Inline functions for state-equation matrices

- To estimate power limits using bisection and ESC model, must define bisection cost function involving $k_{\Delta T}$-second prediction
- ESC-model state equation is linear, with

$$x_n[k+1] = Ax_n[k] + Bu_n[k]$$

- To make some required computations easier, we define inline matrix functions to compute state-space A and B based on input current:

```
A = @(ik) diag([1 exp(-1/(RC)) 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)];
```


5.4.3: How to use bisection to estimate available power using comprehensive cell model

Inline functions for state-equation matrices

- For u_n constant over entire prediction horizon, we have

$$x_n[k+k_{\Delta T}] = A^{k_{\Delta T}} x_n[k] + \left(\sum_{j=0}^{k_{\Delta T}-1} A^{k_{\Delta T}-1-j} B\right) u_n$$

- A is diagonal, so $A^{k_{\Delta T}}$ simply matrix comprising scalar power of diagonal elements
- Similarly, the summation can be written as

$$\begin{aligned}\sum_{j=0}^{k_{\Delta T}-1} A^{k_{\Delta T}-1-j} &= \left(\sum_{j=0}^{k_{\Delta T}-1} A^{-j}\right) A^{k_{\Delta T}-1} = \left(\sum_{j=0}^{k_{\Delta T}-1} \left(A^{-1}\right)^{j}\right) A^{k_{\Delta T}-1} \\ &= \left(I - A^{-1}\right)^{-1}\left(I - A^{-k_{\Delta T}}\right) A^{k_{\Delta T}-1} \\ &= \left(I - A^{-1}\right)^{-1}\left(A^{k_{\Delta T}-1} - A^{-1}\right) = (A-I)^{-1}\left(A^{k_{\Delta T}} - I\right)\end{aligned}$$

5.4.3: How to use bisection to estimate available power using comprehensive cell model

Simulating cell with constant current

- Simplifications allow writing very efficient code to simulate a cell $k_{\Delta T}$ samples into the future

```
% Simulate cell for KDT samples, with input current equal to ik, initial
% state = x0, A and B functions, temperature = T, with model parameters
% R0, R, M and the model structure "model".
function [vDT,xDT] = simCellKDT(ik,x0,A,B,KDT,T,model,R0,R,M,M0)
  Amat = A(ik); Bmat = B(ik); dA = diag(Amat);
  if ik == 0,
    ADT = diag([KDT, (1-dA(2)^KDT)/(1-dA(2)), KDT]);
  else
    ADT = diag([KDT, (1-dA(2)^KDT)/(1-dA(2)), (1-dA(3)^KDT)/(1-dA(3))]);
  end
  xDT = (dA).^KDT.*x0 + ADT*Bmat*[ik; sign(ik)];
  vDT = OCVfromSOCtemp(xDT(1),T,model)+M*xDT(3)+M0*sign(ik)-R*xDT(2)-ik*R0;
end
```

Invoking bisection

- Can now write bisection "cost" functions
- Consider discharge for only terminal voltage and SOC limits:

```
function h = bisectDischarge(ik,x0,A,B,KDT,T,model,R0,R,M,minV,zmin)
  [vDT,xDT] = simCellKDT(ik,x0,A,B,KDT,T,model,R0,R,M);
  h = max(minV - vDT,zmin - xDT(1)); % max must be less than zero
end
```

- Consider charge for same limits:

```
function h = bisectCharge(ik,x0,A,B,KDT,T,model,R0,R,M,maxV,zmax)
  [vDT,xDT] = simCellKDT(ik,x0,A,B,KDT,T,model,R0,R,M);
  h = min(maxV - vDT,zmax - xDT(1)); % min must be greater than zero
end
```

- To use one of these functions, we use code like:

```
h = @(x) bisectDischarge(x,x0,A,B,KDT,T,model,R0,R,M,minV,zmin)
ilimit = bisect(h,0,imax,itol);
```


Summary

- To find current limits, we will use bisection algorithm
- In this lesson, you learned how bisection will be used inside overall algorithm to find current limits (and hence power limits)
- Have seen how to code efficient $k_{\Delta T}$ cell simulation for use in bisection
- Have learned how to create bisection "cost function" and how to invoke bisection
- Next lesson, we will put everything together