An EKF example

- We will look at two examples of implementing EKF
  1. A simple example, with fairly straightforward math
  2. The battery-cell example
- In this lesson, we implement EKF for model having following dynamics:

$$x_{k+1} = f(x_k, u_k, w_k) = \sqrt{5 + x_k} + w_k$$
$$y_k = h(x_k, u_k, v_k) = x_k^3 + v_k$$

with $\Sigma_{\widetilde{w}} = 1$ and $\Sigma_{\widetilde{v}} = 2$

Computing the derivatives

- To implement EKF, we must determine $\hat{A}_k$, $\hat{B}_k$, $\hat{C}_k$, and $\hat{D}_k$

$$\hat{A}_k = \left.\frac{\partial f(x_k, u_k, w_k)}{\partial x_k}\right|_{x_k = \hat{x}_k^+} = \left.\frac{\partial\left(\sqrt{5 + x_k} + w_k\right)}{\partial x_k}\right|_{x_k = \hat{x}_k^+} = \frac{1}{2\sqrt{5 + \hat{x}_k^+}}$$

$$\hat{B}_k = \left.\frac{\partial f(x_k, u_k, w_k)}{\partial w_k}\right|_{w_k = \bar{w}_k} = \left.\frac{\partial\left(\sqrt{5 + x_k} + w_k\right)}{\partial w_k}\right|_{w_k = \bar{w}_k} = 1$$

$$\hat{C}_k = \left.\frac{\partial h(x_k, u_k, v_k)}{\partial x_k}\right|_{x_k = \hat{x}_k^-} = \left.\frac{\partial\left(x_k^3 + v_k\right)}{\partial x_k}\right|_{x_k = \hat{x}_k^-} = 3(\hat{x}_k^-)^2$$

$$\hat{D}_k = \left.\frac{\partial h(x_k, u_k, v_k)}{\partial v_k}\right|_{v_k = \bar{v}_k} = \left.\frac{\partial\left(x_k^3 + v_k\right)}{\partial v_k}\right|_{v_k = \bar{v}_k} = 1$$

EKF initialization code

- Code to implement EKF starts below
  - Define simulation constants; reserve storage

```
% Initialize simulation variables
SigmaW = 1; % Process noise covariance
SigmaV = 2; % Sensor noise covariance
maxIter = 40;

xtrue = 2 + randn(1); % Initialize true system initial state
xhat = 2;             % Initialize Kalman filter initial estimate
SigmaX = 1;           % Initialize Kalman filter covariance
u = 0;                % Unknown initial driving input: assume zero

% Reserve storage for variables we might want to plot/evaluate
xstore = zeros(maxIter+1,length(xtrue)); xstore(1,:) = xtrue;
xhatstore = zeros(maxIter,length(xhat));
SigmaXstore = zeros(maxIter,length(xhat)^2);
```

EKF steps 1a through 1b

- Main EKF loop starts below
  - Also co-simulating system dynamics for sensor inputs

```
for k = 1:maxIter,
  % EKF Step 1a: State estimate time update
  % (First compute Ahat, Bhat: Specifics depend on model!)
  % Note: For this example, x(k+1) = sqrt(5+x(k)) + w(k)
  % Note: You need to insert your system's f(...) equation here
  Ahat = 0.5/sqrt(5+xhat); Bhat = 1;
  xhat = sqrt(5+xhat);

  % EKF Step 1b: Error covariance time update
  SigmaX = Ahat*SigmaX*Ahat' + Bhat*SigmaW*Bhat';

  % [Co-simulate system, with input signal u, and output signal y]
  w = chol(SigmaW)'*randn(1);
  v = chol(SigmaV)'*randn(1);
  ytrue = xtrue^3 + v;  % y is based on present x and u
  xtrue = sqrt(5+xtrue) + w;  % future x is based on present u
```


EKF steps 1c through 2b

- Main EKF loop continues below
  - Notice the “extra” robustness code at end

```
  % EKF Step 1c: Estimate system output
  % (First compute Ahat, Bhat: Specifics depend on model!)
  % Note: For this example, y(k) = x(k)^3
  % Note: You need to insert your system's h(...) equation here
  Chat = 3*xhat^2; Dhat = 1;
  yhat = xhat^3;

  % EKF Step 2a: Compute Kalman gain matrix
  SigmaY = Chat*SigmaX*Chat' + Dhat*SigmaV*Dhat';
  L = SigmaX*Chat'/SigmaY;

  % EKF Step 2b: State estimate measurement update
  xhat = xhat + L*(ytrue - yhat);
  xhat = max(-5,xhat); % don't get square root of negative xhat!
```


EKF step 2c

- Main EKF loop concludes below
  - Includes code to force `SigmaX` to be PSD

```
  % EKF Step 2c: Error covariance measurement update
  SigmaX = SigmaX - L*SigmaY*L';
  [~,S,V] = svd(SigmaX);
  HH = V*S*V';
  SigmaX = (SigmaX + SigmaX' + HH + HH')/4; % Help to keep robust

  % [Store information for evaluation/plotting purposes]
  xstore(k+1,:) = xtrue; xhatstore(k,:) = xhat;
  SigmaXstore(k,:) = (SigmaX(:))';
end;
```

EKF plotting code

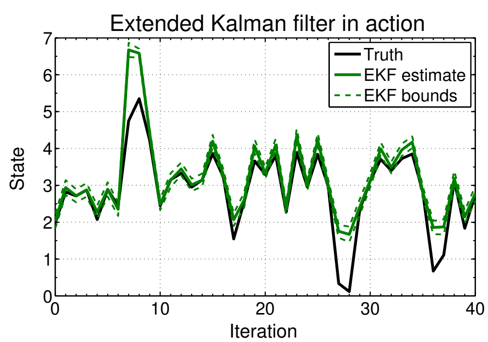
- This is an example showing how to plot the results from this EKF code in two different ways

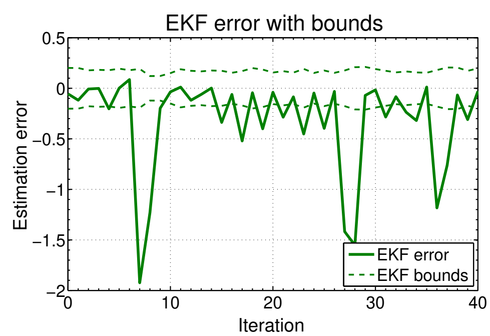
```
subplot(1,2,1);
t = 0:maxIter-1;
plot(t,xstore(1:maxIter),'k-',t,xhatstore,'b--', ...
  t,xhatstore+3*sqrt(SigmaXstore),'m-.',...
  t,xhatstore-3*sqrt(SigmaXstore),'m-.'); grid;
legend('true','estimate','bounds');
xlabel('Iteration'); ylabel('State');
title('Extended Kalman filter in action');

subplot(1,2,2);
plot(t,xstore(1:maxIter)-xhatstore,'b-',t, ...
  3*sqrt(SigmaXstore),'m--',t,-3*sqrt(SigmaXstore),'m--');
grid; legend('Error','bounds',0);
title('EKF Error with bounds');
xlabel('Iteration'); ylabel('Estimation error');
```


Representative results

- Figures below show representative results
  - EKF works well for small states, when system is fairly linear
  - EKF struggles for larger states, when system is more nonlinear





Summary

- Have now seen code to implement EKF on relatively simple nonlinear state-space model
- Finding derivatives was most difficult part to do correctly (but, not too bad for this simple model)
- Actual code was straightforward implementation of steps seen earlier this week
- Results show that EKF provides good estimates and bounds only for operating regimes where the model is nearly linear (as expected)
- Estimates and bounds are poorer when the model is being operated far away from a linear region