Objective of this lesson

- In this lesson, I share Octave code to compute power limits using comprehensive cell model, with state produced by “truth” open-loop simulation or SPKF
- Start with code that computes limits using state produced by open-loop simulation

```
% Bisection Power Estimation: Truth
% Define inline functions that compute A and B for input current = ik
A = @(ik) [1 0 0; 0 exp(-1/(RC)) 0; 0 0 exp(-abs(ik*Gamma/(3600*Q)))];
B = @(ik) [-1/(3600*Q) 0; (1-exp(-1/RC)) 0; 0 (exp(-abs(ik*Gamma/(3600*Q)))-1)];

% First simulate state trajectory for entire profile of current versus time
[vk,irck,hk,zk,OCV] = simCell(current,T,model,1,0,0);

% Reserve storage for results of power calculations
pDisMax = zeros(length(current),1);
pChgMin = zeros(length(current),1);
```


Truth: Main simulation loop (1)

- Start looking at main code loop
- This portion computes discharge power

```
% Loop through profile of current versus time, computing power
for ii = 1:length(current)
  x0 = [zk(ii); irck(ii); hk(ii)]; % cell model state at this point in time

  % Discharge Power Estimation
  g = @(ik) bisectDischarge(ik,x0,A,B,Thorz,T,model,R0,R,M,M0,vmin,zmin);
  iDisMax = 0;
  if zk(ii) > zmin,
    iDisMax = bisect(g,0,imax,0.5); % bisect "g" between 0 and imax with eps=0.5
  end
  pDisMax(ii) = vmin*iDisMax;
```


Truth: Main simulation loop (2)

- Continue looking at main simulation loop
- This portion computes charge power

```
  % Charge Power Estimation
  g = @(ik) bisectCharge(ik,x0,A,B,Thorz,T,model,R0,R,M,M0,vmax,zmax);
  iChgMin = 0;
  if zk(ii) < zmax,
    iChgMin = bisect(g,0,imin,0.5); % bisect "g" between 0 and imin with eps=0.5
  end
  pChgMin(ii) = vmax*iChgMin;
end
truthBisect.pDisMax = pDisMax;
truthBisect.pChgMin = pChgMin;
```

SPKF: Main simulation loop (1)

- The following code is same, except for SPKF producing state
- This portion computes discharge power

```
% Bisection Power Estimation: SPKF
[irck,irckBounds,hk,hkBounds,zk,zkBounds] = SPKFmethod();

pDisMax = zeros(length(current),1);
pChgMin = zeros(length(current),1);
for ii = 1:length(current)
  % Discharge Power Estimation
  x0 = [zk(ii)-zkBounds(ii); irck(ii)+irckBounds(ii); (hk(ii)-hkBounds(ii))];
  g = @(ik) bisectDischarge(ik,x0,A,B,Thorz,T,model,R0,R,M,M0,vmin,zmin);
  iDisMax = 0;
  if zk(ii)-zkBounds(ii)>zmin,
    iDisMax = bisect(g,0,imax,0.5);
  end
  pDisMax(ii) = vmin*iDisMax;
```


SPKF: Main simulation loop (2)

- The following code is same, except for SPKF producing state
- This portion computes charge power

```
  % Charge Power Estimation
  x0 = [zk(ii)+zkBounds(ii); irck(ii)-irckBounds(ii); (hk(ii)+hkBounds(ii))];
  g = @(ik) bisectCharge(ik,x0,A,B,Thorz,T,model,R0,R,M,M0,vmax,zmax);
  iChgMin = 0;
  if zk(ii)+zkBounds(ii) < zmax,
    iChgMin = bisect(g,0,imin,0.5);
  end
  pChgMin(ii) = vmax*iChgMin;
end
spkfBisect.pDisMax = pDisMax;
spkfBisect.pChgMin = pChgMin;
```


Summary

- This lesson looked at code to determine power limits using entire cell model and full model state
- First set of code used open-loop simulation to determine true state values at every point in time; second set of code used SPKF plus conservative bounds
- Next lesson will apply this code to same example you saw last week for HPPC method for comparison purposes