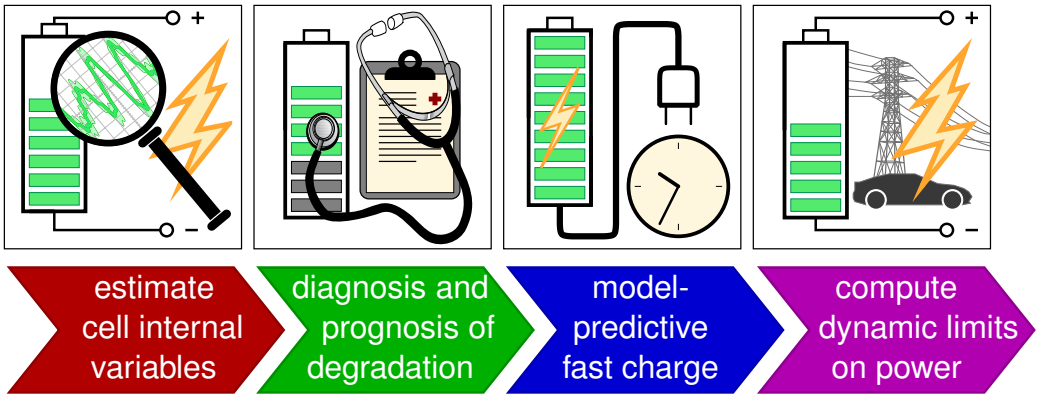
OPTIMAL FAST CHARGE

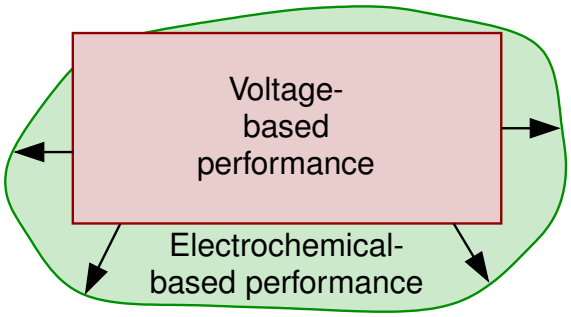
7.1: Fast-charge control problem

- Our progress so far can again be visualized using the roadmap:



- So far, we've seen how PBMs of lithium-ion dynamics can accurately predict a cells internal electrochemical variables as well as externally measured indicators of ideal battery behavior.
- We've discovered how to estimate a cell's internal electrochemical state using measurable quantities and nonlinear Kalman filters.
- We have further seen how PBMs of degradation mechanisms can forecast a loss of performance or shortening of battery lifetime.
- In this and the next chapter, we introduce two practical control applications of PBMs: (1) optimal fast charge, and (2) calculation of operational power limits.
- Present state-of-the-art ECM-based battery controls address short-term performance objectives using voltage-based design limits.

- Although ECMs and voltage limits provide adequate performance for many applications, the goal of charging a lithium-ion cell as quickly as possible is ultimately limited by internal electrochemical effects.
- Exceeding certain internal physical limits—not foreseen by an externally measurable voltage—can cause irreversible damage and capacity loss—degrading long-term performance.
- The figure portrays this idea via a notional cell performance envelope.
- BMS algorithms using ECMs and empirical voltage limits give performance defined by the inner box.



- Physics-based approaches can theoretically move the performance boundary outward by removing conservatism by incorporating knowledge of the values of cell internal electrochemical variables.
  - This enables operation right to the true electrochemical performance limits of the cell.
- Note that it is possible that empirical limits can result in harmful operation outside of the true safe region because, again, the empirical models are unaware of the cell's true internal electrochemical state.
- The widely used constant-current constant-voltage (CCCV) charge profile is based on voltage limits—and may either damage the cell or not take full advantage of cell's true operating range!
- In this chapter, we show potential benefits of an electrochemical-based approach by applying model predictive control (MPC) to the fast-charge of a lithium ion battery.

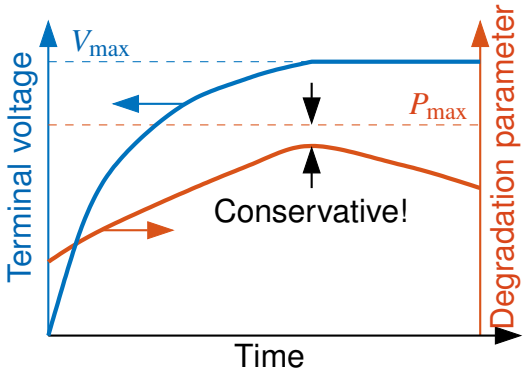
- The proposed method:[1,2]
  - Uses the computationally compact reduced-order, electrochemical model introduced in Ch. 4.
  - Imposes hard constraints on internal cell variables to prevent a severe degradation mechanism—lithium plating.
  - Computes step-wise optimal applied maximum and/or minimum value of current to bring about desired result.
- We hypothesize that battery cell performance—and lifetime—can be extended by designing controls like this, which allow cell performance up to but not exceeding true electrochemical limits.

[1] This chapter introduces the basics of MPC but there is insufficient time to explore all its nuances. If you are interested in a deeper understanding, Prof. Trimboli offers a full semester-long course on this subject, *ECE5590, Model Predictive Control*.

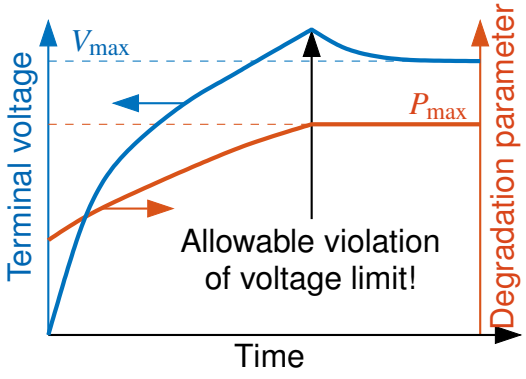
[2] We do not consider it in this chapter, but we imagine that a similar approach could be taken to optimize a fast discharge. The physical constraints that would be applied to the electrochemical variables would be different, but the optimization strategy would be the same.

7.2: Fast-charge limitations

- Battery fast-charging protocols are commonly governed by seeking to maintain voltage less than or equal to an upper limit.
- But as already pointed out, a limit on cell terminal voltage is not a criterion a BMS algorithm designer ought to be concerned about.
- Manufacturer-supplied voltage limits are merely imperfect and indirect indicators of electro/chemical/mechanical cell aging processes.
- While cell voltage is measurable, it does not adequately predict the amount of damage expected to occur at different charging levels at any point in time caused by following various charging protocols.
- Charging rates should ideally be calculated in real-time to effect a tradeoff between time-to-charge and the resulting degradation experienced by the cell, only predictable using PBMs.
- The figure shows a notional charge scheme limited strictly by voltage.
- In this hypothetical case, the actual internal cell electrochemical degradation parameter never reaches its limit $P_{\max}$, indicating that our voltage-limited scheme is conservative.



- However, if we charge instead to the *true* electrochemical-based degradation limit, then the maximum voltage specified on a cell manufacturer's data sheet may be exceeded, but without harm!



- Ch. 6 introduced the idea that lithium-ion battery cells are prone to a wide and complex array of degradation mechanisms.
- No single mechanism is responsible for capacity and power fade; rather, degradation arises from the intricate coupling of many factors.
  - This makes it difficult to isolate specific causes, and research on cell degradation remains active.
- Good news: we don't need complete knowledge of all mechanisms and how they relate to our cell to implement advanced BMS controls.
  - If we are able to model only the most prevalent and damaging of them, we can make big improvements over the state of the art.
- One of the most severe mechanisms if a cell is improperly charged is Li plating, which can ruin both battery life and performance.
- Metallic lithium plates on the electrode surface—impacting performance—and, worse yet, can produce dendrites that can introduce a short-circuit safety concern.
- Since the conditions that bring about this degradation mode are relatively well understood (and modeled), lithium plating provides an excellent starting point for the development of advanced charge-control strategies aiming to improve battery safety and performance.

Lithium plating

- Before we can incorporate degradation concerns into our controls, however, we require mathematical models of their behaviors.
- A model for lithium plating was introduced in ECE 5720 and a model of SEI layer growth was introduced in Ch. 6 of this course.

- In principle, either could be used as the basis for a control strategy aimed at mitigating the effects of these degradation mechanisms.
- For purposes of this chapter, we focus on lithium plating, where the conditions bringing about its formation are particularly easy to specify and thus include in our MPC procedure.
- First, we shall recapture the salient points of the lithium plating model as they relate to the battery control problem.
- Lithium plating occurs due to a side reaction at the negative electrode that may occur during charging; its effect is most acute at cold temperatures where solid diffusion is generally slower.
- During overcharge, lithium is unable to enter the negative electrode and instead plates solid lithium onto the graphite surface.
- This process reduces lithium inventory[3] and inhibits the intercalation of lithium ions into the graphite particles.
- The associated reaction occurs when the solid–electrolyte potential difference at the electrode surface drops below 0 V.
- This condition is more prone to occur at high states-of-charge where the OCP of the negative electrode is relatively low (generally $\lesssim 0.1$ V).
- Additionally, lithium metal can catalyze SEI growth and/or can form a metallic annealing site that promotes growth of metal dendrites, which can penetrate the separator and eventually lead to a potentially dangerous cell short circuit.

[3] The reverse process of *lithium stripping* on discharge can return some lithium inventory. For this development, we assume that the amount of lithium stripped is negligible with respect to the amount plated. A more comprehensive model accounting for lithium stripping could improve the utility of the approach.

- The lithium-plating model from ECE5720 summarized here assumes that the intercalation kinetics of lithium in the negative electrode is expressed using a standard Butler–Volmer equation,[4,5]

$$i_{\mathrm{f}}\left(\tilde{x},t\right)=i_0\left[\exp\left(\frac{\left(1-\alpha\right)F}{RT}\eta(\tilde{x},t)\right)-\exp\left(-\frac{\alpha F}{RT}\eta(\tilde{x},t)\right)\right],$$

driven by the overpotential,

$$\eta\left(\tilde{x},t\right)=\phi_{\mathrm{s}}\left(\tilde{x},t\right)-\phi_{\mathrm{e}}\left(\tilde{x},t\right)-U_{\mathrm{ocp}}\left(\theta_{\mathrm{ss}}\right)-\bar{R}_{\mathrm{f}}i_{\mathrm{f}}(\tilde{x},t),$$

and where i_0 is the exchange current density given by,

$$i_0=\bar{k}_0\left(\theta_{\mathrm{e}}\right)^{1-\alpha}\left(1-\theta_{\mathrm{ss}}\right)^{1-\alpha}\left(\theta_{\mathrm{ss}}\right)^{\alpha}.$$

- Here, $U_{\mathrm{ocp}}(\theta_{\mathrm{ss}})$ gives the equilibrium potential as a function of the normalized solid phase concentration at the particle surface.
- The side-reaction rate of lithium deposition leading to irreversible lithium loss is then given by:

$$i_{\mathrm{sr}}\left(\tilde{x},t\right)=\min\left(0,i_{0,\mathrm{sr}}\left[\exp\left(\frac{\left(1-\alpha_{\mathrm{s}}\right)F}{RT}\eta_{\mathrm{s}}(\tilde{x},t)\right)-\exp\left(-\frac{\alpha_{\mathrm{s}}F}{RT}\eta_{\mathrm{s}}(\tilde{x},t)\right)\right]\right),$$

driven by the side reaction overpotential,

$$\eta_{\mathrm{s}}\left(\tilde{x},t\right)=\phi_{\mathrm{s}}(\tilde{x},t)-\phi_{\mathrm{e}}(\tilde{x},t)-\bar{U}_{\mathrm{s}}-F\bar{R}_{\mathrm{f}}i_{\mathrm{sr}}(\tilde{x},t),\tag{7.1}$$

where the side-reaction exchange current density is $i_{0,\mathrm{sr}}=\bar{k}_{0,\mathrm{sr}}\left(\theta_{\mathrm{e}}\right)^{1-\alpha}$.

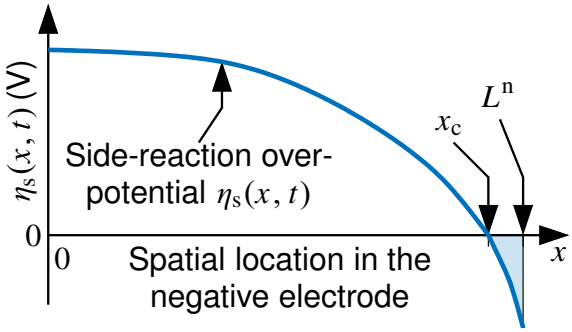
- A key conclusion is the following: *The side reaction occurs only at spatial locations in the negative electrode where* $\eta_{\mathrm{s}}\left(\tilde{x},t\right)<0$.[6]

[4] For simplicity in this treatment, we shall assume a single Butler–Volmer equation for the entire negative electrode; the method can be extended for the MSMR model.

[5] In the following, we omit the negative-electrode superscript "n" for brevity.

[6] While the 0 V value is a generally accepted threshold for the onset of lithium plating, it has been shown to occur at potentials slightly offset from this value in practice, depending in part on the dynamic condition of the cell.

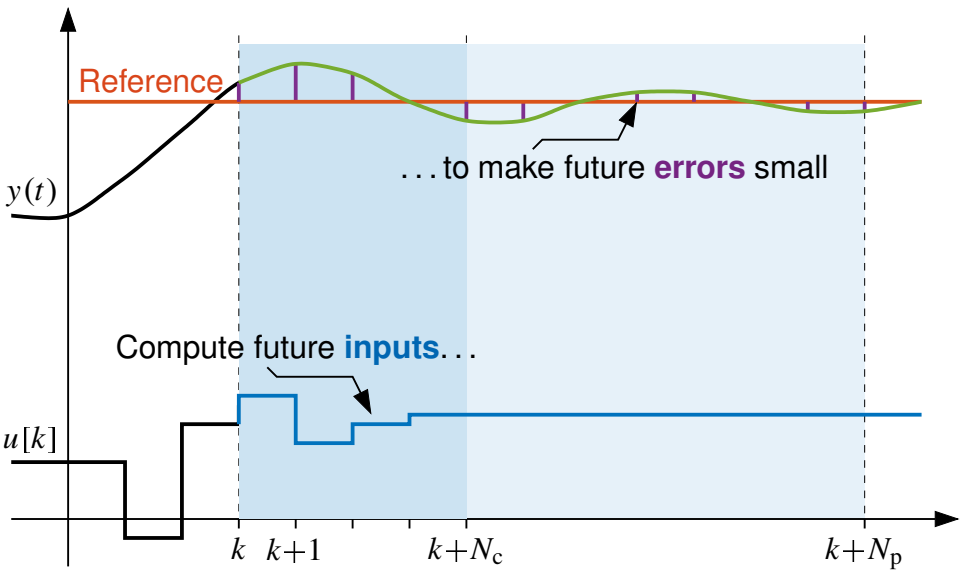
- Discussed in ECE5720, the spatial distribution of overpotential across the negative electrode governs the region in which plating occurs.
- In ECE5720, we also noted that under a persistent charging event, the profile of overpotential with respect to electrode spatial location is monotonic and approximately parabolic in shape, reaching its lowest value at the electrode/separator boundary.
- In addition, during charge, the local value of $\eta_s(\tilde{x}, t)$ decreases over time—again decreasing more quickly near the separator, as shown.



- For the purposes of fast-charge control, we shall be concerned only with the overpotential value computed at the extreme dimension of the negative electrode (i.e., $x = L^n$ or equivalently $\tilde{x} = 1$).
- Computation of the time-dependent overpotential value from the ROM output equation is addressed in Topic 7.7.
- We are nearly in a position to combine our reduced order ideal-cell model with a degradation model for lithium plating to formulate an advanced fast-charge control strategy.
- But first, we introduce a real-time predictive control method, MPC, which we shall use to enforce the conditions required to avoid plating.

7.3: Model predictive control

- MPC belongs to a special class of computer-based control algorithms that make use of an explicit process model to predict the future response of a system and base control actions on those predictions.
- A process model is here defined as a mathematical representation of the process we wish to control (i.e., a model of lithium-ion cell dynamics which includes a description of lithium plating).
- The method employs a look-ahead strategy that attempts to compute a *best* sequence of future inputs that will result in a best output response for the objective at hand.
- Technically speaking, the term *MPC* refers to a family of continuous- or discrete-time control algorithms that adopt this approach; in what follows we shall introduce one such implementation.
- Since our aim is real-time embedded control implemented by an advanced BMS, we consider here only the discrete-time setting and adopt the state-space representation introduced in Ch. 4.
- The figure to the right illustrates the MPC concept—at this point for a generic control problem where we wish to move $y(t)$, a system output, to a reference value.



- We will adapt this idea to the lithium plating fast-charge problem later.

- In the figure, future system behavior is influenced by a sequence of future input variables indicated as a discrete-time sequence of $u[k]$ values, of which only the first is implemented operationally.
- The input sequence is computed by optimizing the future output sequence in some meaningful sense.
- For example, the figure shows an output signal $y(t)$ attempting to reach a desired set-point in the shortest possible time.
- A trivial solution to this problem can be obtained by applying near-infinite input energy thus bringing about near-perfect alignment of our output trajectory with the set-point trace.
- However, reality suggests we must trim our input energy in order to respect the constraints imposed by real-world system capabilities.
- This effects a necessary trade-off between the zero-output-error solution and the realistic input-energy solution depicted in the plot.
- MPC is based on the receding horizon principle, whereby an optimal-control solution is computed for the duration of a finite horizon, yet only the first value of the solution is used in the control.
- The entire solution process is carried out at each discrete sample point, each time taking the first value in the newly computed sequence and discarding the rest.
- This can be viewed notionally as a finite-length window moving forward in time, much like a flashlight beam illuminating a fixed distance ahead as you move steadily forward.
- That is, imagine that you are attempting to navigate a dark room, having only the illumination of a flashlight to help guide you.

- At any point in time, you mentally process what you are able to see and you plan a path forward.
- You take one step of that path and now the flashlight illuminates an area that exposes more of the surroundings than were visible before you took that step.
- So, you discard the rest of your previously planned path and replan a path forward using this new information.
- You take one step of this new planned path and iterate the process.
- Predictive control methods—properly applied—can be extremely effective and deliver excellent overall performance due to their reliance on optimality criteria.
- Nonetheless, the true value of MPC in advanced controls lies in its ability to handle hard constraints in real time during system operation.
- Indeed, MPC forces a dynamic system to conform exactly to multiple constraints imposed on designated problem variables.
  - It is this aspect that makes MPC particularly appealing for the fast-charge problem, where respecting certain parameter limits can be shown to influence both instantaneous and long-term cell performance.
- The next sections will develop the classic MPC algorithm.
  - First we will develop a general solution to the unconstrained problem and then treat the case incorporating hard constraints.

7.4: MPC: The classic algorithm

- We begin by first defining two important time horizons: (1) the *prediction horizon,* N_p, and (2) the *control horizon,* N_c.
- These parameters establish the degrees of freedom available to effect a solution to our control problem.

  PREDICTION HORIZON: How far into the future to predict system behavior.

  CONTROL HORIZON: How long into the future over which to compute optimized control actions.
- For discrete-time systems such as ours, these horizons are denoted by integers signifying a specified number of sampling intervals.
- For example, for a sampling rate of $10\,\mathrm{Hz}$ ($T_\mathrm{s} = 0.1\,\mathrm{s}$) and a prediction horizon of, say, $5\,\mathrm{s}$, we have that $N_\mathrm{p} = 50$.
- The selection of an appropriate prediction horizon length can have an important effect on MPC controller performance, although the design guidance for its selection is largely heuristic, noting that longer prediction horizons generally tend to stabilize controller performance.
- Theoretically, as the $N_\mathrm{p} \to \infty$, the solution to the unconstrained MPC optimization can be shown to approach a linear quadratic optimal solution, which is guaranteed to be stable.
- It would thus seem prudent to select long prediction horizons to leverage this stability property.
- Rough guidelines suggest choosing a value of N_p that captures $80\,\%$ of the dominant system rise time, although in practice many systems perform satisfactorily using prediction horizons well below this value.

- The trade-off relates to resource usage—larger horizons demand larger computational effort.
- In practice, we must find a balance between computational complexity and controller performance, normally requiring trial and error.
- Regarding the control horizon, we normally require $N_\mathrm{c} \leq N_\mathrm{p}$ for simplicity of computation.
  - For systems such as lithium-ion cells that exhibit slow dynamics, controller performance is largely insensitive to choice of N_c.
  - For most implementations, we shall use a control horizon in the range: $2 \leq N_\mathrm{c} \leq 5$.
- Since $N_\mathrm{c} \leq N_\mathrm{p}$, it is interesting to consider what happens with future input variables that lie outside of the control-horizon range.
- In typical implementations, we assume that the control input is held constant at its final value $u[k+N_\mathrm{c}]$ beyond the control horizon and up to the length of the prediction horizon.
- The decision variables in our predictive-control optimization problem are thus the N_c future control choices, $u[k]$, $u[k+1]$,..., $u[k+N_\mathrm{c}]$.[7]
- To see in more detail how MPC works, consider a linearized, discrete-time, state-space model of a general multi-input multi-output system:

$$\begin{aligned} \boldsymbol{x}[k+1] &= \boldsymbol{A}[k]\boldsymbol{x}[k] + \boldsymbol{B}[k]\boldsymbol{u}[k] \\ \boldsymbol{y}[k] &= \boldsymbol{C}[k]\boldsymbol{x}[k] + \boldsymbol{D}[k]\boldsymbol{u}[k], \end{aligned} \tag{7.2}$$

  where $\boldsymbol{x}[k] \in \mathbb{R}^{n\times 1}$, $\boldsymbol{u}[k] \in \mathbb{R}^{m\times 1}$, $\boldsymbol{y}[k] \in \mathbb{R}^{q\times 1}$ are as defined in Ch. 4.

[7] Other strategies exist, including using a linear quadratic regulator solution for the control action in dual-mode MPC, or an exponential control profile.

- $\boldsymbol{A}[k]$, $\boldsymbol{B}[k]$, $\boldsymbol{C}[k]$, and $\boldsymbol{D}[k]$ are (possibly time-varying) matrices defining the state-space model, and index k is the sampling instant.
- Note that it is important to allow for time-varying matrices since the ROM that forms the basis of our linearized model can be expected to vary over the operational space of the battery charging profile.
- To accommodate for dynamics that vary with temperature and SOC in our MPC implementation, we shall incorporate the output blending method introduced in Topic 4.7.
- Classical MPC constructs the state equation in terms of the control-input increment $\Delta\boldsymbol{u}[k+1] = \boldsymbol{u}[k+1] - \boldsymbol{u}[k]$, not control input $\boldsymbol{u}[k]$.
- This formulation implies the inclusion of integral action within the feedback loop and caters directly for eliminating steady-state error.
- To achieve this form, we first define an augmented state vector:

$$\boldsymbol{\chi}[k] = \left[\boldsymbol{x}^T[k] \;\vdots\; \boldsymbol{u}^T[k]\right]^T, \tag{7.3}$$

and rewrite the state equations as:

$$\begin{aligned}\boldsymbol{\chi}[k+1] &= \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \tilde{\boldsymbol{B}}[k]\Delta\boldsymbol{u}[k]\\ \boldsymbol{y}[k] &= \tilde{\boldsymbol{C}}[k]\boldsymbol{\chi}[k],\end{aligned} \tag{7.4}$$

where we formulate the augmented state matrices as:

$$\tilde{\boldsymbol{A}}[k] = \left[\begin{array}{c:c}\boldsymbol{A}[k] & \boldsymbol{B}[k]\\ \hdashline \boldsymbol{0}_{m\times n} & \boldsymbol{I}_{m\times n}\end{array}\right],\; \tilde{\boldsymbol{B}} = \left[\begin{array}{c}\boldsymbol{0}_{n\times m}\\ \hdashline \boldsymbol{I}_{m\times m}\end{array}\right],\; \tilde{\boldsymbol{C}}[k] = \left[\boldsymbol{C}[k] \;\vdots\; \boldsymbol{D}[k]\right].$$

- Note that real-time implementation MPC requires the timely update of the constituent state matrices at the start of each calculation step.
- This in turn requires that we compute the appropriate output-blended solution for $\{\boldsymbol{A}[k], \boldsymbol{B}[k], \boldsymbol{C}[k], \boldsymbol{D}[k]\}$ based on the current state $\boldsymbol{x}[k]$.

- Future values of the augmented state vector can now be obtained by propagating the state equations recursively:

$$\begin{aligned}
\boldsymbol{\chi}[k+1] &= \tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] + \tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k] \\
\boldsymbol{\chi}[k+2] &= \tilde{\boldsymbol{A}}\boldsymbol{\chi}[k+1] + \tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+1] \\
&= \tilde{\boldsymbol{A}}^2\boldsymbol{\chi}[k] + \tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k] + \tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+1] \\
&\vdots \\
\boldsymbol{\chi}[k+N_\mathrm{p}] &= \tilde{\boldsymbol{A}}^{N_\mathrm{p}}\boldsymbol{\chi}[k] + \tilde{\boldsymbol{A}}^{N_\mathrm{p}-1}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k] + \tilde{\boldsymbol{A}}^{N_\mathrm{p}-2}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+1] \\
&\quad \cdots + \tilde{\boldsymbol{A}}^{N_\mathrm{p}-N_\mathrm{c}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+N_\mathrm{c}-1].
\end{aligned}$$

- Similarly, applying the output equation from Eq. (7.4) we can compute future output values as:

$$\begin{aligned}
\boldsymbol{y}[k+1] &= \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k] \\
\boldsymbol{y}[k+2] &= \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k+1] + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+1] \\
&= \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^2\boldsymbol{\chi}[k] + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k] + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+1] \\
&\vdots \\
\boldsymbol{y}[k+N_\mathrm{p}] &= \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{N_\mathrm{p}}\boldsymbol{\chi}[k] + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{N_\mathrm{p}-1}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k] + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{N_\mathrm{p}-2}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+1] \\
&\quad \cdots + \tilde{\boldsymbol{C}}\tilde{\boldsymbol{A}}^{N_\mathrm{p}-N_\mathrm{c}}\tilde{\boldsymbol{B}}\Delta\boldsymbol{u}[k+N_\mathrm{c}-1].
\end{aligned}$$

- At this point, we define special vectors containing the set of future outputs and future control input increments, respectively, as:

$$\begin{aligned}
\underrightarrow{\boldsymbol{Y}}[k+1] &= \begin{bmatrix} \boldsymbol{y}^T[k+1] & \boldsymbol{y}^T[k+2] & \cdots & \boldsymbol{y}^T[k+N_\mathrm{p}+1] \end{bmatrix}^T \\
\underrightarrow{\Delta\boldsymbol{U}}[k+1] &= \begin{bmatrix} \Delta\boldsymbol{u}^T[k+1] & \Delta\boldsymbol{u}^T[k+2] & \cdots & \Delta\boldsymbol{u}^T[k+N_\mathrm{c}] \end{bmatrix}^T,
\end{aligned}$$

where we have introduced the underscore arrow notation to denote sequences of future variables.

- The output prediction equation can now be written more compactly:

$$\underrightarrow{\boldsymbol{Y}}[k+1] = \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \boldsymbol{G}[k]\underrightarrow{\Delta\boldsymbol{U}}[k+1], \tag{7.5}$$

where matrices $\boldsymbol{\Phi}[k] \in \mathbb{R}^{(N_\mathrm{p}\cdot q)\times(n+m)}$ and $\boldsymbol{G}[k] \in \mathbb{R}^{(N_\mathrm{p}\cdot q)\times(N_\mathrm{c}\cdot m)}$ are:

$$\boldsymbol{\Phi}[k] = \begin{bmatrix} \tilde{\boldsymbol{C}}[k] \\ \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{A}}[k] \\ \vdots \\ \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{A}}^{n_p-1}[k] \end{bmatrix}$$

$$\boldsymbol{G}[k] = \begin{bmatrix} \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{B}}[k] & 0 & \cdots & 0 \\ \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{A}}[k]\tilde{\boldsymbol{B}}[k] & \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{B}}[k] & \cdots & 0 \\ \vdots & \vdots & \ddots & \\ \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{A}}^{N_\mathrm{p}-1}[k]\tilde{\boldsymbol{B}}[k] & \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{A}}^{N_\mathrm{p}-2}[k]\tilde{\boldsymbol{B}}[k] & \cdots & \tilde{\boldsymbol{C}}[k]\tilde{\boldsymbol{A}}^{N_\mathrm{p}-N_\mathrm{c}}[k]\tilde{\boldsymbol{B}}[k] \end{bmatrix},$$

and compute the free and forced response components, respectively, of the predicted output.

- Since the original dynamic system is nonstrictly causal (i.e., it has a nonzero D-term), this formulation assumes that at time sample k, the control input $\boldsymbol{u}[k]$ was computed at previous time sample $k-1$.
- This information, together with the current state vector value $\boldsymbol{x}[k]$, is used to compute the set of future input increments $\underrightarrow{\Delta\boldsymbol{U}}[k+1]$ from which the next control input will be computed, namely $\boldsymbol{u}[k+1]$.[8] This process is carried out according to the receding-horizon principle.

[8] This is a nontrivial point for the battery application. Since the battery exhibits an internal ohmic resistance, there is a near-instantaneous voltage change with the application of current. This implies the presence of a direct feedthrough term (D-matrix) in the state-space model. Most classic forms of MPC neglect the D-matrix; however, in our development we adopt the modification of Ordys and Pike to cater for this term.

- Now that we have an expression for the predicted output in terms of the future input decision variables, we next fashion a cost function that penalizes the error between future values of the output (i.e., vector $\underrightarrow{\boldsymbol{Y}}[k+1]$) and future values of a reference.
- For convenience, define the $\left(q \cdot N_{\mathrm{p}}\right) \times 1$ reference vector:

$$\boldsymbol{R}[k+1] = \left[\boldsymbol{r}^T[k+1],\ \boldsymbol{r}^T[k+2], \ldots,\ \boldsymbol{r}^T[k+N_{\mathrm{p}}+1]\right]^T,$$

  where each of the elements $\boldsymbol{r}[j]$ are $q \times 1$ vectors stipulating a references value for each of the q outputs.
- Recall that for realizability (i.e., to avoid the trivial solution), the cost function must place a nonzero penalty on the energy in the input, in this case defined by the magnitude of the elements of the vector of future input increments, $\underrightarrow{\boldsymbol{\Delta U}}[k+1]$.
- The scalar cost function to be minimized can be written as a quadratic form:

$$J[k] = \left(\underrightarrow{\boldsymbol{R}}[k+1] - \underrightarrow{\boldsymbol{Y}}[k+1]\right)^T \left(\underrightarrow{\boldsymbol{R}}[k+1] - \underrightarrow{\boldsymbol{Y}}[k+1]\right) + \underrightarrow{\Delta \boldsymbol{U}}^T[k+1]\bar{\boldsymbol{R}}\underrightarrow{\Delta \boldsymbol{U}}[k+1],$$

  where the vector multiplications bring about a nonnegative sum-of-squares cost measure.
- Here, $\bar{\boldsymbol{R}}$ denotes a weighting matrix that trades off the penalty on the control input increment energy with that on the predicted output error.
- In many cases, there is little justification for differentially weighting the individual input sequence elements; so we normally take $\bar{\boldsymbol{R}} = \rho \boldsymbol{I}$ where ρ is a scalar weight and $\boldsymbol{I}$ is an identity matrix of square dimension $N_{\mathrm{c}} \cdot m$.

- In this context, weighting element ρ serves as another tuning parameter in the optimization problem.
- We now use the prediction equation, Eq. (7.5), and substitute for $\underrightarrow{\boldsymbol{Y}}[k+1]$ to rewrite the cost function as:

$$\begin{aligned} J[k] = &\left(\underrightarrow{\boldsymbol{R}}[k+1] - \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k]\right)^T \left(\underrightarrow{\boldsymbol{R}}[k+1] - \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k]\right) \\ &- 2\left(\underrightarrow{\boldsymbol{R}}[k+1] - \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k]\right)^T \boldsymbol{G}[k]\underrightarrow{\Delta\boldsymbol{U}}[k+1] \\ &+ \underrightarrow{\Delta\boldsymbol{U}}^T[k+1]\left(\boldsymbol{G}^T[k]\boldsymbol{G}[k] + \rho I\right)\underrightarrow{\Delta\boldsymbol{U}}[k+1], \end{aligned} \tag{7.6}$$

  where $J[k]$ is an explicit function of the unknown future input sequence $\underrightarrow{\Delta\boldsymbol{U}}[k+1]$; all other values in the expression are updated at each sample time.
- We shall first develop the solution for the case where all problem variables are unconstrained.
- We have previously noted that most meaningful predictive-control problems include constraints; in fact, this is the principal reason for using a real-time model-predictive control strategy in the first place.
- Nevertheless, during operation, the algorithm will often traverse regions of the functional space where no constraints are active.
- In these instances, the optimal unconstrained solution is the correct solution.

7.5: Optimal MPC solution: unconstrained case

- For notational simplicity in what follows, we shall drop dependence on the sampling index k except where needed for clarity.
- Our task is now to find the future input sequence $\underrightarrow{\boldsymbol{\Delta U}}$ that minimizes our cost function J as defined by Eq. (7.6), repeated here:

$$J = \left(\underrightarrow{\boldsymbol{R}} - \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}\right)^T \left(\underrightarrow{\boldsymbol{R}} - \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}\right) - 2\left(\underrightarrow{\boldsymbol{R}} - \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}\right)^T \boldsymbol{G}\underrightarrow{\Delta \boldsymbol{U}} + \underrightarrow{\Delta \boldsymbol{U}}^T \left(\boldsymbol{G}^T\boldsymbol{G} + \rho I\right)\underrightarrow{\Delta \boldsymbol{U}}.$$

- Since J is a quadratic form, a global optimum is found as the solution to the stationarity condition:

$$\frac{\partial J}{\partial \underrightarrow{\boldsymbol{\Delta U}}} = -2\boldsymbol{G}^T\left(\underrightarrow{\boldsymbol{R}} - \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k]\right) + 2\left(\boldsymbol{G}^T\boldsymbol{G} + \rho\boldsymbol{I}\right)\underrightarrow{\boldsymbol{\Delta U}} = 0. \tag{7.7}$$

- By construction of the quadratic form, matrix $\boldsymbol{G}^T\boldsymbol{G} + \rho\boldsymbol{I}$ is positive definite; thus, the solution to the stationarity condition is a unique minimizing one.
- Solving Eq. (7.7) for $\underrightarrow{\boldsymbol{\Delta U}}$ yields the optimal (unconstrained) control sequence as the $N_c \cdot m \times 1$ vector:

$$\underrightarrow{\boldsymbol{\Delta U}}^* = \left(\boldsymbol{G}^T\boldsymbol{G} + \rho\boldsymbol{I}\right)^{-1}\boldsymbol{G}^T\left(\underrightarrow{\boldsymbol{R}} - \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k]\right). \tag{7.8}$$

- Note that by application of the receding-horizon principle, only the first element of the optimal sequence $\underrightarrow{\boldsymbol{\Delta U}}^*$ is implemented, namely, $\boldsymbol{\Delta u}^*[k+1]$; the process starts over and repeats at the next time sample.
- Practitioners of control theory will recognize Eq. (7.8) as a time-varying full state feedback controller, functionally dependent on

the augmented state vector $\boldsymbol{\chi}[k]$ as:

$$\begin{aligned}\boldsymbol{\chi}[k+1] &= \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \tilde{\boldsymbol{B}}[k]\boldsymbol{\Delta u}[k+1] \\ &= \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \tilde{\boldsymbol{B}}[k]\left(\boldsymbol{G}^T[k]\boldsymbol{G}[k] + \rho\boldsymbol{I}\right)^{-1}\boldsymbol{G}^T[k]r[k] \\ &\quad - \tilde{\boldsymbol{B}}[k]\left(\boldsymbol{G}^T[k]\boldsymbol{G}[k] + \rho\boldsymbol{I}\right)^{-1}\boldsymbol{G}^T[k]\boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] \\ &= \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] - \tilde{\boldsymbol{B}}[k]\boldsymbol{K}[k]\boldsymbol{\chi}[k] \\ &\quad + \tilde{\boldsymbol{B}}[k]\left(\boldsymbol{G}^T[k]\boldsymbol{G}[k] + \rho\boldsymbol{I}\right)^{-1}\boldsymbol{G}^T[k]r[k],\end{aligned}$$

where the time-varying state-feedback gain is given by,

$$\boldsymbol{K}[k] = \left(\boldsymbol{G}^T[k]\boldsymbol{G}[k] + \rho\boldsymbol{I}\right)^{-1}\boldsymbol{G}^T[k]\boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k].$$

- This expression implies that fundamentally, unconstrained MPC implements a closed-loop feedback system, and furthermore, for systems where the state-space model is assumed to be constant over all time, stability may be assured by careful selection of tuning parameters that give a stable set of closed-loop eigenvalues for the constant matrix $\tilde{\boldsymbol{A}} - \tilde{\boldsymbol{B}}\boldsymbol{K}$.
  - Stability assurances for the time-varying case are less straightforward.
- So far, we have not considered applying constraints to the system state or output.
- When problem constraints are active, however, the above solution is no longer valid and we therefore must turn to alternative strategies to find the correct constrained solution.

7.6: Optimal MPC solution: constrained case

- Practical control problems impose constraints to reflect real-world system operation.
- Such constraints may be imposed on three classes of parameters: (1) control inputs; (2) system outputs; and (3) state variables.
- For the fast-charge problem, our control input is the applied current (i.e., $\boldsymbol{u}[k] = i_{\text{app}}[k]$),[9] implying that $m = 1$ and that $u[k]$ is a scalar.
- Hard constraints on input current are normally stipulated by the nature of the charging infrastructure and are imposed accordingly.
  - That is, charging electronics will have a maximum rated current that is independent of cell performance limits and MPC must be informed of these hard limits on $u[k]$.
- Output variables can be further subdivided into both measurable and nonmeasurable outputs of interest.
  - For the battery, voltage and surface temperature are measurable outputs since their values are readily obtained with sensors.
  - The class of unmeasurable outputs are perhaps of greater interest from an advanced-controls point of view as they may harbor information directly related to phenomena that can harm a cell.
  - These might include lithium concentration ratios or potentials at various locations in the cell, for example.

[9] Ambient or flow temperature may also be used as a control input to regulate cell core temperature. In this case, we would adopt a 2×1 input vector $\boldsymbol{u}[k] = \begin{bmatrix} i_{\text{app}}[k] & T_{\text{flow}}[k] \end{bmatrix}^T$ and implement a multivariable form of MPC. This is beyond the scope of this chapter and will not be addressed here.

  - Enforcing constraints on such indicators can address true causes of degradation and will form the basis of our control approach.
  - Clearly, unmeasurable outputs cannot be sensed, but must be estimated using techniques such as discussed in Ch. 5.
- The final class of constraints deals with the dynamic state variables.
  - Depending on the underlying model structure, state variables may or may not have a meaningful physical interpretation.
  - In cases where they do, it is always possible to create a corresponding output variable by proper selection of an output $\boldsymbol{C}$-matrix, in which case they are handled in exactly the same manner as the unmeasurable outputs above.

Quadratic programming

- The constrained optimization result can be found by solving a quadratic programming (QP) problem, defined as one in which the cost function is quadratic and the constraint functions are all linear.
- QP problems can be solved systematically in a finite number of steps and have been shown to be robust and computationally efficient.
- Although quadratic programming routines are generally available (e.g., MATLAB's `quadprog`), we shall present an implementable algorithm herein; this provides the ability to access the code to control error handling and make any necessary algorithm modifications.
- For generality, and to be consistent with prevailing literature, we denote the objective function J in the QP problem in the general form:

$$J[k] = \frac{1}{2}\underrightarrow{\boldsymbol{\Delta U}}^T[k+1]\boldsymbol{H}[k]\underrightarrow{\boldsymbol{\Delta U}}[k+1] + \boldsymbol{g}^T[k]\underrightarrow{\boldsymbol{\Delta U}}[k+1] + f[k], \quad (7.9)$$

where the decision vector $\underrightarrow{\boldsymbol{\Delta U}}[k+1]$ contains the degrees of freedom for the solution.

- Symmetric Hessian matrix $\boldsymbol{H}[k]$, vector $\boldsymbol{g}[k]$, and scalar $f[k]$ are compatible with the objective function defined in Eq. (7.6) where:

$$\boldsymbol{H}[k] = 2\left(\boldsymbol{G}^T[k]\boldsymbol{G}[k] + \rho\boldsymbol{I}\right)$$

$$\boldsymbol{g}[k] = -2\left(\underrightarrow{\boldsymbol{R}}[k+1] - \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k]\right)^T \boldsymbol{G}[k]$$

$$f[k] = \left(\underrightarrow{\boldsymbol{R}}[k+1] - \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k]\right)^T \left(\underrightarrow{\boldsymbol{R}}[k+1] - \boldsymbol{\Phi}[k]\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k]\right).$$

- Note: when $\boldsymbol{H}[k]$ is positive definite $\boldsymbol{x}^*[k]$ is a unique global minimizer.
- Additionally, we shall stack the linear constraint set that we wish to impose on the solution into a convenient linear algebraic inequality as:

$$\boldsymbol{M}\underrightarrow{\boldsymbol{\Delta U}}[k+1] \leq \boldsymbol{\gamma}[k], \tag{7.10}$$

where each constraint equation can be expressed row-wise as:

$$\boldsymbol{m}_i^T \underrightarrow{\boldsymbol{\Delta U}}[k+1] \leq \gamma_i[k].$$

- Here, $\boldsymbol{m}_i^T$ are the rows of $\boldsymbol{M}$ and $\gamma_i[k]$ is the corresponding ith element of $\boldsymbol{\gamma}[k]$.
- For general application to the battery control problem, we consider constraints imposed on: (1) the input, (2) the measured output, and (3) designated unmeasured outputs.
- In what follows, we shall consider a scalar-valued input variable and a vector-valued output variable.
- We first consider the simplest case: constraints imposed on the input increment, $\Delta u[k]$.

- Enforcing a constraint on the input increment effectively limits the rate at which the input can be changed. Defining hard bounds gives:

$$\Delta u_{\min}[k] \leq \Delta u[k] \leq \Delta u_{\max}[k].$$

- In classic MPC, input constraints are generally applied to all future input increments, however this choice is left up to the controls designer as the mathematical approach is unaffected by this choice.
- We next build up a matrix representation that stacks the individual constraint equations imposed at each future sample point as:

$$\left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \\ \hdashline -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{array}\right] \left[\begin{array}{c} \Delta u[k+1] \\ \Delta u[k+2] \\ \vdots \\ \Delta u[k+N_{\rm c}] \end{array}\right] \leq \left[\begin{array}{c} \Delta u_{\max}[k] \\ \Delta u_{\max}[k] \\ \vdots \\ \Delta u_{\max}[k] \\ \hdashline -\Delta u_{\min}[k] \\ -\Delta u_{\min}[k] \\ \vdots \\ -\Delta u_{\min}[k] \end{array}\right],$$

  where we have flipped the sign on the lower half in order to pose the minimum bound as a left-hand inequality.
- This can be written in more compact form as:

$$\left[\begin{array}{c} \boldsymbol{I}_{N_{\rm c}} \\ -\boldsymbol{I}_{N_{\rm c}} \end{array}\right] \underrightarrow{\boldsymbol{\Delta U}}[k+1] \leq \left[\begin{array}{c} \boldsymbol{e}_{N_{\rm c}} \cdot \Delta u_{\max} \\ -\boldsymbol{e}_{N_{\rm c}} \cdot \Delta u_{\min} \end{array}\right],$$

  where $\boldsymbol{I}_{N_{\rm c}}$ denotes an $N_{\rm c}$-dimensioned identity matrix and $\boldsymbol{e}_{N_c} \in \mathbb{R}^{N_c \times 1}$ is a vector of ones.
- This results in a form identical to Eq. (7.10) above.

- Next, we consider constraints imposed directly on the input magnitude $u[k]$ by defining the bounds: $u_{\min}[k] \leq u[k] \leq u_{\max}[k]$.
  - Bounds $u_{\min}[k]$ and $u_{\max}[k]$ may, in general, be time-varying.
- For simplicity of the present development, we shall consider the hard bounds on $u[k]$ to be constant over the control horizon.
- We proceed as before to build a vector-valued inequality by stacking the individual constraint equations for each future sample point as:

$$\left[\begin{array}{cccc} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \\ \hdashline -1 & 0 & \cdots & 0 \\ -1 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & -1 \end{array}\right] \left[\begin{array}{c} \Delta u[k+1] \\ \Delta u[k+2] \\ \vdots \\ \Delta u[k+N_c] \end{array}\right] \leq \left[\begin{array}{c} u_{\max} - u[k] \\ u_{\max} - u[k] \\ \vdots \\ u_{\max} - u[k] \\ \hdashline -u_{\min} + u[k] \\ -u_{\min} + u[k] \\ \vdots \\ -u_{\min} + u[k] \end{array}\right],$$

  or more compactly,

$$\left[\begin{array}{c} \boldsymbol{E}_1 \\ -\boldsymbol{E}_1 \end{array}\right] \underrightarrow{\boldsymbol{\Delta U}} \leq \left[\begin{array}{c} \boldsymbol{e}_{N_c} \cdot u_{\max} \\ -\boldsymbol{e}_{N_c} \cdot u_{\min} \end{array}\right] + \left[\begin{array}{c} -\boldsymbol{e}_{N_c} \\ \boldsymbol{e}_{N_c} \end{array}\right] u[k],$$

  where we've introduced the matrix $\boldsymbol{E}_1 \in \mathbb{R}^{N_c \times N_c}$ as a lower triangular matrix of ones.
- Note that in this formulation, respective values of $\boldsymbol{\gamma}[k]$ are updated each time step using the previously computed input magnitude.
- Output constraints can be addressed by starting with the matrix form of the output equation:

$$\underrightarrow{\boldsymbol{Y}}[k+1] = \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] + \boldsymbol{G}\underrightarrow{\boldsymbol{\Delta U}}[k+1],$$

where we fashion bounds as: $y_{\min} \leq y[k] \leq y_{\max}$.

- We can express this in equivalent vector form as:

$$\boldsymbol{e}_{N_p} y_{\min} \leq \underrightarrow{\boldsymbol{Y}}[k+1] \leq \boldsymbol{e}_{N_p} y_{\max},$$

and proceeding as above we obtain the constraint inequality:

$$\begin{bmatrix} \boldsymbol{G} \\ -\boldsymbol{G} \end{bmatrix} \underrightarrow{\boldsymbol{\Delta U}}[k+1] \leq \begin{bmatrix} \boldsymbol{e}_{N_p} y_{\max} \\ -\boldsymbol{e}_{N_p} y_{\min} \end{bmatrix} - \begin{bmatrix} \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] \\ -\boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] \end{bmatrix},$$

where here $\boldsymbol{e}_{N_\mathrm{p}}$ now represents an $(N_\mathrm{p}+1)$-length vector of ones.

- Note that both constraint inequalities introduced above have time-varying elements on their right-hand sides, requiring that we update their values at each sampling point.
- Also, as mentioned above, we may construct constraint inequalities for state variables by defining them as special outputs. For example,

$$y_z[k] = \boldsymbol{C}_z \boldsymbol{x}[k],$$

by constructing the appropriate output matrix, $\boldsymbol{C}_z$.

- Then, we simply proceed exactly as we did for output constraints.
- Both outputs and state variables can be used to enforce hard constraints on other problem variables of interest (e.g., cell temperature) that are available in the selected model.
- In most practical control problems, we will have a combination of constraints imposed on different problem variables.
- This case is handled by stacking the relevant constraint matrices and vectors into a combined form.

- For example, if our problem imposed both input-magnitude and output constraints, we would assemble these as:

$$\begin{bmatrix} \boldsymbol{E}_1 \\ -\boldsymbol{E}_1 \\ \boldsymbol{G} \\ -\boldsymbol{G} \end{bmatrix} \underrightarrow{\boldsymbol{\Delta U}} \leq \begin{bmatrix} \boldsymbol{e}_{N_c} \cdot u_{\max} \\ \boldsymbol{e}_{N_c} \cdot u_{\min} \\ \boldsymbol{e}_{N_p} y_{\max} \\ -\boldsymbol{e}_{N_p} y_{\min} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{e}_{N_c} \\ \boldsymbol{e}_{N_c} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} u[k] + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] \\ -\boldsymbol{\Phi}\tilde{\boldsymbol{A}}\boldsymbol{\chi}[k] \end{bmatrix},$$

  where by appropriate definition, we arrive at the form of Eq. (7.10),

$$\boldsymbol{M}\boldsymbol{\Delta U} \leq \boldsymbol{\gamma}.$$

- Summarizing to this point, we have seen that the fundamental optimization problem addressed by MPC can be written as:

$$\min_{\underrightarrow{\boldsymbol{\Delta U}}} J[k]$$

  subject to linear constraints, $\boldsymbol{M}\underrightarrow{\boldsymbol{\Delta U}} \leq \boldsymbol{\gamma}$.

- It turns out that solving this constrained-optimization problem in real time can be made significantly easier by solving its corresponding dual form instead.

- A dual method can be used in a quadratic-optimization problem to identify constraints that are not active in optimization so that they can be systematically eliminated from the solution process.

- Assuming feasibility (i.e., that there exists a candidate solution $\boldsymbol{x}$ such that $\boldsymbol{M}\boldsymbol{x} < \boldsymbol{\gamma}$), then the primal problem is equivalent to:

$$\max_{\boldsymbol{\lambda} \geq \boldsymbol{0}} \min_{\boldsymbol{x}} \left[\frac{1}{2}\boldsymbol{x}^T \boldsymbol{H}\boldsymbol{x} + \boldsymbol{g}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{M}\boldsymbol{x} - \boldsymbol{\gamma}) \right],$$

  where here we have adjoined the constraints to the cost function with a vector of Lagrange multipliers, $\boldsymbol{\lambda}$.[10]

[10] Note that the vector inequality $\lambda \geq 0$ means that individual λ_i of λ must satisfy $\lambda_i \geq 0, \forall i$.

- The minimum of this augmented cost function over $\boldsymbol{x}$ is now unconstrained and is attained by:

$$\boldsymbol{x}^{\mathrm{o}} = -\boldsymbol{H}^{-1}(\boldsymbol{g} + \boldsymbol{M}^T\boldsymbol{\lambda}).$$

- Substituting $\boldsymbol{x}^{\mathrm{o}}$ into the above expression gives the dual problem (see App. 7.b for the complete derivation):

$$\max_{\boldsymbol{\lambda}\geq\boldsymbol{0}} \left(-\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{P}\boldsymbol{\lambda} - \boldsymbol{\lambda}^T\boldsymbol{k} - \frac{1}{2}\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g} \right),$$

where:

$$\boldsymbol{P} = \boldsymbol{M}\boldsymbol{H}^{-1}\boldsymbol{M}^T$$

$$\boldsymbol{k} = \boldsymbol{\gamma} + \boldsymbol{M}\boldsymbol{H}^{-1}\boldsymbol{g}.$$

- So, the dual problem turns out to be another quadratic-programming problem, only now with $\boldsymbol{\lambda}$ as the decision variable instead of $\boldsymbol{x}$ and with the original problem constraints baked in to the dual formulation.
- Switching sign, the dual problem is equivalent to the minimization:

$$\min_{\boldsymbol{\lambda}\geq\boldsymbol{0}} \left(\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{P}\boldsymbol{\lambda} + \boldsymbol{\lambda}^T\boldsymbol{k} + \frac{1}{2}\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g} \right).$$

- The solution to this new quadratic programming problem can be accomplished using an iterative routine developed by Clifford Hildreth.
- Hildreth's algorithm employs a primal-dual active set method to solve a linear system of equations using element-by-element reduction, which avoids matrix inversion.
- The algorithm is guaranteed to converge under some mild conditions and has been shown to be robust in practice.
- Details of the Hildreth method are presented in Apps. 7.b and 7.c.

7.7: MPC fast charge of lithium-ion cells

- Application of MPC to the fast-charge problem ultimately utilizes the output-blended HRA ROM from Ch. 4 together with state estimates from Ch. 5.
- Here, though, we simplify by considering a single setpoint for which the model can be written as:

$$\begin{aligned}
\boldsymbol{x}[k+1] &= \boldsymbol{A}[k]\boldsymbol{x}[k] + \boldsymbol{B}[k]i_{\text{app}}[k] \\
\tilde{\boldsymbol{y}}[k] &= \boldsymbol{C}[k]x[k] + \boldsymbol{D}[k]i_{\text{app}}[k] \\
\boldsymbol{y}[k] &= g(\tilde{\boldsymbol{y}}[k], i_{\text{app}}[k]) \\
v[k] &= g_v(\tilde{\boldsymbol{y}}[k], i_{\text{app}}[k]).
\end{aligned}$$

- In this formulation, vector $\tilde{\boldsymbol{y}}[k]$ is the linearized version of the electrochemical variables of interest emerging directly from the ROM.
- We use this linearized model to generate the predictions required by MPC and as the basis for the Kalman filter that will ultimately provide estimates of the cell's internal electrochemical state.
- Importantly, we shall use the estimated variables supplied by the model to enforce hard limits during operation.
- Note that although the ROM is configured in linear state-space form, key outputs of interest are *not* linear functions of the state variables and must be reconfigured to obtain linear approximate forms.
- In order to construct a more accurate simulation of battery behavior, nonlinear corrections are applied; these are computed via the function, $g(\cdot)$ to give corrected output values in $\boldsymbol{y}[k]$.

- Cell terminal voltage is computed as a nonlinear function of electrochemical parameters, $g_v(\cdot)$.
- Leveraging Ch. 4, we now recap SOC and cell-voltage calculations.

State-of-Charge (SOC)

- Cell SOC defined in terms of the negative electrode is given by:

$$z[k] = \frac{\theta^{\mathrm{n}}_{\mathrm{s,avg}}[k] - \theta^{\mathrm{n}}_{0}}{\theta^{\mathrm{n}}_{100} - \theta^{\mathrm{n}}_{0}},$$

  where $\theta^{\mathrm{n}}_{100}$ and θ^{n}_{0} give the electrode stoichiometry at 100 % and 0 % SOC, respectively for the negative electrode.

- Additionally, if the cell model has no double layer effect,[11] then the electrode state of charge $\theta^{\mathrm{n}}_{\mathrm{s,avg}}$ can be computed as:

$$\theta^{\mathrm{n}}_{\mathrm{s,avg}}[k] = \theta^{\mathrm{n}}_{\mathrm{s},0} + \theta^{\mathrm{res0,n}}_{\mathrm{ss}} x_0[k], \tag{7.11}$$

  where $\theta^{\mathrm{n}}_{\mathrm{s},0}$ is the initial stoichiometry, $x_0[k]$ is the integrator state, and $\theta^{\mathrm{res0,n}}_{\mathrm{ss}}$ is the integrator residue for the appropriate TF.

- Initial electrode stoichiometry is $\theta^{\mathrm{n}}_{\mathrm{s},0} = \theta^{\mathrm{n}}_{0} + z_0\left(\theta^{\mathrm{n}}_{100} - \theta^{\mathrm{n}}_{0}\right)$,where z_0 is the initial cell SOC.

- Performing appropriate substitutions, we can express the SOC as:

$$z[k] = \boldsymbol{C}_z[k]\boldsymbol{x}[k] + z_0,$$

  where we have constructed $C_z[k] = \begin{bmatrix} 0 & 0 & \cdots & c_{n_x}[k] \end{bmatrix}$ where $c_{n_x}[k]$ is a function of $\theta^{\mathrm{n}}_{\mathrm{s,avg}}[k]$ and multiplies integrator state $x_0[k]$.[12]

[11] During fast-charge, the double-layer charges to a steady-state level *very* quickly, and since its capacity is negligible with respect to the capacity of the electrode, Eq. (7.11) is still approximately true and can be considered to be accurate.

[12] As before, we assume that the integrator state occupies the last position in $\boldsymbol{x}[k]$.

Cell voltage

- Cell terminal voltage is a somewhat complicated function of a subset of the cell variables contained in the output vector $\boldsymbol{y}_k$:

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

- By linearizing, making appropriate substitutions, and rearranging terms, we can reformulate this equation as an affine relationship:

$$v[k] = C_v[k]\boldsymbol{x}[k] + D_v[k] i_{\mathrm{app}} + b_v[k].$$

Lithium-plating side-reaction overpotential

- From Topic 7.2, we recall that the negative-electrode side-reaction overpotential, η_{s}, can serve as an indicator of the onset of Li plating.
- Starting from Eq. (7.1), we make the following approximation:

$$\eta_{\mathrm{s}}[\tilde{x},k] \approx \phi_{\mathrm{s}}[\tilde{x},k] - \phi_{\mathrm{e}}[\tilde{x},k] = \phi_{\mathrm{s\text{-}e}}[\tilde{x},k], \tag{7.12}$$

  where we have further assumed $\bar{U}_{\mathrm{s}} = 0$, as is the case for Li plating.
- Therefore, for purposes of advanced physics-based MPC, we can approximate the condition for avoiding Li plating as, $\phi_{\mathrm{s\text{-}e}}[\tilde{x},k] \geq 0$.
- ROMs can be used to produce values for the electrochemical variables required to monitor for the lithium plating condition.[13]
- In terms of ROM variables, Ch. 4 showed that:

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

[13] Note here that the ROMs are actually supplied to the EKF to compute estimates of the required variables that will then be introduced to MPC for constraint handling.

- A state-space representation can be written as,

$$\phi_{\text{s-e}}[k, \tilde{x}] = \boldsymbol{C}_{\phi_{\text{s-e}}}[k]\boldsymbol{x}[k] + U_{\text{ocp}}[\theta^{\text{n}}_{\text{s,avg}}[k]], \tag{7.13}$$

  where we have introduced the output matrix $\boldsymbol{C}_{\phi_{\text{s-e}}}$ which linearly combines the elements of the state vector to generate $[\tilde{\phi}_{\text{s-e}}[\tilde{x}, k]]^*$.[14]

- Substituting Eq. (7.13) into Eq. (7.12), and renaming the output matrix $\boldsymbol{C}_{\eta_{\text{s}}} = \boldsymbol{C}_{\phi_{\text{s-e}}}$ gives the expression for the side-reaction overpotential we'll use in the fast-charge algorithm:

$$\eta_{\text{s}}[k, \tilde{x}] = \boldsymbol{C}_{\eta_{\text{s}}}[k]\boldsymbol{x}[k] + U_{\text{ocp}}[\theta^{\text{n}}_{\text{s,avg}}[k]].$$

State-space prediction model

- Combining the results from above, we can now define an augmented state-space prediction model for the physics-based ROM as:

$$\begin{aligned}
\boldsymbol{\chi}[k+1] &= \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \tilde{\boldsymbol{B}}[k]\triangle i_{\text{app}}[k+1] \\
v[k] &= \tilde{\boldsymbol{C}}_v[k]\boldsymbol{\chi}[k] + b_v[k] \\
z[k] &= \tilde{\boldsymbol{C}}_z[k]\boldsymbol{\chi}[k] + z_0 \\
\eta_s[k] &= \tilde{\boldsymbol{C}}_{\eta_s}[k]\boldsymbol{\chi}[k] + U_{\text{ocp}}[\theta^{\text{n}}_{\text{s,avg}}[k]],
\end{aligned}$$

  where the augmented state vector is given by,

$$\boldsymbol{\chi}[k] = \begin{bmatrix} \boldsymbol{x}^T[k] \\ i_{\text{app}}[k] \end{bmatrix}.$$

- Appropriate MPC prediction matrices $\boldsymbol{\Phi}$ and $\boldsymbol{G}$ are built according to the procedure in Topic 7.6, enabling predicted sequences of SOC,

[14] In App. 5.b, these were denoted as $[\boldsymbol{C}^{\phi_{\text{s-e}}[\tilde{x}]}]$. We change the notation slightly here for a more compact presentation.

cell voltage, and side-reaction overpotential to be computed as:

$$\underset{\rightarrow}{\boldsymbol{z}}[k+1] = \boldsymbol{\Phi}_z \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \boldsymbol{G}_z \boldsymbol{\Delta i}_{\text{app}}[k+1] + \boldsymbol{e}_{N_p} z_0$$

$$\underset{\rightarrow}{\boldsymbol{v}}[k+1] = \boldsymbol{\Phi}_v \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \boldsymbol{G}_v \boldsymbol{\Delta i}_{\text{app}}[k+1] + \boldsymbol{e}_{N_p} b_v[k]$$

$$\underset{\rightarrow}{\boldsymbol{\eta}_s}[k+1] = \boldsymbol{\Phi}_{\eta_s} \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \boldsymbol{G}_{\eta_s} \boldsymbol{\Delta i}_{\text{app}}[k+1] + \boldsymbol{e}_{N_p} U_{\text{ocp}}[\theta^{\text{n}}_{\text{s,avg}}[k]].$$

- The construction of the $\boldsymbol{\Phi}$ and $\boldsymbol{G}$ matrices utilizes common $\tilde{\boldsymbol{A}}[k]$ and $\tilde{\boldsymbol{B}}[k]$ state and input matrices, but separate output $\tilde{\boldsymbol{C}}$ matrices corresponding to the output equations for each required variable.
- Now that we have the prediction model in place, we next turn toward constructing the needed constraint matrices.

Defining constraints on PBM variables

- The linear-constraint inequality $\boldsymbol{M}\underset{\rightarrow}{\boldsymbol{\Delta i}}_{\text{app}}[k+1] \le \boldsymbol{\gamma}[k]$ introduced in Topic 7.6 is built from the bounds defined above as:

$$\underbrace{\begin{bmatrix} \boldsymbol{M}_i \\ \boldsymbol{M}_z \\ \boldsymbol{M}_v \\ \boldsymbol{M}_{\eta_s} \end{bmatrix}}_{M} \underset{\rightarrow}{\boldsymbol{\Delta i}}_{\text{app}}[k+1] \le \underbrace{\begin{bmatrix} \boldsymbol{\gamma}_i[k] \\ \boldsymbol{\gamma}_z[k] \\ \boldsymbol{\gamma}_v[k] \\ \boldsymbol{\gamma}_{\eta_s}[k] \end{bmatrix}}_{\boldsymbol{\gamma}},$$

where:

$$\boldsymbol{M}_i = \begin{bmatrix} \boldsymbol{E}_1 \\ -\boldsymbol{E}_1 \end{bmatrix}$$

$$\boldsymbol{\gamma}_i[k] = \begin{bmatrix} \boldsymbol{e}_{N_c}\left(i_{\max} - i_{\text{app}}[k]\right) \\ -\boldsymbol{e}_{N_c}\left(i_{\min} - i_{\text{app}}[k]\right) \end{bmatrix}$$

$$\boldsymbol{M}_v = \boldsymbol{G}_v$$

$$\boldsymbol{M}_v = \boldsymbol{e}_{N_p} v_{\max} - \boldsymbol{\Phi}_v \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] - \boldsymbol{e}_{N_p} b_v[k]$$

$$\boldsymbol{M}_z = \boldsymbol{G}_z$$

$$\boldsymbol{\gamma}_z[k] = \boldsymbol{e}_{N_p} z_{\max} - \boldsymbol{\Phi}_z \tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] - \boldsymbol{e}_{N_p} \mathrm{z}_0$$

$$\boldsymbol{M}_{\eta_s} = -\boldsymbol{G}_{\eta_s}$$

$$\boldsymbol{\gamma}_{\eta_s}[k] = -\boldsymbol{e}_{N_p}\eta_{s,\,\min} + \boldsymbol{\Phi}_{\eta_s}\tilde{\boldsymbol{A}}[k]\boldsymbol{\chi}[k] + \boldsymbol{e}_{N_p} U_{\mathrm{ocp}}[\theta^{\mathrm{n}}_{\mathrm{s,avg}}[k]].$$

Setting up the fast-charge problem

- At this point, we have most of the algorithmic machinery in place to execute the MPC-EKF fast-charge problem.
- What remains is to formulate the optimization objective that will bring about a fastest-time-to-charge profile.
- Strictly speaking, an optimal fast-charge is obtained as the solution to a minimum-time optimal control problem posed as:

$$\min_{\underrightarrow{\boldsymbol{\Delta i}}_{\mathrm{app}}[k+1]} \left\{ T_{\mathrm{charge}} \right\}$$

  subject to constraints.
- In other words, the cost function is $J[k] = T_{\mathrm{charge}}$, defined as the total time interval required to bring the cell from an initial state-of-charge z_0 at time t_0 to a final state-of-charge $\mathrm{z_f}$ at time t_{f}.
- Optimization is carried out over the decision vector $\underrightarrow{\boldsymbol{\Delta i}}_{\mathrm{app}}[k+1]$; that is, the sequence of future input increments of applied current, and must adhere to all stipulated constraints.

- The optimal MPC solution $\underset{\rightarrow}{\boldsymbol{\Delta i}}^*_{\text{app}}[k+1]$ describes the N_p-length input-current profile (i.e., the $\Delta i_{\text{app}}[k]$ sequence) achieving this objective.[15]
- Interestingly, when constraints are imposed simply on the input current and cell terminal voltage, the solution approaches the well-known constant-current-constant-voltage (CCCV) profile.
- In general, optimal minimum-time problems are notoriously difficult to solve and are generally not amenable to real-time computation.
- So for our implementation, we instead fashion what we shall term a pseudo-minimum-time problem:

$$\min_{\underset{\rightarrow}{\boldsymbol{\Delta i}}_{\text{app}}[k+1]} J[k],$$

subject to appropriate constraints, where we define the cost function:

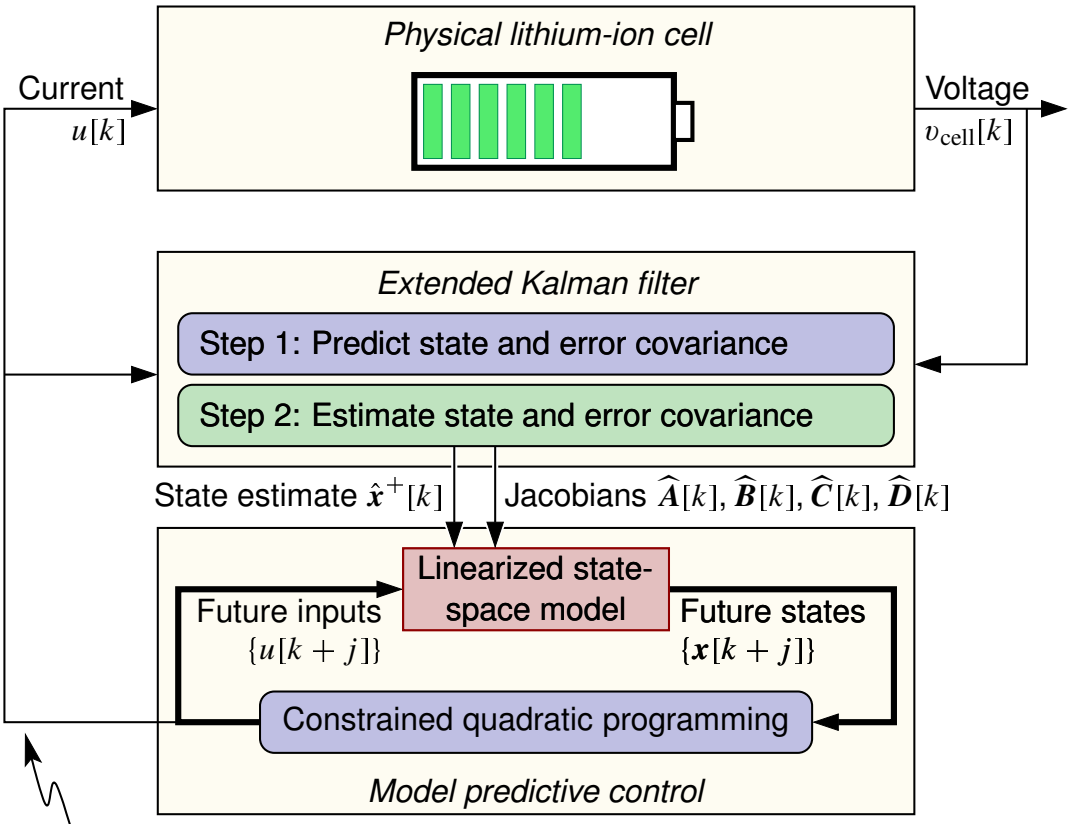
$$J[k] = \left(\underset{\rightarrow}{\boldsymbol{R}}[k+1] - \underset{\rightarrow}{\boldsymbol{Y}}[k+1]\right)^T \left(\underset{\rightarrow}{\boldsymbol{R}}[k+1] - \underset{\rightarrow}{\boldsymbol{Y}}[k+1]\right) + \rho\, \underset{\rightarrow}{\Delta \boldsymbol{U}}^T[k+1]\, \underset{\rightarrow}{\Delta \boldsymbol{U}}[k+1].$$

- The output is defined to be the instantaneous SOC $y[k] = z[k]$, and the reference is the desired final SOC charge value, $r[k] = z_\text{f}$.
- In essence, the cost function is designed to minimize the normed distance between the measured SOC and the output reference while penalizing the magnitude of the input current rate.
- As such, the optimal solution seeks to bring the SOC to its target value as quickly as possible.
- By detuning the input weighting penalty ρ to a small (but nonzero) value, the result can be shown to approach the min-time solution.

[15] Recall that although the optimization produces the entire optimal input sequence, only the first element $\Delta i_{\text{app}}[k]$ is supplied in accordance with the receding-horizon principle.

7.8: MPC implementation

- In our configuration, the state-space physics-based ROM produced in Ch. 4 will serve as the MPC prediction model.
- This model embodies the dynamics needed to describe the internal electrochemical variables of interest.
- But we cannot rely on the model alone to inform our controller since any error in initial conditions—or model accuracy—will propagate as errors in the corresponding variable estimates.
- Instead we employ an extended Kalman filter (EKF) to incorporate measurable outputs and statistical representations of process uncertainty into the computation of estimates of internal variables.
- Figure shows a notional implementation combining EKF and MPC.
- Since both MPC and EKF rely on an underlying model of the process, it is natural (and computationally efficient) to use the same physics-based ROM to drive both methods.

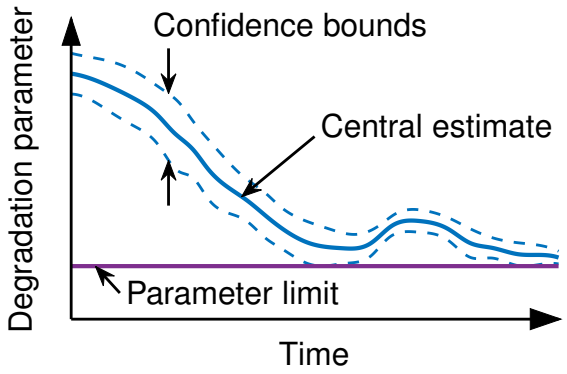


Single input-current value $u[k]$, extracted as the first (zeroth) element of the future-input sequence $\{u[k+j]\}$ optimized by MPC.

- The diagram is centered around the linearized state-space model, comprising the basic set of $\boldsymbol{A}[k]$, $\boldsymbol{B}[k]$, $\boldsymbol{C}[k]$, and $\boldsymbol{D}[k]$ matrices generated from the output-blending procedure applied to stored model sets precomputed by the xRA process of Ch. 4.
- Note that the model is updated at every sample increment to reflect changes in the operational state.
- At the top of the figure is our cell—the process to be controlled.
- Sensors capture externally available quantities (normally voltage and temperature, although only voltage is depicted in the figure)[16] and supply these to the Kalman-filtering algorithm at each time step in order to compute the estimate of the battery's state, $\hat{\boldsymbol{x}}^{+}[k]$.
- Note that when cell surface temperature is supplied, the Kalman filter can be used to estimate the corresponding core (or worst-case) internal temperature on which constraints should be enforced.
- The algorithm further updates the Jacobian matrices $\hat{\boldsymbol{A}}[k]$, $\hat{\boldsymbol{B}}[k]$, $\hat{\boldsymbol{C}}[k]$, and $\hat{\boldsymbol{D}}[k]$ used by the EKF to compute covariances.
- This state estimate is then supplied to the MPC algorithm to initialize its N_{p}-ahead prediction of the state dynamics using the same state-matrices employed by the EKF.
- The trajectory of future state values is indicated by the bold black line on the right-hand side of the MPC block in the figure.

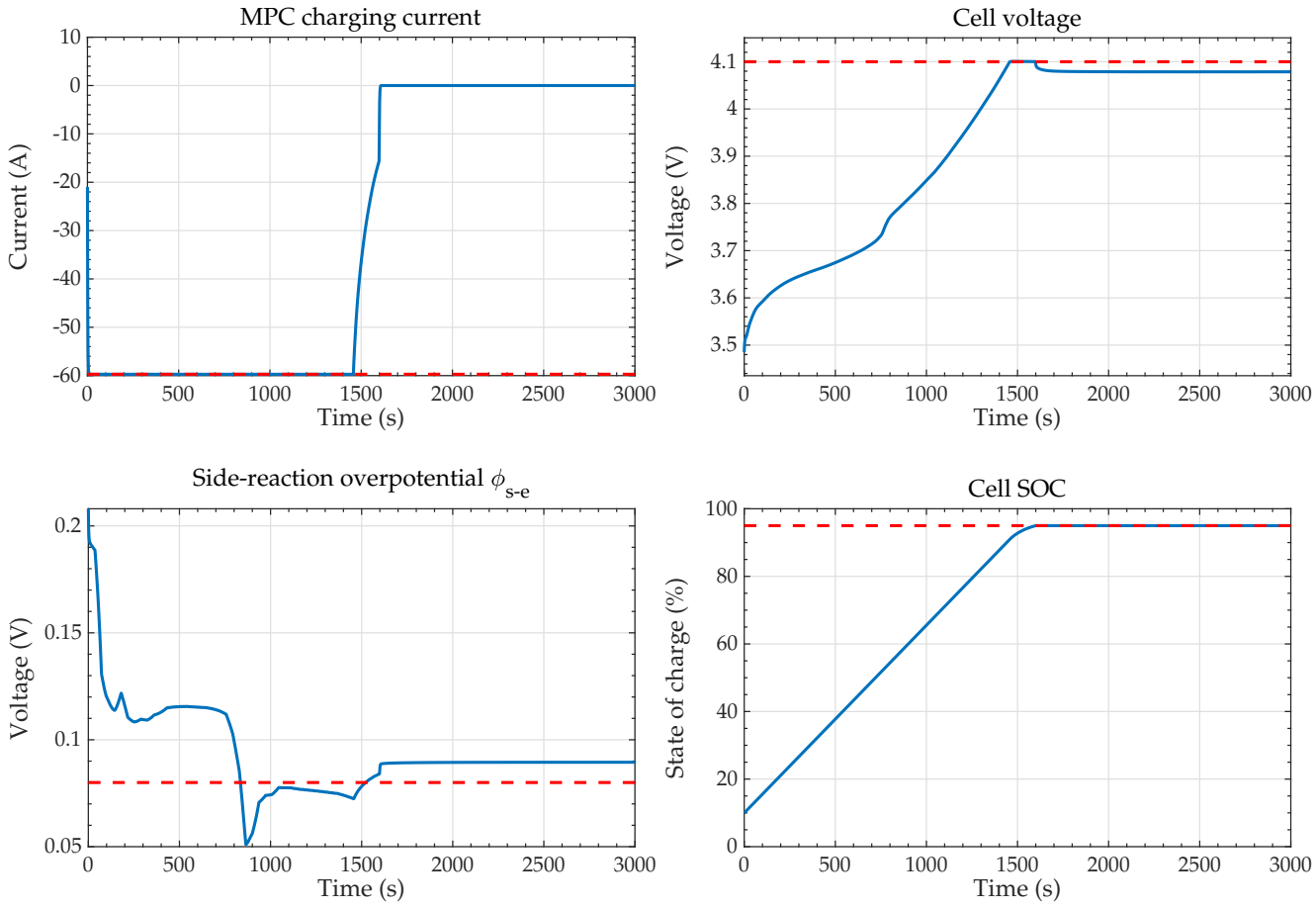
[16] We point out here that cell temperature is an important and limiting factor for fast-charge, and should be included in the constraint set of any practical fast-charge problem. For ease of exposition, we consider only voltage as the measured output here.

- Proceeding into the MPC algorithm, a constrained quadratic program is executed (e.g., the Hildreth algorithm) to produce an optimized sequence of future input values, $\{u[k+j]\}$.
- In accordance with the receding-horizon principle, the first of these values is implemented (and delivered to the EKF), while the others are discarded.
- This completes one loop through the combined MPC-EKF algorithm.
- It is important to point out here that the algorithm just described operates on the central estimate provided by the EKF, although the principles of Kalman filtering tell us that this is only the statistically most likely value with respect to the modeled statistical uncertainty.
- When using such estimates to impose bounds on system variables, it is suggested to fashion those limits instead on statistical bounding envelopes surrounding the central parameter estimates.
- The EKF computes these bounds from its own moving estimate of covariance values; a 95% confidence bound is a typical choice.
- While this imposes a somewhat conservative bound, it is nonetheless a sensible bound that effects a good compromise between performance and safety.
- The figure shows how we might bound a notional measure of cell degradation on estimated Kalman-filter statistical limits.

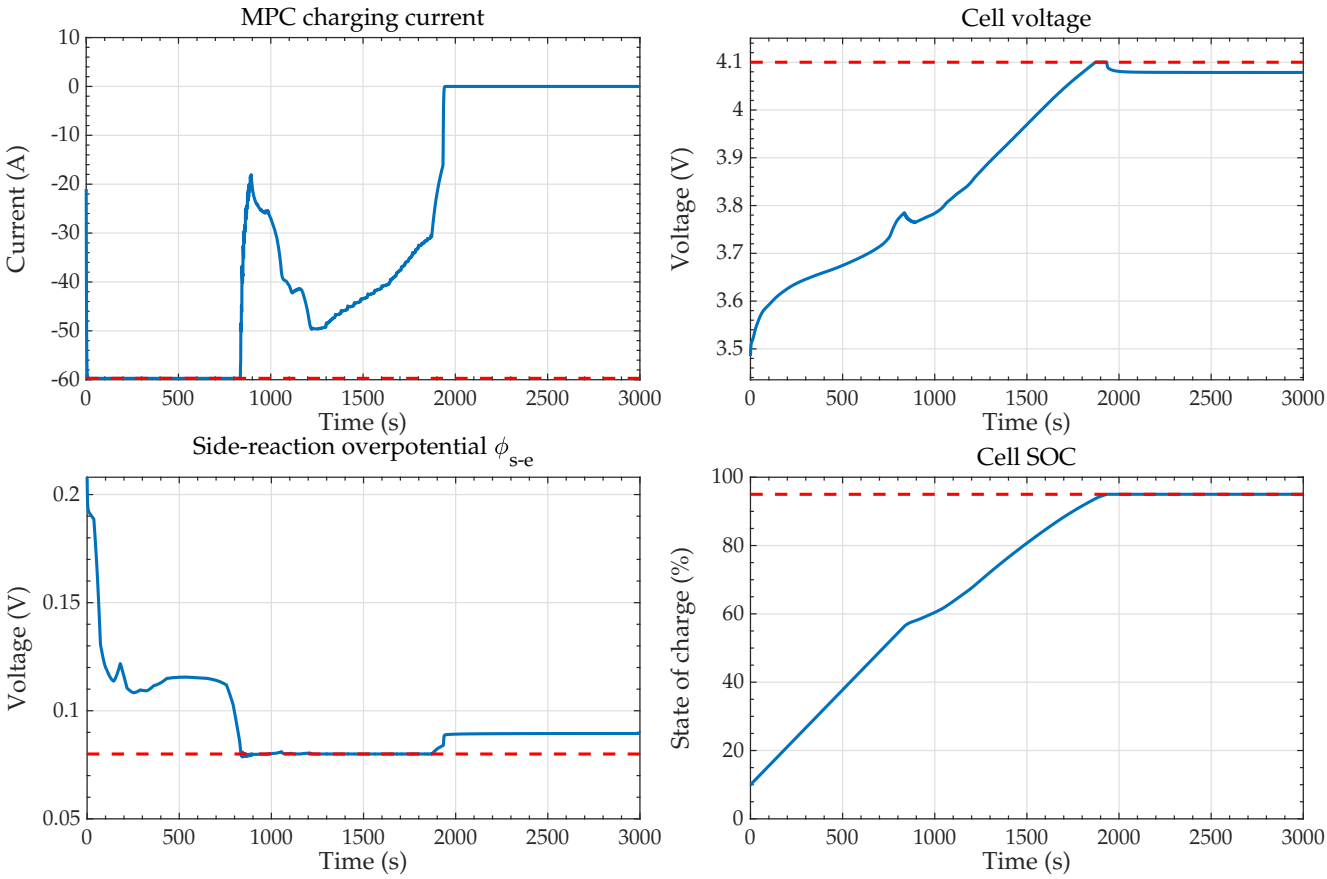


7.9: Example of MPC-EKF fast charge

- The physics-based fast-charge method is next demonstrated by applying the MPC-EKF code to the NMC30 cell model.
- The first case establishes a baseline of performance by imposing limits on maximum applied current and cell terminal voltage:
  - $|i_{\text{app}}| < 59.72\,\text{A}$ (2C rate) and $v_{\max} \leq 4.1\,\text{V}$.
- Additionally, a threshold of 0.08 V is defined for the side-reaction overpotential as a conservative limit for the onset of lithium plating.
- We seek to fast-charge from 10 % SOC to a target value of 95 %.
- MPC is tuned with relatively short horizons, $N_p = 5$ and $N_c = 2$, assuring compact prediction matrices and efficient computation.
- The figures below show simulation results:



- As expected, when constraining only current and voltage, the optimal solution gives the CCCV charge profile.
- Despite adhering to both limits, however, the side-reaction overpotential limit is breached approximately 800 s into the charge and extends to nearly 1500 s, suggesting that the conditions for lithium plating may be attained well within terminal voltage limits.
- The figures below show the same simulation conditions with the addition of a constraint imposed directly on the side-reaction overpotential as estimated by the EKF.



- It is immediately apparent that the current profile is markedly different from the CCCV version.

- In particular, MPC foresees the impending violation of the side-reaction overpotential constraint and forces the current to back off such that the limit is observed.
- Subsequent current calculations enable the fast charge to continue while respecting all imposed constraints.
- Note that in this case, the voltage limit is again reached just prior to achieving the SOC target.
- It is also important to note that lifetime-extending operation within electrochemical limits for this example comes at the cost of some charging time.

7.10: Experimental validation

- Direct experimental validation of the proposed techniques is not easily obtained, due mainly to the inaccessibility of internal electrochemical properties in situ.
- Nonetheless, a number of indirect methods have been devised to show experimentally the viability of the electrochemically constrained fast-charge idea.
- Of note, recent work by Sieg et al. experimentally investigates the fast charge of a commercial 51 Ah lithium-ion pouch cell using a control method similar in principle to that presented here.
- In particular, the authors assemble three-electrode test cells of a high-energy graphite anode pouch cell using lithium metal as a reference electrode in order to measure the overpotential value at the negative electrode-electrolyte interface.
- They employ a current-control strategy different from what is presented here, but having the same objective of maintaining the side-reaction overpotential above 0 V.
- A variety of test setups are utilized; the setup that most closely approximates the fast-charge problem presented in this chapter is shown to extend battery lifetime by more than 30% as measured by normalized capacity obtained by a C/10 discharge.
- The controlled cell reached a remaining capacity of 80 % after 1800 cycles, compared to 1350 cycles for a CC–CV reference.

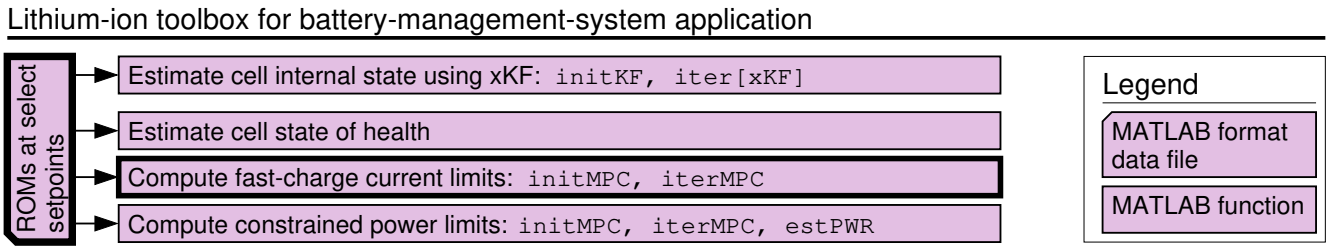
- Yin and Choe further demonstrate the viability of the proposed method using an experimental battery-in-the-loop configuration and a 40 Ah high-energy pouch cell.
- Their work features a multiobjective optimization scheme based on a reduced-order physics-based model where they seek an optimal charging profile for fast-charge that alleviates capacity fade.
- As in our approach, they consider the side-reaction overpotential (as estimated by a nonlinear Kalman filter) for mitigation of lithium-plating.
- They also consider temperature effects, which as noted previously, are easily accommodated in the approach outlined above, as well as discharge pulses to promote lithium stripping.
- Upon cycling, the controlled case reaches 95 % of original capacity after 140 cycles, whereas the reference case (CC–CV) reached the same level after just 25 cycles.
- In summary, this chapter has outlined an advanced battery control approach that exploits the rich information content of a PBM.
- Unmeasurable quantities are estimated using a nonlinear Kalman filter applied to a computationally compact reduced-order form of the full electrochemical description.
- Finally, MPC is employed to obtain optimal charge current profiles that adhere to hard constraints chosen to mitigate certain degradation effects and thereby extend life.
- The potential viability of this approach is inferred from experimental results obtained for strategies similar in principle to that presented here.

Where to from here?

- This chapter presented a practical implementation of MPC combined with EKF state estimation to bring about the safe fast charge of a lithium-ion cell within limits imposed by internal battery states.
- Use of statistical confidence bounds enhances safety of operation.
- The most important implications of this approach are:
  - Computationally compact physics-based models can form the basis of an effective battery management and control system.
  - Kalman filter estimation can be used in synchronization with predictive control to produce a workable algorithm for real-time application.
- It turns out that predictive fast-charge methods may also be used to manage energy flow into and out of a battery cell during standard operation; this has important implications for power limits estimation and will be the topic of Ch. 8.

7.11: MATLAB toolbox

- This chapter's MPC-EKF algorithms have been incorporated into the MATLAB toolbox that accompanies the course text.
- As with Chaps. 5–6, this chapter and the next are applications of the reduced order physics-based models developed in Ch. 4.
- The model-predictive control components of the toolbox are encapsulated by the MATLAB functions `initMPC` and `iterMPC`.



- To run an MPC simulation, we must first initialize the algorithm. This is done by calling `initMPC`, whose contents are:

```
% Establish MPC tuning parameters:
mpcData.Np = Np;          % Prediction horizon
mpcData.Nc = Nc;          % Control horizon
mpcData.ref = targetSOC; % Target value for final SOC
mpcData.Sigma = tril(ones(mpcData.Nc,mpcData.Nc));
mpcData.Nsim = Nsim;      % Number of simulation points
mpcData.uk_1 = 0;         % Initialization value
mpcData.SOCk_1 = 0;
mpcData.DUk_1 = 0;
mpcData.maxHild = 100;    % Maximum iterations for Hildreth
mpcData.lambda = [];
mpcData.SOC0 = SOC0;      % Initialization value
mpcData.Ts = 1;           % Sampling interval [sec]
Iapp_max  = -ROM.cellData.function.const.Q(0.5)*Crate; % Max current
Iapp_min  = 0;

% Establish MPC constraints
mpcData.const.constraints = constraints; % Enter desired conditions
if constraints(1) == 1
```



```
  mpcData.const.u_max =   Iapp_min;    % Minimum applied current
  mpcData.const.u_min =   Iapp_max;    % Maximum applied current
  mpcData.const.du_max =  50;          % Maximum increment
  mpcData.const.du_min = -50;          % Minimum increment
end
if constraints(2) == 1
  mpcData.const.v_min = V_min;
  mpcData.const.v_max = V_max;
end
if constraints(3) == 1
  mpcData.const.phise_min = Phise_min;
end
```

- After initialization, we execute MPC to compute the constrained optimal solution at each sampling interval using `iterMPC`.

```
%ITERMPC This function computes one iteration of the MPC charging protocol
% Inputs:
%   xk:        present state vector
%   xk_1:      previous timestep state vector
%   cellState: structure, contains cell information from simStep
%   mpcData:   structure, contains MPC information
% Outputs:
%   uk:        optimal input current
%   mpcData:   structure, contains updated MPC information
function [uk, mpcData] = iterMPC(xk,xk_1,cellState, mpcData)

  % Load MPC data
  Np = mpcData.Np;
  Nc = mpcData.Nc;
  uk_1 = mpcData.uk_1;
  Sigma = mpcData.Sigma;
  Ref = mpcData.ref*ones(Np,1)/100;

  % Obtain matrices from Kalman Filter
  Am = cellState.MPC.A;
  Bm = cellState.MPC.B;
  Csoc = cellState.MPC.Csoc;
  Dsoc = cellState.MPC.Dsoc;

  % Compute SOC prediction matrices
  [Phi_soc,G_soc] = predMat(Am,Bm,Csoc,Dsoc,Np,Nc);
```



```
  % Build augmented state vector
  dx = [(xk - xk_1); mpcData.SOCk_1];

  % Compute values needed by hildreth
  F = -2*G_soc'*(Ref - Phi_soc*dx );
  [~,S,~] = svd(G_soc'*G_soc);
  [m,n] = size(S);
  Ru = (norm(F,2)/(2*mpcData.const.du_max*sqrt(Nc)))-(S(m,n));
  Ru = Ru*eye(mpcData.Nc,mpcData.Nc);
  mpcData.Ru = Ru;

  E = 2*(G_soc'*G_soc + Ru);
  DU = -E\F;

  [M,gamma] = constraintsMPC(dx,cellState,mpcData);

  % Check if constraints are violated; if so, run Hildreth
  if sum(M*DU - gamma > 0) > 0
    [DU,lambda,nexec] = hildreth(E,F,M,gamma,mpcData.lambda, ...
                         mpcData.maxHild);
    mpcData.lambda = lambda;
    mpcData.nexec = nexec;
  else
    mpcData.nexec = 0;
  end

  du = DU(1);
  uk = du + mpcData.uk_1;

  mpcData.uk_1 = uk;
  mpcData.xk_1 = xk;
  mpcData.SOCk_1 = cellState.MPC.prior_SOC;
  mpcData.DUk_1 = DU;
end
```


Appendix 7.a: Summary of variables

- $b_v[k]$, affine term in voltage output equation.
- $i_{\text{app}}[k]$, applied input current value [A].
- k, sampling index.
- $J[k]$, quadratic programming objective function.
- N_{c}, MPC control horizon.
- N_{p}, MPC prediction horizon.
- T_{charge}, the charge-time interval $t_f - t_0$ [s].
- T_{s} , sampling period of discrete-time model [s].
- $u_{\max}$, input variable maximum bound.
- $u_{\min}$, input variable minimum bound.
- $\Delta u_{\max}$, input-increment variable maximum bound.
- $\Delta u_{\min}$, input-increment variable minimum bound.
- η_s, side-reaction overpotential [V].
- ρ, input increment weighting factor for MPC cost function.
- $\boldsymbol{A}[k]$, time-varying state-transition matrix of state-space model.
- $\tilde{\boldsymbol{A}}[k]$, time-varying state-transition matrix of augmented state-space model.
- $\boldsymbol{B}[k]$, time-varying input matrix of state-space model.
- $\tilde{\boldsymbol{B}}[k]$, time-varying input matrix of augmented state-space model.
- $\boldsymbol{C}[k]$, time-varying output matrix of state-space model.
- $\tilde{\boldsymbol{C}}[k]$, time-varying output matrix of augmented state-space model.
- $\boldsymbol{C}_z[k]$, time-varying SOC output matrix.
- $\boldsymbol{C}_{\eta_s}[k]$, time-varying side-reaction overpotential output matrix.
- $\boldsymbol{C}_v[k]$, time-varying voltage output matrix.
- $\boldsymbol{C}_{\phi_{\text{s-e}}}[k]$, time-varying phase potential difference output matrix.
- $\boldsymbol{D}[k]$, time-varying direct-feedthrough matrix of state-space model.
- $\boldsymbol{D}_v[k]$, time-varying voltage direct-feedthrough matrix.

- $\boldsymbol{E}_1$, a lower triangular matrix of ones.
- $\boldsymbol{H}[k]$, symmetric Hessian matrix in quadratic program objective function.
- $\boldsymbol{G}[k]$, time-and-input-dependent prediction matrix for MPC.
- $\boldsymbol{M}$, matrix whose rows contain linear constraint equation coefficients.
- $\boldsymbol{P}[k]$, time-varying Hessian matrix of dual quadratic program objective function.
- $\boldsymbol{\Phi[k]}$, time-and-state-dependent prediction matrix for MPC.
- $\boldsymbol{e}_N$, an $N \times 1$ dimension vector of ones.
- $f[k]$, time-varying scalar value in quadratic program objective function.
- $\boldsymbol{g}[k]$, time-varying gradient vector in quadratic program objective function.
- $\boldsymbol{k}[k]$, time-varying gradient vector in dual quadratic program objective function.
- $\Delta\boldsymbol{u}[k]$, input-increment vector for MPC.
- $\boldsymbol{\chi[k]}$, MPC augmented state-vector.
- $\boldsymbol{x}[k]$, time-varying state vector of state-space model.
- $\tilde{\boldsymbol{y}}[k]$, vector of linearized system outputs from ROM.
- $\boldsymbol{y}[k]$, vector of nonlinear-corrected system outputs.
- $\boldsymbol{\gamma[k]}$, vector of time-varying bounds in linear constraint equations.

Appendix 7.b: Derivation of the dual optimization problem

- Starting with the primal problem:

$$J^* = \max_{\boldsymbol{\lambda}\geq 0} \min_{\boldsymbol{x}} \left[{}^1\!/_2 \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x} + \boldsymbol{\lambda}^T (\boldsymbol{M}\boldsymbol{x} - \boldsymbol{\gamma}) \right],$$

  we solve for the unconstrained minimizer via the stationarity condition:

$$\frac{\partial J}{\partial \boldsymbol{x}} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda} = 0,$$

  giving the solution,

$$\boldsymbol{x}^o = -\boldsymbol{H}^{-1}(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}).$$

- Substituting $\boldsymbol{x}^o$ into the primal problem, we obtain:

$$\begin{aligned} J^* = \max_{\boldsymbol{\lambda}\geq 0} \Big\{ & {}^1\!/_2 \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right)^T \boldsymbol{H}^{-1} \boldsymbol{H} \boldsymbol{H}^{-1} \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right) \\ & - \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right)^T \boldsymbol{H}^{-1} \boldsymbol{g} + \boldsymbol{\lambda}^T \left(\boldsymbol{M} \left(-\boldsymbol{H}^{-1}(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda})\right) - \boldsymbol{\gamma}\right) \Big\}. \end{aligned}$$

- Expanding and cancelling terms:

$$\begin{aligned} J^* = & \max_{\boldsymbol{\lambda}\geq 0} \Big\{ {}^1\!/_2 \left(\boldsymbol{\lambda}^T \boldsymbol{M} + \boldsymbol{g}^T\right) H^{-1} \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right) \\ & - \left(\boldsymbol{\lambda}^T \boldsymbol{M} + \boldsymbol{g}^T\right) \boldsymbol{H}^{-1} \boldsymbol{g} + \boldsymbol{\lambda}^T \left[-\boldsymbol{M}\boldsymbol{H}^{-1} \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right) - \boldsymbol{\gamma}\right] \Big\} \\ = & \max_{\boldsymbol{\lambda}\geq 0} \Big\{ {}^1\!/_2 \left[\boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right) + \boldsymbol{g}^T \boldsymbol{H}^{-1} \left(\boldsymbol{g} + \boldsymbol{M}^T \boldsymbol{\lambda}\right)\right] \\ & - \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{g} - \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g} + \boldsymbol{\lambda}^T \left[-\boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{g} - \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{M}^T \boldsymbol{\lambda} - \boldsymbol{\gamma}\right] \Big\} \\ = & \max_{\boldsymbol{\lambda}\geq 0} \Big\{ {}^1\!/_2 \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{g} + {}^1\!/_2 \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{M}^T \boldsymbol{\lambda} \\ & + {}^1\!/_2 \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g} + {}^1\!/_2 \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{M}^T \boldsymbol{\lambda} - \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{g} - \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g} \\ & - \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{g} - \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{M}^T \boldsymbol{\lambda} - \boldsymbol{\lambda}^T \boldsymbol{\gamma} \Big\}. \end{aligned}$$

- And finally, we achieve:

$$J^* = \max_{\boldsymbol{\lambda}\geq 0} \left\{ -\boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{g} - \frac{1}{2} \boldsymbol{\lambda}^T \boldsymbol{M}\boldsymbol{H}^{-1} \boldsymbol{M}^T \boldsymbol{\lambda} - \frac{1}{2} \boldsymbol{g}^T \boldsymbol{H}^{-1} \boldsymbol{g} - \boldsymbol{\lambda}^T \boldsymbol{\gamma} \right\}.$$

Appendix 7.c: Hildreth's algorithm

- Hildreth's quadratic programming procedure is a simple variant of the Gauss–Seidel method for solving a linear set of equations.
- Gauss–Seidel is an iterative technique for solving equations of the general form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ for a square matrix $\boldsymbol{A}$.
- It proceeds by decomposing $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{L}_A + \boldsymbol{U}_A$, where $\boldsymbol{L}_A$ is a lower triangular matrix and $\boldsymbol{U}_A$ is strictly upper triangular.
- By doing so, we can write:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

$$\left(\boldsymbol{L}_A + \boldsymbol{U}_A\right)\boldsymbol{x} = \boldsymbol{b}$$

$$\boldsymbol{L}_A\boldsymbol{x} = -\boldsymbol{U}_A\boldsymbol{x} + \boldsymbol{b}.$$

- Gauss–Seidel then solves for $\boldsymbol{x}$ on the left-hand side using previously computed values of $\boldsymbol{x}$ on the right-hand side:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{L}_A^{-1}\left(-\boldsymbol{U}_A\boldsymbol{x}^{(k)} + \boldsymbol{b}\right).$$

- By exploiting the triangular nature of $\boldsymbol{L}_A$ and $\boldsymbol{U}_A$ we avoid inversion and compute the elements of $\boldsymbol{x}$ by substitution,

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[-\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} + b_i\right].$$

- The Hildreth procedure utilizes this method to solve for the individual elements of $\boldsymbol{\lambda}$ that satisfy the dual optimization problem,

$$\min_{\boldsymbol{\lambda}\geq 0} J = \left(\frac{1}{2}\boldsymbol{\lambda}^T\boldsymbol{P}\boldsymbol{\lambda} + \boldsymbol{\lambda}^T\boldsymbol{k} + \frac{1}{2}\boldsymbol{g}^T\boldsymbol{H}^{-1}\boldsymbol{g}\right).$$

- The stationarity condition is obtained from $\partial J/\partial \boldsymbol{\lambda} = \boldsymbol{P}\boldsymbol{\lambda} + \boldsymbol{k} = 0$, which gives the linear matrix equation $\boldsymbol{P}\boldsymbol{\lambda} = -\boldsymbol{k}$.
- In particular, the method computes individual λ_i by setting $\partial J/\partial \lambda_i = 0$ while holding all other components fixed at their previous values.
  - Positive λ_i values are retained; negative values replaced by $\lambda_i = 0$.
- The Hildreth procedure implements the following algorithm:

$$\lambda_i^{(m+1)} = \max\left(0,\ w_i^{(m+1)}\right),$$

  where:

$$w_i^{(m+1)} = -\frac{1}{P_{ii}}\left[\sum_{j=1}^{i-1} P_{ij}\lambda_j^{(m+1)} + \sum_{j=i+1}^{n_\lambda} P_{ij}\lambda_j^{(m)} + k_i\right],$$

  for $i = 1,\ 2,\ \ldots,\ n_\lambda$. Here, P_{ij} is the ijth element of the matrix $\boldsymbol{P} = \boldsymbol{M}\boldsymbol{H}^{-1}\boldsymbol{M}^T$, k_i is the ith element of the vector $\boldsymbol{k}$, and n_λ is the length of $\boldsymbol{\lambda}$.
- Some features of the algorithm:
  - $\boldsymbol{\lambda}$ is systematically varied one component at a time and the λ_i elements chosen to minimize the objective function.
  - The objective function can be regarded as a quadratic function in each λ_i component.
  - Previously calculated values of λ_i are incorporated into subsequent calculations.
  - The Hildreth procedure, when convergent, approaches the solution asymptotically.
  - If constraint matrix $\boldsymbol{M}$ has full row rank, then $\boldsymbol{P}$ is positive definite and the problem has a unique solution.