An SPKF example

- We will look at two examples of implementing SPKF
  1. A simple example, with fairly straightforward math
  2. The battery-cell example
- In this lesson, we implement SPKF 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 $\Sigma_{\tilde{v}} = 2$
- Same example as was used when introducing EKF

SPKF initialization code

- Code to implement SPKF starts below
  - Define SPKF constants

```
% Define size of variables in model
Nx = 1;      % state  = 1x1 scalar
Nxa = 3;     % augmented state has also w(k) and v(k) contributions
Ny = 1;      % output = 1x1 scalar

% Some constants for the SPKF algorithm. Use standard values for
% cases with Gaussian noises.  (These are the weighting matrices
% comprising the values of alpha(c) and alpha(m) organized in a
% way to make later computation efficient).
h = sqrt(3);
Wmx(1) = (h*h-Nxa)/(h*h); Wmx(2) = 1/(2*h*h); Wcx=Wmx;
Wmxy = [Wmx(1) repmat(Wmx(2),[1 2*Nxa])]';
```


SPKF initialization code

- Code to implement SPKF 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

% 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);
```

SPKF step 1a

- Main SPKF loop starts below
  - Augment state estimate and covariance first

```
for k = 1:maxIter,
  % SPKF Step 1a: State estimate time update
  % 1a-i: Calculate augmented state estimate, including ...
  xhata = [xhat; 0; 0]; % process and sensor noise mean
  % 1a-ii: Get desired Cholesky factor
  Sigmaxa = blkdiag(SigmaX,SigmaW,SigmaV);
  sSigmaxa = chol(Sigmaxa,'lower');
  % 1a-iii: Calculate sigma points (strange indexing of xhat to avoid
  % "repmat" call, which is very inefficient in Matlab)
  X = xhata(:,ones([1 2*Nxa+1])) + h*[zeros([Nxa 1]), ...
                                    sSigmaxa, -sSigmaxa];
  % 1a-iv: Calculate state equation for every element
  % Hard-code equation here for efficiency
  Xx = sqrt(5+X(1,:)) + X(2,:);
  xhat = Xx*Wmxy;
```


SPKF steps 1b–1c

- Main SPKF loop continues below
  - Also co-simulating system dynamics for sensor inputs

```
  % SPKF Step 1b: Covariance of prediction
  Xs = Xx - xhat(:,ones([1 2*Nxa]));
  SigmaX = Xs*diag(Wmxy)*Xs';

  % [Implied operation of system in background, 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

  % SPKF Step 1c: Create output estimate
  % Hard-code equation here for efficiency
  Y = Xx.^3 + X(3,:);
  yhat = Y*Wmxy;
```


SPKF steps 2a–2b

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

```
  % SPKF Step 2a: Estimator gain matrix
  Ys = Y - yhat*ones([1 2*Nxa]);
  SigmaXY = Xs*diag(Wmxy)*Ys';
  SigmaY  = Ys*diag(Wmxy)*Ys';
  Lx= SigmaXY/SigmaY;

  % SPKF Step 2b: Measurement state update
  xhat = xhat + Lx*(ytrue-yhat); % update prediction to estimate
  xhat = max(-5,xhat); % don't get square root of negative
```

SPKF step 2c

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

```
  % SPKF Step 2c: Measurement covariance update
  SigmaX = SigmaX - Lx*SigmaY*Lx';
  [~,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
```


SPKF plotting code

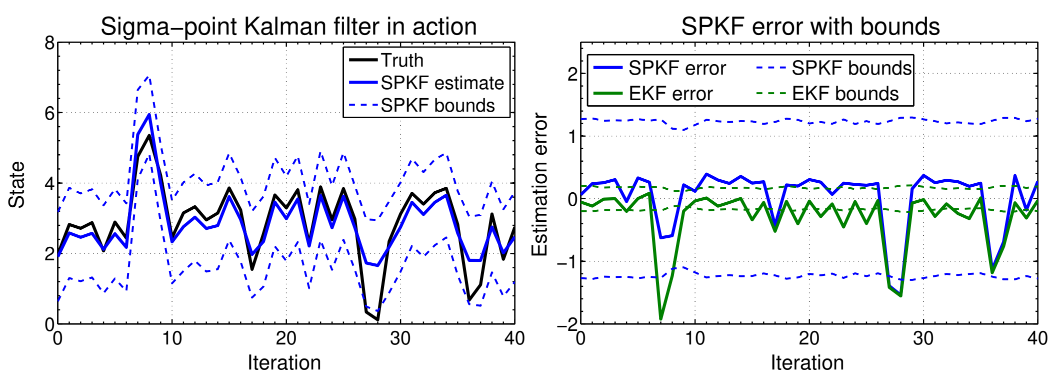
- This is an example showing how to plot the results from this SPKF code in two different ways

```
subplot(1,2,1);
plot(0:maxIter-1,xstore(1:maxIter),'k-',0:maxIter-1,xhatstore,'b--', ...
  0:maxIter-1,xhatstore+3*sqrt(SigmaXstore),'m-.',...
  0:maxIter-1,xhatstore-3*sqrt(SigmaXstore),'m-.'); grid on;
legend('true','estimate','bounds');
title('Sigma-point Kalman filter in action');
xlabel('Iteration'); ylabel('State');

subplot(1,2,2);
plot(0:maxIter-1,xstore(1:maxIter)-xhatstore,'-',0:maxIter-1, ...
  3*sqrt(SigmaXstore),'--',0:maxIter-1,-3*sqrt(SigmaXstore),'--');
grid on; legend('Error','bounds',0); title('SPKF Error with bounds');
xlabel('Iteration'); ylabel('Estimation Error');
```


SPKF representative results

- Figures below show representative results
  - Estimation accuracy improved over EKF
  - Error bounds greatly improved over EKF



Summary

- Have now seen code to implement SPKF on relatively simple nonlinear state-space model
- Actual code was straightforward implementation of steps seen earlier this week
- Results show that SPKF provides somewhat better estimates than EKF
- Results also show SPKF provides much better error-bounds estimates than EKF
- Improvements are greatest when model is being operated far away from a linear region