Problem 1: Optimizing plug-in charging

- Plug-in charging can be optimized by nonlinear programming
  e.g., sequential quadratic programming, Octave's `fmincon.m`
- Nonlinear programming attempts to find solutions to problems posed in framework

$$x^* = \arg\min f(x), \quad \text{such that} \begin{cases} c(x) \le 0, & Ax \le b \\ c_{\text{eq}}(x) = 0, & A_{\text{eq}}x = b_{\text{eq}} \\ lb \le x, & x \le ub, \end{cases}$$

where we wish to minimize $f(x)$ by choosing optimum input vector x^* and

- Nonlinear in/equality constraint vector functions $c(x) \le 0$ and $c_{\text{eq}}(x) = 0$ satisfied
- Linear in/equality constraint vector functions $Ax \le b$ and $A_{\text{eq}}x = b_{\text{eq}}$ satisfied
- Bounds $lb \le x \le ub$ for all entries in vector x are satisfied

for user-specified $f(x)$, $c(x)$, $c_{\text{eq}}(x)$, A, b, A_{eq}, b_{eq}, lb, and ub

Cost function for plug-in charge

- Choose x as vector of cell applied current vs. time, $f(x)$ to be estimate of cell degradation caused by that applied current, and the other functions and matrices to make the problem work
- For example, we might want to find

$$i^* = \arg\min \sum_{k=0}^{K-1} -J_s(i_k, z_k, T_k) \quad \text{such that} \begin{cases} z_{\min} \le z_k \le z_{\max} \\ z_K = z_{\text{end}} \\ -I_{\max} \le i_k \le I_{\max} \\ z_k = z_0 - \sum_{j<k} i_j \Delta t / Q \end{cases}$$

- That is, desire to minimize capacity loss when charging, starting at SOC z_0, ending at SOC z_{end} over period of K sampling intervals, where current is limited between $\pm I_{\max}$, SOC is limited between $z_{\min}$ and $z_{\max}$ and standard SOC equation holds

Recasting in nonlinear-programming framework

- Not difficult to recast this problem in the right framework
- First, note we can rewrite SOC equation in vector form as:

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_K \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}_{CV} z_0 - \frac{\Delta t}{Q} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}}_{LT} \underbrace{\begin{bmatrix} i_0 \\ i_1 \\ \vdots \\ i_{K-1} \end{bmatrix}}_{x}$$

- Using this formulation, can write the z_K constraint

$$z_K = z_0 - (\Delta t / Q) \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \end{bmatrix} x = z_{\text{end}}$$

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \end{bmatrix}}_{A_{\text{eq}}} x = \underbrace{(z_0 - z_{\text{end}})\, Q / \Delta t}_{b_{\text{eq}}}$$

Continuing to recast constraints

- The limit $z_{\min} \le z_k$ can be written as (where LT is a "lower-triangular" matrix and CV is a "column-vector" of ones)

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} z_{\min} \le \underbrace{\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}}_{CV} z_0 - \frac{\Delta t}{Q} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}}_{LT} \underbrace{\begin{bmatrix} i_0 \\ i_1 \\ \vdots \\ i_{K-1} \end{bmatrix}}_{x}$$

$$(CV)(z_{\min} - z_0) \le -\frac{\Delta t}{Q} (LT)\, x$$

$$(LT)\, x \le \frac{Q}{\Delta t}(CV)(z_0 - z_{\min}).$$

Finalizing constraints

- Finally, $z_k \le z_{\max}$ can be written as

$$-(LT)x \le \frac{Q}{\Delta t}(CV)(z_{\max} - z_0)$$

- Putting the last two constraints together gives

$$\underbrace{\begin{bmatrix} LT \\ -LT \end{bmatrix}}_{A} x \le \underbrace{\frac{Q}{\Delta t}\begin{bmatrix} (CV)(z_0 - z_{\min}) \\ (CV)(z_{\max} - z_0) \end{bmatrix}}_{b}$$

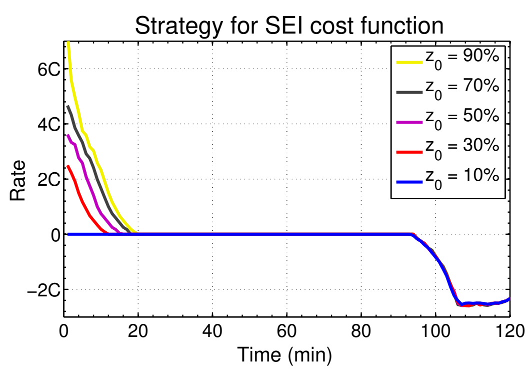
- The constraints on input current can be satisfied by setting

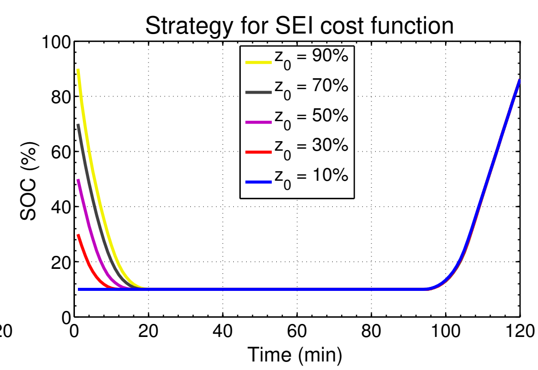
$$lb = -I_{\max}(CV), \qquad \text{and} \qquad ub = I_{\max}(CV)$$

- There are no nonlinear constraints in this problem

Example 1 using SEI cost function

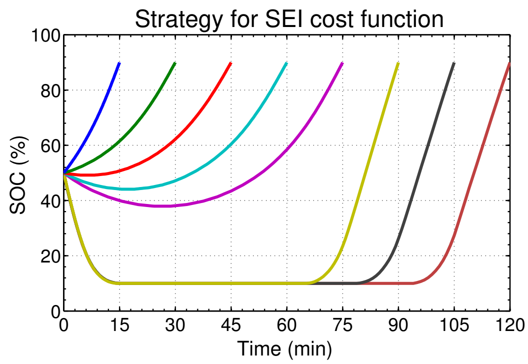
- Here, we use SEI growth model for cost $f(x)$, $z_{\min} = 10\,\%$, $z_{\max} = 90\,\%$, z_0 varied, time allowed of 2 h, i_k unconstrained

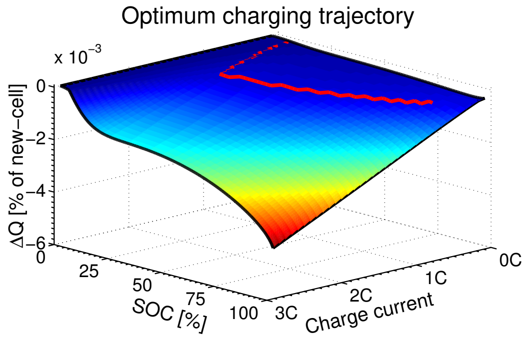




Example 2 using SEI cost function

- Same as before, except $z_0 = 50\,\%$ and allowed time varied





Problem 2: Optimizing dynamic power limits

- Compute limits on dis/charge power over next ΔT s, observing design limits on SOC, maximum power or current, degradation
- Handle by looking for maximum dis/charge current, then convert to power
- Can use Model Predictive Control (MPC), where idea is to:
  - Find N-length input sequence—using cell model to predict future performance—that will cause model's controlled variables to converge toward desired values
  - Implement only the first of these N signals
  - Repeat
- Allows us, for example, to predict constant-current input that would not violate limits and would optimize a cost function if applied for the full ΔT s (N sample periods), but only implement the first of these, then repeat

Modified method

- Here, I use a similar idea to MPC, leading up to same form of quadratic optimization used by MPC, assuming system model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k + Du_k, \end{aligned}$$

  where y_k are performance variables we desire to control to some limit, or maintain within hard constraints (may be different from outputs we've previously called y_k)
- We'll look at this idea really quickly
- First, define the vectors:

$$Y = \begin{bmatrix} y_k & y_{k+1} & \cdots & y_{k+N} \end{bmatrix}^T \quad \text{and} \quad U = \begin{bmatrix} u_k & u_{k+1} & \cdots & u_{k+N} \end{bmatrix}^T$$

Helpful matrix/vector equation

- With these definitions, we can "unwrap" system dynamics as a matrix/vector equation

$$\underbrace{\begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+N} \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^N \end{bmatrix}}_{F} x_k + \underbrace{\begin{bmatrix} D & & & 0 \\ CB & D & & \\ CAB & CB & & \\ \vdots & \vdots & \ddots & \\ CA^{N-1}B & CA^{N-2}B & \cdots & D \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N} \end{bmatrix}}_{U}$$

$$Y = Fx_k + \Phi U$$

- This compact notation helpfully shows linear-algebra structure of model

Cost function for power-limits

- Now, we define a cost function that we wish to optimize:

$$\begin{aligned} J &= (R_s - Y)^T Q (R_s - Y) + U^T R U \\ &= (R_s - [Fx_k + \Phi U])^T Q (R_s - [Fx_k + \Phi U]) + U^T R U \\ &= [R_s^T Q R_s - 2R_s^T Q F x_k + x_k^T F^T Q F x_k] \quad \text{(not a function of } U\text{)} \\ &\quad + 2[x_k^T F^T Q \Phi - R_s^T Q \Phi] U + U^T [\Phi^T Q \Phi + R] U \end{aligned}$$

- Can write in quadratic form ($H = 2[\Phi^T Q \Phi + R]$, $f^T = 2(x_k^T F^T Q \Phi - R_s^T Q \Phi)$)

$$J = \frac{1}{2} U^T H U + f^T U + \text{constant}$$

Constraints on power-limits problem

- Can put constraints on Y via $Y_{\max} \le Fx_k + \Phi U \le Y_{\max}$, which can be written as

$$\begin{aligned} \Phi U &\le [Y_{\max} - Fx_k] \\ -\Phi U &\le [Fx_k - Y_{\min}] \end{aligned}$$

- Can combine in the matrix inequality $A_{\text{ineq}} U \le b_{\text{ineq}}$, where

$$\underbrace{\begin{bmatrix} \Phi \\ -\Phi \end{bmatrix}}_{A_{\text{ineq}}} U \le \underbrace{\begin{bmatrix} Y_{\max} - Fx_k \\ Fx_k - Y_{\min} \end{bmatrix}}_{b_{\text{ineq}}}$$

Solving power-limits problem

- Have now defined vectors/matrices H, f^T, A_{ineq}, and b_{ineq} that match a quadratic programming problem, which is:

$$U^* = \arg\min \frac{1}{2} U^T H U + f^T U$$

such that $A_{\text{ineq}} U \le b_{\text{ineq}}$ (in Octave, solution found via `quadprog.m`)

- Note, can use $U = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T u$ to make fast single-variable optimization problem, to give maximum dis/charge current value that would apply to all times
- But, what values to use?
  - Reference R_s value for model SOC state on charging set to 1.0;
  - Reference R_s value for model SOC state on discharging set to 0.0;
  - Soft constraints to slow degradation (how?), hard constraints to prohibit Li plating

Where from here?

- We've reached edge of knowledge for battery management
- There's plenty of work yet to do:
  - How do we efficiently implement the optimized power controls, and how do we tune to accommodate various aging mechanisms and cost tradeoffs?
  - How do we perform system identification of physics-based model parameters to give a good enough model to match a real cell well?
  - How do we model new degradation mechanisms in efficient ways, for implementation in embedded systems?
  - And many more we haven't even thought of yet
- I hope some of this material has sparked your imagination, and I hope you will be able to contribute to making battery management systems of the future even better!