EKF iteration code, load model

- Implementation of EKF on ESC model refactors code
  - “Wrapper” code, coordinates the entire simulation process
  - Initialization routine (`initEKF.m`), called once at startup
  - Update routine (`iterEKF.m`), called every sample interval, which starts with:

```
function [zk,zkbnd,ekfData] = iterEKF(vk,ik,Tk,deltat,ekfData)
  model = ekfData.model;
  % Load the cell model parameters
  Q  = getParamESC('QParam',Tk,model);
  G  = getParamESC('GParam',Tk,model);
  M  = getParamESC('MParam',Tk,model);
  M0 = getParamESC('M0Param',Tk,model);
  RC = exp(-deltat./abs(getParamESC('RCParam',Tk,model)))';
  R  = getParamESC('RParam',Tk,model)';
  R0 = getParamESC('R0Param',Tk,model);
  eta = getParamESC('etaParam',Tk,model);
  if ik<0, ik=ik*eta; end;
```

EKF iteration code, load covariances/states

- EKF iteration code continues
  - Load covariances, states from prior iteration

```
  % Get data stored in ekfData structure
  I = ekfData.priorI;
  SigmaX = ekfData.SigmaX;
  SigmaV = ekfData.SigmaV;
  SigmaW = ekfData.SigmaW;
  xhat = ekfData.xhat;
  irInd = ekfData.irInd;
  hkInd = ekfData.hkInd;
  zkInd = ekfData.zkInd;
  if abs(ik)>Q/100, ekfData.signIk = sign(ik); end;
  signIk = ekfData.signIk;
```

EKF iteration code, steps 1a–1b

- EKF iteration code continues
  - Compute $\widehat{A}_{k-1}$, $\widehat{B}_{k-1}$, and implement steps 1a and 1b

```
  % Step 1a: Compute Ahat[k-1], Bhat[k-1]; Then, state estimate time update
  nx = length(xhat); Ahat = zeros(nx,nx); Bhat = zeros(nx,1);
  Ahat(zkInd,zkInd) = 1; Bhat(zkInd) = -deltat/(3600*Q);
  Ahat(irInd,irInd) = diag(RC); Bhat(irInd) = 1-RC(:);
  Ah   = exp(-abs(I*G*deltat/(3600*Q)));  % hysteresis factor
  Ahat(hkInd,hkInd) = Ah;
  B = [Bhat, 0*Bhat];
  Bhat(hkInd) = -abs(G*deltat/(3600*Q))*Ah*(1+sign(I)*xhat(hkInd));
  B(hkInd,2) = Ah-1;

  xhat = Ahat*xhat + B*[I; sign(I)];

  % Step 1b: Error covariance time update
  % sigmaminus(k) = Ahat(k-1)*sigmaplus(k-1)*Ahat(k-1)' + ...
  %                 Bhat(k-1)*sigmawtilde*Bhat(k-1)'
  SigmaX = Ahat*SigmaX*Ahat' + Bhat*SigmaW*Bhat';
```

EKF iteration code, steps 1c–2a

- EKF iteration code continues
  - □ Output estimate; estimator gain matrix

```
% Step 1c: Output estimate
yhat = OCVfromSOCtemp(xhat(zkInd),Tk,model) + M0*signIk + ...
       M*xhat(hkInd) - R*xhat(irInd) - R0*ik;

% Step 2a: Estimator gain matrix
Chat = zeros(1,nx);
Chat(zkInd) = dOCVfromSOCtemp(xhat(zkInd),Tk,model);
Chat(hkInd) = M;
Chat(irInd) = -R;
Dhat = 1;
SigmaY = Chat*SigmaX*Chat' + Dhat*SigmaV*Dhat';
L = SigmaX*Chat'/SigmaY;
```

EKF iteration code, steps 2b–2c

- EKF iteration code continues
  - □ State and covariance measurement updates

```
% Step 2b: State estimate measurement update
r = vk - yhat; % residual.  Use to check for sensor errors...
if r^2 > 100*SigmaY, L(:)=0.0; end
xhat = xhat + L*r;
xhat(hkInd) = min(1,max(-1,xhat(hkInd))); % Help maintain robustness
xhat(zkInd) = min(1.05,max(-0.05,xhat(zkInd)));

% Step 2c: Error covariance measurement update
SigmaX = SigmaX - L*SigmaY*L';
if r^2 > 4*SigmaY, % bad voltage estimate by 2 std. devs, bump SigmaX
  fprintf('Bumping SigmaX\n');
  SigmaX(zkInd,zkInd) = SigmaX(zkInd,zkInd)*ekfData.Qbump;
end
[~,S,V] = svd(SigmaX);
HH = V*S*V';
SigmaX = (SigmaX + SigmaX' + HH + HH')/4; % Help maintain robustness
```

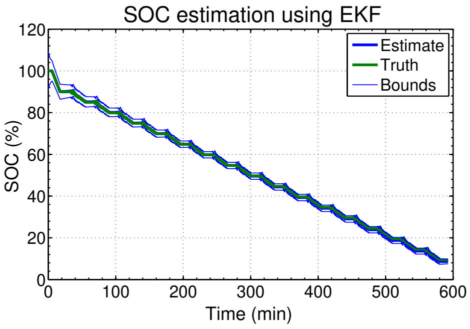
EKF iteration code, cleanup

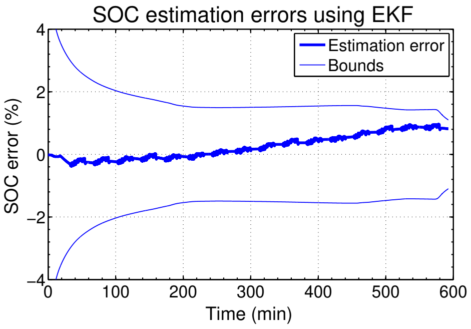
- EKF iteration code continues
  - □ Store results for next time through loop

```
  % Save data in ekfData structure for next time...
  ekfData.priorI = ik;
  ekfData.SigmaX = SigmaX;
  ekfData.xhat = xhat;
  zk = xhat(zkInd);
  zkbnd = 3*sqrt(SigmaX(zkInd,zkInd));
end
```

Example EKF on ESC results

- In following example, EKF was executed for a test having dynamic profiles from 100 % SOC down to around 10 % SOC
  - □ RMS SOC estimation error = 0.46 %
  - □ Percent of time error outside bounds = 0 %


SOC estimation using EKF


SOC estimation errors using EKF

Summary

- Implementation of EKF on ESC model refactors code
  - □ Initialization routine (`initEKF.m`), called once at startup
  - □ Update routine (`iterEKF.m`), called every sample interval
  - □ “Wrapper” code, coordinates the entire simulation process
- You have now seen the details of the entire codeset plus some example results
- EKF works quite well as SOC estimator using ESC model