calculation is nonlinear in $i_{\text{app}}[k]$.

The idea is as follows: we compute, iteration-by-iteration, the exact required applied current for that iteration to achieve the desired terminal voltage. The net effect is that $v_{\text{cell}}[k]$ *appears* to be the input since we are adjusting the actual model input to enforce the desired $v_{\text{cell}}[k]$, making $i_{\text{app}}[k]$ (and the electrochemical variables) the output.

To see how to do this, note that we can write present voltage as a nonlinear function of the present state

and present input:

$$v_{\text{cell}}[k] = g_v(\mathfrak{X}[k], i_{\text{app}}[k]).$$

The present state $\mathfrak{X}[k]$, however, is not a function of the present input. It is a function only of prior inputs. So, the present state simply creates a bias point for cell voltage, and the present applied current modulates the present actual voltage around this bias.

We can think of the operation of computing $i_{\text{app}}[k]$ as a kind of inverse of $g_v(\cdot)$ for the particular present value of $\mathfrak{X}[k]$:

$$i_{\text{app}}[k] = g_v^{-1}(v_{\text{cell}}[k]; \mathfrak{X}[k]).$$

How might we compute this inverse? Digging into the equations that compute $v_{\text{cell}}[k]$, we have not been able to find a closed-form expression. So, instead, we use a line-search nonlinear optimization to search for the value $i_{\text{app}}[k]$ that causes $v_{\text{cell}}[k] = v_{\text{desired}}[k]$. We will look at an example shortly.

Simulating constant power: We can use a similar strategy to implement a simulation where power is the input (i.e., cell power is held constant for one or more time intervals). Recall that power is equal to applied current multiplied by terminal voltage. In terms of the analysis to date, $p_{\text{cell}}[k] = v_{\text{cell}}[k] i_{\text{app}}[k]$. To simulate constant power, we use nonlinear optimization to search for $i_{\text{app}}[k]$ that causes $p_{\text{cell}}[k] = p_{\text{desired}}[k]$.

Note that there will be two solutions! Both solutions satisfy $p_{\text{cell}}[k] = p_{\text{desired}}[k]$; however, one solution computes a negative cell voltage. We need to be careful to find the solution that produces positive $v_{\text{cell}}[k]$. Once we recognize this detail, the solution is straightforward: if we desire negative power, we constrain $i_{\text{app}}[k]$ to be negative; if we desire positive power, we constrain $i_{\text{app}}[k]$ to be positive.

Example: Fig. 3 shows example output from this method. The cell was initialized with SOC equal to $50\,\%$. Either constant current or constant power was requested from the cell until the maximum voltage limit of $4.2\,\text{V}$ was reached. After that point, the simulation logic automatically switched over to CV.

The value of power for the CP/CV simulation was chosen to make the two simulations roughly comparable. Therefore, we see very little difference in the evolution of SOC and voltage over time. Small differences are seen in the figures showing current and power versus time. As expected, the first segment

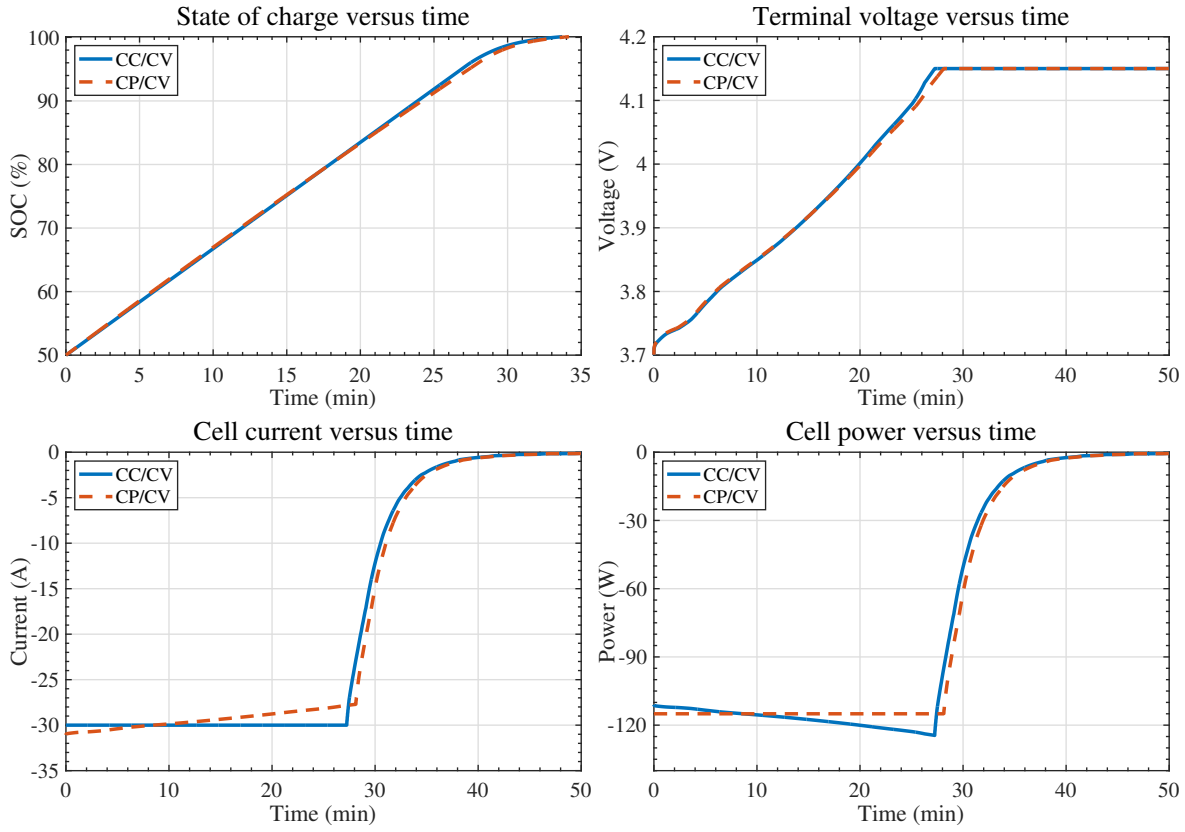

Figure 3: An example of CC/CV and CP/CV charging.

of the simulation demonstrates constant current for the CC/CV simulation and constant power for the CP/CV simulation.

6 Simulating battery packs

Until now, our focus has been on simulating battery *cells*. Now, we briefly consider simulating battery *packs* comprising many cells. The framework that we have developed to this point—especially the new concepts for CV simulations—can be extended readily to add this new capability.

Series-connected cells: Cells connected in series each experience the same current. If all cells have the same initial state and identical parameters, then all cells have exactly the same state and voltage at all times, so we need to simulate only one cell (the others will be identical). This is not generally true, however, so we can simulate all cells' dynamics by keeping separate state and parameter information for every cell, updating each cell's state once per sample interval. We can also include a per-cell "interconnect" resistance term R_{int} that describes terminal resistances and weld resistances and so forth when we compute pack voltage. If there are N_{s} cells wired in series, we compute pack voltage $v_{\mathrm{pack}}[k]$ based on cell voltages for all cells and the interconnect resistance as:

$$v_{\mathrm{pack}}[k] = \left(\sum_{j=1}^{N_{\mathrm{s}}} v_{\mathrm{cell},j}[k] \right) - N_{\mathrm{s}} R_{\mathrm{int}} i_{\mathrm{app}}[k]. \tag{17}$$

Logically modular battery packs: We might need to build battery packs from multiple cells for a variety of reasons. To achieve high power, we require either high voltage or high current (or both). Since parasitic resistive power losses scale with the square of the magnitude of the applied current, it is often preferable to construct a high-voltage pack by wiring many cells in series. Such series-connected packs are common for low-energy, high-power applications, such as HEV. On the other hand, high-energy applications such as EV usually have cells and even entire sub-packs that are connected in parallel to increase the pack's total capacity.

When constructing battery packs having cells wired in series and/or in parallel, it is common to employ a modular approach. One extreme is where all cells in a module are connected in parallel, forming a "parallel cell module" (PCM). These PCMs are then themselves wired in series to construct the battery pack. This is illustrated in the top frame of Fig. 1. Another extreme is where all cells in a module are connected in series, forming a "series-cell module" (SCM). These SCMs are then themselves wired in parallel to construct a pack. This is illustrated in the bottom frame of Fig. 1. The features of these design choices are described in [2].

We would like to be able to simulate battery packs comprising either PCMs or SCMs using physics-based reduced-order models. If cells differ in their internal states and/or parameter values, we must simulate all cells individually and then combine the results to be able to predict the evolution of all electrochemical variables and voltages over time. But, how do we do so?

Simulating PCM-based packs: We begin by considering how to simulate a single PCM. At every point in time, a cell's voltage depends in part on its state vector $\mathfrak{X}[k]$, which itself depends only on prior inputs $i_{\mathrm{app}}[m]$ for $m < k$. The present sample of current $i_{\mathrm{app}}[k]$ does not affect the value of $\mathfrak{X}[k]$ (but of course it will affect $\mathfrak{X}[m]$ for $m > k$). The remaining part of the cell's voltage depends directly on $i_{\mathrm{app}}[k]$. For example, the ROM's linear outputs are directly connected to $i_{\mathrm{app}}[k]$ via the $\mathfrak{D}$-matrix term. Therefore, we consider that a cell's voltage comprises a "fixed" part that does not depend on the present cell current, and a "variable" part that does depend on present cell current. By Kirchhoff's voltage law (KVL), all terminal voltages for cells connected in parallel must be equal. By Kirchhoff's current law (KCL), the sum of branch currents within a PCM must equal the total battery-pack current. When we applied these principles to equivalent-circuit models in [2], we found that we could solve for all branch-current and voltage values in closed form. However, due to the nonlinear voltage equation in our physics-based ROMs, we cannot and so must use nonlinear optimization again.

The procedure is as follows. We assume that total pack applied current $i_{\mathrm{app}}[k]$ is given. Then, for every PCM:

1. We initialize PCM cell currents $i_{\mathrm{app},j}[k] = i_{\mathrm{app}}[k]/N_{\mathrm{p}}$ for all cells j in the PCM.
2. We compute cell voltages $v_{\mathrm{cell},j}[k]$ for each PCM cell using the cell's individual ROM and $i_{\mathrm{app},j}[k]$.

3. We revise cell currents and loop from Step 2 until the maximum absolute difference between all cell voltages in the PCM is minimized, with the constraint that $\sum_{j=1}^{N_\mathrm{p}} i_{\mathrm{app},j}[k] = i[k]$.

The "revise" part of this process in Step 3 is accomplished via a nonlinear optimization, such as `fmincon` in MATLAB.
This procedure gives us all cell currents and all PCM voltages. We then update the state of each cell using its individual ROM and $i_{\mathrm{app},j}[k]$ and proceed to the next iteration. Battery-pack voltage is computed by summing PCM voltages, in a similar fashion to Eq. (17), but where $v_{\mathrm{cell},j}[k]$ is replaced by the common cell voltage of all cells in the jth PCM.

Simulating SCM-based packs: The approach to simulating an SCM is very similar to simulating a PCM. Each cell in an SCM has its own "fixed" and "variable" parts. All "fixed" parts sum to give an equivalent voltage source; all "variable" parts sum to give an equivalent (nonlinear) resistance. The procedure for simulating a battery pack based on SCMs is similar to the one for simulating a PCM. We assume that total pack applied current $i_{\mathrm{app}}[k]$ is given. Then,

1. We initialize SCM currents $i_{\mathrm{app},j}[k] = i_{\mathrm{app}}[k]/N_\mathrm{p}$ for all SCMs j in the pack.
2. We compute cell voltages $v_{\mathrm{cell},j}[k]$ for each cell in the pack using the cell's individual ROM and its input current $i_{\mathrm{app},j}[k]$.
3. We compute SCM voltage by summing voltages of all cells in that SCM in a similar fashion to Eq. (17).
4. We revise SCM currents and loop from Step 2 until the maximum absolute difference between all SCM voltages is minimized, with the constraint that $\sum_{j=1}^{N_\mathrm{p}} i_{\mathrm{app},j}[k] = i[k]$.

Again, the "revise" part of this process in Step 4 is done via nonlinear optimization. This procedure gives us all branch currents and the pack voltage. We update the state of each cell using its individual ROM and $i_{\mathrm{app},j}[k]$ and proceed to the next iteration. In this case, the pack voltage is equal to the common voltage shared by all SCMs, except perhaps for the inclusion of a pack-level interconnect-resistance.

7 Example simulations of PCM-based and SCM-based packs

We briefly consider a simulation comparison of a PCM-based battery pack versus an SCM-based pack. Both packs had $N_\mathrm{s} = 7$ and $N_\mathrm{p} = 3$. The twenty-one cells comprising the packs had individual ROMs that were generated by slightly perturbing the parameter values from a default set, giving each ROM somewhat different characteristics. All cells were initialized to $80\,\%$ SOC. The pack input current cycled the cells in the pack at a nominal 1C rate twice between nominal SOCs of $80\,\%$ and $20\,\%$ before a final discharge to bring the nominal SOC of each cell to $50\,\%$. The pack then rested.
PCM-based battery-pack results are shown in Fig. 4. The voltages of all cells within the same PCM are identical so there are only three distinct voltage profiles for this simulation. However, every cell's input current can be different, depending on each cell's present state and parameters, which is what we observe in the simulation.
SCM-based battery-pack results are shown in Fig. 5. In this case, the currents for all cells within the same SCM are identical so there are only three distinct current profiles for this simulation. However, every cell's voltage can be different, depending on each cell's present state and parameters, which is what we observe.
Comparing Figs. 4 and 5, we make an interesting observation. In a PCM, the parallel connection of all cells tends to average out differences in individual cell characteristics naturally since their voltages are forced to be equal. In an SCM, there is no such cell-to-cell averaging. The overall SCM voltages must be equal, but the individual cell voltages may be quite different. We conclude that without real-time balancing, the SCM architecture appears to be more prone to larger cell-voltage swings than does the PCM architecture.
The primary benefit of using physics-based models versus equivalent-circuit models is that we can also investigate cell internal variables. One variable of particular interest is the side-reaction overpotential η_s, which predicts the rate of SEI-layer growth and the onset of lithium plating. To avoid lithium plating, this value must never become negative. Fig. 6 compares values of η_s for the PCM-based pack to those of the SCM-based pack for the same simulations. Since the SCM-based architecture does not naturally average cell differences the same way as does the PCM-based pack, we see somewhat greater fluctuation in η_s

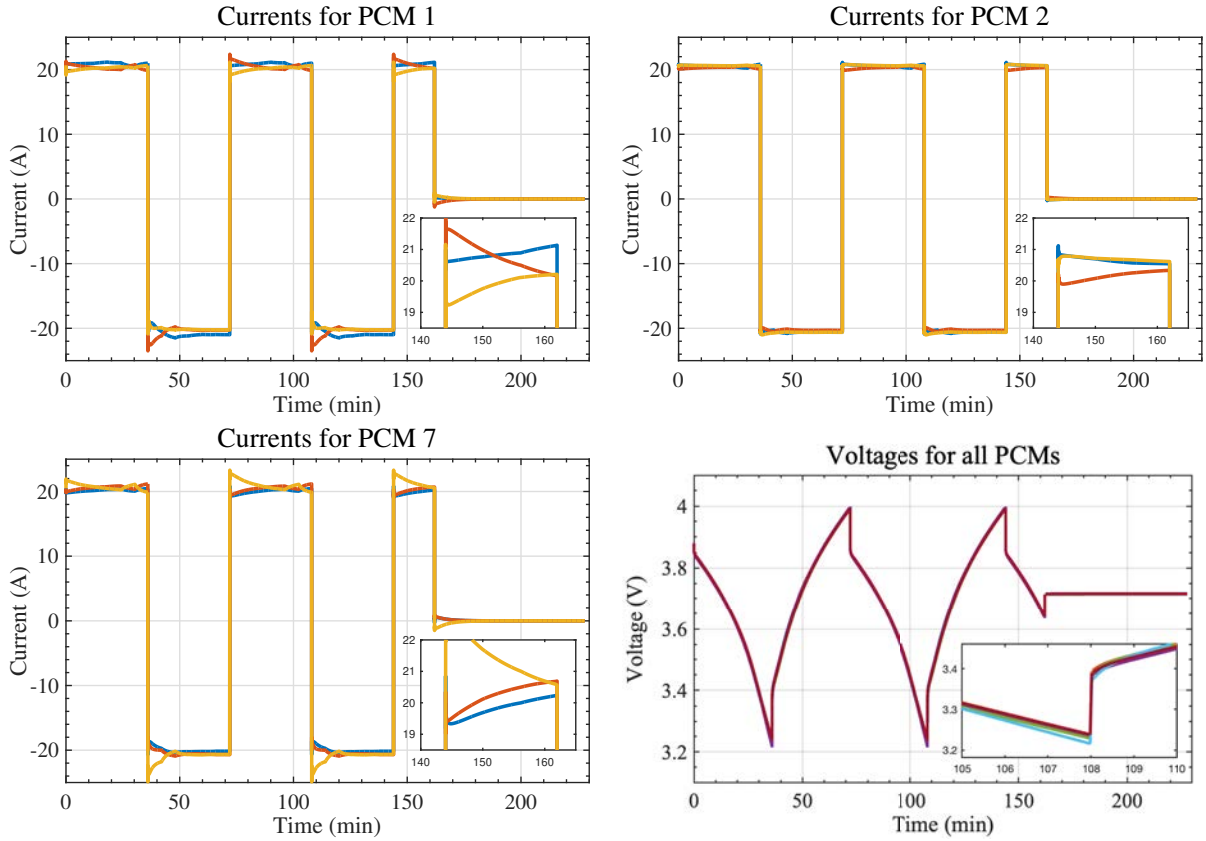

Figure 4: Simulation results for a PCM-based battery pack with $N_\mathrm{s} = 7$ and $N_\mathrm{p} = 3$. Insets show detail.

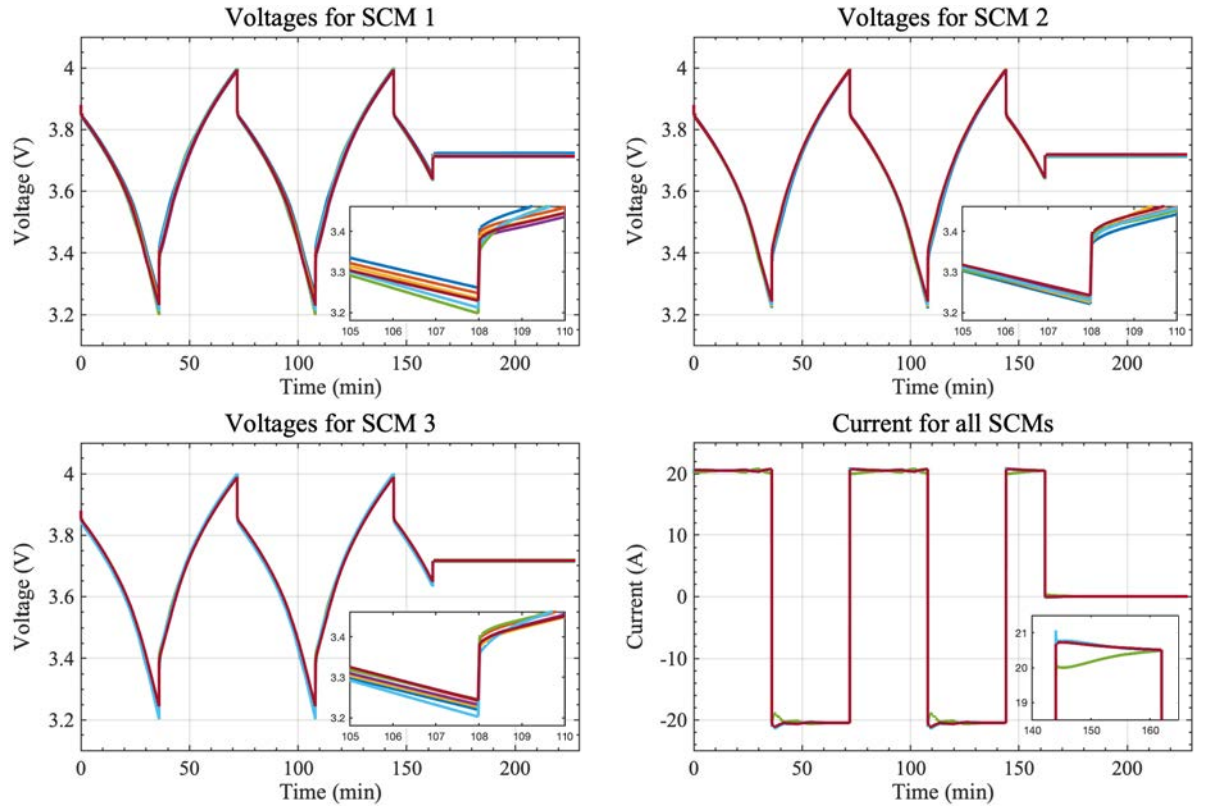

Figure 5: Simulation results for an SCM-based battery pack with $N_\mathrm{s} = 7$ and $N_\mathrm{p} = 3$. Insets show detail.

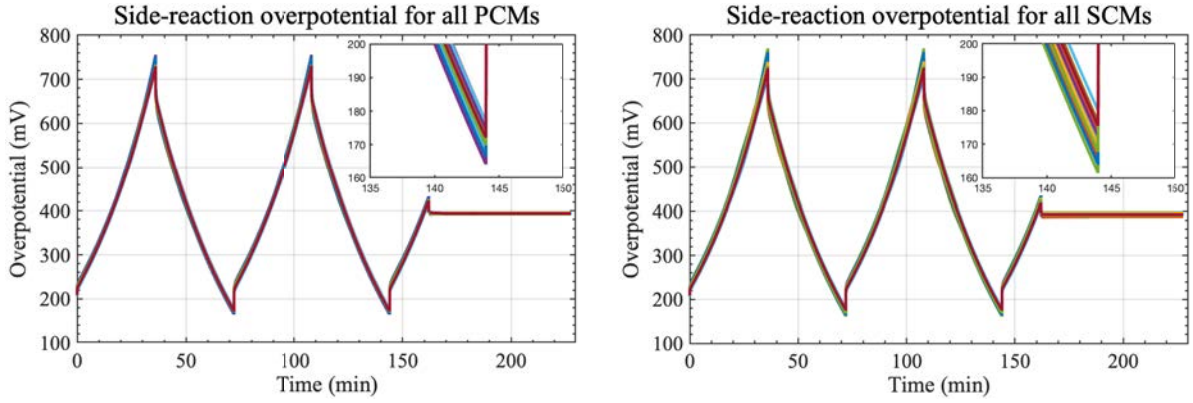

Figure 6: Illustrating differences between side-reaction overpotentials based on pack architecture.

for the SCM pack. For these particular simulations, there is no risk of lithium plating for either PCM or SCM. However, we conclude that the SCM-based pack is more susceptible to lithium plating than the PCM-based pack since its side-reaction overpotential has greater variation if there are large parameter differences between different cells in the pack.

8 Summary

This paper has proposed a method for simulating multicell battery packs using PB-ROMs. We reviewed a transfer-function method for creating a ROM around a specific SOC and temperature setpoint. We showed how to apply nonlinear corrections to the output of the ROM to improve its predictions of cell internal variables and voltage. We then discussed an output-blending approach to extend the operating range of the ROMs over a wide range of SOC and temperature.

These ROMs most naturally accept the cell's current $i_{\text{app}}[k]$ as input and produce voltage $v[k]$ as output. However, we showed how to use the ROMs in an optimization loop that could also simulate constant-voltage and constant-power events. We then extended this approach to allow simulating multicell battery packs where cells can be wired together in parallel and/or in series. Some sample results illustrate interesting features of the PCM and SCM tactics to modularizing a battery pack.

While simulating multicell battery packs using the PB-ROMs is not as simple as it was when using ECMs, it is still very feasible to do so, and a great deal of insight into pack operations can be gleaned. Further, we can predict the internal electrochemical variables of every cell that might lead to premature degradation and pack failure. This, in our opinion, is the primary advantage of using PB-ROMs instead of ECMs.

Acknowledgments

The information, data, or work presented herein was funded in part by the National Science Foundation under award to the ASPIRE Engineering Research Center headquartered at Utah State University in Logan, UT.

References

[1] G. L. Plett and M. J. Klein, "Simulating battery packs comprising parallel cell modules and series cell modules," in *Proc. of EVS*, 2009, pp. 1–17.

[2] G. L. Plett, *Battery Management Systems, Volume 2: Equivalent-Circuit Methods*. Artech House, 2015.

[3] M. Dubarry, N. Vuillaume, and B. Y. Liaw, "From li-ion single cell model to battery pack simulation," in *2008 IEEE International Conference on Control Applications*. IEEE, 2008, pp. 708–713.

[4] ——, "From single cell model to battery pack simulation for li-ion batteries," *Journal of Power Sources*, vol. 186, no. 2, pp. 500–507, 2009.

[5] M. Broussely, P. Biensan, F. Bonhomme, P. Blanchard, S. Herreyre, K. Nechev, and R. Staniewicz, "Main aging mechanisms in li ion batteries," *Journal of power sources*, vol. 146, no. 1-2, pp. 90–96, 2005.

[6] J. Vetter, P. Novák, M. Wagner, C. Veit, K.-C. Möller, J. Besenhard, M. Winter, M. Wohlfahrt-Mehrens, C. Vogler, and A. Hammouche, "Ageing mechanisms in lithium-ion batteries," *Journal of power sources*, vol. 147, no. 1, pp. 269–281, 2005.

[7] A. Barré, B. Deguilhem, S. Grolleau, M. Gérard, F. Suard, and D. Riu, "A review on lithium-ion battery ageing mechanisms and estimations for automotive applications," *Journal of Power Sources*, vol. 241, pp. 680–689, 2013.

[8] M. Doyle, T. F. Fuller, and J. Newman, "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell," *J. Electrochemical Society*, vol. 140, no. 6, p. 1526, 1993.

[9] J. Newman and K. E. Thomas-Alyea, *Electrochemical systems*. John Wiley & Sons, 2012.

[10] G. L. Plett, *Battery Management Systems, Volume 1: Battery Modeling*. Artech House, 2015.

[11] A. Rodriguez, G. L. Plett, and M. S. Trimboli, "Improved transfer functions modeling linearized lithium-ion battery-cell internal electrochemical variables," *Journal of Energy Storage*, vol. 20, pp. 560–575, 2018.

[12] A. Rodríguez, G. L. Plett, and M. S. Trimboli, "Comparing four model-order reduction techniques, applied to lithium-ion battery-cell internal electrochemical transfer functions," *eTransportation*, vol. 1, p. 100009, 2019.

[13] G. L. Plett and M. S. Trimboli, "Process for lumping parameters to enable nondestructive parameter estimation for lithium-ion physics-based models," in *Proc. of EVS*, 2022.

[14] J. L. Lee, L. L. Aldrich, K. D. Stetzel, and G. L. Plett, "Extended operating range for reduced-order model of lithium-ion cells," *Journal of Power Sources*, vol. 255, pp. 85–100, 2014.

Authors

Dr. Plett received his Ph.D. in Electrical Engineering from Stanford University and is now Professor of Electrical and Computer Engineering at the University of Colorado Colorado Springs. His research focuses on topics in control systems as applied to the management of high-capacity battery systems, such as found in electric vehicles. Current research efforts include: physics-based reduced-order modeling of ideal lithium-ion dynamics and degradation mechanisms; nondestructive parameter estimation for physics-based models; estimation of cell electrochemical and degradation state; state-of-charge and state-of-health estimation; life-extending power-prediction methods.

Dr. Trimboli received his Ph.D. in Control Engineering from Oxford University and is now Associate Professor of Electrical and Computer Engineering at the University of Colorado Colorado Springs. His research focuses on topics in control systems as applied to the management of high-performance battery systems. Current research efforts include: physics-based reduced-order modeling of ideal lithium-ion dynamics; MPC for battery fast-charge and power limit estimation for life extension; battery state estimation using Kalman filters; life-extending power-prediction methods.