What you will learn in this lesson

- In a previous lesson, you learned how to write equations for simulation of a battery pack comprising multiple SCMs
- In this lesson, you will see some Octave/MATLAB code that simulating SCMs, and will see how it works step-by-step
- This code (optionally) allows every cell to have different parameter values, and allows for an arbitrary number of cells in parallel and in series
- It even (optionally) allows simulation of packs containing cell(s) having open-circuit or short-circuit faults

Initialization code

- The following code loads model, begins configuration

```
% ------------------------------------------------------------------------
% simSCM: Simulate series-cell-module packs (cells are connected in series
% to make modules; these modules are connected in parallel to make packs)
% The parameter values for each cell may be different
clear; close all; clc;

% Initialize some pack configuration parameters...
load E2model;    % creates var. "model" with E2 cell parameter values
Ns = 12;         % Number of modules connected in series to make a pack
Np = 3;          % Number of cells connected in parallel in each module

% Initialize some simulation configuration parameters...
maxtime = 3600; % Simulation run time in simulated seconds
t0 = 2700;       % Pack rests after time t0
storez = zeros([maxtime Ns Np]);  % create storage for SOC
storei = zeros([maxtime Ns Np]);  % create storage for current
```


Code for default initialization

- The following code initializes all states and parameters

```
% Initialize states for ESC cell model
z   = 0.25*ones(Ns,Np); % SOC for each cell
irc = zeros(Ns,Np);     % R-C resistor currents for each cell
h   = zeros(Ns,Np);     % dynamic hysteresis for each cell
s   = zeros(Ns,Np);     % current sign for each cell

% Default initialization for cells within the pack
T    = 25; % Default temperature
q    = getParamESC('QParam',T,model)*ones(Ns,Np);
rc   = exp(-1./abs(getParamESC('RCParam',T,model)))'*ones(Ns,Np);
r    = (getParamESC('RParam',T,model))';
m    = getParamESC('MParam',T,model)*ones(Ns,Np);
m0   = getParamESC('M0Param',T,model)*ones(Ns,Np);
g    = getParamESC('GParam',T,model)*ones(Ns,Np);
r0   = getParamESC('R0Param',T,model)*ones(Ns,Np);
rt   = 0.000125; % 125 microOhm resistance for each tab
```

Code to modify cell parameter values

- The following code (optionally) modifies some parameters

```
% Modified initialization for cell variability

% e.g., set individual random "initial SOC" values
if 1, % set to "if 1," to execute, or "if 0," to skip this code
  z=0.30+0.40*rand([Ns Np]); % rand. init. SOC for ea. cell
end

% e.g., set individual random cell-capacity values
if 1, % set to "if 1," to execute, or "if 0," to skip this code
  q=4.5+rand([Ns Np]);      % random capacity for ea. cell
end

% e.g., set individual random cell-resistance relationships
if 1, % set to "if 1," to execute, or "if 0," to skip this code
  r0 = 0.005+0.020*rand(Ns,Np);
end
r0 = r0 + 2*rt; % add tab resistance to cell resistance
```


Code to add faults, set up simulation

- The following code (optionally) adds faults, then sets up sim

```
% Add faults to pack: cells faulted open- and short-circuit

% To delete a PCM (open-circuit fault), set a resistance to Inf
%r0(1,1) = Inf; % for example...

% To delete a cell from a PCM (short-circuit fault), set its SOC to NaN
%z(1,2) = NaN; % for example, delete cell 2 in PCM 1
Rsc = 0.0025; % Resistance value to use for cell whose SOC < 0%

% Get ready to simulate... first compute pack capacity in Ah
totalCap = sum(min(q)); % pack capacity = minimum module capacity
I = 10*totalCap; % cycle at 10C... not realistic, faster simulation


% Okay... now to simulate pack performance using ESC cell model...
```


Main code loop to simulate SCMs

```
for k = 1:maxtime,
  v = OCVfromSOCtemp(z,T,model); % get OCV for Ns * Np cells
  v = v + m.*h +m0.*s - r.*irc; % add hysteresis, diffusion
  r0(isnan(z)) = Rsc; % s-c fault has "short-circuit" resistance
  V = (sum(sum(v,1)./sum(r0,1),2)-I)./sum(1./sum(r0,1),2); % Bus V
  ik = (sum(v,1)-repmat(V,1,Np))./sum(r0,1); % 1 * Np cell currents
  ik = repmat(ik,Ns,1);
  z = z - (1/3600)*ik./q;  % Update each cell SOC
  z(z<0) = NaN; % set over-discharged cells to short-circuit fault
  irc = rc.*irc + (1-rc).*ik; % Update diffusion currents
  Ah = exp(-abs(g.*ik)./(3600*q));
  h = Ah.*h + (1-Ah).*sign(ik); % Update hysteresis voltages
  s(abs(ik)>1e-3) = sign(ik(abs(ik)>1e-3)); % Update current sign
  if min(z(:)) < 0.05, I = -abs(I); end  % stop discharging
  if max(z(:)) > 0.95, I = abs(I);  end  % stop charging
  if k>t0, I = 0; end % rest
  storez(k,:,:) = z; storei(k,:,:) = ik; % Store SOC, current for later plotting
end % for k
```

Example of how to plot results

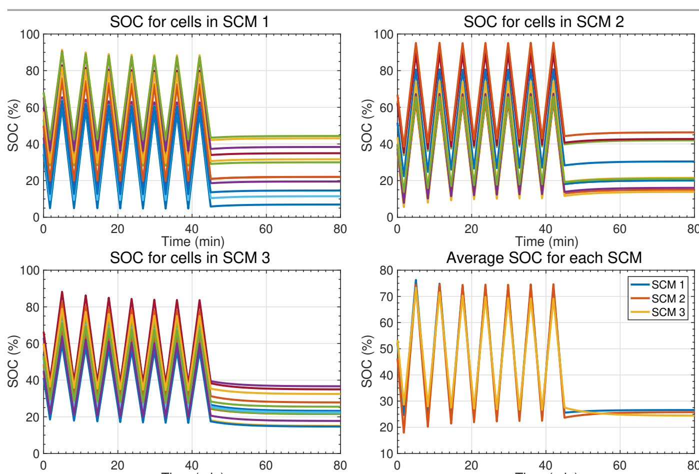
```
% Plot individual cell SOC vs. time for all cells
% in all series PCMs. There is one subplot for each SCM.
figure; clf; t = (0:(length(storez(:,:,1))-1))/60;
xplots = round(ceil(sqrt(Np))); yplots = ceil(Np/xplots);
for k = 1:Np,
  zr=squeeze(100*storez(:,:,k)); subplot(yplots,xplots,k); plot(t,zr);
  axis([0 ceil(maxtime/60) 0 100]); title(sprintf('Cells in SCM %d',k));
  ylabel('SOC (%)'); xlabel('Time (min)');
end

% Plot individual cell current vs. time for all cells in all series PCMs
figure; clf; t = (0:(length(storei(:,:,1))-1))/60;
for k = 1:Np,
  zr=squeeze(storei(:,:,k)); subplot(yplots,xplots,k); plot(t,zr);
  axis([0 ceil(maxtime/60) -101 101]); title(sprintf('Cells in SCM %d',k));
  ylabel('Current (A)'); xlabel('Time (min)');
end
```


Example simulation scenario

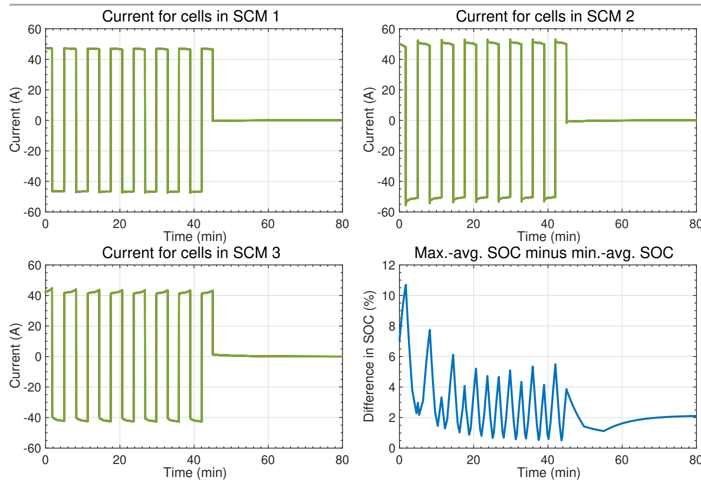
- We look at an example simulation with $N_s = 12$ and $N_p = 3$
- Cell SOCs are randomly initialized between 30 % and 70 %
- Cell R_0 values are randomly initialized between 5 mΩ and 25 mΩ
- Cell capacities are randomly initialized between 4.5 Ah and 5.5 Ah
- The pack is repeatedly
  - Discharged at 10C rate until lowest cell hits 5 %
  - Charged at 10C rate until highest cell hits 95 %
- After 45 min, the pack rests

Results for cell SOCs



- Representative results for individual cell SOCs and SCM-average SOCs are shown

Results for cell currents



- Corresponding representative results for individual cell currents and dispersion between SCM-average SOCs are shown

Summary

- In this lesson, you have learned how to write Octave/MATLAB code to simulate battery packs built from SCMs
- This code allows for arbitrarily sized packs, and (optionally) allows for every cell to have different parameter values, and even to be faulted open- or short-circuit
- The code agrees with the math presented in a prior lesson
- Results show interesting and valuable insight into the individual cell responses in a battery pack comprising SCMs that would be difficult to measure in an actual pack