Kalman-filter smoothing

- Smoothing is the estimation of the system state at a time m amid the data interval. That is (where $m < N$),

$$\hat{x}^+_{m|N} = \mathbb{E}[x_m \mid \mathbb{Z}_N].$$

- There are three different smoothing scenarios:
  - Fixed-point smoothing: Find $\hat{x}^+_{m|k}$ where m is fixed, but k is changing as more data becomes available;
  - Fixed-lag smoothing: Find $\hat{x}^-_{k-L|k}$ where L is a fixed lag time;
  - Fixed-interval smoothing: Find $\hat{x}^+_{m|N}$ where k is fixed, but m can take on multiple past values.
- We will discuss all three in this lesson but focus on fixed-interval smoothing; the others use a variation of this idea.

Fixed-interval smoothing: State estimate

- The fixed-interval-smoothing algorithm consists of a forward recursive pass followed by a backward pass.
- The forward pass uses a Kalman filter and saves the intermediate results $\hat{x}^-_k$, $\hat{x}^+_k$, $\Sigma^-_{\tilde{x},k}$, and $\Sigma^+_{\tilde{x},k}$.
- The backward pass starts at time N of the last measurement, and computes the smoothed state estimate using the results obtained from the forward pass.
- The recursive equations of the backward sweep for estimating the state are:

$$\hat{x}^+_{m|N} = \hat{x}^+_m + \lambda_m \left(\hat{x}^+_{m+1|N} - \hat{x}^-_{m+1}\right)$$
$$\lambda_m = \Sigma^+_{\tilde{x},m} A^T_m \left(\Sigma^-_{\tilde{x},m+1}\right)^{-1}$$

where $m = N-1, N-2, \ldots, 0$. Note, $\hat{x}^+_{N|N} = \hat{x}^+_N$ to start backward pass.

Fixed-interval smoothing: Covariance estimate

- We can also compute the smoothed estimation-error covariance matrix (for developing confidence bounds).
- The error covariance matrix for the smoothed estimate is

$$\Sigma^+_{\tilde{x},m|N} = \Sigma^+_{\tilde{x},m} + \lambda_m \left[\Sigma^+_{\tilde{x},m+1|N} - \Sigma^-_{\tilde{x},m+1}\right] \lambda^T_m.$$

- Notice (from the prior equations) that we are not required to compute this quantity to be able to perform the backward pass.
- Also note that the term in the square brackets is negative semi-definite, so the covariance of the smoothed estimate is “smaller” than for the filtered estimate only.

Fixed point smoothing

- Here, m is fixed, and the final point k keeps increasing.

$$\hat{x}^+_{m|k} = \hat{x}^+_{m|k-1} + \mu_k \left(\hat{x}^+_k - \hat{x}^-_k\right) \quad \text{where} \quad \mu_k = \prod_{i=m}^{k-1} \lambda_i,$$

  where the product multiplies on the left as i increases.
  - For example, for $k = m + 1$,

$$\hat{x}^+_{m|m+1} = \hat{x}^+_m + \mu_{m+1} \left(\hat{x}^+_{m+1} - \hat{x}^-_{m+1}\right)$$
$$\mu_{m+1} = \lambda_m = \Sigma^+_{\tilde{x},m} A^T_m \left(\Sigma^-_{\tilde{x},m+1}\right)^{-1}.$$

  - Then, for $k = m + 2$, (and so forth for $k > m + 2$)

$$\hat{x}^+_{m|m+2} = \hat{x}^+_{m|m+1} + \mu_{m+2} \left(\hat{x}^+_{m+2} - \hat{x}^-_{m+2}\right)$$
$$\mu_{m+2} = \Sigma^+_{\tilde{x},m+1} A^T_{m+1} \left(\Sigma^-_{\tilde{x},m+2}\right)^{-1} \mu_{k+1}.$$

Fixed-lag smoothing

- Here, we seek to estimate the state vector at a fixed time interval lagging the time of the current measurement.
  - This type of smoothing trades off estimation latency for more accuracy.
  - The fixed interval smoothing algorithm could be used to perform fixed-lag smoothing when the number of backward steps equals the time lag
  - This is fine as long as the number of backward steps is small.
  - Fixed-lag smoothing algorithm has a startup problem: Cannot run until enough data are available.

Octave code for fixed-interval smoothing

```
clearvars
load simOut.mat Ad Bd Cd Dd SigmaV SigmaW dT t u x z

% Initialize simulation variables
[nx,nt] = size(x); [nz,~] = size(z);

xhat = zeros(nx,1);      % Initialize Kalman filter initial estimate
SigmaX = zeros(nx,nx);   % Initialize Kalman filter covariance

% Reserve storage for variables we might want to plot/evaluate
xhatstore = zeros(nx,nt);         % storage for the forward state estimate
boundstore = xhatstore;           % storage for state-estimate bounds
xhatMstore = zeros(nx,nt);        % xhat-minus storage, used by backward pass
SigmaXstore = zeros(nx,nx,nt);    % Sigma-plus storage, used by backward pass
SigmaXMstore = zeros(nx,nx,nt);   % Sigma-minus storage, used by backward pass
xhatstore(:,1) = xhat;            % store xhat (i.e., xhat-plus) at time zero
xhatMstore(:,1) = xhat;           % same with xhat-minus
SigmaXstore(:,:,1) = SigmaX;      % store Sigma-plus at time zero
SigmaXMstore(:,:,1) = SigmaX;     % store Sigma-minus at time zero
```

Octave code for fixed-interval smoothing

- The main program loop begins:

```
for k = 2:nt
  % KF Step 1a: State prediction time update
  xhat = Ad*xhat + Bd*u(:,k-1); % use prior value of "u"
  % KF Step 1b: Prediction-error covariance time update
  SigmaX = Ad*SigmaX*Ad' + SigmaW;

  % Store prediction data for smoothing
  xhatMstore(:,k) = xhat;        % store xhat-minus
  SigmaXMstore(:,:,k) = SigmaX; % store sigma-minus

  % KF Step 1c: Estimate system output
  zhat = Cd*xhat + Dd*u(k);
  % KF Step 2a: Compute Kalman gain matrix
  L = SigmaX*Cd'/(Cd*SigmaX*Cd' + SigmaV);
  % KF Step 2b: State estimate measurement update
  xhat = xhat + L*(z(k) - zhat);
  % KF Step 2c: Estimation-error covariance measurement update
  SigmaX = SigmaX - L*Cd*SigmaX;
```


Octave code for fixed-interval smoothing

```
  % Store estimate data for smoothing and KF output
  SigmaXstore(:,:,k) = SigmaX; % store sigma-plus
  xhatstore(:,k) = xhat;       % store xhat (i.e., xhat-plus)
  boundstore(:,k) = 3*sqrt(diag(SigmaX)); % store xhat bounds
end

% % Now, do backward pass...
xhatSstore = xhatstore;       % smoothed estimate
boundSstore = boundstore;     % smoothed bounds
SigmaXSstore = SigmaXstore;   % smoothed covariance
for k = nt-1:-1:1
  Sp = SigmaXstore(:,:,k);    Sm = SigmaXMstore(:,:,k+1);
  lambda = Sp*Ad'/Sm;
  xhatSstore(:,k) = xhatstore(:,k) + ... % compute smoothed estimate
    lambda*(xhatSstore(:,k+1) - xhatMstore(:,k+1)); % and store it
  Spp = SigmaXSstore(:,:,k+1);
  Sp = Sp + lambda*(Spp - Sm)*lambda';    % compute smoothed covariance
  SigmaXSstore(:,:,k) = Sp;               % store smoothed covariance
  boundSstore(:,k) = 3*sqrt(diag(Sp));    % store smoothed estimate bounds
end
```


Octave code for fixed-interval smoothing

- Plot states, estimates, and smoothed estimates:

```
CL = lines;
figure(1); clf;
t2 = [t fliplr(t)]; % Prepare for plotting bounds via "fill"
x2 = [xhatstore-boundstore fliplr(xhatstore+boundstore)];
x3 = [xhatSstore-boundSstore fliplr(xhatSstore+boundSstore)];
h1 = fill(t2,x2,CL(1,:),'FaceAlpha',0.15,'LineStyle','none'); hold on; grid on
h3 = fill(t2,x3,CL(5,:),'FaceAlpha',0.20,'LineStyle','none');
set(gca,'ColorOrderIndex',1);
h2 = plot(t,x(1:2,:)',t,xhatstore(1:2,:)','--');
h4 = plot(t,xhatSstore(1:2,:)','--');
legend([h2;h4;h1(1);h3(1)],{'True posn.','True vel.','Posn. est.',...
  'Vel. est.','Posn. smooth','Vel. smooth','KF bounds','Smooth bounds'},...
    'NumColumns',3);
title('KF state estimates with smoothing');
xlabel('Time (s)'); ylabel('State (m or m/s)');
```

Octave code for fixed-interval smoothing

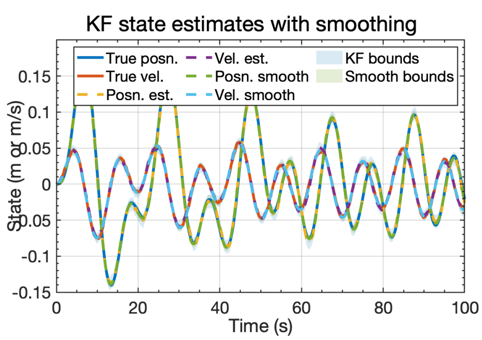
- Plot estimation errors and smoothed-estimate errors:

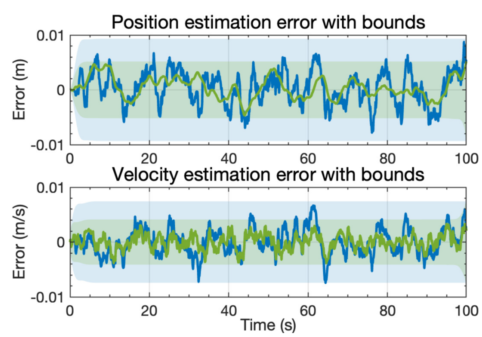
```
figure(2); clf;
xerr = x - xhatstore; xSerr = x - xhatSstore;
subplot(2,1,1);
fill([t fliplr(t)],[-boundstore(1,:) fliplr(boundstore(1,:))],CL(1,:),...
  'FaceAlpha',0.15,'LineStyle','none'); hold on; grid on;
fill([t fliplr(t)],[-boundSstore(1,:) fliplr(boundSstore(1,:))],CL(5,:),...
  'FaceAlpha',0.20,'LineStyle','none');
plot(t,xerr(1,:),'Color',CL(1,:)); plot(t,xSerr(1,:),'Color',CL(5,:));
title('Position estimation error with bounds'); ylabel('Error (m)');

subplot(2,1,2);
fill([t fliplr(t)],[-boundstore(2,:) fliplr(boundstore(2,:))],CL(1,:),...
  'FaceAlpha',0.15,'LineStyle','none'); hold on; grid on;
fill([t fliplr(t)],[-boundSstore(2,:) fliplr(boundSstore(2,:))],CL(5,:),...
  'FaceAlpha',0.20,'LineStyle','none');
plot(t,xerr(2,:),'Color',CL(1,:)); plot(t,xSerr(2,:),'Color',CL(5,:));
title('Velocity estimation error with bounds');
xlabel('Time (s)'); ylabel('Error (m/s)');
```


Results from the Kalman smoother

- The figures show results of running the smoother.
- The improvement is most noticeable in the error plots where the blue lines (and shading) display the forward estimation error (and bounds) and the green lines (and shading) display the backward smoothed error (and bounds).





Summary

- The Kalman filter can be extended to use “future” data to compute an improved estimate of a “past” state.
- Three common scenarios: Fixed-point smoothing, fixed-lag smoothing, fixed-interval smoothing.
- We focused here on fixed-interval smoothing:
  - A standard KF is run in the “forward” direction and signals are saved.
  - A backward pass is made to refine the KF’s state estimates and bounds.
- You learned how to implement a fixed-interval Kalman smoother in Octave code.
- An example showed that errors are smaller and bounds are tighter using the smoother than simply using a standard Kalman filter.