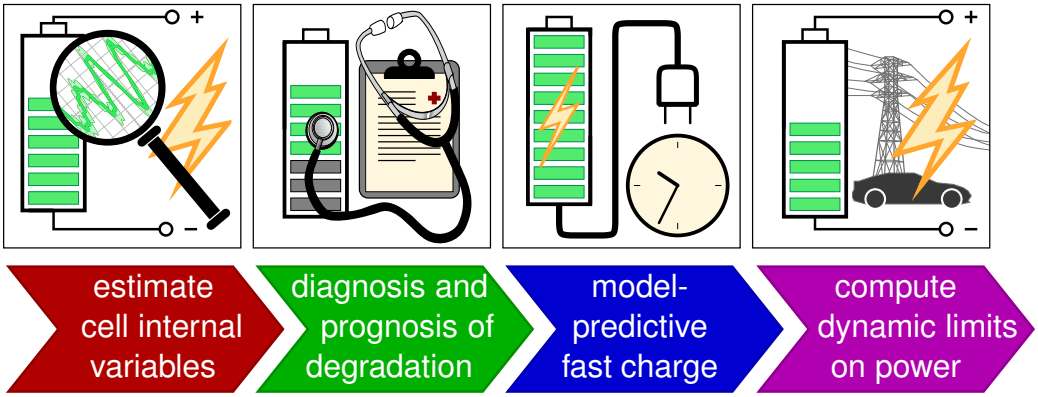
ELECTROCHEMICAL INTERNAL VARIABLES ESTIMATION

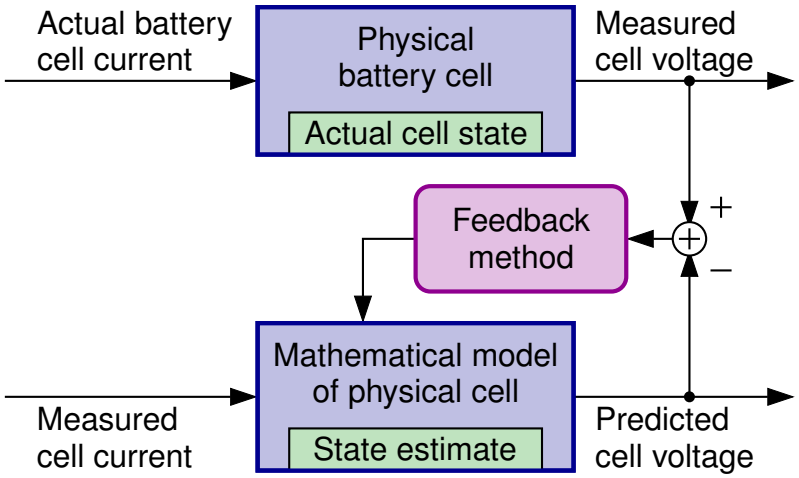
5.1: Introduction

- In ECE5720, you learned that a BMS must produce real-time estimates of SOC for all cells in the battery pack.
- You also learned that accurate (but conservative) voltage-based power-limit estimates could be made if estimates of all elements the ESC-model state vector (plus confidence bounds) were available.
- The same principle is true when using electrochemical models.



- In this chapter, you will learn how to:
  - Make real-time estimates of PBM state (with confidence bounds);
  - How to leverage state estimates to compute estimates of SOC and any desired set of cell electrochemical internal electrochemical variables (e.g., θ_s, θ_e, ϕ_s, ϕ_e, i_{f+dl}), also with confidence bounds.
- These estimates will be used in the following chapters in computations needed for fast charge and for power-limits estimates.

- As in ECE5720, we will use a model-based approach.
- The PBM—combined with input (current) and output (voltage) measurements—will be used to adapt a state estimate.
- Unlike the ESC model, however, the PBM's state vector $\boldsymbol{x}[k]$ is nonphysical.



- Therefore, we will need to extend the methods from ECE5720 to compute—from the state-vector—estimates of the internal electrochemical variables of interest (which are physical).
- In this chapter,
  - We will first review the idea of sequential probabilistic inference, and will add a necessary additional step to the process.
  - We will then review approaches for estimating statistics of a random variable that is the output of a nonlinear function of an input random variable.
  - We will apply these two concepts to the output-blended ROM to develop an internal-variables estimator.
- The topics we cover initially are review of Ch. 3 in ECE5720.
  - I will only summarize those topics here—you may wish to refer back to your ECE5720 notes if you wish a more detailed review.
  - The primary intent is to springboard off of these known topics to apply the methods to the PBROM.

5.2: Review of sequential probabilistic inference

- When applying model-based estimation, we start by assuming a general, possibly nonlinear, cell model:

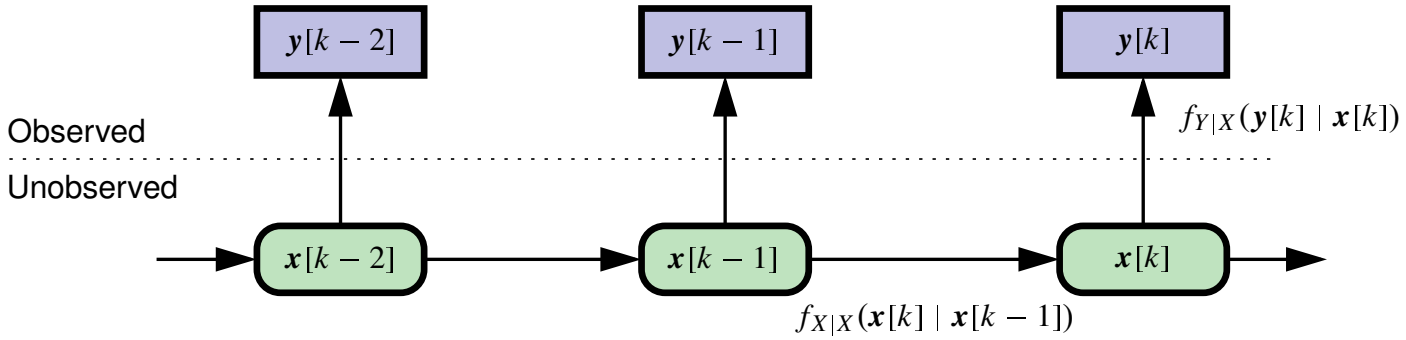
$$\boldsymbol{x}[k] = \boldsymbol{f}(\boldsymbol{x}[k-1], \boldsymbol{u}[k-1], \boldsymbol{w}[k-1])$$

$$\boldsymbol{y}[k] = \boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{v}[k]),$$

  where $\boldsymbol{u}[k]$ is a known (deterministic/measured) input signal, $\boldsymbol{w}[k]$ is a process-noise random input, and $\boldsymbol{v}[k]$ is sensor-noise random input.[1]

- We note that $\boldsymbol{f}(\cdot)$ and $\boldsymbol{h}(\cdot)$ may be time-varying, but we generally omit the time dependency from the notation for ease of understanding.

- Probabilistic sequential inference is a process for estimating a dynamic system's present state $\boldsymbol{x}[k]$ using the set of measurements $\mathbb{Y}[k] = \{\boldsymbol{y}[0], \cdots, \boldsymbol{y}[k]\}$ (also implicitly assuming knowledge of the set of inputs $\mathbb{U}[k] = \{\boldsymbol{u}[0], \cdots, \boldsymbol{u}[k]\}$).



- The observations allow us to "peek" at what is happening in the true system. Based on observations and our model, we estimate the state.
  - However, process-noise and sensor-noise randomness cause us never to be able to compute the state exactly.

[1] This overall topic depends heavily on topics in probability theory, covered in ECE5720. I won't spend time reviewing these topics here, but you may wish to do so on your own.

- In the notation that we use
  - A superscript "$-$" indicates a predicted quantity based only on past measurements.
  - A superscript "$+$" indicates an estimated quantity based on both past and present measurements.
  - A "hat" ($\hat{\cdot}$) symbol indicates a predicted or estimated quantity.
  - A "tilde" ($\tilde{\cdot}$) symbol indicates an error: the difference between a true and predicted or estimated quantity.
  - A "Σ" symbol is used to denote the correlation between the two arguments in its subscript (autocorrelation if only one is given):

$$\boldsymbol{\Sigma}_{\boldsymbol{x}\boldsymbol{y}} = \mathbb{E}[\boldsymbol{x}\boldsymbol{y}^T] \qquad \text{and} \qquad \boldsymbol{\Sigma}_{\boldsymbol{x}} = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^T].$$

  - Furthermore, if the arguments are zero mean (as they often are in the quantities we talk about), then this represents covariance:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}}} = \mathbb{E}[\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}}^T] = \mathbb{E}[(\tilde{\boldsymbol{x}} - \mathbb{E}[\tilde{\boldsymbol{x}}])(\tilde{\boldsymbol{y}} - \mathbb{E}[\tilde{\boldsymbol{y}}])^T],$$

    for zero-mean $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$.

- We seek a state estimate that minimizes the "mean-squared error":

$$\hat{\boldsymbol{x}}^{\text{MMSE}}[k](\mathbb{Y}[k]) = \arg\min_{\hat{\boldsymbol{x}}[k]} \left(\mathbb{E}\Big[\left\| \boldsymbol{x}[k] - \hat{\boldsymbol{x}}^+[k] \right\|_2^2 \mid \mathbb{Y}[k] \Big] \right).$$

- For this metric, the optimal state estimate is $\hat{\boldsymbol{x}}^+[k] = \mathbb{E}\Big[\boldsymbol{x}[k] \mid \mathbb{Y}[k]\Big]$.
- If all random variables have a Gaussian distribution—as we assume—this estimate can be computed via a predict/correct mechanism:

$$\hat{\boldsymbol{x}}^+[k] = \hat{\boldsymbol{x}}^-[k] + \boldsymbol{L}[k]\tilde{\boldsymbol{y}}[k].$$

  - The state prediction is $\hat{\boldsymbol{x}}^-[k] = \mathbb{E}\Big[\boldsymbol{x}[k] \mid \mathbb{Y}[k-1]\Big]$.

- State prediction error is $\tilde{\boldsymbol{x}}^-[k] = \boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k]$.
  - Error is always "truth minus prediction" or "truth minus estimate."
  - We can't compute error in practice, as truth value is not known.
  - But, we can prove statistical results using this definition that give an algorithm for estimating the truth using measurable values.
- Measurement innovation (what is new or unexpected in measurement) is $\tilde{\boldsymbol{y}}[k] = \boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]$, where $\hat{\boldsymbol{y}}[k] = \mathbb{E}\big[\boldsymbol{y}[k] \mid \mathbb{Y}[k-1]\big]$.

■ Note that $\boldsymbol{L}[k]$ is a function of $\boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{x}}}[k]$, which may be computed as:

$$\boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{x}}}[k] = \boldsymbol{\Sigma}^-_{\tilde{\boldsymbol{x}}}[k] - \boldsymbol{L}[k]\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}[k]\boldsymbol{L}^T[k].$$

■ Overall, the output of this process has two components:

1. The state estimate. Every iteration, we compute our best guess of the present state value, which is $\hat{\boldsymbol{x}}^+[k]$.
2. The covariance estimate. The covariance matrix $\boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{x}}}[k]$ gives the uncertainty of $\hat{\boldsymbol{x}}^+[k]$, and can be used to compute error bounds.

■ Summarizing, the generic Gaussian sequential probabilistic inference recursion becomes:

$$\begin{aligned}\hat{\boldsymbol{x}}^+[k] &= \hat{\boldsymbol{x}}^-[k] + \boldsymbol{L}[k]\big(\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]\big) = \hat{\boldsymbol{x}}^-[k] + \boldsymbol{L}[k]\tilde{\boldsymbol{y}}[k] \\ \boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{x}}}[k] &= \boldsymbol{\Sigma}^-_{\tilde{\boldsymbol{x}}}[k] - \boldsymbol{L}[k]\Sigma_{\tilde{\boldsymbol{y}}}[k]\boldsymbol{L}^T[k],\end{aligned}$$

where

$$\hat{\boldsymbol{x}}^-[k] = \mathbb{E}\big[\boldsymbol{x}[k] \mid \mathbb{Y}[k-1]\big], \quad \boldsymbol{\Sigma}^-_{\tilde{\boldsymbol{x}}}[k] = \mathbb{E}\big[(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k])(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k])^T\big] = \mathbb{E}\big[(\tilde{\boldsymbol{x}}^-[k])(\tilde{\boldsymbol{x}}^-[k])^T\big]$$

$$\hat{\boldsymbol{x}}^+[k] = \mathbb{E}\big[\boldsymbol{x}[k] \mid \mathbb{Y}[k]\big], \quad \boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{x}}}[k] = \mathbb{E}\big[(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^+[k])(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^+[k])^T\big] = \mathbb{E}\big[(\tilde{\boldsymbol{x}}^+[k])(\tilde{\boldsymbol{x}}^+[k])^T\big]$$

$$\hat{\boldsymbol{y}}[k] = \mathbb{E}\big[\boldsymbol{y}[k] \mid \mathbb{Y}[k-1]\big], \quad \boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}[k] = \mathbb{E}\big[(\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k])(\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k])^T\big] = \mathbb{E}\big[(\tilde{\boldsymbol{y}}[k])(\tilde{\boldsymbol{y}}[k])^T\big]$$

$$\boldsymbol{L}[k] = \mathbb{E}\big[(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k])(\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k])^T\boldsymbol{\Sigma}^{-1}_{\tilde{\boldsymbol{y}}}[k]\big] = \boldsymbol{\Sigma}^-_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}}}[k]\boldsymbol{\Sigma}^{-1}_{\tilde{\boldsymbol{y}}}[k].$$

■ Note that this is a linear recursion, even if the system is nonlinear(!)

5.3: The eight-step process

- The generic Gaussian sequential probabilistic inference solution can be divided into two main steps, each having three sub-steps.
  - We add one more main step having two-sub-steps.

General step 1a: State prediction time update.

- Each time step, compute an updated prediction of the present value of $\boldsymbol{x}[k]$, based on prior information and the system model:

$$\hat{\boldsymbol{x}}^-[k] = \mathbb{E}\Big[\boldsymbol{x}[k] \mid \mathbb{Y}[k-1]\Big] = \mathbb{E}\Big[\boldsymbol{f}(\boldsymbol{x}[k-1], \boldsymbol{u}[k-1], \boldsymbol{w}[k-1]) \mid \mathbb{Y}[k-1]\Big].$$

General step 1b: Prediction-error covariance time update.

- Determine the predicted state-estimate error covariance matrix $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^-[k]$ based on prior information and the system model.
- We compute $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^-[k] = \mathbb{E}\Big[(\tilde{\boldsymbol{x}}^-[k])(\tilde{\boldsymbol{x}}^-[k])^T\Big]$, where $\tilde{\boldsymbol{x}}^-[k] = \boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k]$.

General step 1c: Predict system output (i.e., cell voltage).

- Predict the system's output using prior information:

$$\hat{\boldsymbol{y}}[k] = \mathbb{E}\Big[\boldsymbol{y}[k] \mid \mathbb{Y}[k-1]\Big] = \mathbb{E}\Big[\boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{v}[k]) \mid \mathbb{Y}[k-1]\Big].$$

General step 2a: Estimator gain matrix $\boldsymbol{L}[k]$.

- Compute estimator gain matrix $\boldsymbol{L}[k]$ as $\boldsymbol{L}[k] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}}}^-[k]\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}^{-1}[k]$.

General step 2b: State estimate measurement update.

- Compute the posterior state estimate by updating the prediction using the $\boldsymbol{L}[k]$ and the innovation $\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]$:

$$\hat{\boldsymbol{x}}^+[k] = \hat{\boldsymbol{x}}^-[k] + \boldsymbol{L}[k](\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]).$$

General step 2c: Estimation-error covariance measurement update.

- Compute the posterior error covariance matrix:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^{+}[k] = \mathbb{E}\Big[(\tilde{\boldsymbol{x}}^{+}[k])(\tilde{\boldsymbol{x}}^{+}[k])^{T}\Big] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^{-}[k] - \boldsymbol{L}[k]\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}[k]\boldsymbol{L}^{T}[k].$$

Additional step 3a: Estimate internal variables.

- The standard Gaussian sequential probabilistic inference solution has only Steps 1a through 2c.
- However, the state of our model is not directly physical.
  - We desire to leverage the state estimate to compute estimates of internal electrochemical physical variables.
- We model these additional physical variables as nonlinear functions of the state and the input:

$$\boldsymbol{z}[k] = \boldsymbol{g}_{z}(\boldsymbol{x}[k], u[k]).$$

- Then, this step estimates the variables as:

$$\hat{\boldsymbol{z}}^{+}[k] = \mathbb{E}\Big[\boldsymbol{z}[k] \mid \mathbb{Y}[k]\Big] = \mathbb{E}\Big[\boldsymbol{g}_{z}(\boldsymbol{x}[k], u[k]) \mid \mathbb{Y}[k]\Big].$$

Additional step 3b: Internal variables estimation-error covariance.

- Finally, we compute an estimate of the estimation-error covariance:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{+}[k] = \mathbb{E}\Big[(\tilde{\boldsymbol{z}}^{+}[k])(\tilde{\boldsymbol{z}}^{+}[k])^{T}\Big].$$

KEY POINT: The estimator output comprises the variables estimate $\hat{\boldsymbol{z}}^{+}[k]$ and estimation-error covariance estimate $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{+}[k]$.

- As a result, we have high confidence that the true $\boldsymbol{z}[k]$ lies within $\hat{\boldsymbol{z}}^{+}[k] \pm 3\sqrt{\operatorname{diag}\left(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{+}[k]\right)}$.
- The estimator then waits until the next sample interval, updates k, and proceeds to Step 1a.

- We can summarize the process as:

Prediction

Step 1a: State prediction time update

$$\hat{\boldsymbol{x}}^-[k] = \mathbb{E}\big[\boldsymbol{x}[k] \mid \mathbb{Y}[k-1]\big] = \mathbb{E}\big[\boldsymbol{f}(\boldsymbol{x}[k-1], \boldsymbol{u}[k-1], \boldsymbol{w}[k-1]) \mid \mathbb{Y}[k-1]\big].$$

Step 1b: Prediction-error covariance time update

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^-[k] = \mathbb{E}\big[(\tilde{\boldsymbol{x}}^-[k])(\tilde{\boldsymbol{x}}^-[k])^T\big] = \mathbb{E}\big[(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k])(\boldsymbol{x}[k] - \hat{\boldsymbol{x}}^-[k])^T\big].$$

Step 1c: Predict system output

$$\hat{\boldsymbol{y}}[k] = \mathbb{E}\big[\boldsymbol{y}[k] \mid \mathbb{Y}[k-1]\big] = \mathbb{E}\big[\boldsymbol{h}(\boldsymbol{x}[k], \boldsymbol{u}[k], \boldsymbol{v}[k]) \mid \mathbb{Y}[k-1]\big].$$

Correction

Step 2a: Estimator gain matrix

$$\boldsymbol{L}[k] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}\tilde{\boldsymbol{y}}}^-[k]\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}[k]^{-1}.$$

Step 2b: State estimate measurement update

$$\hat{\boldsymbol{x}}^+[k] = \hat{\boldsymbol{x}}^-[k] + \boldsymbol{L}[k](\boldsymbol{y}[k] - \hat{\boldsymbol{y}}[k]).$$

Step 2c: Estimation-error covariance measurement update

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^+[k] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}^-[k] - \boldsymbol{L}[k]\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}[k]\boldsymbol{L}[k]^T.$$

Extension

Step 3a: Estimate internal variables

$$\hat{\boldsymbol{z}}^+[k] = \mathbb{E}\big[\boldsymbol{z}[k] \mid \mathbb{Y}[k]\big] = \mathbb{E}\big[\boldsymbol{g}_z(\boldsymbol{x}[k], \boldsymbol{u}[k]) \mid \mathbb{Y}[k]\big].$$

Step 3b: Internal variables estimation-error covariance

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^+[k] = \mathbb{E}\big[(\tilde{\boldsymbol{z}}^+[k])(\tilde{\boldsymbol{z}}^+[k])^T\big].$$

- The key to implementing these steps is to find procedural ways to compute the expected values in these equations.
- Extended and sigma-point Kalman filters (EKF, SPKF) use analytic and numeric linearization, respectively, to do so.
- We will present both approaches applied to output-blended ROMs. . . but first we need to review those ROMs and introduce some notation that will help see how to reduce computational requirements.

5.4: Setup for xKF with output-blended models

- Before applying xKFs for state and internal-variables estimation, we review the output-blending method and refine some notation.
- Each PBROM has its own state vector, modeled as:

$$\mathfrak{X}_j[k] = \mathfrak{A}_j \mathfrak{X}_j[k-1] + \mathfrak{B}_j u[k-1],$$

  for all $1 \leq j \leq N$, where $\mathfrak{X}_j[k]$ is the state of integrator-augmented model j, $\mathfrak{A}_j$ and $\mathfrak{B}_j$ are the constant state-equation matrices for precomputed integrator-augmented model j, and N is the number of precomputed models.
- Since all models will have identical integral states, the integration should be factored out of all models and computed separately.
- For example, if $\mathfrak{X}_j[k]$ is 5×1, when we remove integration state and retain only transient states, we have 4×1 $\boldsymbol{x}_j[k]$.
- Similarly, the 5×5 $\mathfrak{A}_j$ matrix becomes 4×4 $\boldsymbol{A}_j$ and the 5×1 $\mathfrak{B}_j$ vector becomes a 4×1 $\boldsymbol{B}_j$:

$$\boldsymbol{x}_j[k] = \boldsymbol{A}_j \boldsymbol{x}_j[k-1] + \boldsymbol{B}_j u[k-1].$$

- The scalar integral state $x_0[k-1]$ common to all models is updated as:

$$x_0[k] = x_0[k-1] + u[k-1].$$

- Taken altogether, this is the same as:

$$\underbrace{\left[\begin{array}{c} \boldsymbol{x}_1[k] \\ \hdashline \vdots \\ \hdashline \boldsymbol{x}_N[k] \\ \hdashline x_0[k] \end{array}\right]}_{\mathbf{X}[k]} = \underbrace{\left[\begin{array}{c:c:c:c} \boldsymbol{A}_1 & & & \mathbf{0} \\ \hdashline & \ddots & & \\ \hdashline & & \boldsymbol{A}_N & \\ \hdashline \mathbf{0} & & & 1 \end{array}\right]}_{\mathbf{A}} \underbrace{\left[\begin{array}{c} \boldsymbol{x}_1[k-1] \\ \hdashline \vdots \\ \hdashline \boldsymbol{x}_N[k-1] \\ \hdashline x_0[k-1] \end{array}\right]}_{\mathbf{X}[k-1]} + \underbrace{\left[\begin{array}{c} \boldsymbol{B}_1 \\ \hdashline \vdots \\ \hdashline \boldsymbol{B}_N \\ \hdashline 1 \end{array}\right]}_{\mathbf{B}} u[k-1].$$

- Note that we are using somewhat nonstandard notation. $\mathbf{X}[k]$ is a vector quantity, but we use a capital letter—normally used only for matrices—to emphasize that this is a block-vector.

- Next, output blending uses bilinear interpolation to combine the linear outputs of the models corresponding to the four setpoints closest to the present operating SOC and temperature.

- If we define:

$$\theta = \frac{\text{SOC} - \text{SOC}_0}{\text{SOC}_1 - \text{SOC}_0}, \quad \bar{\theta} = 1 - \theta, \quad \psi = \frac{T - T_0}{T_1 - T_0}, \quad \text{and} \quad \bar{\psi} = 1 - \psi,$$

  then the linear ROM output vector is:

$$\boldsymbol{y}[k] = \bar{\psi}(\bar{\theta}\boldsymbol{y}_{0,0}[k] + \theta\boldsymbol{y}_{1,0}[k]) + \psi(\bar{\theta}\boldsymbol{y}_{0,1}[k] + \theta\boldsymbol{y}_{1,1}[k]) \tag{5.1}$$

$$= \gamma_{0,0}\boldsymbol{y}_{0,0}[k] + \gamma_{0,1}\boldsymbol{y}_{0,1}[k] + \gamma_{1,0}\boldsymbol{y}_{1,0}[k] + \gamma_{1,1}\boldsymbol{y}_{1,1}[k] \tag{5.2}$$

$$= \sum_{j=1}^{N} \gamma_j \boldsymbol{y}_j[k]. \tag{5.3}$$

- Notice that 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 weighting constants $\{\gamma_j\}$ for $1 \leq j \leq N$ in Eq. (5.3), 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. (5.2).
  - This reduces the required computation.

- The linear outputs described by Eq. (5.1) could be computed using the full-sized state vectors as:

$$\boldsymbol{y}_j[k] = \mathfrak{C}_j\boldsymbol{x}_j[k] + \mathfrak{D}_j u[k].$$

- Can also use integrator-removed state vectors if we recall that $\boldsymbol{C}_j$ is the first n columns of $\mathfrak{C}_j$ and define $\boldsymbol{C}_0$ to be the final column. Then,

$$\boldsymbol{y}_j[k] = \left[\begin{array}{c:c} \boldsymbol{C}_j & \boldsymbol{C}_0 \end{array}\right] \left[\begin{array}{c} \boldsymbol{x}_j[k] \\ \hdashline x_0[k] \end{array}\right] + \boldsymbol{D}_j u[k].$$

  - The $\boldsymbol{C}_0$ vector comprises the "residues" of the integration state for each linearized model output.
  - Most entries in $\boldsymbol{C}_0$ are zero since most electrochemical variables do not integrate the input current.
  - The residues are weak functions of SOC and temperature; we choose to compute them using analytic expressions during execution rather than relying on $\boldsymbol{C}_0$ from precomputed ROMs.
  - But, we will retain $\boldsymbol{C}_0$ in our notation for now.

- Expanding Eq. (5.3), the output-blended final result is then:

$$\boldsymbol{y}[k] = \sum_{i=1}^{N} \gamma_j \left(\boldsymbol{C}_j \boldsymbol{x}_j[k] + \boldsymbol{C}_{0,j} x_0[k] + \boldsymbol{D}_j u[k]\right). \tag{5.4}$$

- The summation is written for $1 \leq j \leq N$ but since only four values of γ_j are nonzero, computation is reduced by considering only the nonzero terms in the summation.

- Internal electrochemical variables and cell voltage are computed by applying nonlinear corrections to $\boldsymbol{y}[k]$:

$$\boldsymbol{z}[k] = \boldsymbol{g}_z(\boldsymbol{y}[k]). \tag{5.5}$$

- In particular, we can combine Eqs. (5.4) and (5.5) to compute the cell voltage, writing:

$$v_{\text{cell}}[k] = g_v(\mathbf{X}[k], u[k]).$$

5.5: Principles of EKF and SPKF

- We seek to apply Gaussian sequential probabilistic inference to the problem of estimating a cell's internal states and variables.
- The challenge is that some of our model equations are nonlinear:
  - When the inputs are random variables having known mean and covariance, we desire to compute the mean and covariance of the outputs of these equations.
- There is no general solution that does so. Instead, we must make approximations.
- The EKF and SPKF approximate the mean and covariance of the output of a nonlinear function in different ways.

The extended Kalman filter (EKF)

- The EKF makes two simplifying assumptions when adapting the general sequential inference equations to a nonlinear system:
  - When computing means of the output of a nonlinear function, EKF assumes $\mathbb{E}[\mathrm{fn}(\boldsymbol{x})] \approx \mathrm{fn}(\mathbb{E}[\boldsymbol{x}])$, which is not true in general;
  - When computing covariance estimates, EKF uses Taylor-series expansion to linearize the system equations around the present operating point.
- We will apply the EKF assumptions to the equations for an output-blended ROM in the next section. Here, we see generic examples of how the two assumptions are applied.
- Consider a nonlinear function having vector input $\boldsymbol{x}$ and vector output $\boldsymbol{y}$. Specifically, consider: $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$.

- The EKF approach approximates the expected value of $\boldsymbol{y}$ as follows:

$$\begin{aligned}\bar{\boldsymbol{y}} &= \mathbb{E}[\boldsymbol{f}(\boldsymbol{x})] \\ &\approx \boldsymbol{f}(\mathbb{E}[\boldsymbol{x}]) = \boldsymbol{f}(\bar{\boldsymbol{x}}).\end{aligned}$$

- To see how the EKF approach approximates the covariance of $\boldsymbol{y}$, we first make an approximation for $\tilde{\boldsymbol{y}}$:

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} - \bar{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\bar{\boldsymbol{x}}).$$

- The first term is expanded as a Taylor series around $\bar{\boldsymbol{x}}$:

$$\boldsymbol{y} \approx \boldsymbol{f}(\bar{\boldsymbol{x}}) + \underbrace{\left.\frac{\mathrm{d}\boldsymbol{f}(\boldsymbol{x}])}{\mathrm{d}\boldsymbol{x}}\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}}}_{\text{Defined as } \widehat{\boldsymbol{A}}} \tilde{\boldsymbol{x}},$$

  where $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}$. This gives $\tilde{\boldsymbol{y}} \approx \widehat{\boldsymbol{A}}\tilde{\boldsymbol{x}}$.

- Substituting this to find the covariance:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}} = \mathbb{E}[\tilde{\boldsymbol{y}}\tilde{\boldsymbol{y}}^T] \approx \widehat{\boldsymbol{A}}\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}\widehat{\boldsymbol{A}}^T.$$

- Same basic two approaches are used throughout the EKF derivation.

The sigma-point Kalman filter (SPKF)

- The SPKF uses weighted averages of function output values corresponding to carefully chosen function input values to estimate means and covariances.
- It chooses a set of sigma points $\boldsymbol{\mathcal{X}}$ so that the (possibly weighted) mean and covariance of the points exactly matches the mean $\bar{\boldsymbol{x}}$ and covariance $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}$ of the *a priori* random variable being modeled.
- These points are then passed through the nonlinear function, resulting in a transformed set of points $\boldsymbol{\mathcal{Y}}$.

- The *a posteriori* mean $\bar{\boldsymbol{y}}$ and covariance $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{y}}}$ are then approximated by the mean and covariance of these transformed points $\boldsymbol{\mathcal{Y}}$.
- Specifically, if input RV $\boldsymbol{x}$ has dimension L, mean $\bar{\boldsymbol{x}}$, and covariance $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}$, then $p+1 = 2L+1$ sigma points are generated as the set:

$$\boldsymbol{\mathcal{X}} = \left\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}} + \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}}, \bar{\boldsymbol{x}} - \gamma\sqrt{\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}}}\right\},$$

  with members of $\boldsymbol{\mathcal{X}}$ indexed from 0 to p, and where the matrix square root $\boldsymbol{R} = \sqrt{\boldsymbol{\Sigma}}$ computes a result such that $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{R}^T$.
  - Usually, the efficient *Cholesky decomposition* is used, resulting in lower-triangular $\boldsymbol{R}$. (Take care: MATLAB, by default, returns an upper-triangular matrix that must be transposed.)
- The weighted mean and covariance of $\boldsymbol{\mathcal{X}}$ are equal to the original mean and covariance of $\boldsymbol{x}$ for some $\{\gamma, \alpha^{(\mathrm{m})}, \alpha^{(\mathrm{c})}\}$ if we compute:

$$\bar{\boldsymbol{x}} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{X}}_i \qquad \text{and} \qquad \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} (\boldsymbol{\mathcal{X}}_i - \bar{\boldsymbol{x}})(\boldsymbol{\mathcal{X}}_i - \bar{\boldsymbol{x}})^T,$$

  where $\boldsymbol{\mathcal{X}}_i$ is the ith member of $\boldsymbol{\mathcal{X}}$, and both $\alpha_i^{(m)}$ and $\alpha_i^{(\mathrm{c})}$ are real scalars where $\alpha_i^{(m)}$ and $\alpha_i^{(\mathrm{c})}$ must both sum to one.
  - Different sigma-point methods use different weighting constants; popular choices are listed below.[2]

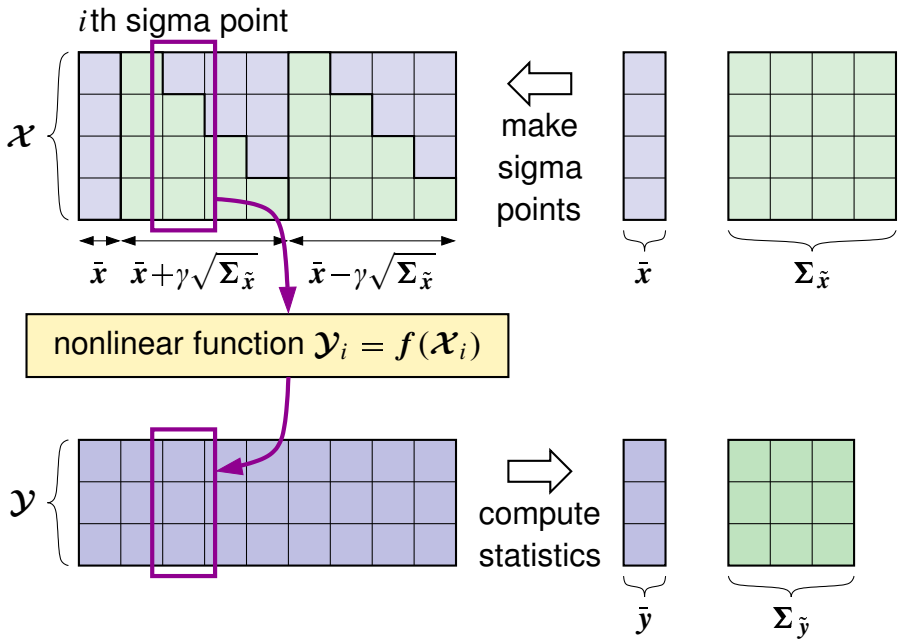
Method	γ	$\alpha_0^{(\mathrm{m})}$	$\alpha_k^{(\mathrm{m})}$	$\alpha_0^{(\mathrm{c})}$	$\alpha_k^{(\mathrm{c})}$
Unscented Kalman filter (UKF)	$\sqrt{L+\lambda}$	$\dfrac{\lambda}{L+\lambda}$	$\dfrac{1}{2(L+\lambda)}$	$\dfrac{\lambda}{L+\lambda} + (1-\alpha^2+\beta)$	$\dfrac{1}{2(L+\lambda)}$
Central-difference Kalman filter (CDKF)	h	$\dfrac{h^2-L}{h^2}$	$\dfrac{1}{2h^2}$	$\dfrac{h^2-L}{h^2}$	$\dfrac{1}{2h^2}$

[2] $\lambda = \alpha^2(L+\kappa) - L$ is a scaling parameter, with $(10^{-2} \leq \alpha \leq 1)$. Note that this α is different from $\alpha^{(\mathrm{m})}$ and $\alpha^{(\mathrm{c})}$. κ is either 0 or $3-L$. Also, β incorporates prior information. For Gaussian RVs, $\beta = 2$. h may take any positive value. For Gaussian RVs, $h = \sqrt{3}$.

- Output sigma points are computed: $\mathcal{Y}_i = f(\mathcal{X}_i)$. Then, the output mean and covariance are computed as well:

$$\bar{y} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{m})} \mathcal{Y}_i \qquad \text{and} \qquad \Sigma_{\tilde{y}} = \sum_{i=0}^{p} \alpha_i^{(\mathrm{c})} (\mathcal{Y}_i - \bar{y})(\mathcal{Y}_i - \bar{y})^T.$$

- The diagram illustrates the overall process, with the sets $\mathcal{X}$ and $\mathcal{Y}$ stored compactly with each set member a column in a matrix:



- The input variables $\bar{x}$ and $\Sigma_{\tilde{x}}$ representing random variable x are used to create sigma points $\mathcal{X}_i$.
- Sigma points $\mathcal{X}_i$ are input to the nonlinear function $f(\cdot)$ one at a time to produce output sigma points $\mathcal{Y}_i$.
- The mean and covariance of $\mathcal{Y}_i$ are computed to be $\bar{y}$ and $\Sigma_{\tilde{y}}$.
- These two output variables represent the mean and covariance of output random variable y.

5.6: EKF with the output-blended model

- Application of EKF to the output-blended ROM is now described.
- Some computational simplifications result from the state equation being linear for the ROM we are using.

Step 1a: State prediction time update.

- The first step of the EFK, every iteration, is to predict the present value of the state using only prior information.
- We define $\widehat{\mathbf{X}}^-[k] = \mathbb{E}[\mathbf{X}[k] \mid v_{\text{cell}}[0] \ldots v_{\text{cell}}[k-1]]$.
- We modify our state equation to consider "process noise" $w[k]$.
- We assume that this is white Gaussian zero-mean current-sensor noise added directly to $u[k]$.

$$\mathbf{X}[k] = \mathbf{A}\mathbf{X}[k-1] + \mathbf{B}\left(u[k-1] + w[k-1]\right).$$

- The state prediction we wish to compute is (for *all* models):

$$\begin{aligned}\widehat{\mathbf{X}}^-[k] &= \mathbb{E}\Big[\mathbf{X}[k] \mid \mathbb{V}_{\text{cell}}[k-1]\Big] \\ &= \mathbb{E}\Big[\mathbf{A}\mathbf{X}[k-1] + \mathbf{B}\left(u[k-1] + w[k-1]\right) \mid \mathbb{V}_{\text{cell}}[k-1]\Big]. \\ &= \mathbf{A}\widehat{\mathbf{X}}^+[k-1] + \mathbf{B}u[k-1],\end{aligned}$$

where $\widehat{\mathbf{X}}^+[k-1] = \mathbb{E}[\mathbf{X}[k-1] \mid v_{\text{cell}}[0] \ldots v_{\text{cell}}[k-1]]$ is the prior state estimate produced by the EKF.[3]

- This can also be implemented on a per-model basis, $1 \leq j \leq N$:

$$\hat{\boldsymbol{x}}_j^-[k] = \boldsymbol{A}_j\hat{\boldsymbol{x}}_j^+[k-1] + \boldsymbol{B}_j u[k-1]$$

$$\hat{x}_0^-[k] = \hat{x}_0^+[k-1] + u[k-1].$$

[3] Noticing that $\mathbf{A}$ is diagonal and that $\mathbf{B}$ is a vector of units values can greatly simplify the computational complexity of an implementation in many of the EKF steps. For example, Hadamard products can be used to compute quantities like $\mathbf{A}\widehat{\mathbf{X}}^+[k-1]$.

Step 1b: Prediction-error covariance time update.

- The next step of the EKF, every iteration, is to compute the covariance (uncertainty) of the state prediction.
- We define:

$$\mathbf{\Sigma}^{-}_{\tilde{\mathbf{X}}}[k] = \mathbb{E}[(\mathbf{X}[k] - \widehat{\mathbf{X}}^{-}[k])(\mathbf{X}[k] - \widehat{\mathbf{X}}^{-}[k])^T],$$

  which leads to:

$$\mathbf{\Sigma}^{-}_{\tilde{\mathbf{X}}}[k] = \mathbf{A}\mathbf{\Sigma}^{+}_{\tilde{\mathbf{X}}}[k-1]\mathbf{A}^T + \mathbf{B}\Sigma_{\tilde{w}}\mathbf{B}^T,$$

  where $\mathbf{\Sigma}^{+}_{\tilde{\mathbf{X}}}[k-1] \in \mathbb{R}^{(Nn+1)\times(Nn+1)}$ is the prior covariance (uncertainty) matrix of the state estimation error produced by the EKF and $\Sigma_{\tilde{w}} \in \mathbb{R}$ is the process-noise covariance.
- This can also be implemented on a per-model basis, $1 \le j \le N$:

$$\mathbf{\Sigma}^{-}_{\tilde{\boldsymbol{x}}_j}[k] = \boldsymbol{A}_j\mathbf{\Sigma}^{+}_{\tilde{\boldsymbol{x}}_j}[k-1]\boldsymbol{A}_j^T + \boldsymbol{B}_j\Sigma_{\tilde{w}}\boldsymbol{B}_j^T$$

$$\Sigma^{-}_{\tilde{x}_0}[k] = \Sigma^{+}_{\tilde{x}_0}[k-1] + \Sigma_{\tilde{w}}.$$

Step 1c: Predict system output.

- We next predict the voltage we will measure. That is, we compute:

$$\hat{v}_{\text{cell}}[k] = \mathbb{E}\Big[v_{\text{cell}}[k] \mid \mathbb{V}_{\text{cell}}[k-1]\Big].$$

- Before continuing, we modify the model's voltage equation to add white Gaussian zero-mean measurement noise $v[k]$:

$$v_{\text{cell}}[k] = g_v(\mathbf{X}[k], u[k]) + v[k]. \tag{5.6}$$

- In fact, $v_{\text{cell}}[k]$ depends only on the four models being combined by output blending, not on the entire model set.
- Therefore, we define a new vector,

$$\mathbf{X}_\gamma[k] = \begin{bmatrix} \boldsymbol{x}_{0,0}[k] \\ \boldsymbol{x}_{0,1}[k] \\ \boldsymbol{x}_{1,0}[k] \\ \boldsymbol{x}_{1,1}[k] \\ x_0[k] \end{bmatrix},$$

where the subscripts "0,0" (etc.) denote the model being blended in the same fashion as used to describe $\boldsymbol{y}$ in Eq. (5.1).

- Then, we can also write $v_{\text{cell}}[k] = g_v(\mathbf{X}_\gamma[k], u[k]) + v[k]$, and since the noise is zero-mean and additive:

$$\hat{v}_{\text{cell}}[k] = \mathbb{E}\Big[g_v(\mathbf{X}_\gamma[k], u[k]) \mid \mathbb{V}_{\text{cell}}[k-1]\Big].$$

- Since $g_v(\cdot)$ is nonlinear, we now implement EKF assumption 1.
- That is,

$$\hat{v}_{\text{cell}}[k] \approx g_v(\widehat{\mathbf{X}}_\gamma^-[k], u[k]),$$

where $\widehat{\mathbf{X}}_\gamma^-$ is a subset of $\widehat{\mathbf{X}}^-$ from Step 1a.

- This means that:
  - We use the four individual state estimates from $\widehat{\mathbf{X}}^-$ to produce four different linear-variable estimates $\hat{\boldsymbol{y}}_{0,0}$, $\hat{\boldsymbol{y}}_{0,1}$, $\hat{\boldsymbol{y}}_{1,0}$, and $\hat{\boldsymbol{y}}_{1,1}$.
  - We use output blending to combine these four estimates into a single linearized output vector $\hat{\boldsymbol{y}}$.
  - We apply nonlinear corrections to $\hat{\boldsymbol{y}}$ to compute $\hat{v}_{\text{cell}}$.

Step 2a: Estimator gain matrix.

- We have now predicted the present state and present measurement, and have computed the covariance of the state-prediction error using only prior information.
- The next steps update the prediction using present information to compute the state estimate and its uncertainty.

- A key feature is the computation of a time-varying estimator gain matrix $\boldsymbol{L}[k] = \boldsymbol{\Sigma}^{-}_{\tilde{\mathbf{X}}_\gamma \tilde{v}_{\text{cell}}}[k]\Sigma^{-1}_{\tilde{v}_{\text{cell}}}[k]$.
- We begin by computing the covariance of the voltage-prediction error, $\Sigma_{\tilde{v}_{\text{cell}}}[k]$, by employing EKF assumption 2.
- First, we compute $\widehat{\boldsymbol{C}}_{0,0}$, $\widehat{\boldsymbol{C}}_{0,1}$, $\widehat{\boldsymbol{C}}_{1,0}$, and $\widehat{\boldsymbol{C}}_{1,1}$, and $\widehat{C}_0$ (cf. App. 5.b for derivative calculations):

$$\widehat{\boldsymbol{C}}_j = \frac{\mathrm{d}g_v(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}\boldsymbol{x}_j[k]} \quad \text{and} \quad \widehat{C}_0 = \frac{\mathrm{d}g_v(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}x_0[k]},$$

  where all derivatives are evaluated at $\mathbf{X}_\gamma[k] = \widehat{\mathbf{X}}^{-}_\gamma[k]$. Then,

$$\Sigma_{\tilde{v}_j}[k] = \widehat{\boldsymbol{C}}_j \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j} \widehat{\boldsymbol{C}}^T_j + \Sigma_{\tilde{v}}, \quad \text{and} \quad \Sigma_{\tilde{v}_0}[k] = \widehat{C}_0 \Sigma_{\tilde{x}_0} \widehat{C}^T_0 + \Sigma_{\tilde{v}}.$$

- Next, we compute the cross covariance $\boldsymbol{\Sigma}^{-}_{\tilde{\mathbf{X}}_\gamma \tilde{v}_{\text{cell}}}[k]$, again using EKF assumption 2. We find,

$$\boldsymbol{\Sigma}^{-}_{\tilde{\mathbf{X}}_\gamma \tilde{v}_j}[k] = \widehat{\boldsymbol{C}}_j \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}, \quad \text{and} \quad \Sigma^{-}_{\tilde{\mathbf{X}}_\gamma \tilde{v}_0}[k] = \widehat{C}_0 \Sigma_{\tilde{x}_0}.$$

- This allows us to compute:

$$\boldsymbol{L}_j[k] = \widehat{\boldsymbol{C}}_j \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j} \left(\widehat{\boldsymbol{C}}_j \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j} \widehat{\boldsymbol{C}}^T_j + \Sigma_{\tilde{v}} \right)^{-1}$$

  for $\boldsymbol{L}_{0,0}[k]$, $\boldsymbol{L}_{0,1}[k]$, $\boldsymbol{L}_{1,0}[k]$, $\boldsymbol{L}_{1,1}[k]$, and $L_0[k]$.

Step 2b: State estimate measurement update.

- The state prediction is now updated using the measured value of voltage to become a state estimate.
- This is done only for the four models that contribute to the output-blended voltage equation.
- For these models,

$$\hat{\boldsymbol{x}}^{+}_j[k] = \hat{\boldsymbol{x}}^{-}_j[k] + \boldsymbol{L}_j[k] \left(v_{\text{cell}}[k] - \hat{v}_{\text{cell}}[k] \right)$$

$$\hat{x}^{+}_0[k] = \hat{x}^{-}_0[k] + L_0[k] \left(v_{\text{cell}}[k] - \hat{v}_{\text{cell}}[k] \right).$$

- For all other models, $\hat{\boldsymbol{x}}_j^+[k] = \hat{\boldsymbol{x}}_j^-[k]$.

Step 2c: Estimation-error covariance measurement update.

- To complete the process, the estimation-error covariance matrix is updated so all the necessary values for the next iteration are computed.
- For the four models contributing to the output-blended voltage,

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^+[k] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^-[k] - \boldsymbol{L}_j[k]\Sigma_{\tilde{v}_j}[k]\boldsymbol{L}_j^T[k]$$

$$\Sigma_{\tilde{x}_0}^+[k] = \Sigma_{\tilde{x}_0}^-[k] - L_0[k]\Sigma_{\tilde{v}_{\text{cell}}}[k]L_0^T[k].$$

- For the remaining models, $\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^+[k] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^-[k]$.

- At this point, we have updated the state estimate and its covariance (uncertainty) matrix.
- In many cases, we would also like to compute estimates of the cell's electrochemical internal variables and their uncertainties.
- In this case, we add an additional major step having two sub-steps.

Step 3a: Estimate internal variables.

- Cell internal variables $z[k]$ are nonlinear functions of the state.
- We again use EKF assumption 1 to estimate their values.
- We wish to compute:

$$\hat{\boldsymbol{z}}^+[k] = \mathbb{E}\Big[\boldsymbol{g}_z(\mathbf{X}_\gamma[k], u[k]) \mid \mathbb{V}_{\text{cell}}[k-1]\Big]$$

$$\approx \boldsymbol{g}_z(\widehat{\mathbf{X}}_\gamma^+[k], u[k]),$$

  where $\widehat{\mathbf{X}}_\gamma^+$ is a subset of $\widehat{\mathbf{X}}^+$ from Step 2b.
- This means that:

- We use the four individual state estimates from $\widehat{\mathbf{X}}^+$ to produce four different linear-variable estimates $\hat{\boldsymbol{y}}_{0,0}$, $\hat{\boldsymbol{y}}_{0,1}$, $\hat{\boldsymbol{y}}_{1,0}$, and $\hat{\boldsymbol{y}}_{1,1}$.
- We use output blending to combine these four estimates into a single linearized output vector $\hat{\boldsymbol{y}}$.
- We apply nonlinear corrections to $\hat{\boldsymbol{y}}$ to compute $\hat{\boldsymbol{z}}$.

Step 3b: Internal-variables estimation-error covariance.

- Finally, we compute the covariance of the internal variable (useful for computing confidence intervals of the estimate).
- We begin by computing the covariance of the voltage-prediction error, $\Sigma_{\tilde{v}_{\text{cell}}}[k]$, by employing EKF assumption 2.
- First, we compute $\widehat{\boldsymbol{C}}_{0,0}$, $\widehat{\boldsymbol{C}}_{0,1}$, $\widehat{\boldsymbol{C}}_{1,0}$, and $\widehat{\boldsymbol{C}}_{1,1}$, and $\widehat{\boldsymbol{C}}_0$ (cf. App. 5.b for derivative calculations):

$$\widehat{\boldsymbol{C}}_j = \frac{\mathrm{d}\boldsymbol{g}_z(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}\boldsymbol{x}_j[k]} \quad \text{and} \quad \widehat{\boldsymbol{C}}_0 = \frac{\mathrm{d}\boldsymbol{g}_z(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}x_0[k]},$$

  where all derivatives are evaluated at $\mathbf{X}_\gamma[k] = \widehat{\mathbf{X}}^+_\gamma[k]$.
- Then,

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}_j}[k] = \widehat{\boldsymbol{C}}_j \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j} \widehat{\boldsymbol{C}}_j^T, \quad \text{and} \quad \boldsymbol{\Sigma}_{\tilde{z}_0}[k] = \widehat{\boldsymbol{C}}_0 \Sigma_{\tilde{x}_0} \widehat{\boldsymbol{C}}_0^T.$$

- Finally,

$$\boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{z}}}[k] = \boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}_{0,0}}[k] + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}_{0,1}}[k] + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}_{1,0}}[k] + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}_{1,1}}[k] + \boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}_0}[k].$$

- At the output of this process, we have high confidence that the true variable is in the range $\hat{\boldsymbol{z}}^+[k] \pm 3\sqrt{\mathrm{diag}\left(\boldsymbol{\Sigma}^+_{\tilde{\boldsymbol{z}}}[k]\right)}$.

5.7: SPKF with the output-blended model

- Application of SPKF to the output-blended ROM is now described.
- Some computational simplifications result from the state equation being linear for the ROM we are using.
- In fact, SPKF Steps 1a and 1b are identical to EKF Steps 1a and 1b.

Step 1a: State prediction time update.

- The state prediction we wish to compute is (for *all* models):

$$\widehat{\mathbf{X}}^{-}[k] = \mathbf{A}\widehat{\mathbf{X}}^{+}[k-1] + \mathbf{B}u[k-1].$$

- This can also be implemented on a per-model basis, $1 \leq j \leq N$:

$$\hat{\boldsymbol{x}}_j^{-}[k] = \boldsymbol{A}_j \hat{\boldsymbol{x}}_j^{+}[k-1] + \boldsymbol{B}_j u[k-1]$$

$$\hat{x}_0^{-}[k] = \hat{x}_0^{+}[k-1] + u[k-1].$$

Step 1b: Prediction-error covariance time update.

- The covariance (uncertainty) of the state prediction is computed as:

$$\boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}}^{-}[k] = \mathbf{A}\boldsymbol{\Sigma}_{\widetilde{\mathbf{X}}}^{+}[k-1]\mathbf{A}^T + \mathbf{B}\Sigma_{\widetilde{w}}\mathbf{B}^T.$$

  This can also be implemented on a per-model basis, $1 \leq j \leq N$:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^{-}[k] = \boldsymbol{A}_j \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^{+}[k-1]\boldsymbol{A}_j^T + \boldsymbol{B}_j \Sigma_{\widetilde{w}} \boldsymbol{B}_j^T$$

$$\Sigma_{\tilde{x}_0}^{-}[k] = \Sigma_{\tilde{x}_0}^{+}[k-1] + \Sigma_{\widetilde{w}}.$$

Step 1c: Predict system output.

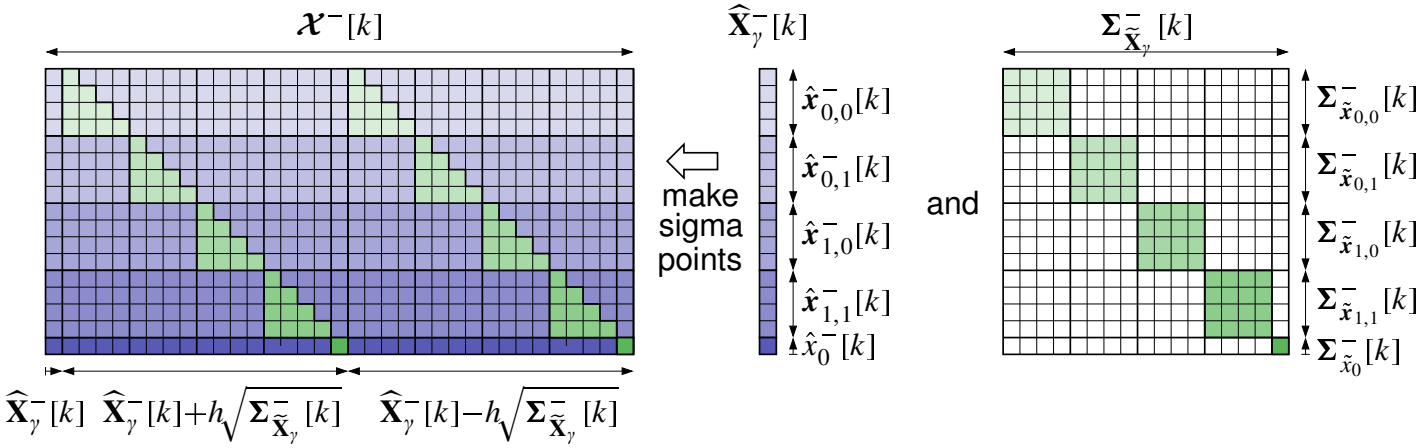
- We adopt the voltage equation Eq. (5.6) from the EKF section, so we can write:

$$\hat{v}_{\text{cell}}[k] = \mathbb{E}\Big[g_v(\mathbf{X}_\gamma[k], u[k]) \mid \mathbb{V}_{\text{cell}}[k-1]\Big].$$

- Since $g_v(\cdot)$ is nonlinear, we now turn to the sigma-point approach to approximate the mean and uncertainty of a random variable computed using a nonlinear equation.
- The mean and error-covariance of the state prediction are used to form a set of sigma points based on the $\widehat{\mathbf{X}}_\gamma^-$ subset of $\widehat{\mathbf{X}}^-$ (and corresponding subset $\mathbf{\Sigma}_{\widetilde{\mathbf{X}}_\gamma}^-$ of $\mathbf{\Sigma}_{\widetilde{\mathbf{X}}}^-$):

$$\boldsymbol{\mathcal{X}}^-[k] = \left\{ \widehat{\mathbf{X}}_\gamma^-[k],\ \widehat{\mathbf{X}}_\gamma^-[k] + h\sqrt{\mathbf{\Sigma}_{\widetilde{\mathbf{X}}_\gamma}^-[k]},\ \widehat{\mathbf{X}}_\gamma^-[k] - h\sqrt{\mathbf{\Sigma}_{\widetilde{\mathbf{X}}_\gamma}^-[k]} \right\}. \quad (5.7)$$

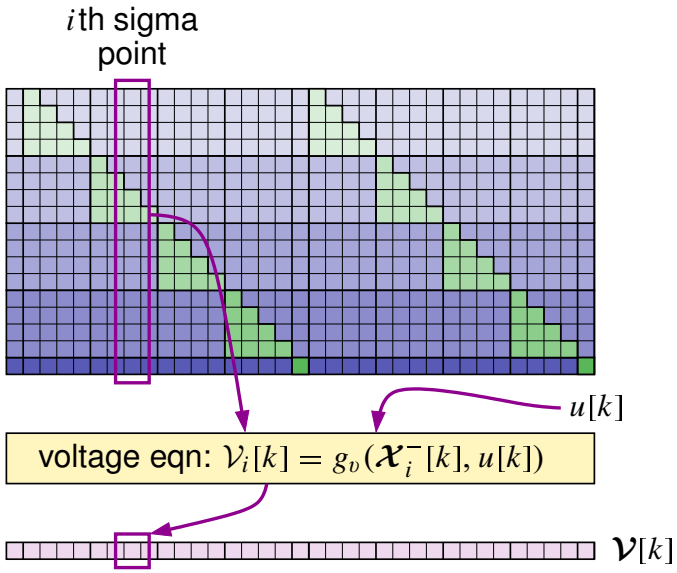
- Here, h is a SPKF tuning variable (often $h = \sqrt{3}$),
  - $\sqrt{(\cdot)}$ is the lower-triangular matrix square root of its argument (usually computed using a Cholesky decomposition),
  - The atypical notation of Eq. (5.7) indicates that the zeroth member of set $\boldsymbol{\mathcal{X}}^-$ is the vector $\widehat{\mathbf{X}}_\gamma^-[k]$, the next member is $\widehat{\mathbf{X}}_\gamma^-[k]$ plus h times the first column of $\sqrt{\mathbf{\Sigma}_{\widetilde{\mathbf{X}}_\gamma}^-[k]}$, the next member is $\widehat{\mathbf{X}}_\gamma^-[k]$ plus h times the second column of $\sqrt{\mathbf{\Sigma}_{\widetilde{\mathbf{X}}_\gamma}^-[k]}$, and so forth.[4]
- Can be organized in convenient matrix form:



[4] The total number of elements in $\boldsymbol{\mathcal{X}}^-$ is $1 + 2(4n+1) = 8n+3$ elements, indexed from 0 to $8n+2$.

- For each element $\boldsymbol{\mathcal{X}}_i^-[k]$ in $\boldsymbol{\mathcal{X}}^-[k]$, we compute linear-model output via Eq. (5.4).
- Then, apply nonlinear corrections to produce voltage prediction based on that element.
- That is, overall we find nonlinear-output sigma points:

$$\mathcal{V}_i[k] = g_v(\boldsymbol{\mathcal{X}}_i^-[k], u[k]).$$



- The weighted mean of these points is the voltage prediction we are seeking to compute:

$$\hat{v}_{\text{cell}}[k] = \sum_{i=0}^{8n+2} \alpha_i^{(\text{m})} \mathcal{V}_i[k],$$

where $\alpha_i^{(\text{m})}$ are SPKF tuning variables when computing a mean.

- This can be computed as an inner product between: 1) a vector $\boldsymbol{\mathcal{V}}[k]$ that stores all component values $\mathcal{V}_i$, and 2) a vector storing the tuning variables $\alpha_i^{(\text{m})}$.

Step 2a: Estimator gain matrix.

- We have now predicted the present state and present measurement, and have computed the covariance of the state-prediction error using only prior information.
- The next steps update the prediction using present information to compute the state estimate and its uncertainty.
- A key feature is the computation of a time-varying estimator gain matrix $\boldsymbol{L}[k] = \boldsymbol{\Sigma}_{\tilde{\mathbf{x}}_\gamma \tilde{v}_{\text{cell}}}^-[k] \Sigma_{\tilde{v}_{\text{cell}}}^{-1}[k]$.

- We begin by computing the covariance of the voltage-prediction error:

$$\Sigma_{\tilde{v}_{\text{cell}}}[k] = \sum_{i=0}^{8n+2} \alpha_i^{(\text{c})} \left(\hat{v}_{\text{cell}}[k] - \mathcal{V}_i[k]\right)\left(\hat{v}_{\text{cell}}[k] - \mathcal{V}_i[k]\right) + \Sigma_{\tilde{v}},$$

where $\alpha_i^{(\text{c})}$ are tuning variables for the SPKF when computing covariances.[5]

- This can be computed efficiently if we define the vector $\widetilde{\boldsymbol{\mathcal{V}}}[k]$ having elements $\widetilde{\mathcal{V}}_i[k] = \hat{v}_{\text{cell}}[k] - \mathcal{V}_i[k]$. Then:

$$\Sigma_{\tilde{v}_{\text{cell}}}[k] = \widetilde{\boldsymbol{\mathcal{V}}}^T[k]\,\text{diag}(\alpha_i^{(\text{c})})\,\widetilde{\boldsymbol{\mathcal{V}}}[k] + \Sigma_{\tilde{v}}.$$

- We also compute the cross covariance between the state-prediction error and the voltage-prediction error:

$$\boldsymbol{\Sigma}^{-}_{\widetilde{\mathbf{X}}_\gamma\,\tilde{v}_{\text{cell}}}[k] = \sum_{i=0}^{8n+2} \alpha_i^{(\text{c})} \left(\widehat{\mathbf{X}}_\gamma^{-}[k] - \boldsymbol{\mathcal{X}}_i^{-}[k]\right)\left(\hat{v}_{\text{cell}}[k] - \mathcal{V}_i[k]\right).$$

- If we further define the matrix $\widetilde{\boldsymbol{\mathcal{X}}}[k]$ having columns $\widetilde{\boldsymbol{\mathcal{X}}}_i[k] = \widehat{\mathbf{X}}_\gamma^{-}[k] - \boldsymbol{\mathcal{X}}_i^{-}[k]$, then this can be computed as:

$$\boldsymbol{\Sigma}^{-}_{\widetilde{\mathbf{X}}_\gamma\,\tilde{v}_{\text{cell}}}[k] = \widetilde{\boldsymbol{\mathcal{X}}}[k]\,\text{diag}(\alpha_i^{(\text{c})})\,\widetilde{\boldsymbol{\mathcal{V}}}[k].$$

- After these are computed, the state-estimator gain matrix can be found as:

$$\boldsymbol{L}[k] = \boldsymbol{\Sigma}^{-}_{\widetilde{\mathbf{X}}_\gamma\,\tilde{v}_{\text{cell}}}[k]\left(\Sigma_{\tilde{v}_{\text{cell}}}[k]\right)^{-1}.$$

Step 2b: State estimate measurement update.

- The state prediction is now updated using the measured value of voltage to become a state estimate.

[5] Note that $\Sigma_{\tilde{v}_{\text{cell}}}[k]$ on the left-hand-side is the time-varying innovation covariance and $\Sigma_{\tilde{v}}$ on the right-hand-side is the (often time-invariant) sensor-noise covariance.

- This is done only for the four models that contribute to the output-blended voltage equation.
- Denote this set of four models as $\mathcal{S}$. Then,

$$\hat{\boldsymbol{x}}_j^+[k] = \begin{cases} \hat{\boldsymbol{x}}_j^-[k] + \boldsymbol{L}_j[k]\left(v_{\text{cell}}[k] - \hat{v}_{\text{cell}}[k]\right), & j \in \mathcal{S} \\ \hat{\boldsymbol{x}}_j^-[k], & j \notin \mathcal{S} \end{cases} \tag{5.8}$$

$$\hat{x}_0^+[k] = \hat{x}_0^-[k] + L_0[k]\left(v_{\text{cell}}[k] - \hat{v}_{\text{cell}}[k]\right). \tag{5.9}$$

where $\boldsymbol{L}_j[k]$ is the portion of $\boldsymbol{L}[k]$ corresponding to model j and $L_0[k]$ is the portion of $\boldsymbol{L}[k]$ corresponding to the integration state.

Step 2c: Estimation-error covariance measurement update.

- To complete the process, the estimation-error covariance matrix is updated so all the necessary values for the next iteration are computed:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^+[k] = \begin{cases} \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^-[k] - \boldsymbol{L}_j[k]\Sigma_{\tilde{v}_{\text{cell}}}[k]\boldsymbol{L}_j^T[k], & j \in \mathcal{S} \\ \boldsymbol{\Sigma}_{\tilde{\boldsymbol{x}}_j}^-[k], & j \notin \mathcal{S} \end{cases} \tag{5.10}$$

$$\Sigma_{\tilde{x}_0}^+[k] = \Sigma_{\tilde{x}_0}^-[k] - L_0[k]\Sigma_{\tilde{v}_{\text{cell}}}[k]L_0^T[k]. \tag{5.11}$$

- At this point, we have updated the state estimate and its covariance.
- In many cases, we would also like to compute estimates of the cell's electrochemical internal variables and their uncertainties.
- In this case, we add an additional major step having two sub-steps

Step 3a: Estimate internal variables.

- The cell's internal variables $\boldsymbol{z}[k]$ are nonlinear functions of the state, so once again, we must use the sigma-point method to find their estimates and uncertainties.

- To estimate the internal variables, we will compute new sigma points based on the present state estimate and uncertainty:

$$\boldsymbol{\mathcal{X}}^{+}[k] = \left\{ \widehat{\mathbf{X}}_{\gamma}^{+}[k], \widehat{\mathbf{X}}_{\gamma}^{+}[k] + h\sqrt{\boldsymbol{\Sigma}_{\tilde{\mathbf{X}}_{\gamma}}^{+}[k]}, \widehat{\mathbf{X}}_{\gamma}^{+}[k] - h\sqrt{\boldsymbol{\Sigma}_{\tilde{\mathbf{X}}_{\gamma}}^{+}[k]} \right\}. \quad (5.12)$$

- Each of these sigma points comprises only those states that participate in computing the cell voltage: $\boldsymbol{\mathcal{X}}_i^{+}[k]$ has dimension $4n+1$.
- Then, for each of the sigma points in the set $\boldsymbol{\mathcal{X}}_i^{+}$, we produce an output sigma point:

$$\boldsymbol{\mathcal{Z}}_i[k] = g_z(\boldsymbol{\mathcal{X}}_i^{+}[k], u[k]).$$

  Note that $\boldsymbol{\mathcal{Z}}_i[k]$ has the same size and organization as the ROM linear-output vector $\boldsymbol{y}[k]$; output blending and all nonlinear corrections are applied to every element.
- Finally, we compute the estimate of the variable of interest as:

$$\hat{\boldsymbol{z}}^{+}[k] = \sum_{i=0}^{8n+2} \alpha_i^{(\mathrm{m})} \boldsymbol{\mathcal{Z}}_i[k].$$

Step 3b: Internal-variables estimation-error covariance.

- Finally, we compute the covariance of the internal variable (useful for computing confidence intervals of the estimate) as:

$$\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{+}[k] = \sum_{i=0}^{8n+2} \alpha_i^{(\mathrm{c})} (\hat{\boldsymbol{z}}^{+}[k] - \boldsymbol{\mathcal{Z}}_i^{+}[k])(\hat{\boldsymbol{z}}^{+}[k] - \boldsymbol{\mathcal{Z}}_i^{+}[k])^T.$$

- At the output of this process, we have high confidence that the true variable is in the range $\hat{\boldsymbol{z}}^{+}[k] \pm 3\sqrt{\mathrm{diag}\left(\boldsymbol{\Sigma}_{\tilde{\boldsymbol{z}}}^{+}[k]\right)}$.

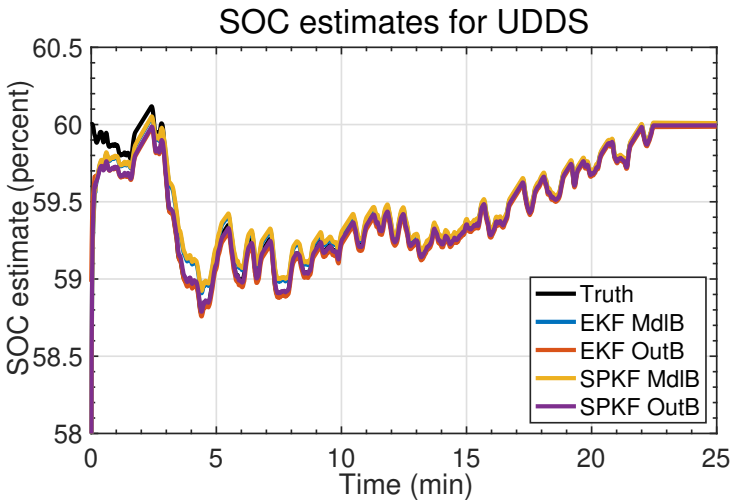
COMMENT: Notice that during operation of either xKF, only four pre-computed models are used to predict cell voltage at any point in time.

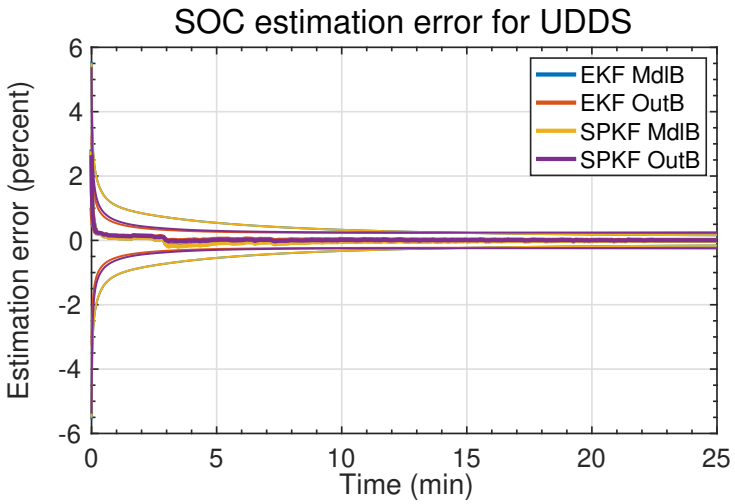
  - So only those same four models are updated using the measured-voltage feedback in Steps 2b and 2c.
  - However, all models are updated in Steps 1a and 1b.

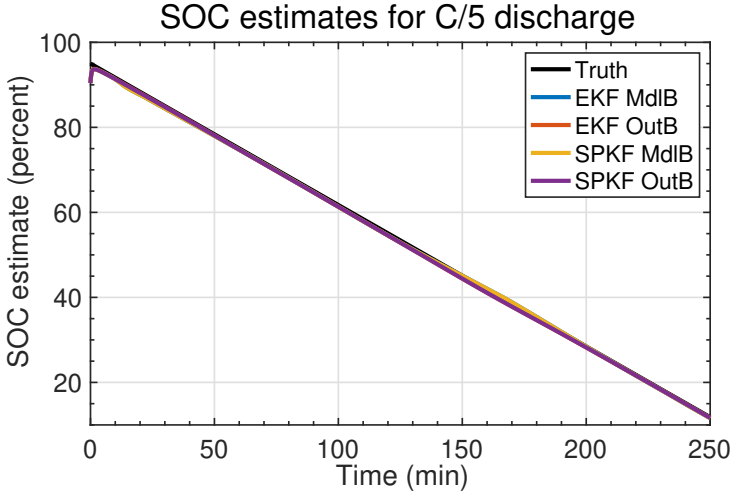
- Since Step 1b has the effect of increasing the uncertainty of model states and Step 2c has the effect of decreasing the uncertainty, many models will have their uncertainty increase over time without having measurements that will then decrease their uncertainty.
- As the actual cell being monitored has a time-varying temperature and SOC, we expect that different sets of four precomputed models will contribute to the voltage prediction at different points in time.
- Any time that this set of models changes, we will begin to blend in the effect of a precomputed model that possibly has a large covariance (large uncertainty of its states) since that model has not been updated using Steps 2b and 2c for some time.
- That is, we might expect a singularity when either SOC and/or temperature change such that we blend together a different set of four precomputed models from the set we have been using most recently.
- We do see this singularity as a kind of "scalloping" in the error bounds of estimates, but the degree of this effect is low since both SOC and temperature change relatively slowly.
- When we start to blend in a new model, the γ value associated with that model is near zero, and so the large uncertainty of that model does not have significant impact on the voltage prediction.
- Might remove scalloping if we update every model in Steps 2b and 2c using $\boldsymbol{y}[k]$ from the four updated models as the "truth/measurement."

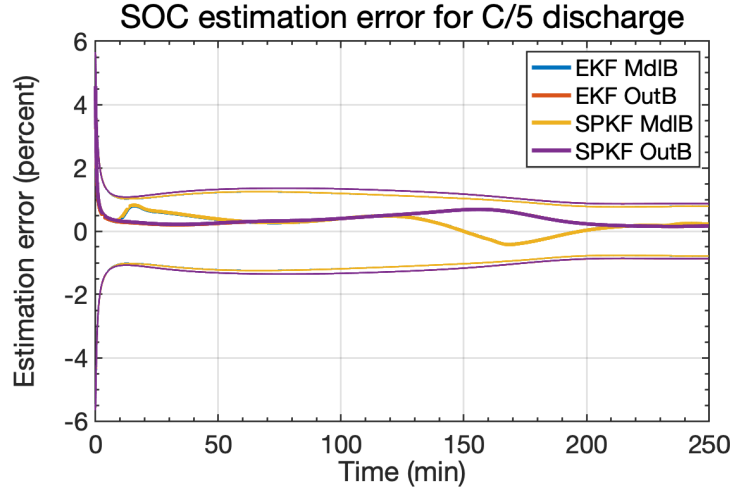
5.8: Example of xKF code in operation

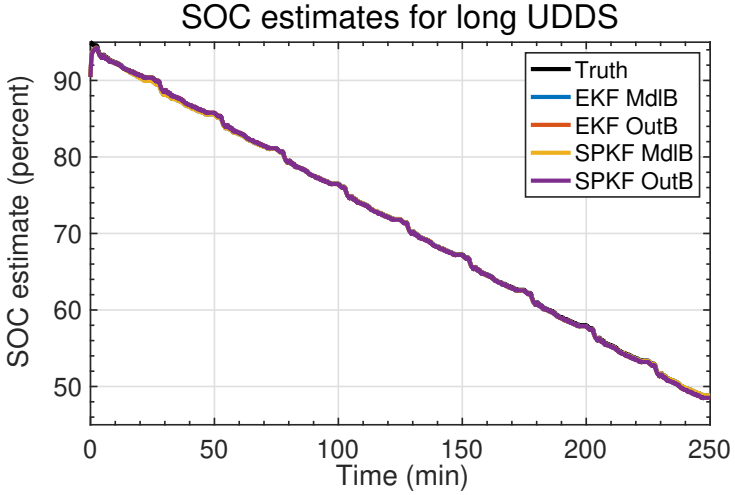
- I ran xKF code on the three load profiles from Ch. 4.
- In each case, the xKF initial SOC was set to 95 % of its true value to demonstrate the xKF's ability to recover from a poor initial estimate via its built-in feedback mechanism.

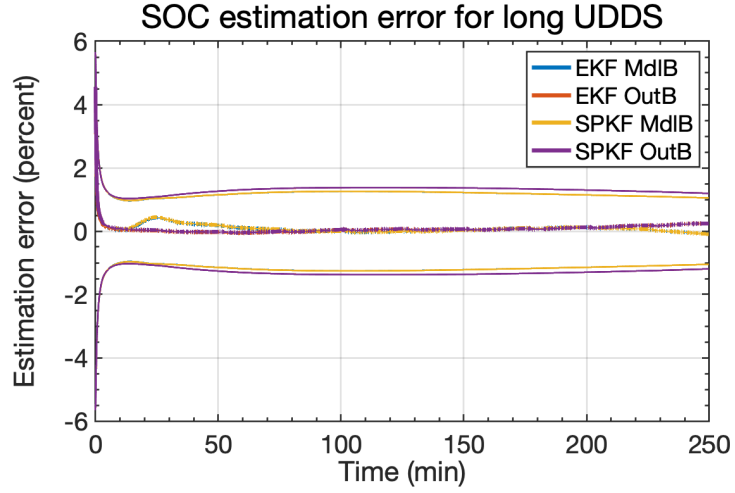






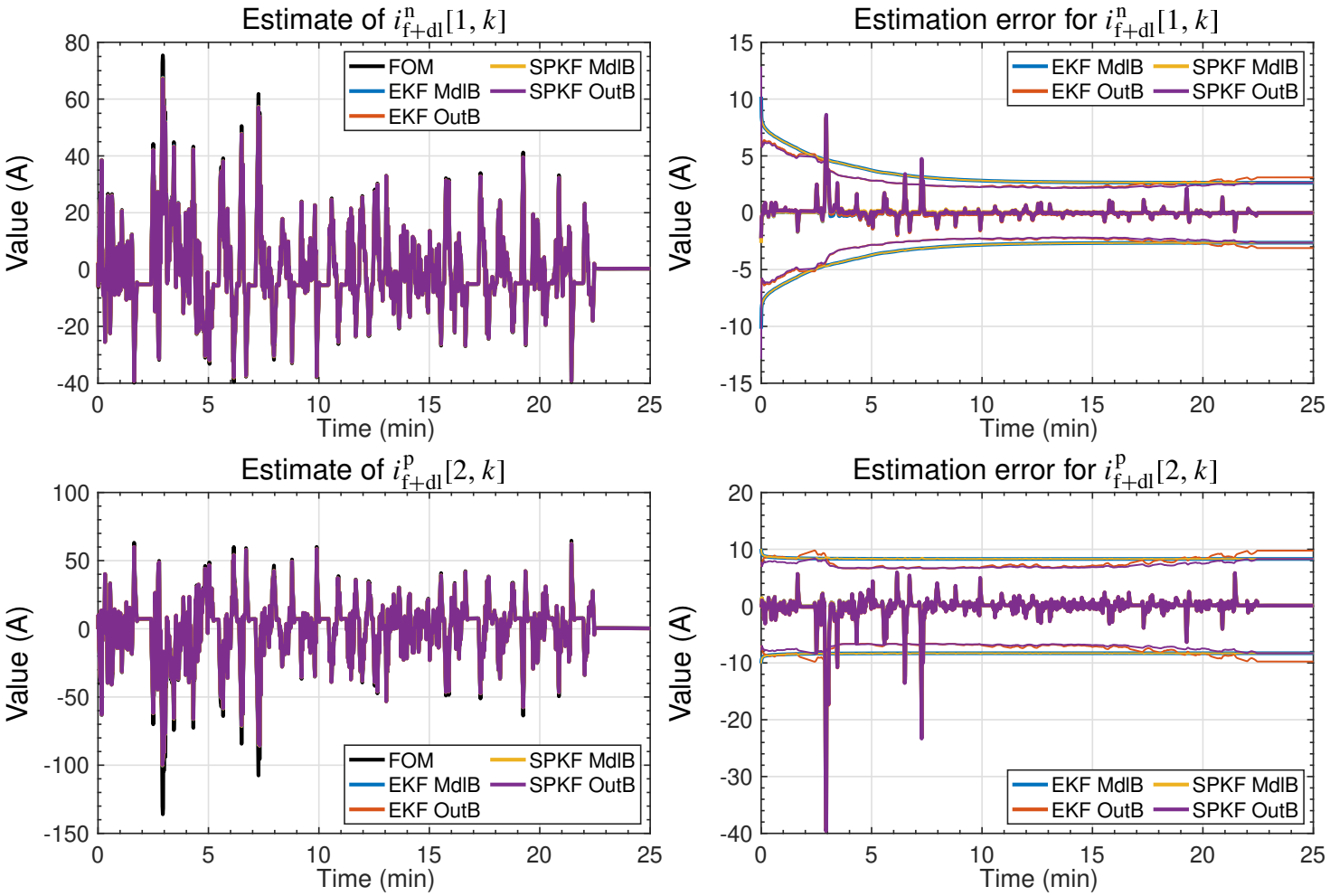






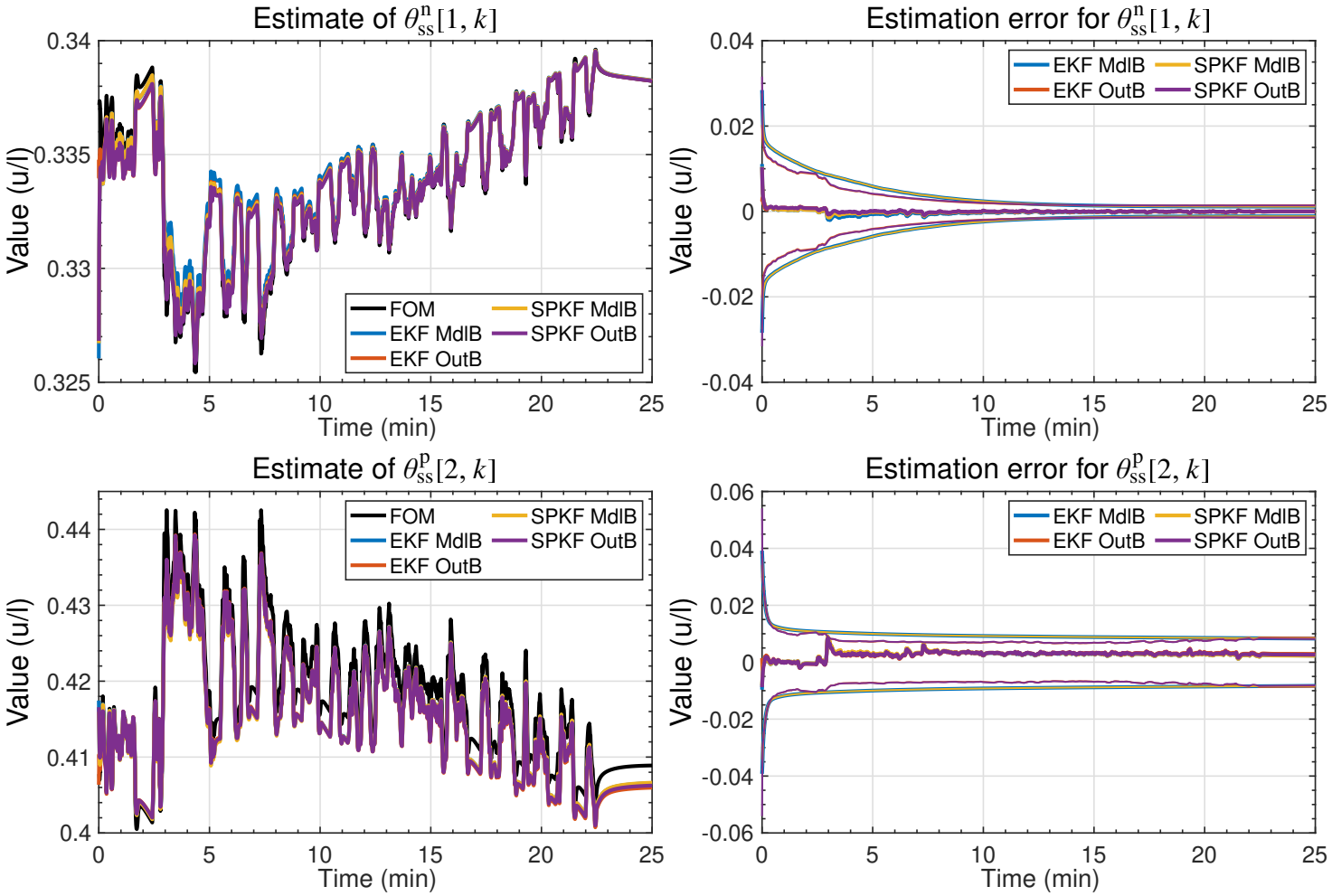
- All xKFs worked very well for estimating SOC (RMSE $< 1\ \%$).[6]
  - Results from model-blended EKF and SPKF were nearly identical;
  - Results from output-blended EKF and SPKF were nearly identical;
  - Output-blended xKF very slightly better than model-blended xKF.
- Thin lines on error plots show the $\pm 3\sigma$ confidence bounds:
  - SOC-estimation error is always within xKF bounds; meaning,
  - True SOC also always within the xKF confidence bounds.

UDDS internal-variable estimation results



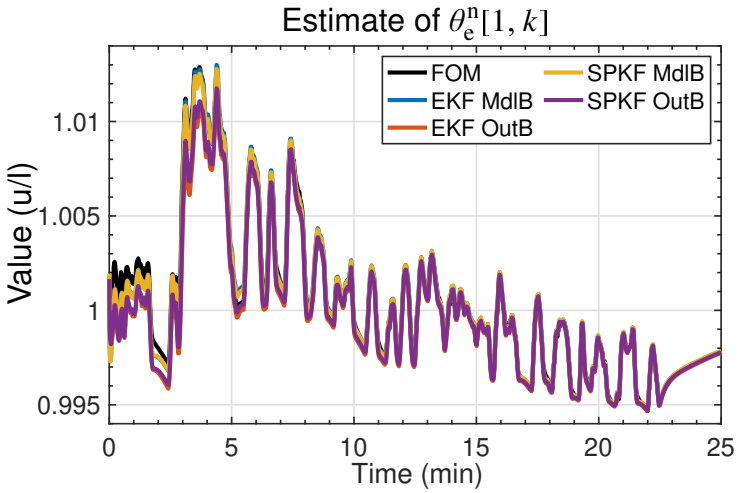
[6] You can reproduce the results by running the example code in the toolbox; it implements EKF and SPKF for both model-blending and output-blending assumptions.

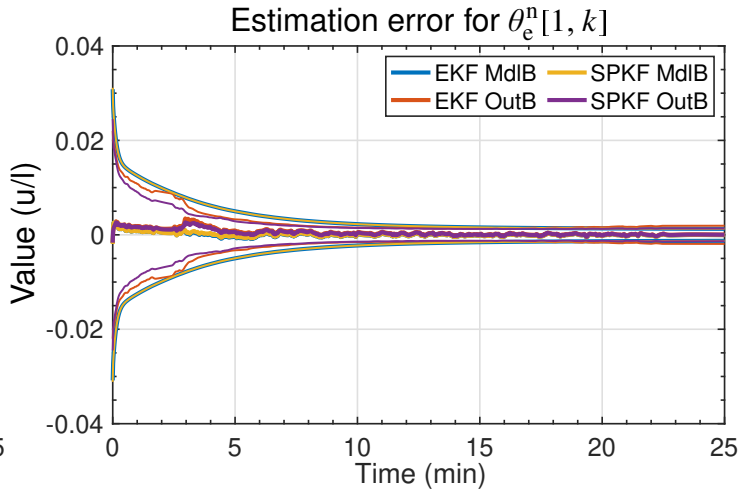
- Showing results for electrode/separator boundaries since variables at those locations are most difficult to estimate.
- Estimates of $i^{\mathrm{n}}_{\mathrm{f+dl}}[\tilde{x}, k]$ are generally good; confidence bounds are not always as reliable as for SOC.
- Next, we look at estimates of solid surface concentrations.

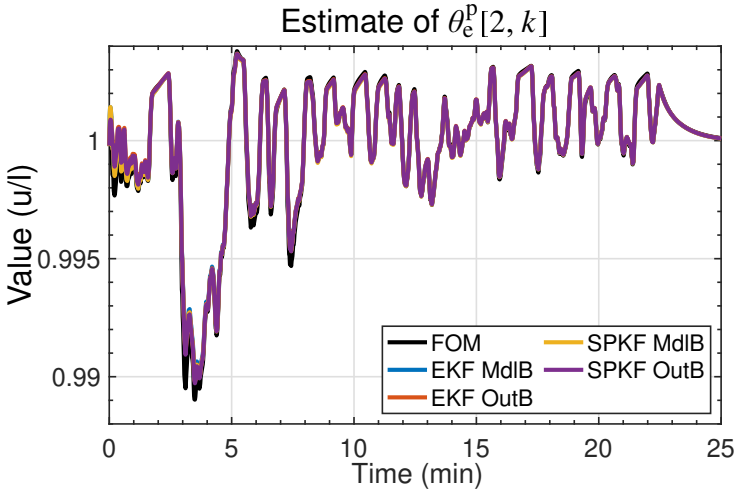


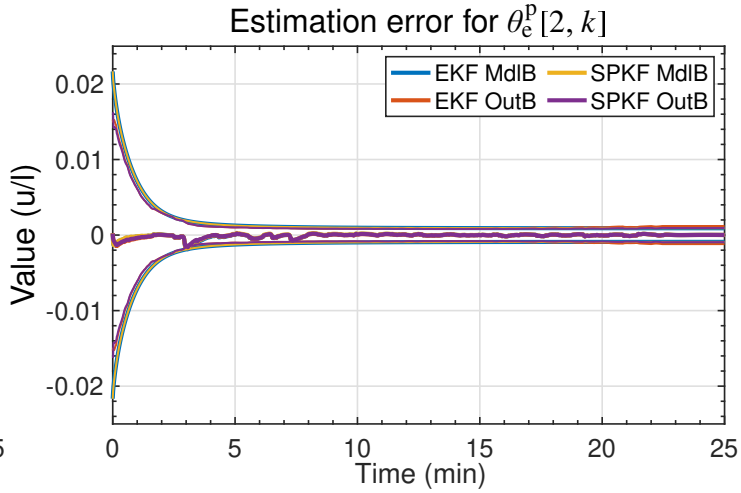
- Estimates are reasonable; confidence bounds are okay.
- We will see, in general, that closed-loop xKF estimates of internal variables are similar to the open-loop predictions.
- The exception is when the xKF is initialized with a bad SOC estimate; then, the xKF recovers quickly from the bad initial estimate. . . to the trajectory that aligns with the open-loop predictions.

- Next, the electrolyte concentration ratios.



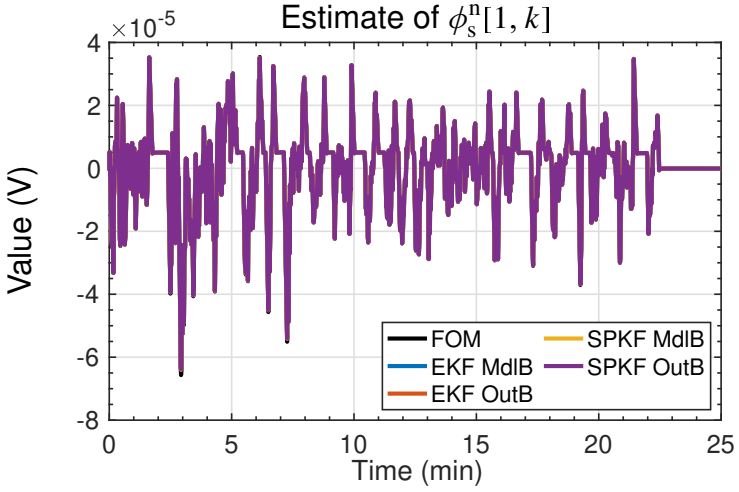


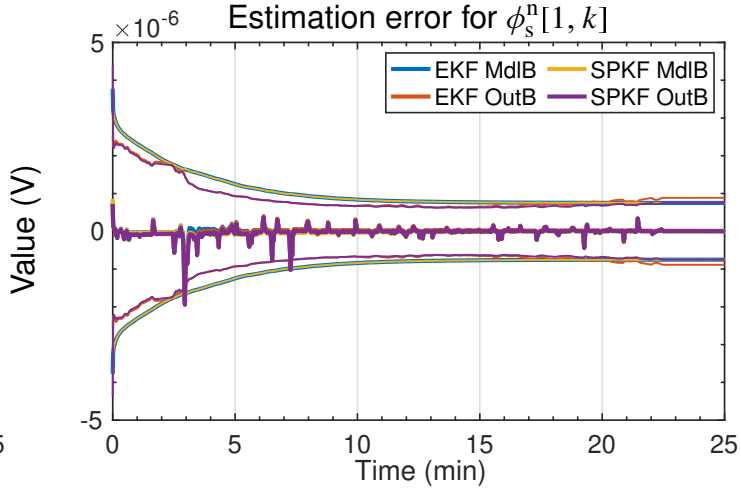


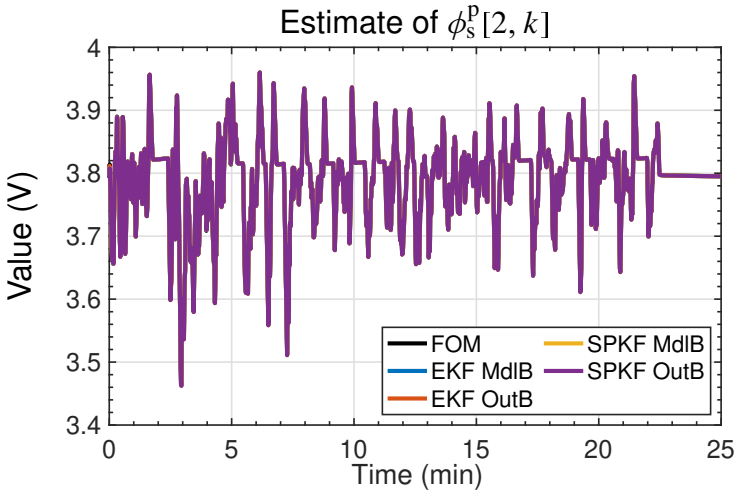


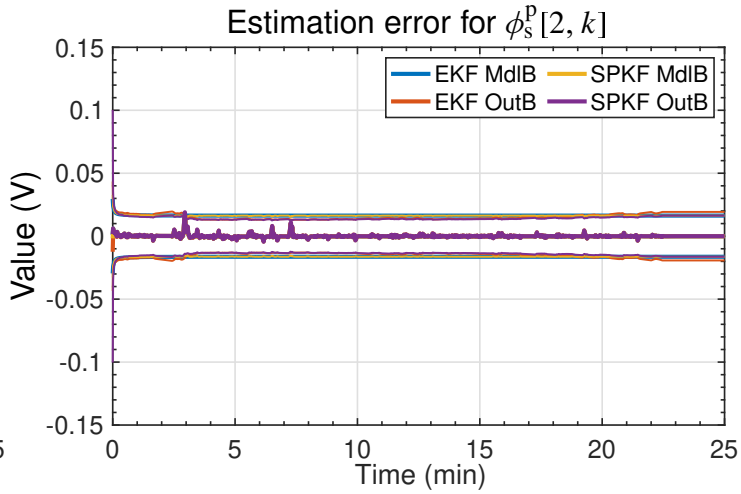
- Once again, estimates are reasonable; SPKF has tighter confidence than EKF.

- Next, the solid potentials.

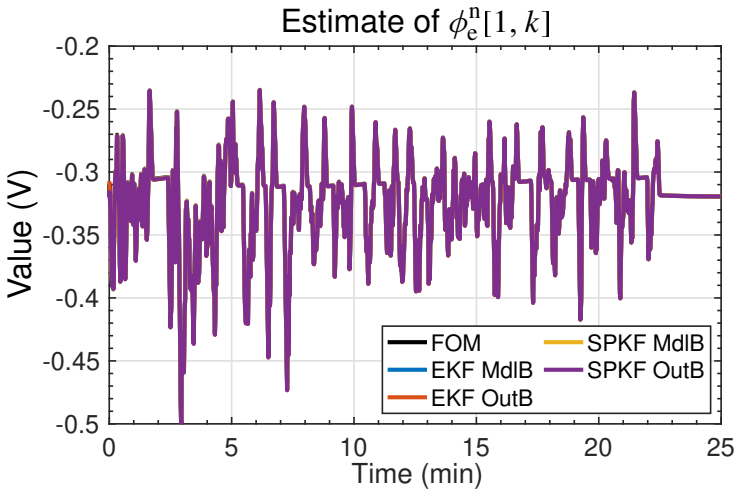


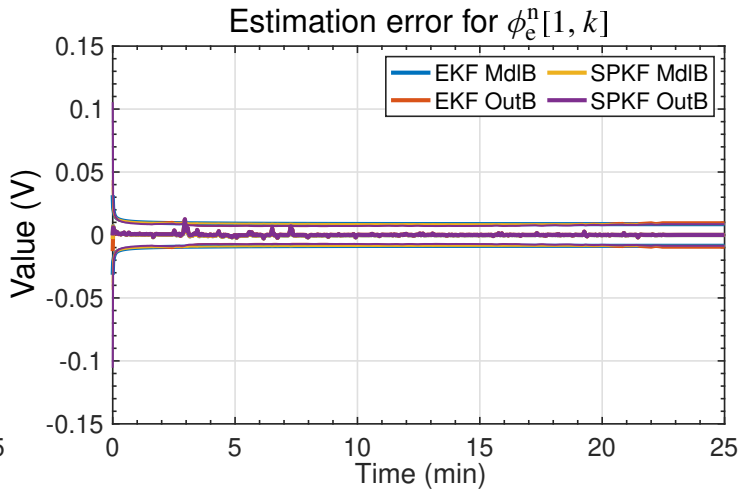


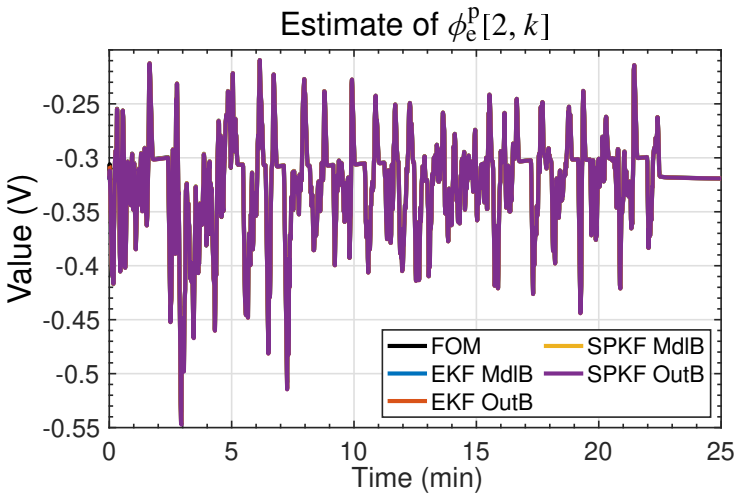


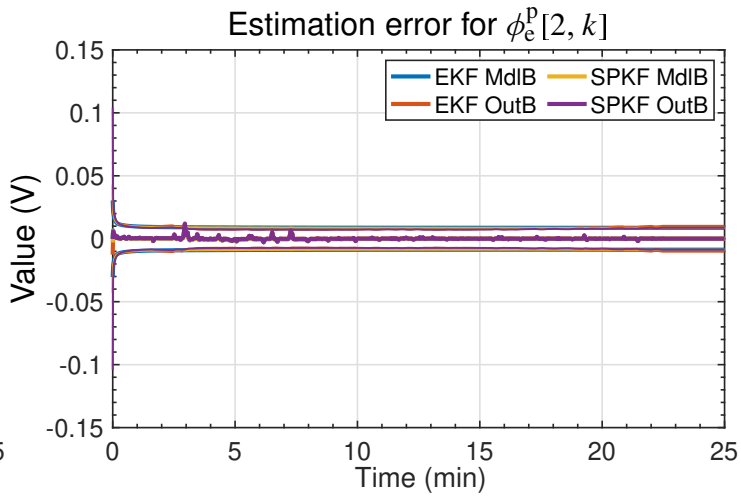


- Electrolyte potential:

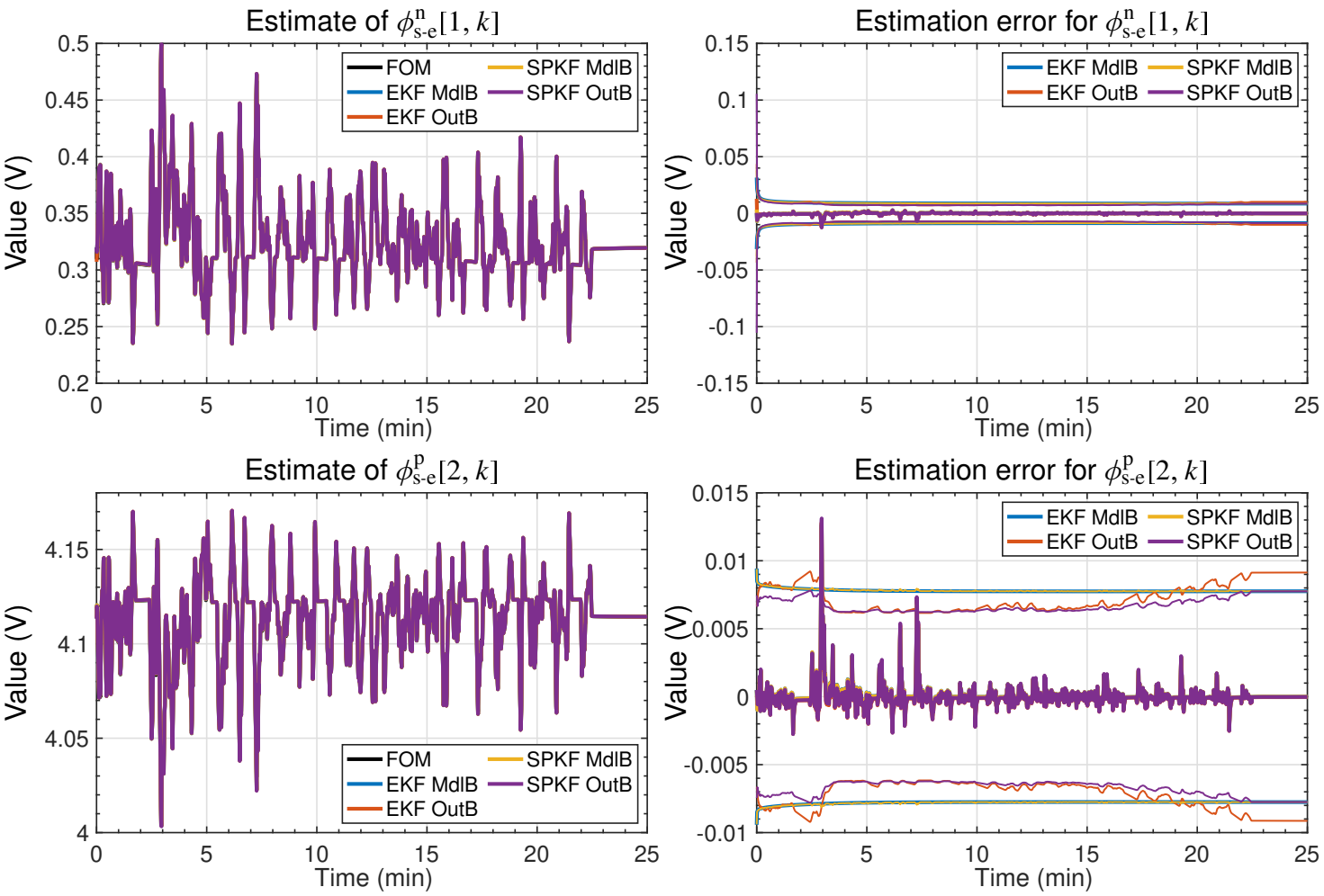






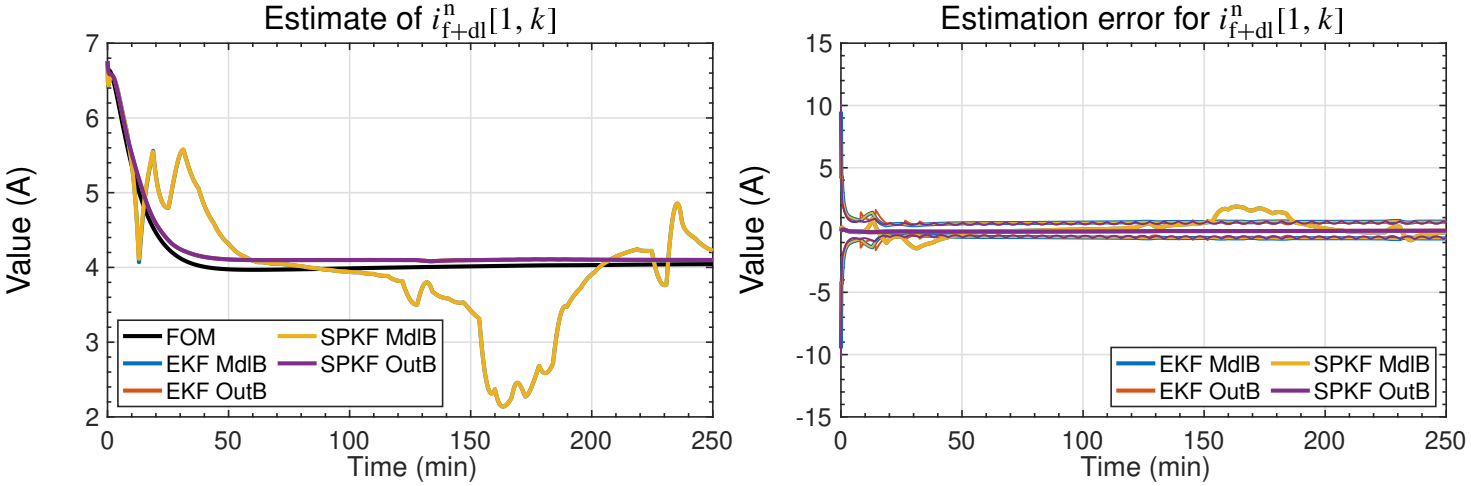


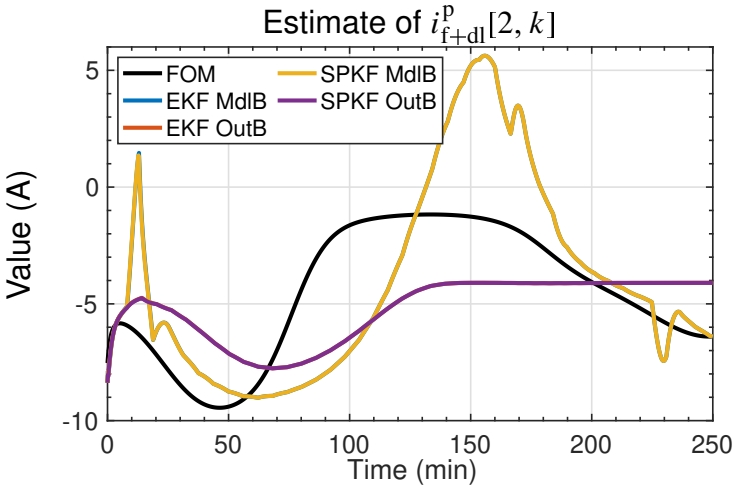
- Finally, interphase potential difference:

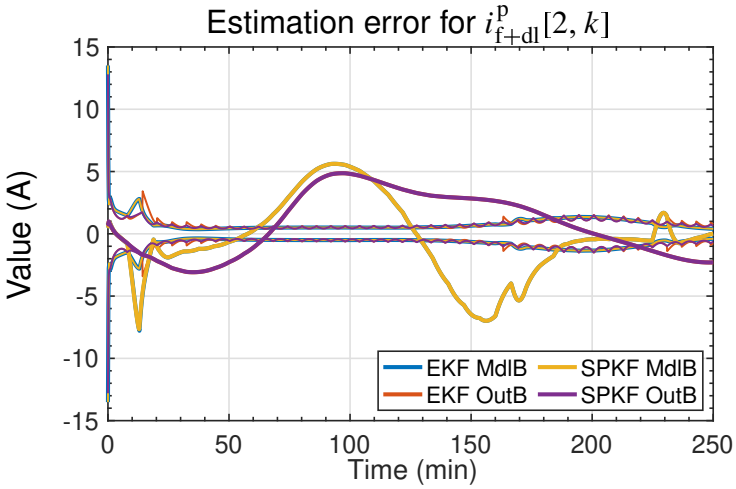


C/5 discharge profile internal-variable estimation results

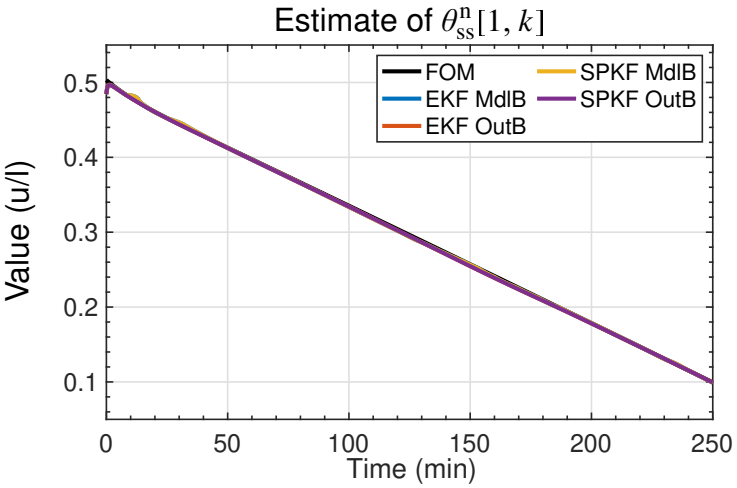
- Next, we look at corresponding results for the C/5 discharge profile.

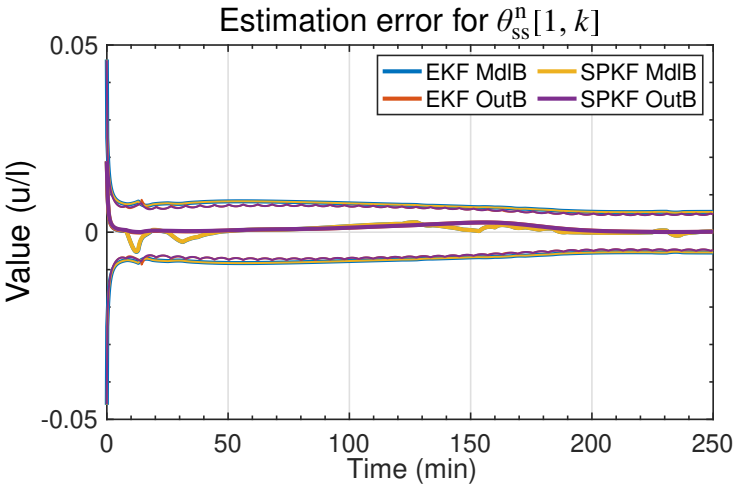


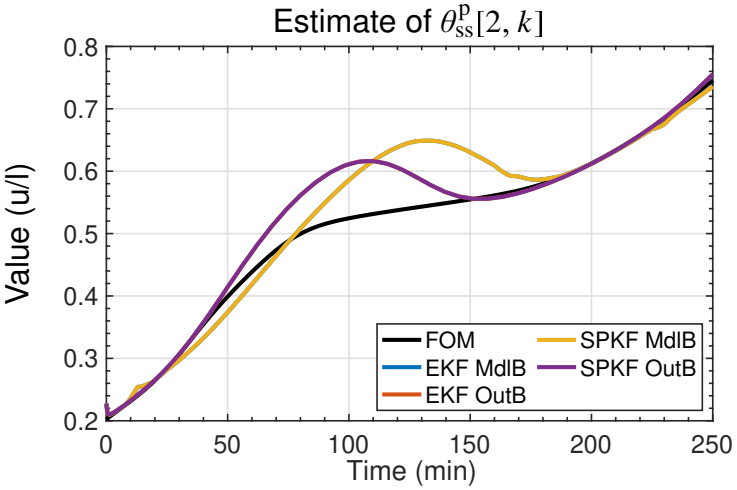


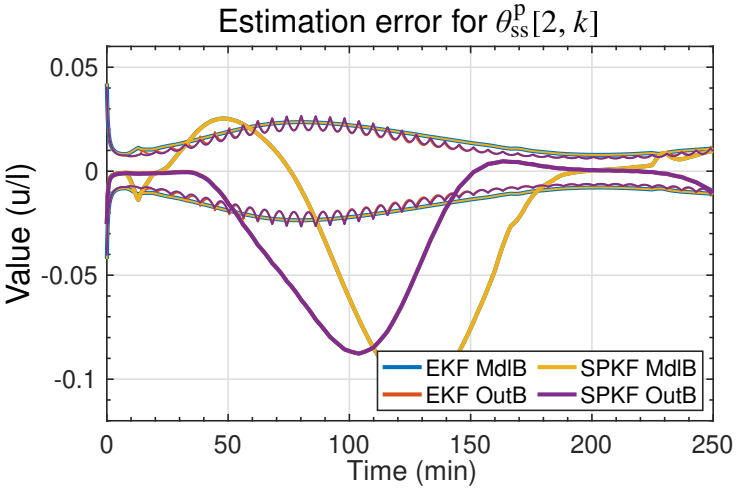


- Output blending clearly better than model blending in negative electrode. Neither is excellent in positive electrode.
- Closed-loop estimates turn out to be close to open-loop estimates.
- Next, we look at estimates of solid surface concentrations.

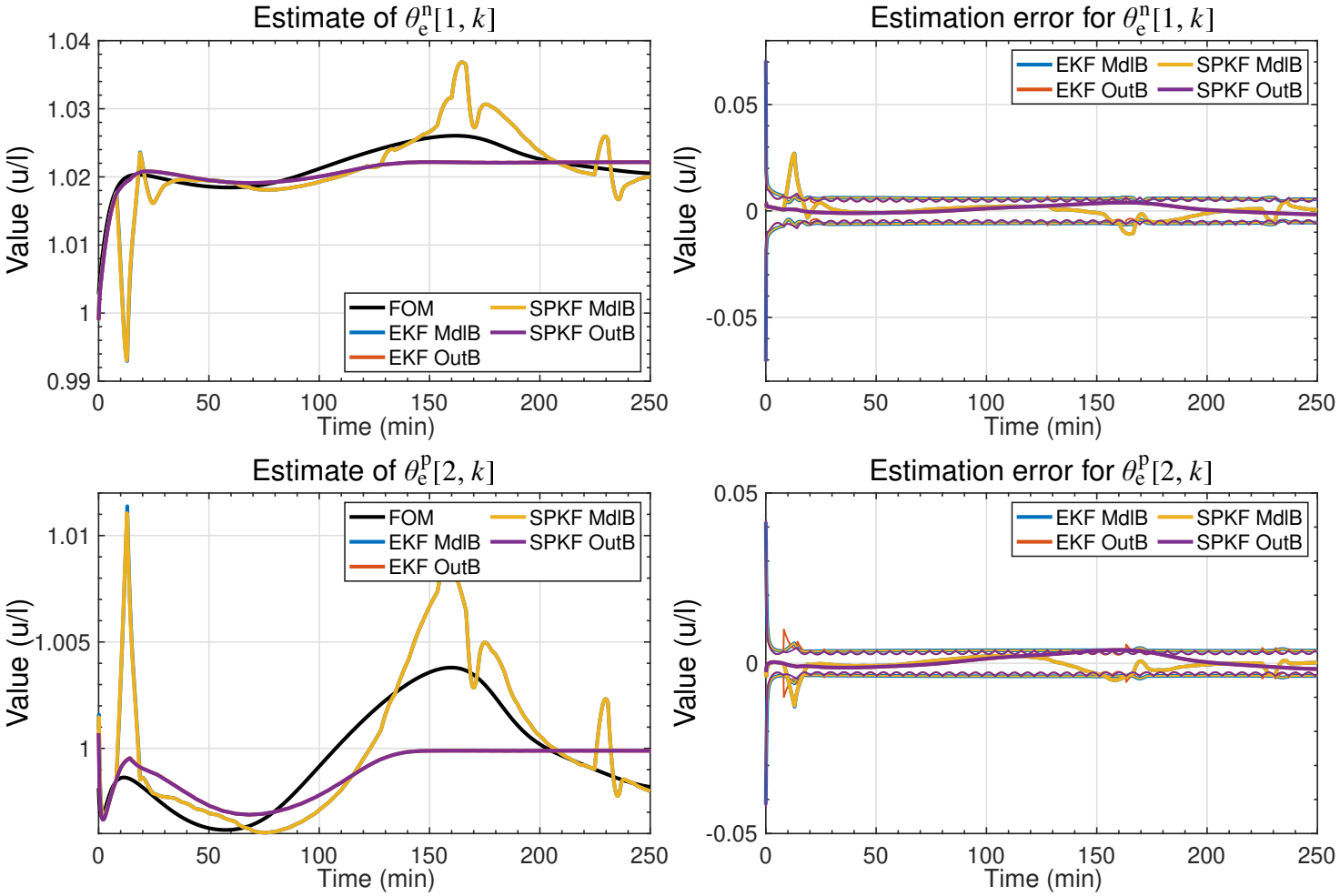






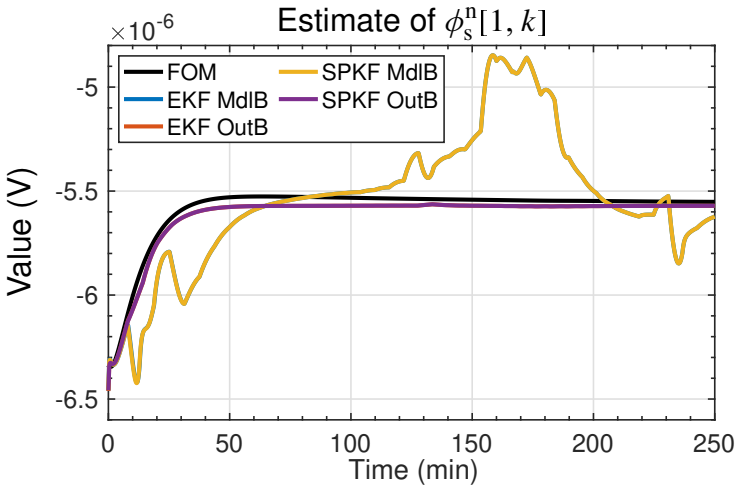


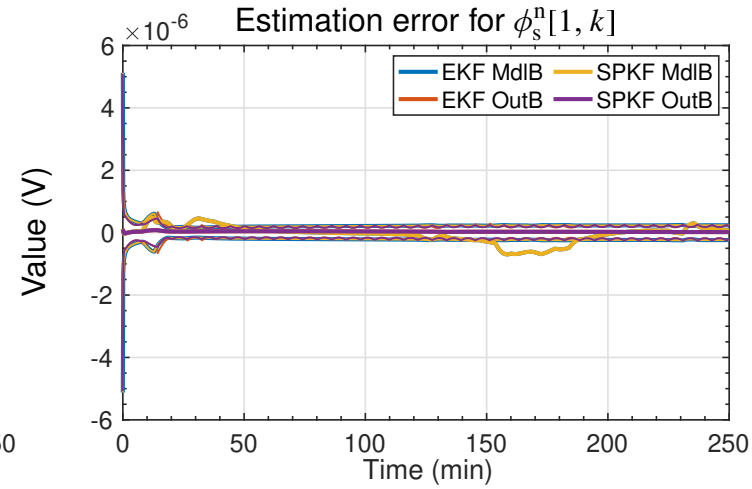
- Notice “scalloping” of confidence bounds for output-blending.
  - Caused by blending into new models whose covariances are still relatively high, but which get adapted downward as measurements are made.
- Next, the electrolyte concentration ratios.

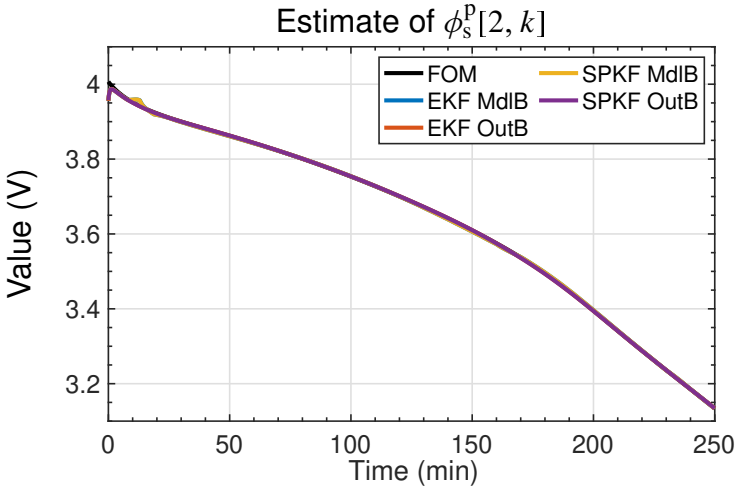


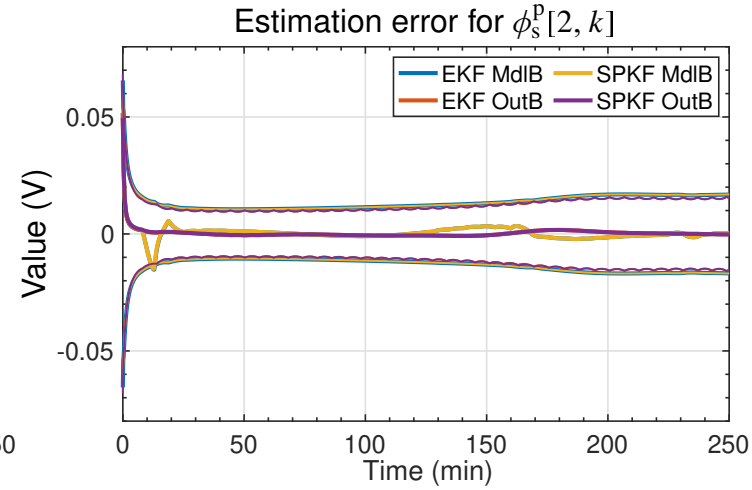
- Again, closed-loop estimates are similar to open-loop predictions. Model blending has less smooth results, and we consider it to be less robust than output blending.

- Next, the solid potentials.

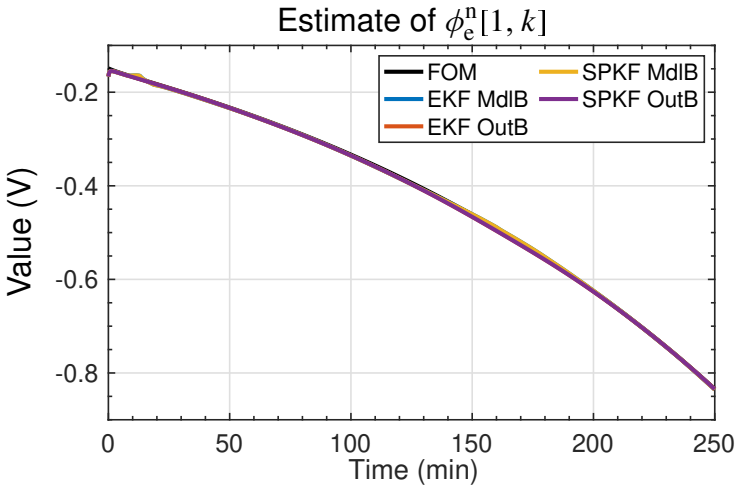


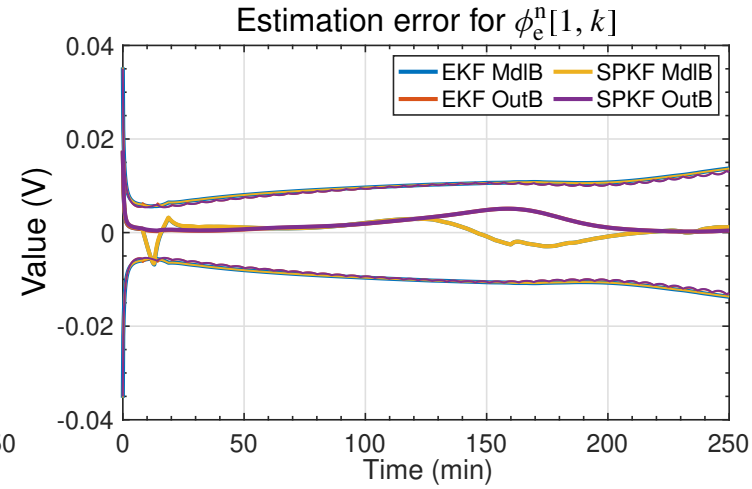


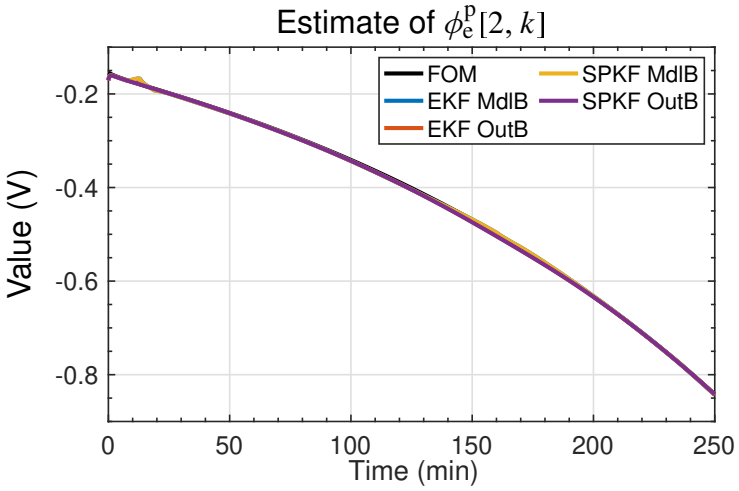


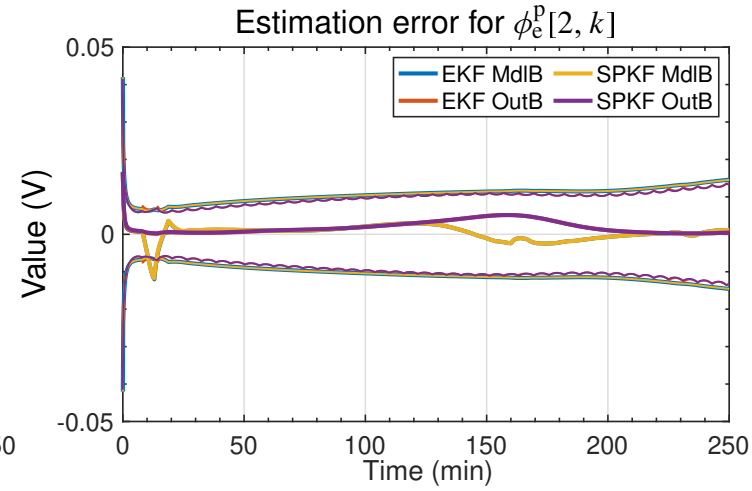


- Electrolyte potential:

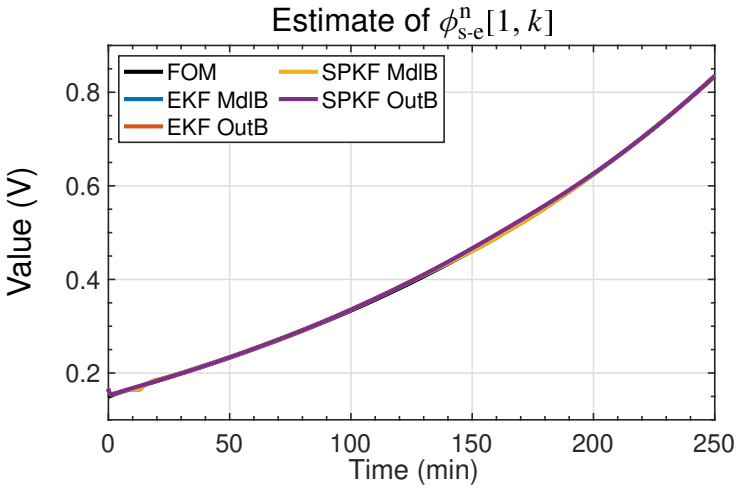


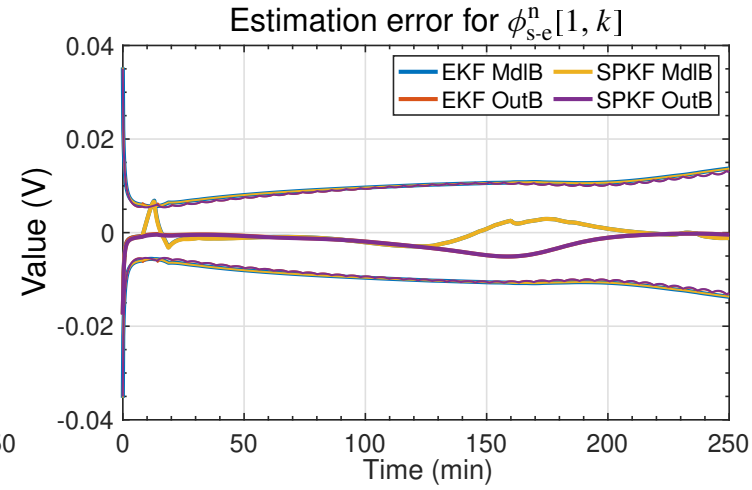


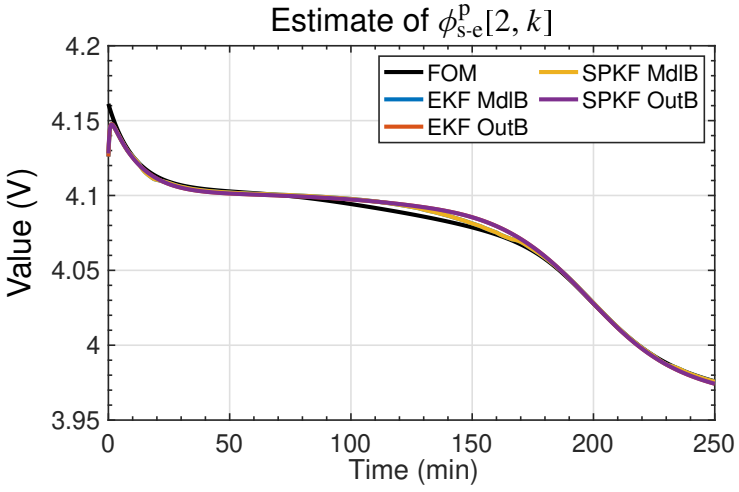


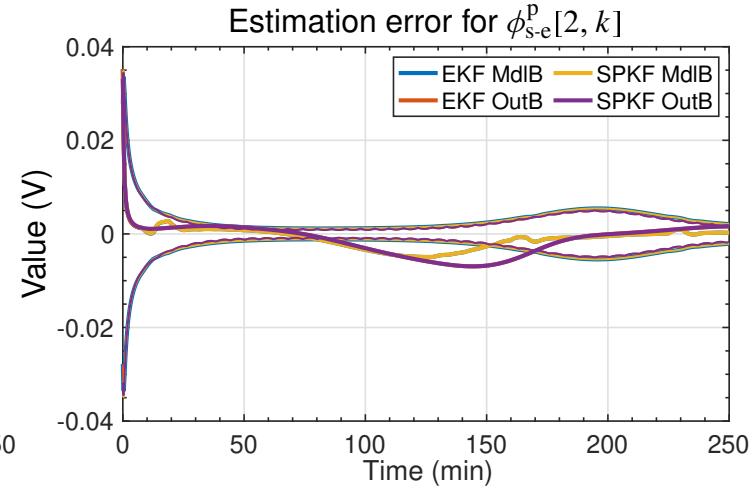


- Finally, interphase potential difference:



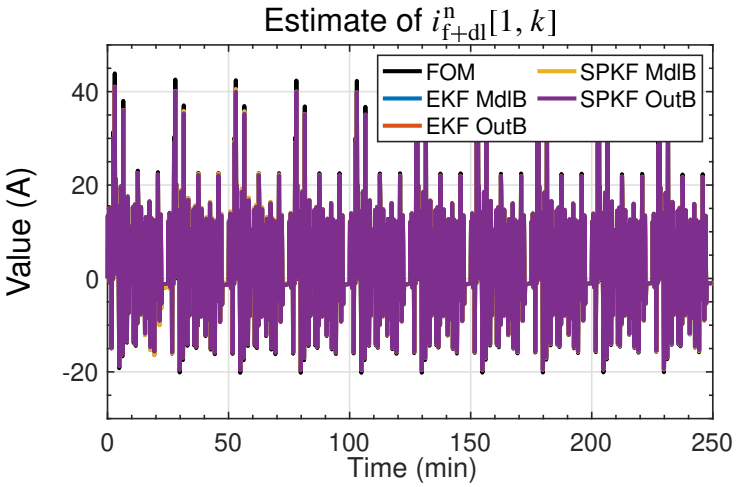


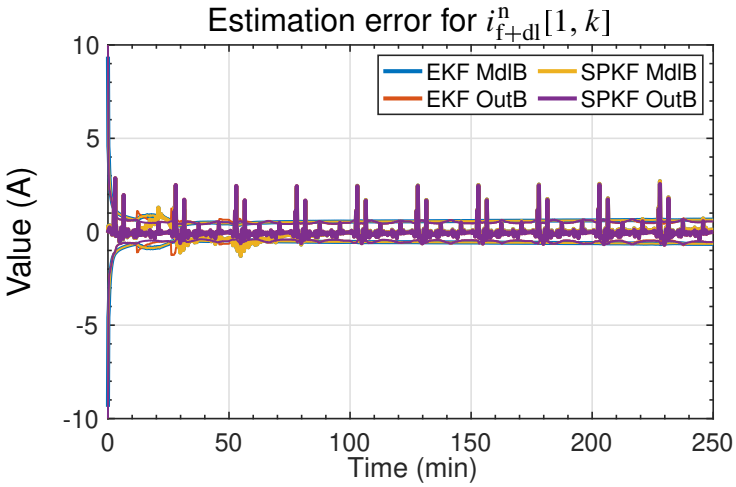


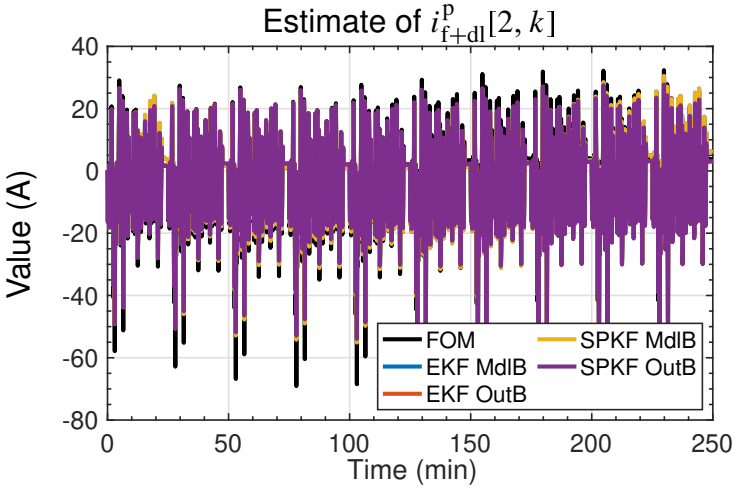


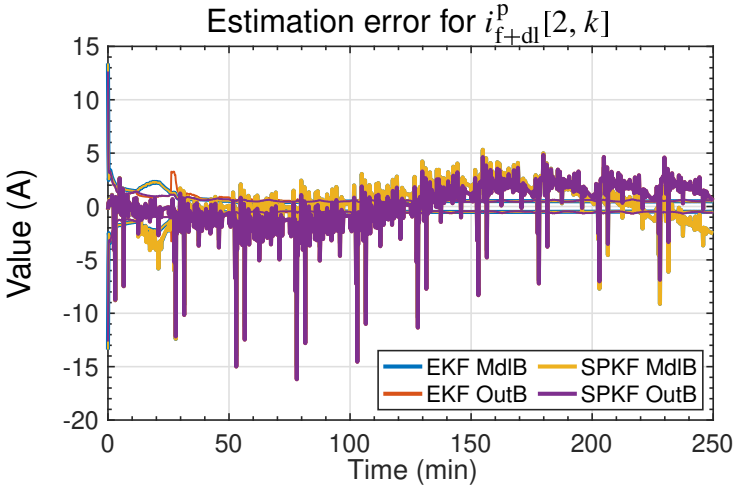
Long charge-depleting UDDS profile internal-variable estimation results

- Next, we look at corresponding results for sequence of ten charge-depleting UDDS profiles. First, lithium flux:

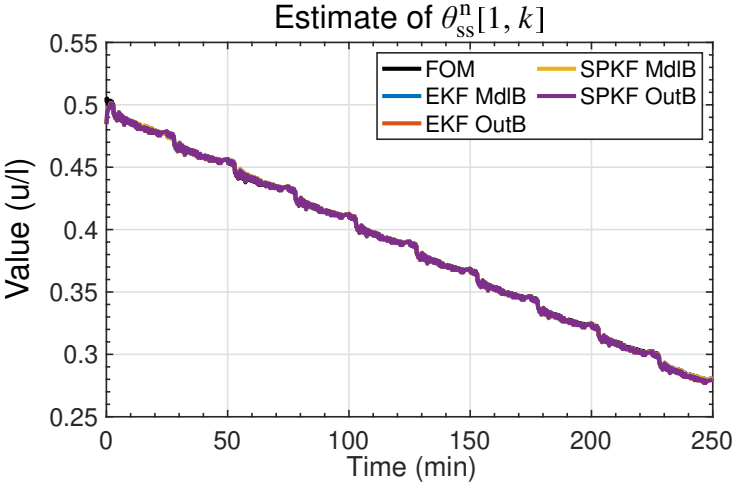


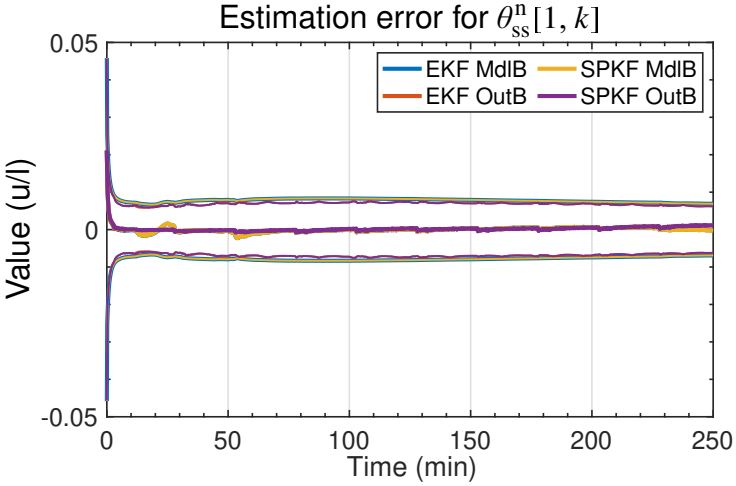


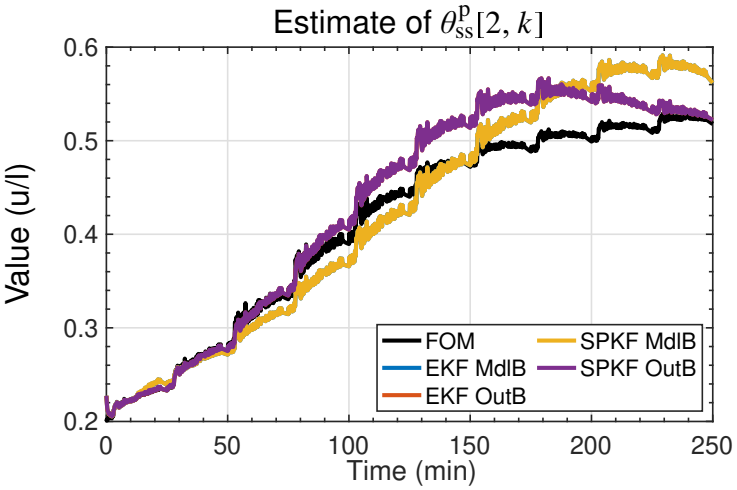


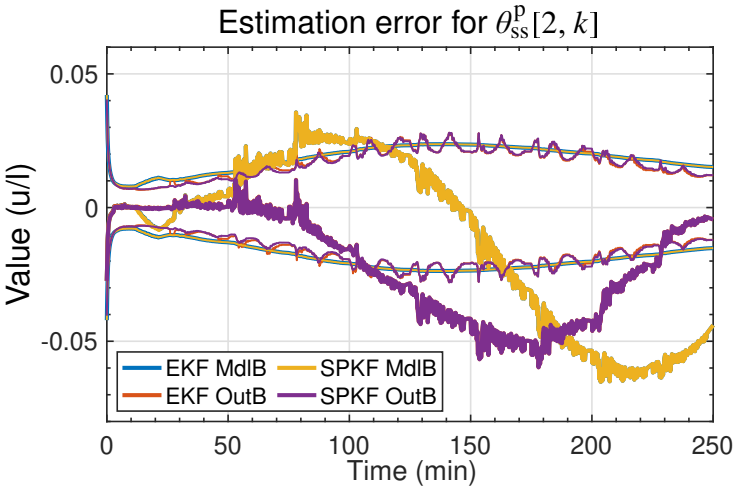


- Next, we look at estimates of solid surface concentrations.

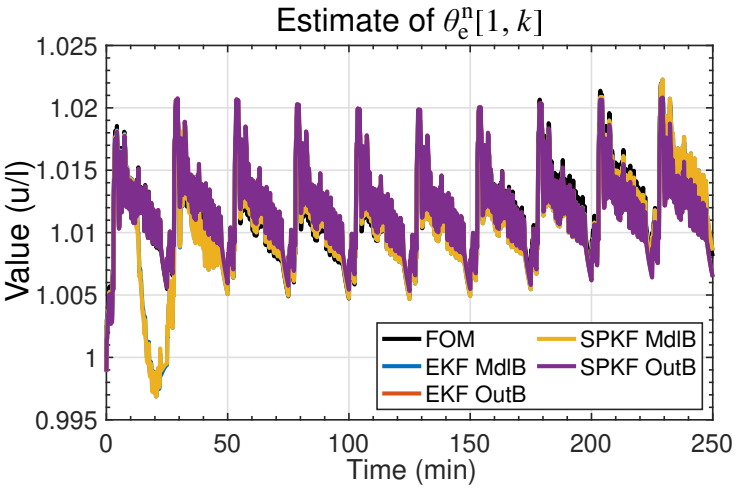


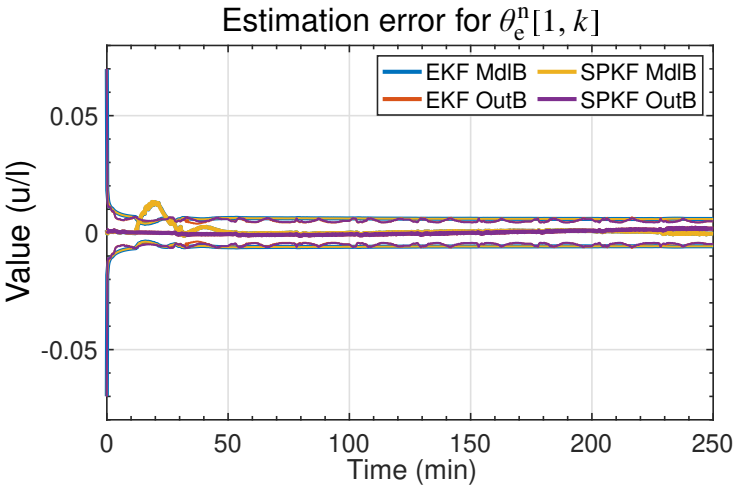


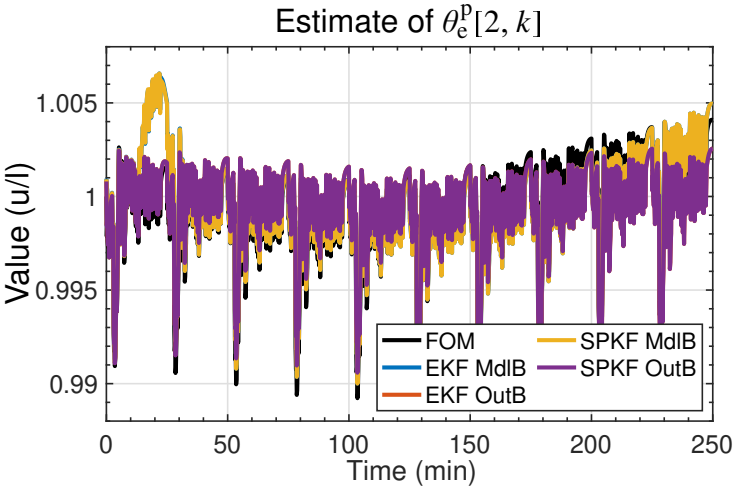


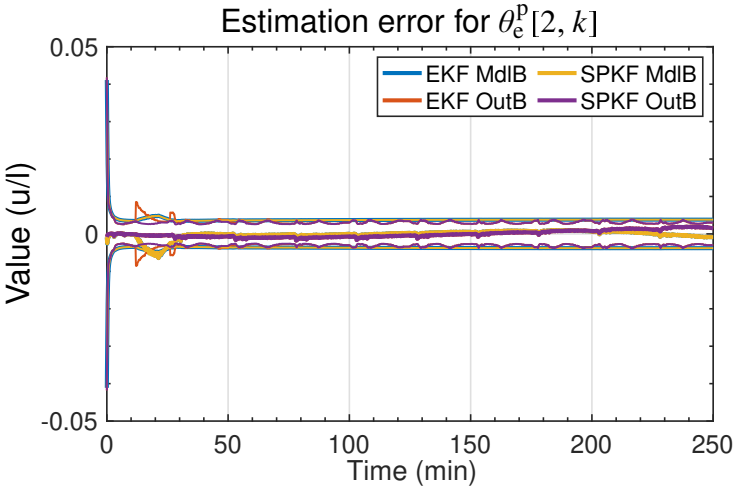


- Next, the electrolyte concentration ratios.

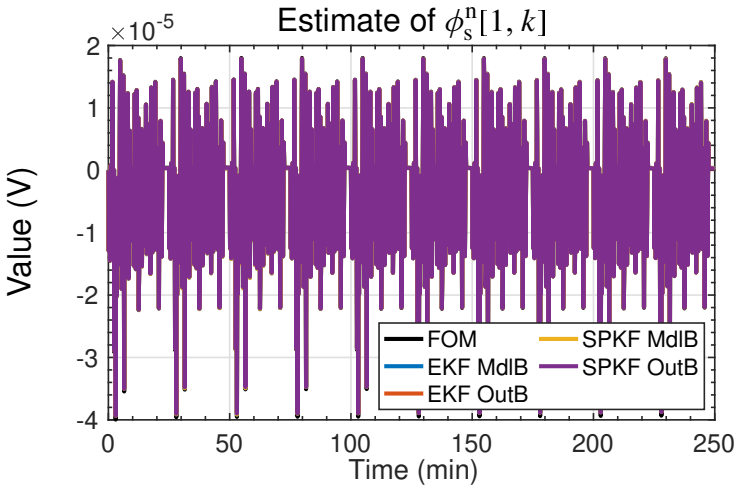


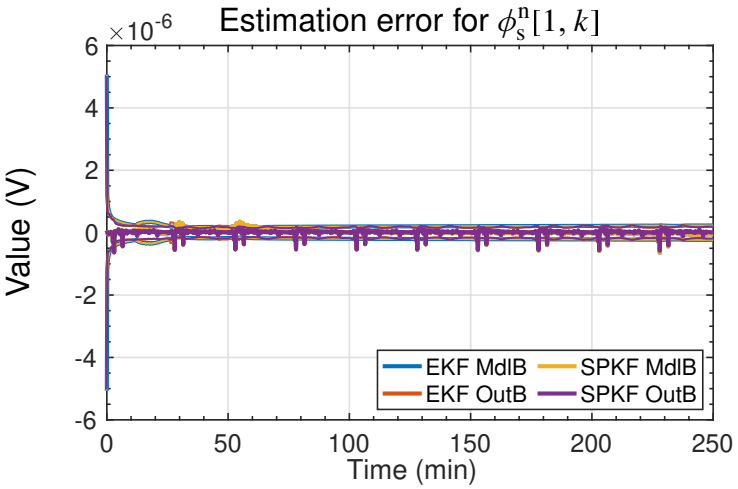


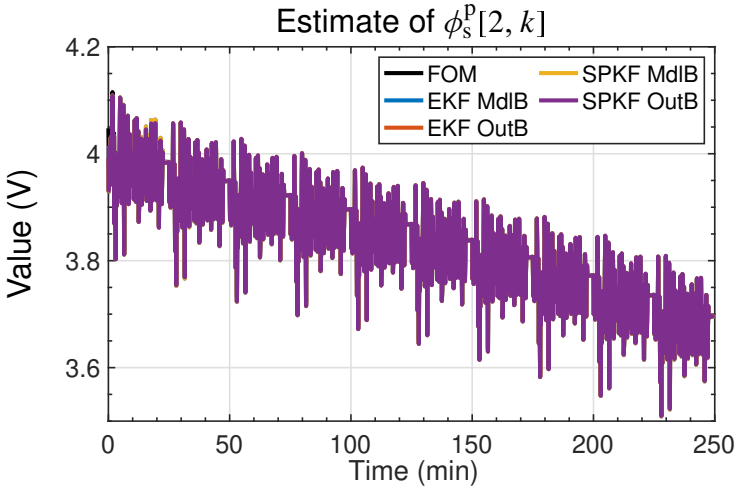


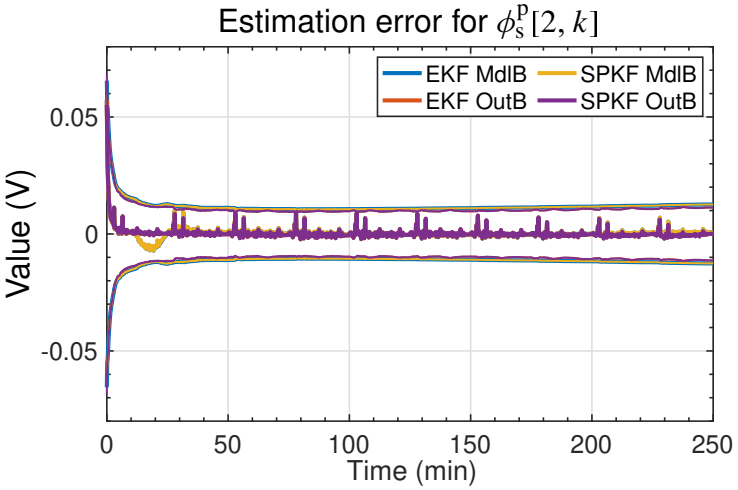


- Next, the solid potentials.

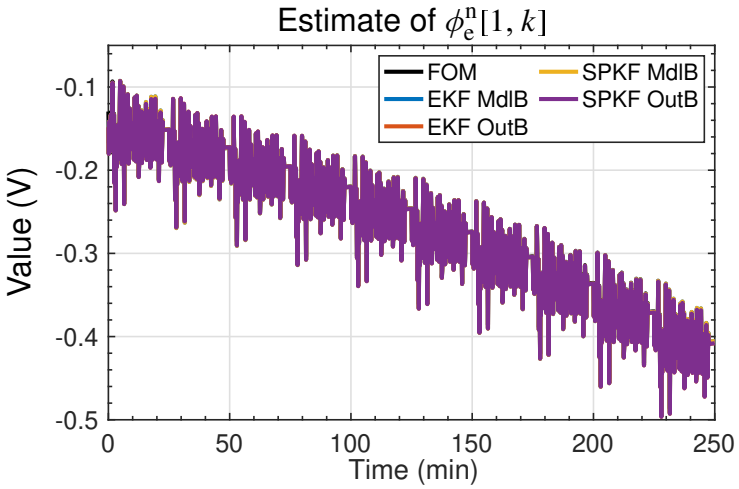


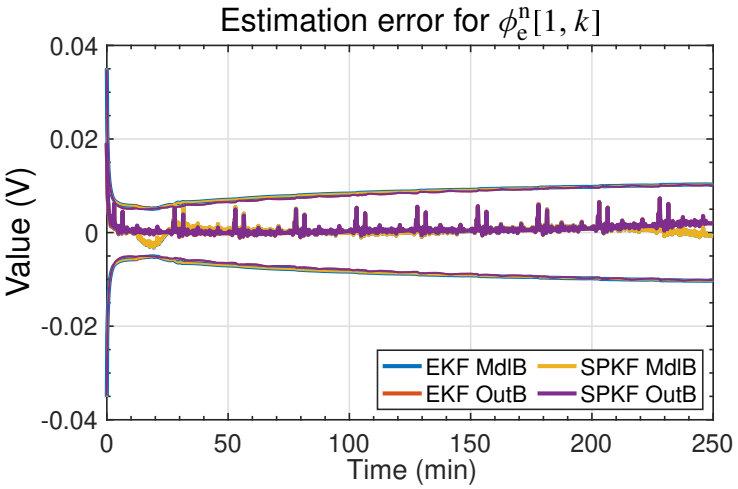


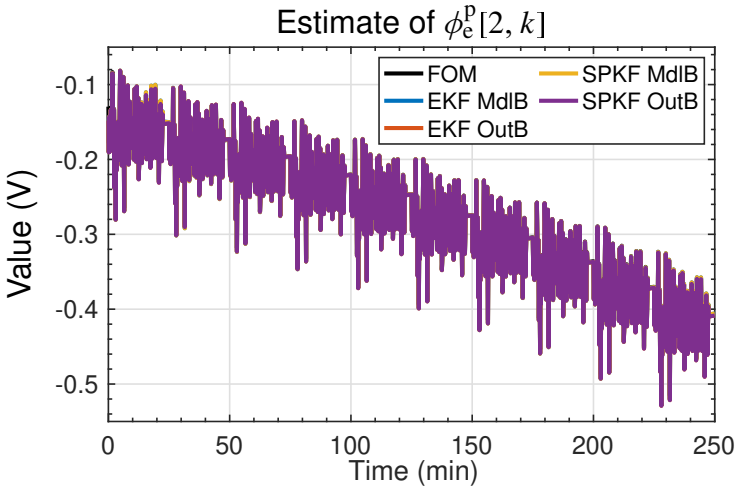


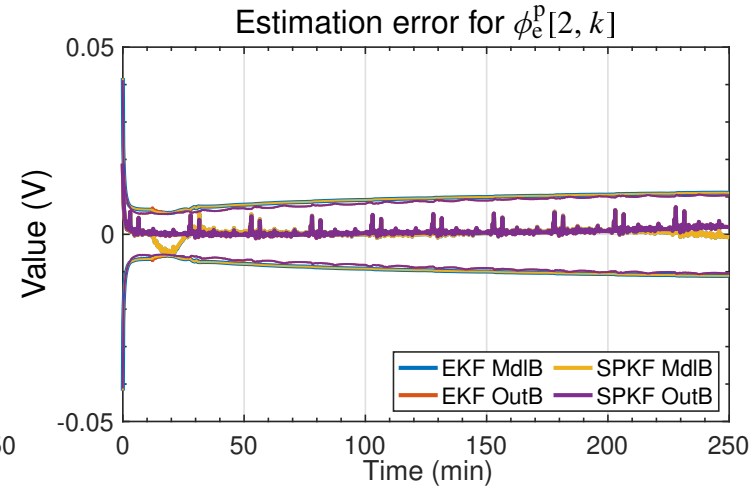


- Electrolyte potential:

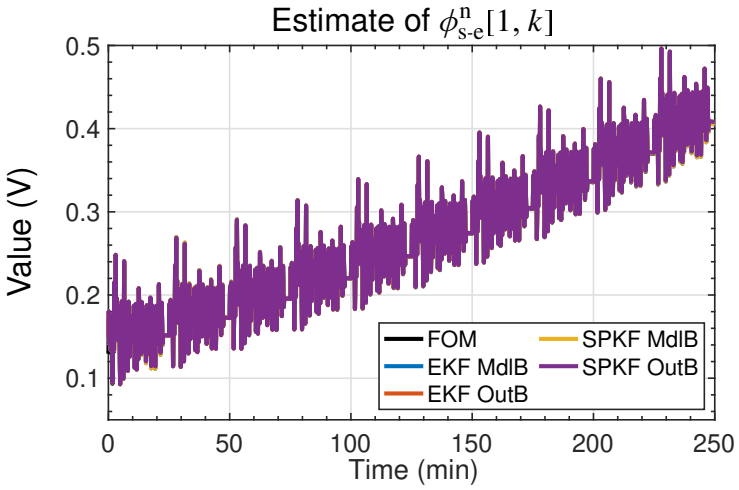


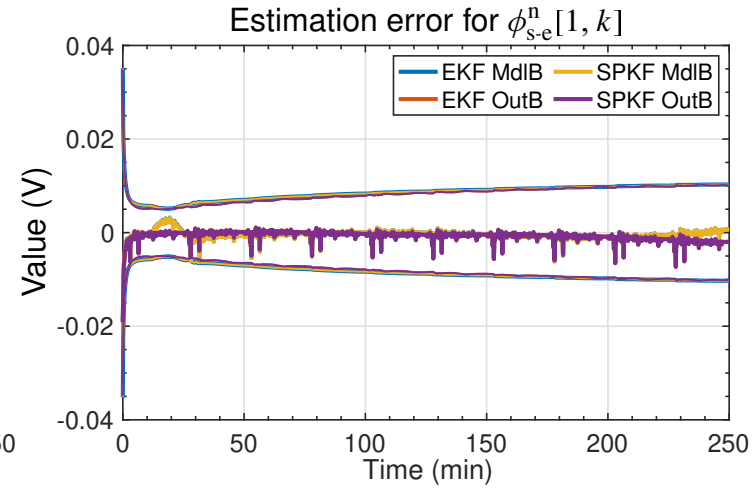


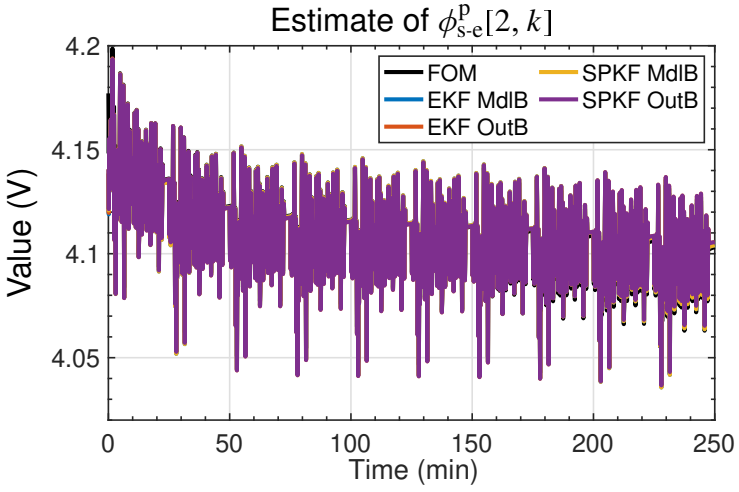


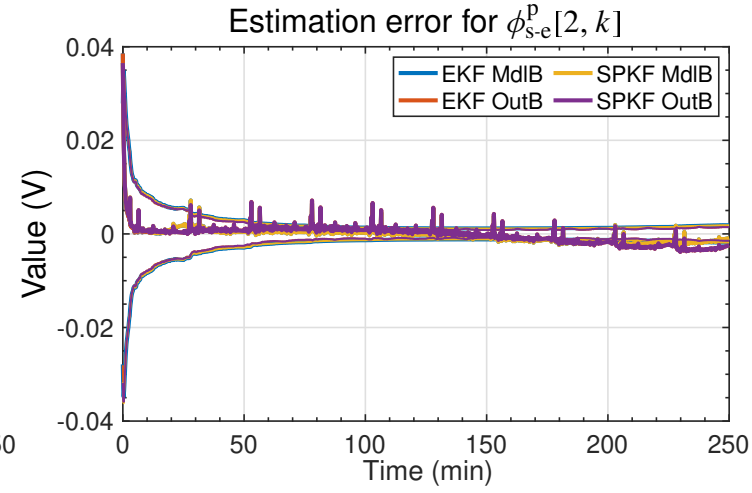


- Finally, interphase potential difference:



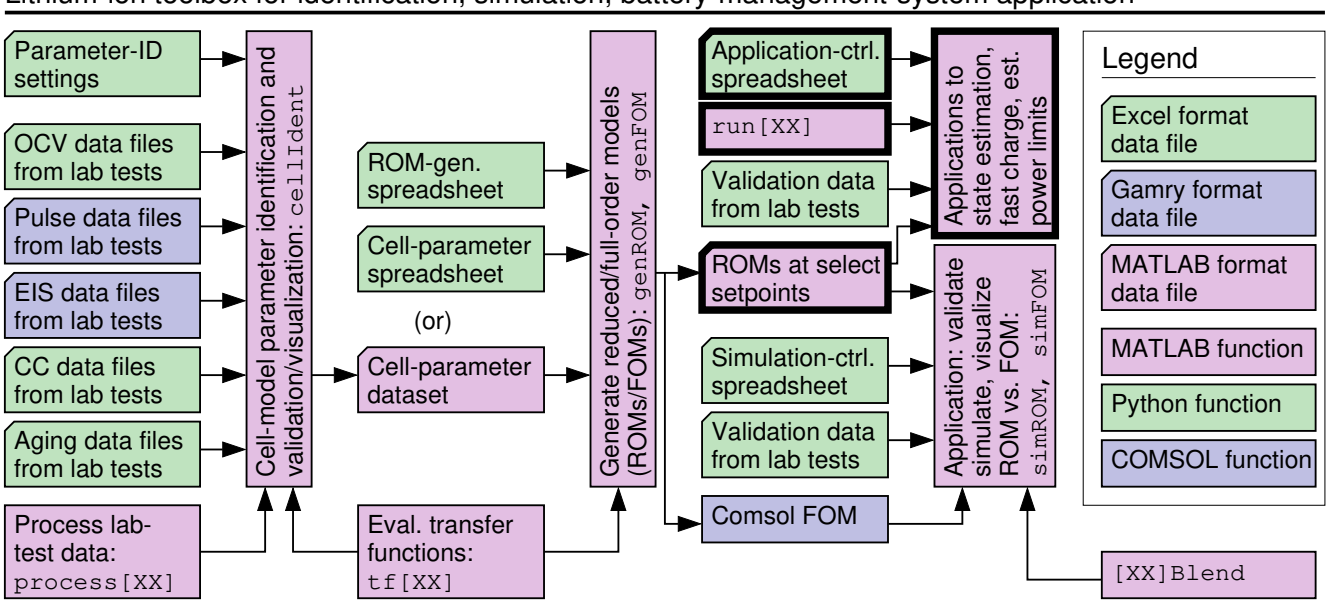






5.9: MATLAB toolbox

- We return to the overview of the MATLAB toolbox from Ch. 2.



- To run an xKF, we first initialize the algorithm via `initKF.m`.

```
% First, set up covariances of initial state, current-sensor noise,
% and voltage-sensor noise (n=8 for this ROM)
SigmaX0 = diag([1e4*ones(1,8) 2e6]); % uncertainty of initial state
SigmaW = 1e0;   % uncertainty of current sensor, state equation
SigmaV = 1e-4;  % uncertainty of voltage sensor, output equation
% Set the filename where the ROMs generated by the HRA are stored
ROMfile    = 'ROM_DOYLEDL_HRA.mat'; % generated ROM
% Set the filename of the FOM simulation output ("truth" data)
FOMoutFile = 'OUT_DOYLEDL_UDDS.mat'; % FOM simulated output
% Set the initial true SOC of the simulation
SOC0 = 60;

% Set the blending method: either 'MdlB' or 'OutB'
blend = 'OutB';

% Load the data files
load(ROMfile,'ROM'); % Load the HRA-produced ROMs
load(FOMoutFile,'FOMout'); % Load the "truth" data

% Initialize the xKF (you can set "SOC0 = []" if you do not know
% the true initial SOC; the xKF will estimate it for you
xkfData = initKF(SOC0,SigmaX0,SigmaV,SigmaW,blend,ROM);
```


- Then, we update the filter once for every measurement using either `iterEKF` (for the EKF method) or `iterSPKF` (for the SPKF method).

```
% reserve storage for SPKF results... including voltage and SOC (the "+2")
zkEst = NaN(spkfData.nz+2,length(FOMout.Iapp));
zkBound = zkEst;

for k = 1:length(FOMout.Iapp)
  vk = FOMout.Vcell(k);
  ik = FOMout.Iapp(k);
  Tk = FOMout.T(k); % degC
  % To EKF, replace "iterSPKF" in next line with "iterEKF"
  [zk,zbk,xkfData] = iterSPKF(vk,ik,Tk,xkfData);
  zkEst(:,k)=zk; zkBound(:,k)=zbk;
end
% Output data structure has one column for every timestep of the sim.
% Each column is organized in same order as "y" outputs of ROMs, plus two
% additional rows: voltage and SOC
```

- The toolbox has examples showing how to run xKFs and plot results.

Where to from here?

- You have now learned how to implement EKF and SPKF on output-blended ROM to estimate internal variables and confidence bounds.
- The most important applications of these estimates are:
  - To monitor cell activity for conditions that might lead to premature aging and failure (and then respond);
  - To compute power limits which, if obeyed, guarantee that these conditions will never occur.
- The remaining chapters investigate:
  - PBM-based diagnosis and prognosis (which use xKF estimates),
  - PBM-based fast-charge and power-limits computations that jointly optimize life and performance. . . also based on xKF estimates.

Appendix 5.a: Summary of variables

Roman

- $\mathfrak{A}_j$, full state-transition matrix of the jth linear model being simulated
- $\boldsymbol{A}_j$, state-transition matrix of the jth linear model being simulated after integrator state removed
- $\mathbf{A}$, state-transition matrix for combined states of all models (note that $\mathbf{A}$ is typeset in an upright font, not italic)
- $\mathfrak{B}_j$, full input matrix of the jth linear model being simulated
- $\boldsymbol{B}_j$, input matrix of the jth linear model being simulated after integrator state removed
- $\mathbf{B}$, input matrix for combined states of all models (note that $\mathbf{B}$ is typeset in an upright font, not italic)
- $\mathfrak{C}_j$, full output matrix of the jth linear model being simulated
- $\boldsymbol{C}_j$, output matrix of the jth linear model being simulated after integrator state removed
- $\boldsymbol{C}_0$, output matrix for the integrator state common to all linear models
- $\boldsymbol{D}_j$, direct-feedthrough matrix of the jth linear model being simulated
- $\mathbb{E}[\cdot]$, expected value of the argument
- $\boldsymbol{f}(\boldsymbol{x}[k], u[k], w[k])$, nonlinear state-transition function producing $\boldsymbol{x}[k+1]$
- $g_v(\boldsymbol{x}[k], u[k])$, nonlinear function converting state to voltage $v_{\text{cell}}[k]$
- $\boldsymbol{g}_z(\boldsymbol{x}[k], u[k])$, nonlinear function converting state to electrochemical variables $\boldsymbol{z}[k]$
- $\boldsymbol{L}[k]$, estimator gain
- $u[k]$, generic model input ($i_{\text{app}}[k]$) for us
- $v[k]$, measurement (voltage-sensor) noise
- $\boldsymbol{\mathcal{V}}$, set of sigma points representing possible cell voltages
- $v_{\text{cell}}[k]$, voltage
- $\mathbb{V}_{\text{cell}}[k] = \{v_{\text{cell}}[0], \cdots, v_{\text{cell}}[k]\}$, sequence of voltage measurements up to time k
- $w[k]$, process (current-sensor) noise

- $\mathfrak{X}[k]$, state of a single model at time k
- $\boldsymbol{x}_j[k]$, the state of a single model, model j, without the integrator, at time k
- $x_0[k]$, the state of the integrator, shared in common by all models, at time k
- $\mathbf{X}[k]$, the combined states of all models at time k (note that $\mathbf{X}$ is typeset in an upright font, not italic)
- $\mathbf{X}_\gamma[k]$, the combined states of all models being blended presently at time k (note that $\mathbf{X}_\gamma$ is typeset in an upright font, not italic)
- $\mathcal{X}$, set of sigma points representing possible system states
- $\mathbb{Y}[k] = \{y[0], \cdots, y[k]\}$, sequence of measurements up to time k
- $\tilde{\boldsymbol{y}}[k]$, the innovation (difference between measurement and its prediction)
- $\boldsymbol{y}[k]$, linear output of ROM after blending but before nonlinear corrections
- $\boldsymbol{y}_j[k]$, linear output of the jth linear ROM (before blending and nonlinear corrections)
- $\mathcal{Z}$, set of sigma points representing possible values of internal electrochemical variables of interest
- $\boldsymbol{z}[k]$, the vector of electrochemical variables desired to be estimated

Greek

- $\alpha_i^{(\mathrm{c})}$, weighting values when determining covariance from sigma points
- $\alpha_i^{(\mathrm{m})}$, weighting values when determining mean from sigma points
- γ_j, blending factor for jth ROM in output-blending approach
- Σ, correlation between the two arguments in its subscript (covariance if the arguments are zero mean)

Symbols

- $(\cdot)^-$, superscript indicates a predicted quantity based only on past measurements
- $(\cdot)^+$, superscript indicates an estimated quantity based on both past and present measurements
- $\hat{(\cdot)}$, "hat" indicates a predicted or estimated quantity
- $\tilde{(\cdot)}$, "tilde" indicates the difference between a true and predicted/estimated quantity

Appendix 5.b: EKF derivative matrices

Computing derivatives of cell voltage with respect to states

- EKF Step 2a requires $\widehat{C}_0$ and derivative matrices $\widehat{\boldsymbol{C}}_{0,0}$, $\widehat{\boldsymbol{C}}_{0,1}$, $\widehat{\boldsymbol{C}}_{1,0}$, $\widehat{\boldsymbol{C}}_{1,1}$:

$$\widehat{\boldsymbol{C}}_j = \frac{\mathrm{d}g_v(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}\boldsymbol{x}_j[k]} \quad \text{and} \quad \widehat{C}_0 = \frac{\mathrm{d}g_v(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}x_0[k]},$$

- We expand in terms of the electrochemical variables:

$$\frac{\mathrm{d}g_v(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}\boldsymbol{x}_j[k]} = \frac{\partial g_v(\mathbf{X}_\gamma[k], u[k])}{\partial \boldsymbol{z}[k]} \frac{\mathrm{d}\boldsymbol{z}[k]}{\mathrm{d}\boldsymbol{x}_j[k]}.$$

- From Ch. 4, we know:

$$v_{\mathrm{cell}}[k] = \left(\eta^{\mathrm{p}}[3,k] - \eta^{\mathrm{n}}[0,k]\right) + \tilde{\phi}_{\mathrm{e}}^{\mathrm{p}}[3,k] + \left(U_{\mathrm{ocp}}^{\mathrm{p}}(\theta_{\mathrm{ss}}^{\mathrm{p}}[3,k]) - U_{\mathrm{ocp}}^{\mathrm{n}}(\theta_{\mathrm{ss}}^{\mathrm{n}}[0,k])\right)$$
$$+ \left(\bar{R}_{\mathrm{f}}^{\mathrm{p}} i_{\mathrm{f+dl}}^{\mathrm{p}}[3,k] - \bar{R}_{\mathrm{f}}^{\mathrm{n}} i_{\mathrm{f+dl}}^{\mathrm{n}}[0,k]\right).$$

- Therefore, to find the derivative matrices, we must be able to find the derivative of each of these terms with respect to a model state.
- We also need to remember that the output-blended linear model outputs are a weighted summation of multiple models (cf. Eq. (5.4)):

$$\boldsymbol{y}[k] = \sum_{j=1}^{N} \gamma_j \left(\boldsymbol{C}_j \boldsymbol{x}_j[k] + \boldsymbol{C}_{0,j} x_0[k] + \boldsymbol{D}_j u[k]\right).$$

  - Therefore, there will generally be γ_j terms in each derivative.

- Denote the $1 \times n$ row of $\boldsymbol{C}_j$ corresponding to electrochemical variable "var" at location $\tilde{x}$ as $[\boldsymbol{C}_j^{\mathrm{var}[\tilde{x}]}]$. Then, for example,

$$\frac{\mathrm{d}i_{\mathrm{f+dl}}^{\mathrm{n}}[0,k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j \left[\boldsymbol{C}_j^{i_{\mathrm{f+dl}}^{\mathrm{n}}[0]}\right] \quad \text{and} \quad \frac{\mathrm{d}i_{\mathrm{f+dl}}^{\mathrm{p}}[3,k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j \left[\boldsymbol{C}_j^{i_{\mathrm{f+dl}}^{\mathrm{p}}[3]}\right].$$

- We use a simplified model of overpotential for the EKF update,

$$\eta^{\mathrm{r}}[\tilde{x},k] \approx \bar{R}_{\mathrm{ct}}^{\mathrm{r}} i_{\mathrm{f}}^{\mathrm{r}}[\tilde{x},k] = \frac{RT}{\bar{i}_0 F} i_{\mathrm{f}}^{\mathrm{r}}[\tilde{x},k],$$

where $\bar{i}_0^{\mathrm{r}} = \bar{k}_0^{\mathrm{r}} \theta_{\mathrm{s},0}^{\mathrm{r}} (1 - \theta_{\mathrm{s},0}^{\mathrm{r}})$. Therefore,

$$\frac{\mathrm{d}\eta^{\mathrm{n}}[0,k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j \bar{R}_{\mathrm{ct}}^{\mathrm{n}}\left[\boldsymbol{C}_j^{i_{\mathrm{f}}^{\mathrm{n}}[0]}\right] \quad \text{and} \quad \frac{\mathrm{d}\eta^{\mathrm{p}}[3,k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j \bar{R}_{\mathrm{ct}}^{\mathrm{p}}\left[\boldsymbol{C}_j^{i_{\mathrm{f}}^{\mathrm{p}}[3]}\right].$$

- The derivatives

$$\frac{\mathrm{d}U_{\mathrm{ocp}}^{\mathrm{r}}(\theta_{\mathrm{ss}}^{\mathrm{r}}[\tilde{x},k])}{\mathrm{d}\boldsymbol{x}_j[k]} = \frac{\partial U_{\mathrm{ocp}}^{\mathrm{r}}}{\partial \theta_{\mathrm{ss}}^{\mathrm{r}}} \frac{\mathrm{d}\theta_{\mathrm{ss}}^{\mathrm{r}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j} = \gamma_j [U_{\mathrm{ocp}}^{\mathrm{r}}]' \left[\boldsymbol{C}_j^{\tilde{\theta}_{\mathrm{ss}}^{\mathrm{r}}[\tilde{x}]}\right],$$

where $[U_{\mathrm{ocp}}^{\mathrm{r}}]'$ is computed from $U_{\mathrm{ocp}}^{\mathrm{r}}$ at the present operating point.

- In summary, $\widehat{\boldsymbol{C}}_{0,0}$, $\widehat{\boldsymbol{C}}_{0,1}$, $\widehat{\boldsymbol{C}}_{1,0}$, and $\widehat{\boldsymbol{C}}_{1,1}$ are computed via:

$$\widehat{\boldsymbol{C}}_j = \gamma_j \Bigg[\bar{R}_{\mathrm{ct}}^{\mathrm{p}}\left[\boldsymbol{C}_j^{i_{\mathrm{f}}^{\mathrm{p}}[3]}\right] - \bar{R}_{\mathrm{ct}}^{\mathrm{n}}\left[\boldsymbol{C}_j^{i_{\mathrm{f}}^{\mathrm{n}}[0]}\right] + \left[\boldsymbol{C}_j^{\tilde{\phi}_{\mathrm{e}}^{\mathrm{p}}[3]}\right] + [U_{\mathrm{ocp}}^{\mathrm{p}}]'\left[\boldsymbol{C}_j^{\tilde{\theta}_{\mathrm{ss}}^{\mathrm{p}}[3]}\right]$$

$$- [U_{\mathrm{ocp}}^{\mathrm{n}}]'\left[\boldsymbol{C}_j^{\tilde{\theta}_{\mathrm{ss}}^{\mathrm{n}}[0]}\right] + \bar{R}_{\mathrm{f}}^{\mathrm{p}}\left[\boldsymbol{C}_j^{i_{\mathrm{f+dl}}^{\mathrm{p}}[3]}\right] - \bar{R}_{\mathrm{f}}^{\mathrm{n}}\left[\boldsymbol{C}_j^{i_{\mathrm{f+dl}}^{\mathrm{n}}[0]}\right] \Bigg].$$

- The term relating to the integrator depends on $\boldsymbol{C}_0$, the column of $\mathfrak{C}$ containing integration residues. Since the only portion of the output equation that has integration dynamics is $\theta_{\mathrm{ss}}^{\mathrm{r}}$, scalar

$$\widehat{C}_0 = [U_{\mathrm{ocp}}^{\mathrm{p}}]'\left[\boldsymbol{C}_0^{\tilde{\theta}_{\mathrm{ss}}^{\mathrm{p}}[3]}\right] - [U_{\mathrm{ocp}}^{\mathrm{n}}]'\left[\boldsymbol{C}_0^{\tilde{\theta}_{\mathrm{ss}}^{\mathrm{n}}[0]}\right].$$

Computing derivatives of electrochemical variables with respect to states

- EKF Step 3b requires derivative matrices:

$$\widehat{\boldsymbol{C}}_j = \frac{\mathrm{d}\boldsymbol{g}_{\boldsymbol{z}}(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}\boldsymbol{x}_j[k]} \quad \text{and} \quad \widehat{\boldsymbol{C}}_0 = \frac{\mathrm{d}\boldsymbol{g}_{\boldsymbol{z}}(\mathbf{X}_\gamma[k], u[k])}{\mathrm{d}x_0[k]},$$

  - Note: Matrices have same names as those used in Step 2a, but different definitions. Usage should be clear by context.
  - If the dimension of $\boldsymbol{z}[k]$ is $q \times 1$, then the dimensions of $\widehat{\boldsymbol{C}}_{0,0}$, $\widehat{\boldsymbol{C}}_{0,1}$, $\widehat{\boldsymbol{C}}_{1,0}$, and $\widehat{\boldsymbol{C}}_{1,1}$, are $q \times n$, and the dimension of $\widehat{\boldsymbol{C}}_0$ is $q \times 1$.

- We compute these matrices one row at a time, where each row is the derivative component relating to the corresponding element in $\boldsymbol{z}[k]$.
- Derivatives of elements of $\boldsymbol{z}[k]$ corresponding to $i^{\mathrm{r}}_{\mathrm{f+dl}}[\tilde{x},k]$ are:

$$\frac{\mathrm{d}i^{\mathrm{r}}_{\mathrm{f+dl}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{i^{\mathrm{r}}_{\mathrm{f+dl}}[\tilde{x}]}\right].$$

- Apply same principle to find derivatives of $i^{\mathrm{r}}_{\mathrm{f}}[\tilde{x},k]$ and $i^{\mathrm{r}}_{\mathrm{dl}}[\tilde{x},k]$:

$$\frac{\mathrm{d}i^{\mathrm{r}}_{\mathrm{f}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{i^{\mathrm{r}}_{\mathrm{f}}[\tilde{x}]}\right] \quad \text{and} \quad \frac{\mathrm{d}i^{\mathrm{r}}_{\mathrm{dl}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{i^{\mathrm{r}}_{\mathrm{dl}}[\tilde{x}]}\right].$$

- When computing derivatives corresponding to $\theta^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k]$, recall (Ch. 4):

$$\begin{aligned}\theta^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k] &= \tilde{\theta}^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k] + \theta^{\mathrm{r}}_{\mathrm{s},0} \\ &= \left[\tilde{\theta}^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k]\right]^* - \frac{[\theta^{\mathrm{r}}_{100} - \theta^{\mathrm{r}}_0]T_{\mathrm{s}}}{3600Q - \bar{C}^{\mathrm{r}}_{\mathrm{dl,eff}}\left|\theta^{\mathrm{r}}_{100} - \theta^{\mathrm{r}}_0\right|\ [U^{\mathrm{r}}_{\mathrm{ocp}}]'}x_0[k] + \theta^{\mathrm{r}}_{\mathrm{s},0},\end{aligned}$$

  where $\left[\tilde{\theta}^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k]\right]^*$ corresponds to the integrator-removed debiased $\theta^{\mathrm{r}}_{\mathrm{ss}}$.
- We simplify the integration residue somewhat, trusting the EKF to adapt to track any error introduced by doing so.

$$\theta^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k] \approx \left[\tilde{\theta}^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k]\right]^* - \frac{[\theta^{\mathrm{r}}_{100} - \theta^{\mathrm{r}}_0]T_{\mathrm{s}}}{3600Q}x_0[k] + \theta^{\mathrm{r}}_{\mathrm{s},0}, \tag{5.13}$$

  and so (because integrator state is not considered in this derivative):

$$\frac{\mathrm{d}\theta^{\mathrm{r}}_{\mathrm{ss}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{[\tilde{\theta}^{\mathrm{r}}_{\mathrm{ss}}]^*[\tilde{x}]}\right].$$

- The derivatives of $\phi^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x},k]$ are made in a similar way. Recall,

$$\phi^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x},k] = [\tilde{\phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x},k]]^* + U^{\mathrm{r}}_{\mathrm{ocp}}[\theta^{\mathrm{r}}_{\mathrm{s,avg}}[k]]. \tag{5.14}$$

- So, since $\theta^{\mathrm{r}}_{\mathrm{s,avg}}[k]$ depends only on the integrator state x_0,

$$\frac{\mathrm{d}\phi^{\mathrm{r}}_{\mathrm{s\text{-}e}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{[\tilde{\phi}^{\mathrm{r}}_{\mathrm{s\text{-}e}}]^*[\tilde{x}]}\right].$$

- The derivatives of $\phi_{\mathrm{s}}^{\mathrm{n}}[\tilde{x},k]$ are simply:

$$\frac{\mathrm{d}\phi_{\mathrm{s}}^{\mathrm{n}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{\tilde{\phi}_{\mathrm{s}}^{\mathrm{n}}[\tilde{x}]}\right].$$

- In the positive electrode, we must add the derivatives of $v_{\mathrm{cell}}[k]$ with respect to the state (computed in the first part of this appendix). So,

$$\frac{\mathrm{d}\phi_{\mathrm{s}}^{\mathrm{p}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{\tilde{\phi}_{\mathrm{s}}^{\mathrm{p}}[\tilde{x}]}\right] + \frac{\mathrm{d}g_v(\mathbf{X}_\gamma[k],u[k])}{\mathrm{d}\boldsymbol{x}_j[k]}.$$

- The derivatives of $\phi_{\mathrm{e}}^{\mathrm{n}}[\tilde{x},k]$ are computed by recognizing:

$$\phi_{\mathrm{e}}^{\mathrm{r}}[\tilde{x},k] = \tilde{\phi}_{\mathrm{e}}^{\mathrm{r}}[\tilde{x},k] - [\tilde{\phi}_{\mathrm{s\text{-}e}}^{\mathrm{n}}[0,k]]^* - U_{\mathrm{ocp}}^{\mathrm{n}}[\theta_{\mathrm{s,avg}}^{\mathrm{n}}[k]]. \tag{5.15}$$

- Therefore, using prior results, and recognizing that the OCP term depends only on the integration state, we have:

$$\frac{\mathrm{d}\phi_{\mathrm{e}}^{\mathrm{r}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{\tilde{\phi}_{\mathrm{e}}^{\mathrm{r}}[\tilde{x}]}\right] - \gamma_j\left[\boldsymbol{C}_j^{[\tilde{\phi}_{\mathrm{s\text{-}e}}^{\mathrm{n}}]^*[0]}\right]$$

- Finally, the derivates corresponding to $\theta_{\mathrm{e}}^{\mathrm{r}}[\tilde{x},k]$ are simply:

$$\frac{\mathrm{d}\theta_{\mathrm{e}}^{\mathrm{r}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_j[k]} = \gamma_j\left[\boldsymbol{C}_j^{\tilde{\theta}_{\mathrm{e}}^{\mathrm{r}}[\tilde{x}]}\right].$$

- We now find derivatives of each variable w.r.t. the integrator state.
- Default value of every $\widehat{\boldsymbol{C}}_0$ entry is zero since most variables do not integrate. Exceptions are (cf. Eqs. (5.13), (5.14), and (5.15)):

$$\frac{\mathrm{d}\theta_{\mathrm{ss}}^{\mathrm{r}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_0[k]} = -\frac{[\theta_{100}^{\mathrm{r}} - \theta_0^{\mathrm{r}}]T_{\mathrm{s}}}{3600Q}$$

$$\frac{\mathrm{d}\phi_{\mathrm{s\text{-}e}}^{\mathrm{r}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_0[k]} = -[U_{\mathrm{ocp}}^{\mathrm{r}}]'\frac{[\theta_{100}^{\mathrm{r}} - \theta_0^{\mathrm{r}}]T_{\mathrm{s}}}{3600Q}$$

$$\frac{\mathrm{d}\phi_{\mathrm{e}}^{\mathrm{r}}[\tilde{x},k]}{\mathrm{d}\boldsymbol{x}_0[k]} = [U_{\mathrm{ocp}}^{\mathrm{r}}]'\frac{[\theta_{100}^{\mathrm{n}} - \theta_0^{\mathrm{n}}]T_{\mathrm{s}}}{3600Q}.$$

