

CMPE 58K

Bayesian Statistics and Machine Learning

Lecture 9

Inference in Linear Dynamical Systems, Kalman Filter and Smoother



Department of Computer Engineering,
Boğaziçi University, Istanbul, Turkey
Instructor: A. Taylan Cemgil

Fall 2009

Kalman Filter Models, Linear Dynamical Systems

- The latent variables s_k and observations y_k are continuous
- The transition and observations models are linear
 - Example: a point moving on the real line
 - A deterministic dynamical system with two state variables

$$\mathbf{s}_k = \begin{pmatrix} \text{position} \\ \text{velocity} \end{pmatrix}_k = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{s}_{k-1} = \mathbf{A}\mathbf{s}_{k-1}$$

$$y_k = \text{position}_k = \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{s}_k = \mathbf{C}\mathbf{s}_k$$

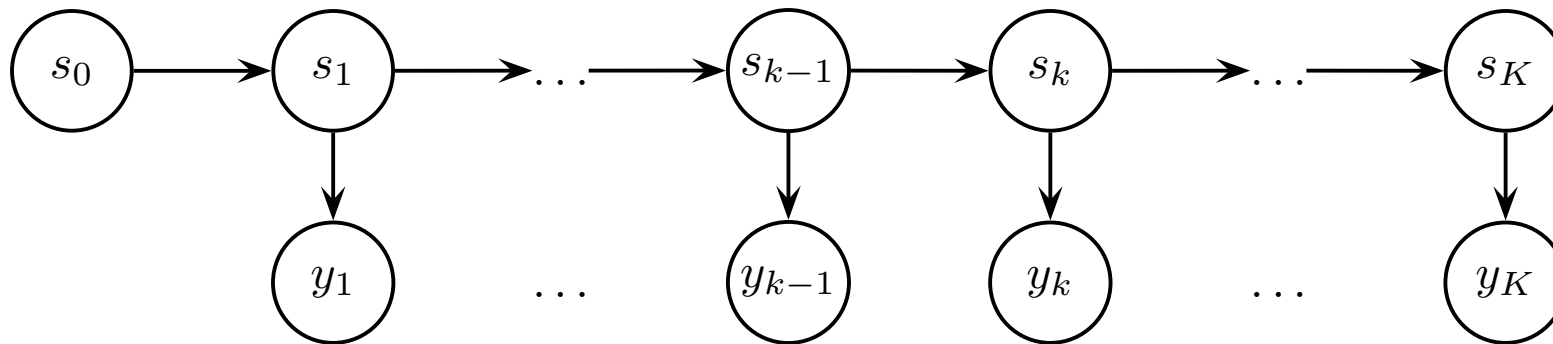
LDS Example

- We allow random (unknown) accelerations

$$\begin{aligned}\mathbf{s}_k &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \mathbf{s}_{k-1} + \epsilon_k \\ &= \mathbf{A}\mathbf{s}_{k-1} + \epsilon_k\end{aligned}$$

$$\begin{aligned}y_k &= \begin{pmatrix} 1 & 0 \end{pmatrix} \mathbf{s}_k + \nu_k \\ &= \mathbf{C}\mathbf{s}_k + \nu_k\end{aligned}$$

LDS Example



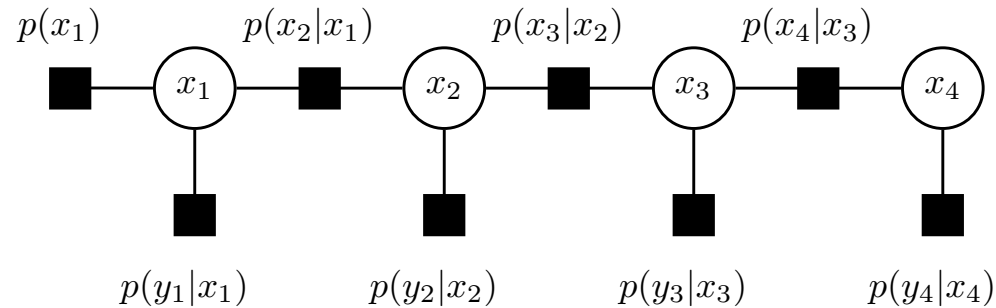
- In generative model notation

$$s_k \sim \mathcal{N}(s_k; \mathbf{A}s_{k-1}, Q)$$

$$y_k \sim \mathcal{N}(y_k; \mathbf{C}s_k, R)$$

- Tracking = estimating the latent state of the system = Kalman filtering

Kalman Filtering and Smoothing (two filter formulation)



- Forward Pass (Kalman Filter)

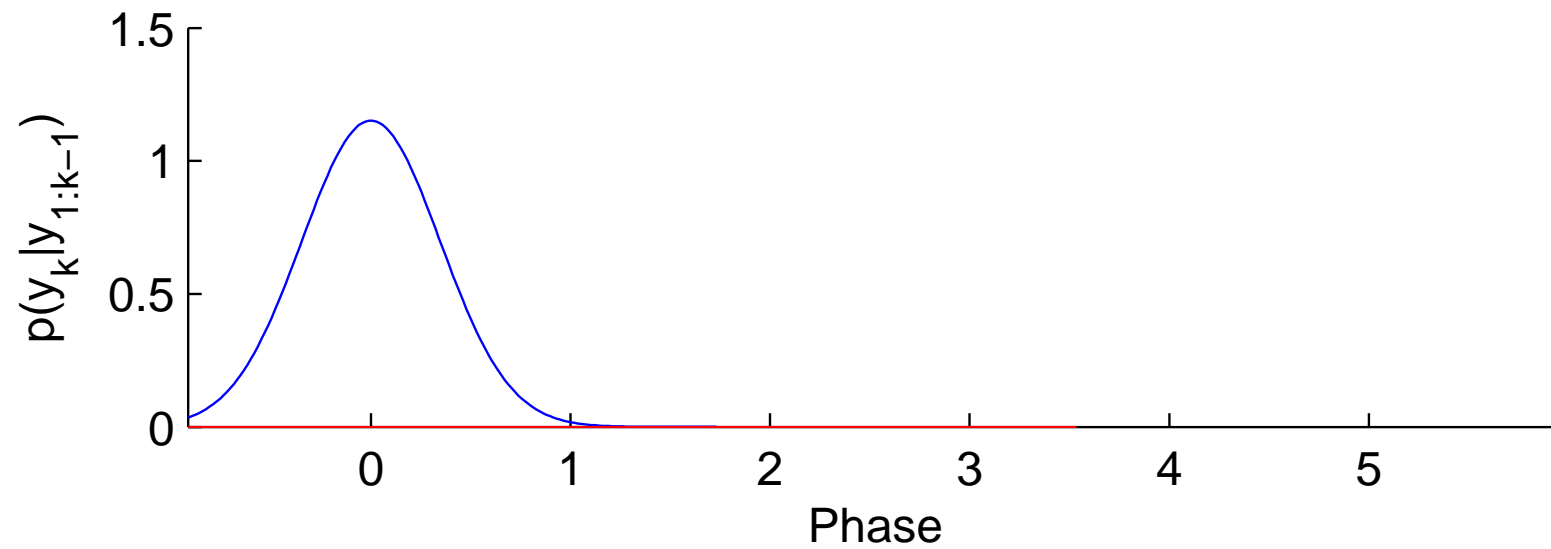
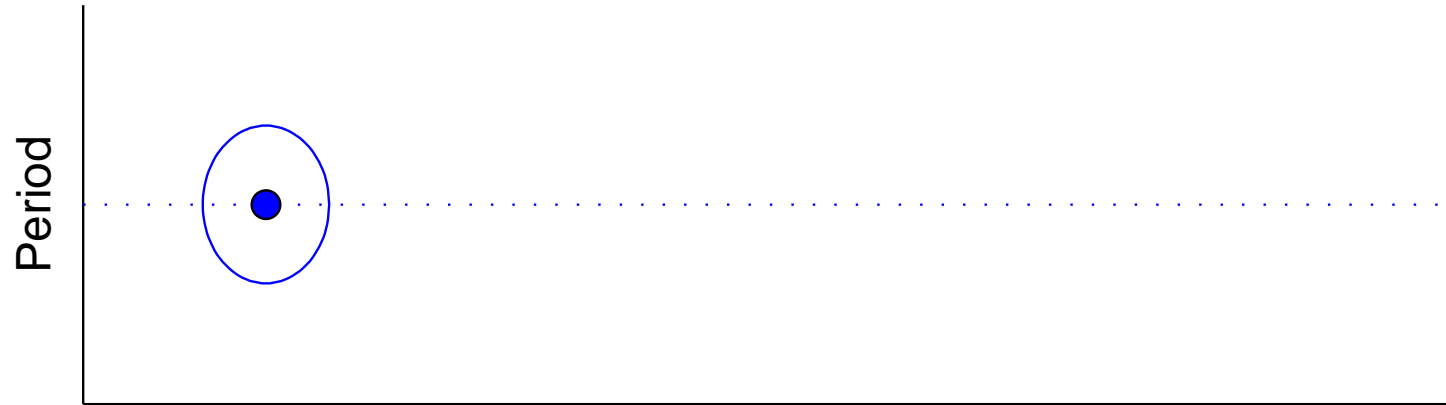
$$p(y_{1:K}) = \underbrace{\int_{x_K} p(y_T|x_K) \int_{x_{K-1}} p(x_K|x_{K-1}) \dots \int_{x_2} p(x_3|x_2) p(y_2|x_2)}_{\alpha_K} \underbrace{\int_{x_1} p(x_2|x_1) p(y_1|x_1)}_{\alpha_2} \underbrace{p(x_1)}_{\alpha_1|0}$$

- Backward Pass

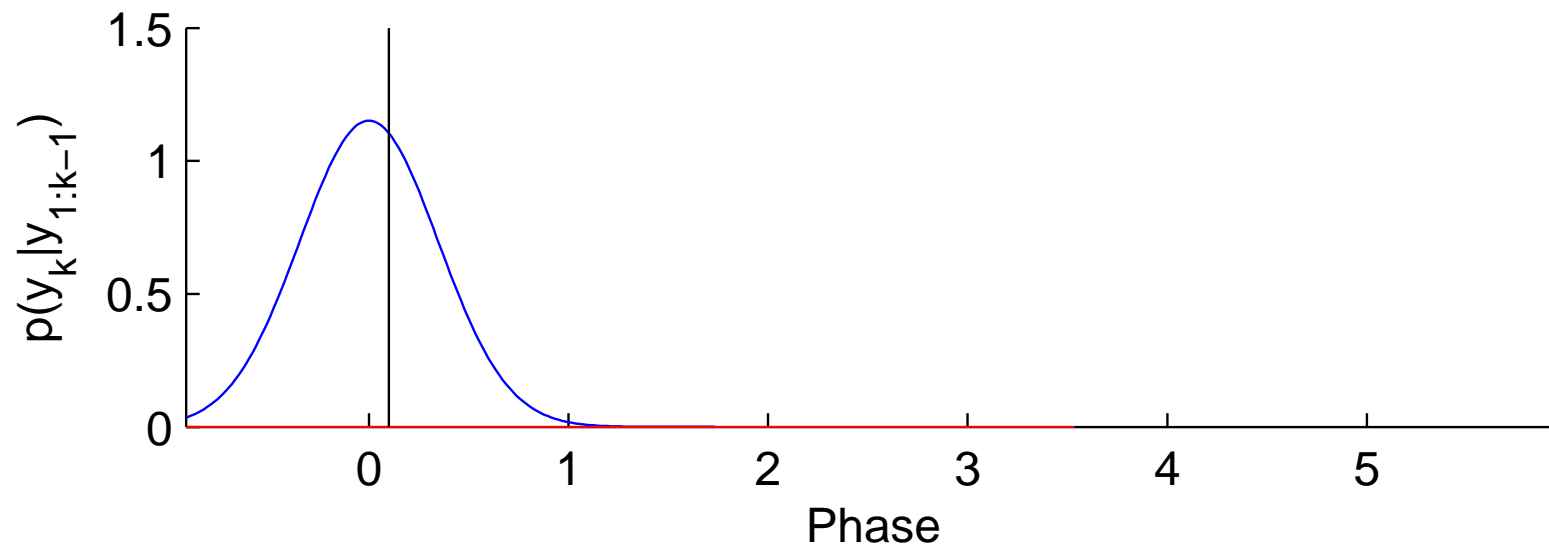
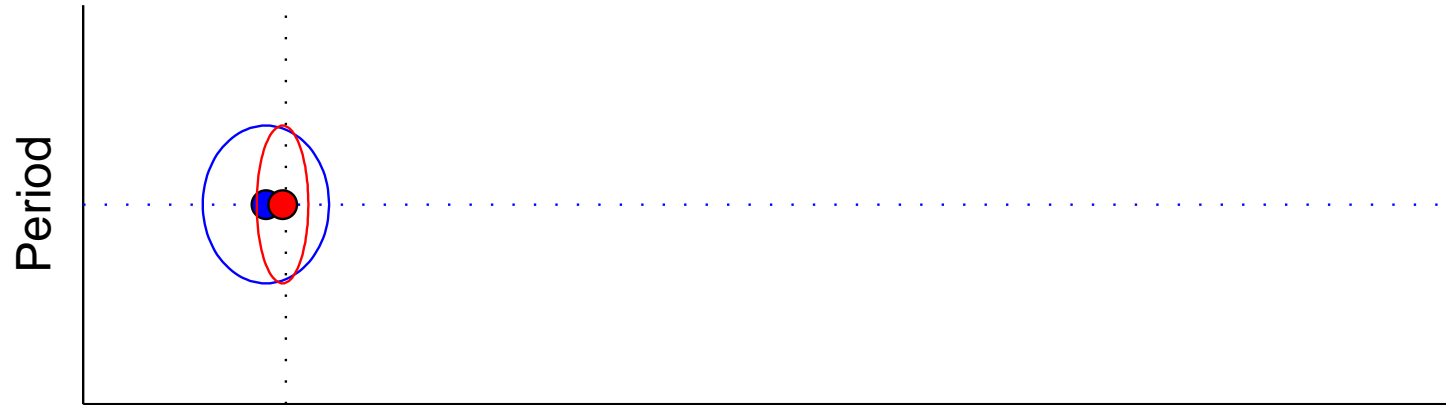
$$p(y_{1:K}) = \int_{x_1} p(x_1) p(y_1|x_1) \dots \underbrace{\int_{x_{K-1}} p(x_{K-1}|x_{K-2}) p(y_{K-1}|x_{K-1})}_{\beta_{K-2}} \underbrace{\int_{x_K} p(x_K|x_{K-1}) p(y_K|x_K)}_{\beta_{K-1}} \underbrace{1}_{\beta_K}$$

- Replace summation by integration

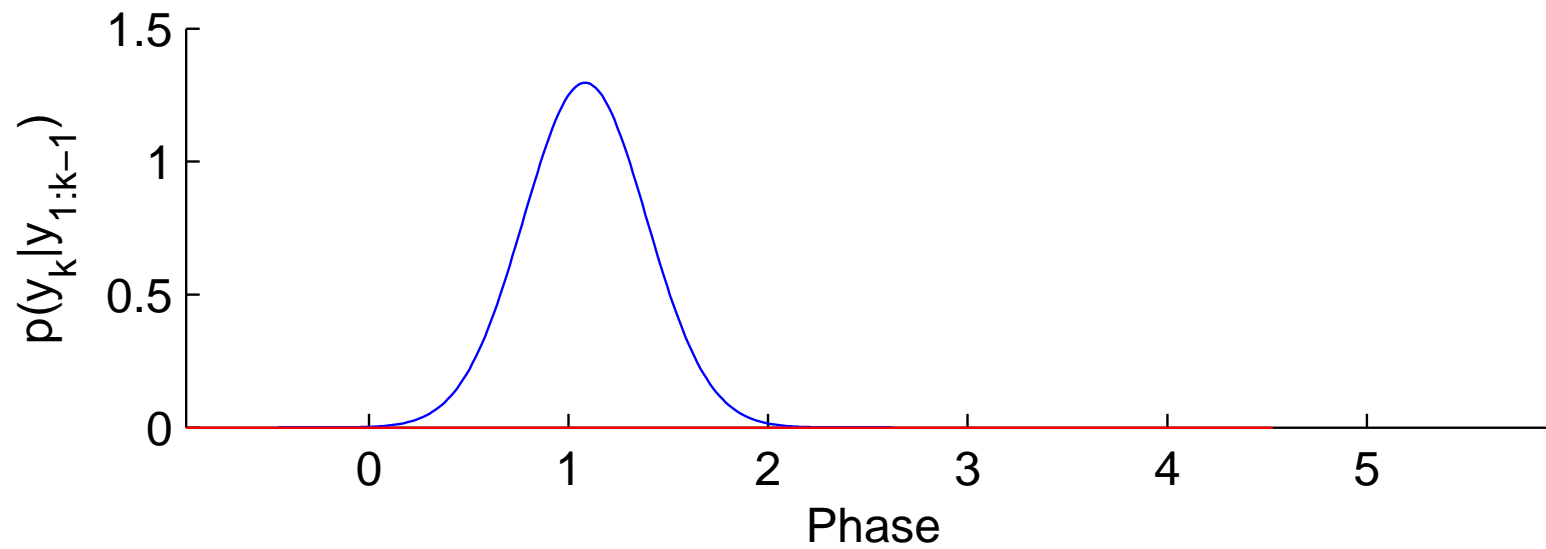
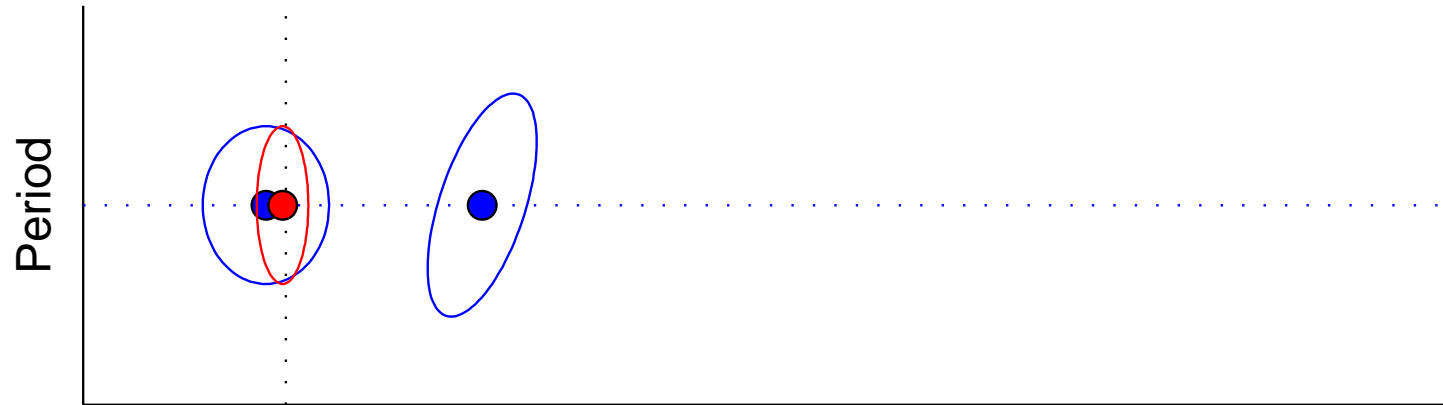
$$p(s_1)$$



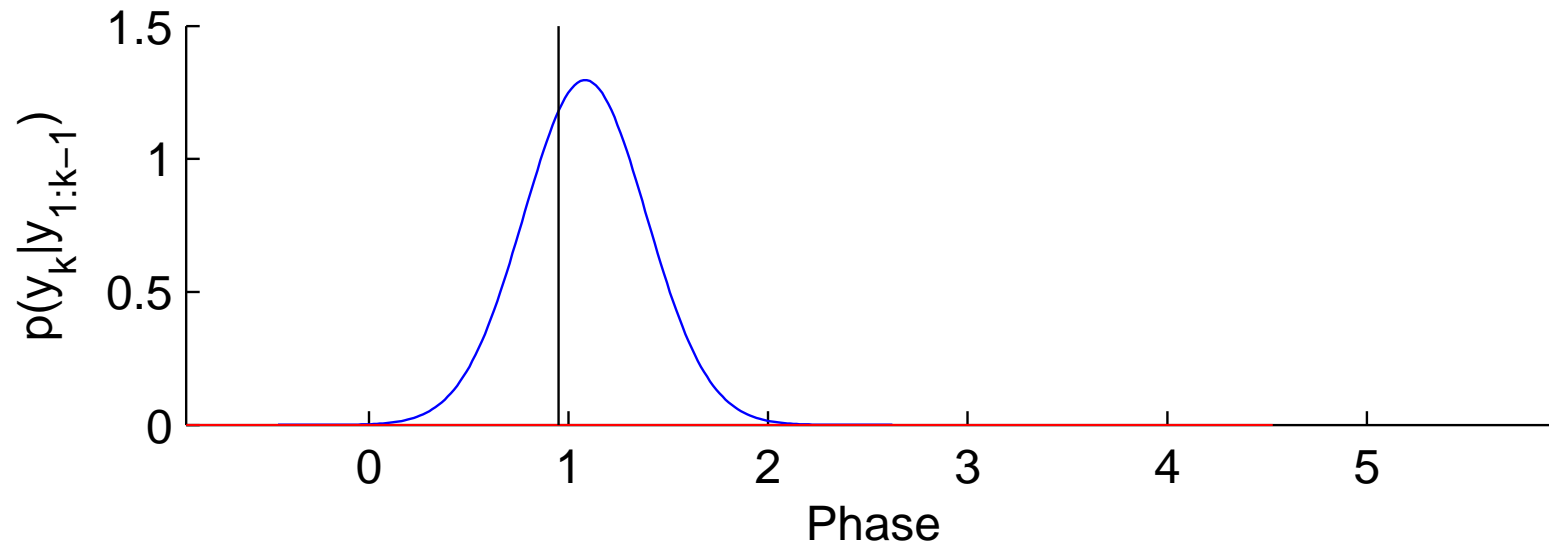
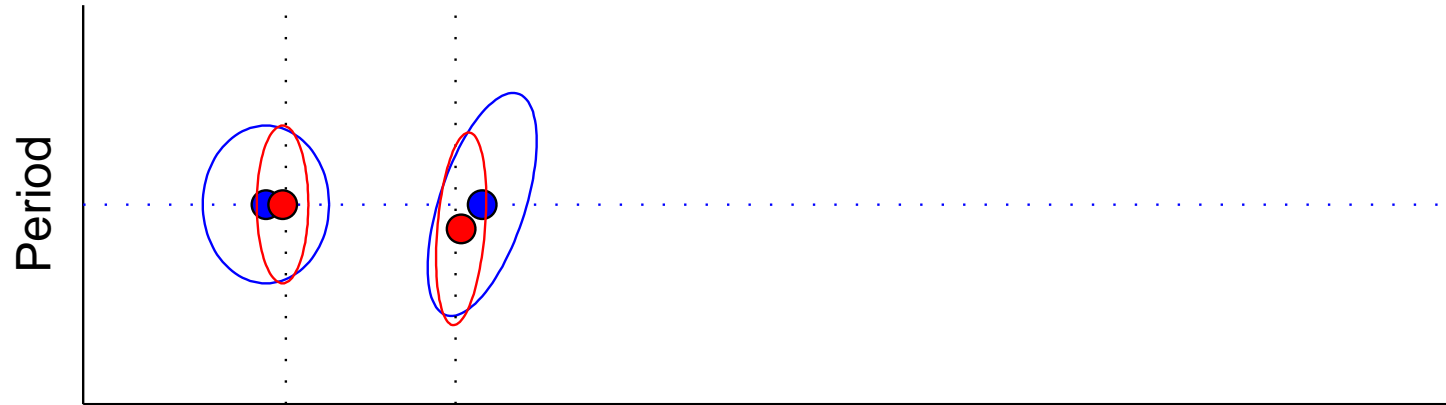
$$p(y_1|s_1)p(s_1)$$



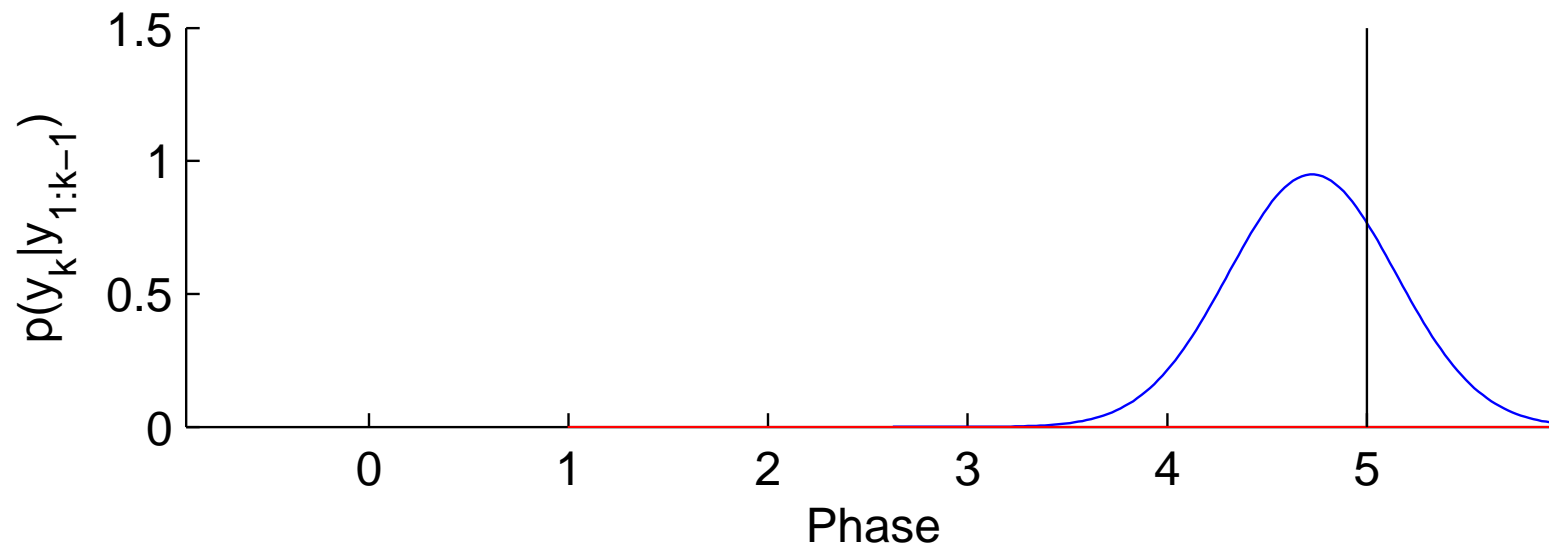
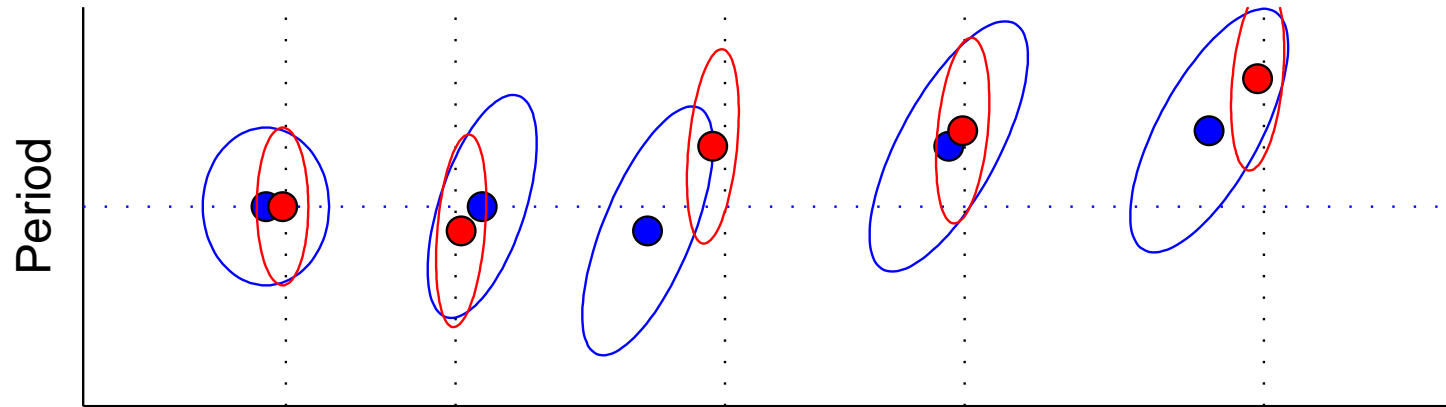
$$p(s_2|y_1) \propto \int ds_1 p(s_2|s_1)p(y_1|s_1)p(s_1)$$



$$p(y_2|s_2)p(s_2|y_1)$$



$$p(s_5 | y_{1:5})$$



The Kalman Filter: Moment form

$$x_k | x_{k-1} \sim \mathcal{N}(x_k; Ax_{k-1}, Q)$$

$$y_k | s_k \sim \mathcal{N}(y_k; Cx_k, R)$$

The Kalman Filter: Moment form

- Forward Pass (Kalman Filter)

$$p(y_{1:K}) = \underbrace{\int_{x_K} p(y_T|x_K) \int_{x_{K-1}} p(x_K|x_{K-1}) \dots \int_{x_2} p(x_3|x_2) p(y_2|x_2)}_{\alpha_K} \underbrace{\int_{x_1} p(x_2|x_1) p(y_1|x_1)}_{\alpha_2} \underbrace{p(x_1)}_{\alpha_1}$$

The Kalman Filter: Moment form

- Moment form: We represent the α messages in *moment* form

$$\begin{aligned}\alpha_{k|k-1} &\equiv p(y_{1:k-1}, x_k) \\ &= p(y_{1:k-1})p(x_k|y_{1:k-1}) \\ &= \exp(l_{k-1})\mathcal{N}(x_k; \mu_{k|k-1}, \Sigma_{k|k-1})\end{aligned}$$

- Note that

$$p(y_{1:k-1}) = \int dx_k p(y_{1:k-1}, x_k) = \exp(l_{k-1})$$

The Kalman Filter: Update equations (Moment form)

- We need to compute

$$\begin{aligned}\alpha_{k|k} &\equiv p(y_{1:k}, x_k) \\ &= \exp(l_k) \mathcal{N}(x_k; \mu_{k|k}, \Sigma_{k|k})\end{aligned}$$

- A product of Gaussian potentials

$$\begin{aligned}\alpha_{k|k} &= p(y_k | x_k) \alpha_{k|k-1} \\ &= p(y_k | x_k) p(y_{1:k-1}) p(x_k | y_{1:k-1}) \\ &= p(y_{1:k-1}) p(y_k | x_k) p(x_k | y_{1:k-1}) \\ &= p(y_{1:k-1}) p(y_k, x_k | y_{1:k-1}) \\ &= \exp(l_{k-1}) \mathcal{N}(y_k; Cx_k, R) \mathcal{N}(x_k; \mu_{k|k-1}, \Sigma_{k|k-1})\end{aligned}$$

Finding $p(y_k, x_k | y_{1:k-1})$

- We know

$$\begin{aligned} p(y_k, x_k | y_{1:k-1}) &= p(y_k | x_k) p(x_k | y_{1:k-1}) \\ &= \mathcal{N}(y_k; Cx_k, R) \mathcal{N}(x_k; \mu_{k|k-1}, \Sigma_{k|k-1}) \end{aligned}$$

- We need the other factorisation

$$p(y_k, x_k | y_{1:k-1}) = p(y_k | y_{1:k-1}) p(x_k | y_k, y_{1:k-1})$$

Finding $p(y_k, x_k | y_{1:k-1})$

- We will consider the vector

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix}$$

- First, we will show that we have the joint density is

$$p(x_k, y_k | y_{1:k-1}) = \mathcal{N} \left(\begin{pmatrix} x_k \\ y_k \end{pmatrix}; \begin{pmatrix} \mu_{k|k-1} \\ C\mu_{k|k-1} \end{pmatrix}, \begin{pmatrix} \Sigma_{k|k-1} & \Sigma_{k|k-1}C^\top \\ C\Sigma_{k|k-1} & C\Sigma_{k|k-1}C^\top + R \end{pmatrix} \right)$$

- Then, we will compute the factorisation

$$p(y_k, x_k | y_{1:k-1}) = p(y_k | y_{1:k-1})p(x_k | y_k, y_{1:k-1})$$

Derivation of $p(y_k, x_k | y_{1:k-1})$

- Remember

$$\begin{aligned}y_k &= Cx_k + \epsilon_k \\ \epsilon_k &\sim \mathcal{N}(\epsilon_k; 0, R)\end{aligned}$$

- Expectations are conditional on observing $y_{1:k-1}$.

$$\begin{aligned}\langle y_k \rangle &= \langle Cx_k + \epsilon_k \rangle = C \langle x_k \rangle + \langle \epsilon_k \rangle = C \langle x_k \rangle \\ &= C \mu_{k|k-1}\end{aligned}$$

Derivation of $p(y_k, x_k | y_{1:k-1})$

- Cross terms

$$\begin{aligned}\text{Cov}[y_k, x_k] &= \langle y_k x_k^\top \rangle - \langle y_k \rangle \langle x_k \rangle^\top \\ y_k x_k^\top &= (C x_k + \epsilon_k) x_k^\top \\ &= C x_k x_k^\top + \epsilon_k x_k^\top \\ \text{Cov}[y_k, x_k] &= C \langle x_k x_k^\top \rangle + \langle \epsilon_k \rangle \langle x_k \rangle^\top - C \mu_{k|k-1} \mu_{k|k-1}^\top \\ &= C \Sigma_{k|k-1}\end{aligned}$$

Derivation of $p(y_k, x_k | y_{1:k-1})$

- Covariance terms

$$\begin{aligned}\text{Cov}[y_k] &= \langle y_k y_k^\top \rangle - \langle y_k \rangle \langle y_k \rangle^\top \\ y_k y_k^\top &= (C x_k + \epsilon_k)(x_k^\top C^\top + \epsilon_k^\top) \\ &= C x_k x_k^\top C^\top + 2 \epsilon_k x_k^\top C^\top + \epsilon_k \epsilon_k^\top \\ \langle y_k y_k^\top \rangle &= \langle C x_k x_k^\top C^\top \rangle + 2 \langle \epsilon_k x_k^\top \rangle C^\top + \langle \epsilon_k \epsilon_k^\top \rangle \\ &= C \langle x_k x_k^\top \rangle C^\top + 2 \langle \epsilon_k \rangle \langle x_k \rangle^\top C^\top + \langle \epsilon_k \epsilon_k^\top \rangle \\ &= C(\text{Cov}[x_k] + \langle x_k \rangle \langle x_k \rangle^\top) C^\top + \text{Cov}[\epsilon_k] + \langle \epsilon_k \rangle \langle \epsilon_k \rangle^\top \\ &= C(\Sigma_{k|k-1} + \mu_{k|k-1} \mu_{k|k-1}^\top) C^\top + R \\ \text{Cov}[y_k] &= C(\Sigma_{k|k-1} + \mu_{k|k-1} \mu_{k|k-1}^\top) C^\top + R - C \mu_{k|k-1} \mu_{k|k-1}^\top C^\top \\ &= C \Sigma_{k|k-1} C^\top + R\end{aligned}$$

Factorisation of $p(y_k, x_k | y_{1:k-1})$

- We compute

$$p(y_k, x_k | y_{1:k-1}) = p(y_k | y_{1:k-1})p(x_k | y_{1:k})$$

- From the factorisation theorem

$$p(y_k | y_{1:k-1}) = \mathcal{N}(y_k; C\mu_{k|k-1}, C\Sigma_{k|k-1}C^\top + R)$$

$$p(x_k | y_{1:k}) = \mathcal{N}(x_k; \mu_{k|k}, \Sigma_{k|k})$$

$$\mu_{k|k} = \mu_{k|k-1} + \Sigma_{k|k-1}C^\top (C\Sigma_{k|k-1}C^\top + R)^{-1}(y_k - C\mu_{k|k-1})$$

$$\Sigma_{k|k} = \Sigma_{k|k-1} - \Sigma_{k|k-1}C^\top (C\Sigma_{k|k-1}C^\top + R)^{-1}C\Sigma_{k|k-1}$$

Computing l_k

- Remember

$$\begin{aligned}\exp l_k &= p(y_{1:k}) \\ &= \int dx_k p(y_{1:k}, x_k) \\ &= p(y_{1:k-1}) \int dx_k p(y_k | x_k) p(x_k | y_{1:k-1}) \\ &= p(y_{1:k-1}) p(y_k | y_{1:k-1})\end{aligned}$$

- Hence

$$\begin{aligned}S_k &= C \Sigma_{k|k-1} C^\top + R \\ l_k &= l_{k-1} - \frac{1}{2} (y_k - C \mu_{k|k-1})^\top S_k^{-1} (y_k - C \mu_{k|k-1}) - \frac{1}{2} \log |2\pi S_k|\end{aligned}$$

The Kalman Filter: Update equations (Moment form)

- Compute prediction error and covariance

$$e_k = y_k - C\mu_{k|k-1}$$
$$S_k = C\Sigma_{k|k-1}C^\top + R$$

- Compute the so called Kalman Gain matrix

$$G_k = \Sigma_{k|k-1}C^\top S_k^{-1}$$

- These simplify the update equations

$$l_k = l_{k-1} - \frac{1}{2}e_k^\top S_k^{-1}e_k - \frac{1}{2}\log |2\pi S_k|$$
$$\mu_{k|k} = \mu_{k|k-1} + G_k e_k$$
$$\Sigma_{k|k} = \Sigma_{k|k-1} - G_k C \Sigma_{k|k-1}$$

The Kalman Filter: Prediction equations (Moment form)

- We need to compute

$$\begin{aligned}\alpha_{k|k-1} &\equiv p(y_{1:k-1}, x_k) \\ &= \exp(l_{k-1}) \mathcal{N}(x_k; \mu_{k|k-1}, \Sigma_{k|k-1})\end{aligned}$$

- Marginalisation over Gaussian potentials

$$\begin{aligned}\alpha_{k|k-1} &= \int dx_{k-1} p(x_k | x_{k-1}) \alpha_{k-1|k-1} \\ &= \int dx_{k-1} p(x_k | x_{k-1}) p(x_{k-1} | y_{1:k-1}) p(y_{1:k-1}) \\ &= p(y_{1:k-1}) \int dx_{k-1} p(x_k | x_{k-1}) p(x_{k-1} | y_{1:k-1}) \\ &= \exp(l_{k-1}) \int dx_{k-1} \mathcal{N}(x_k; Ax_{k-1}, Q) \mathcal{N}(x_{k-1}; \mu_{k-1|k-1}, \Sigma_{k-1|k-1})\end{aligned}$$

Finding $p(x_k, x_{k-1} | y_{1:k-1})$

- We will consider the vector

$$\begin{pmatrix} x_{k-1} \\ x_k \end{pmatrix}$$

- First, we show that the joint density $p(x_{k-1}, x_k | y_{1:k-1})$ is

$$\mathcal{N} \left(\begin{pmatrix} x_{k-1} \\ x_k \end{pmatrix}; \begin{pmatrix} \mu_{k-1|k-1} \\ A\mu_{k-1|k-1} \end{pmatrix}, \begin{pmatrix} \Sigma_{k-1|k-1} & \Sigma_{k-1|k-1}A^\top \\ A\Sigma_{k-1|k-1} & A\Sigma_{k-1|k-1}A^\top + Q \end{pmatrix} \right)$$

- Then, we compute the marginal

$$p(x_k | y_{1:k-1}) = \int dx_{k-1} p(x_k | x_{k-1}) p(x_{k-1} | y_{1:k-1})$$

Prediction Equations (Moment Form)

$$p(x_k | y_{1:k-1}) = \mathcal{N}(x_k; \mu_{k|k-1}, \Sigma_{k|k-1})$$

$$l_{k|k-1} = l_{k-1}$$

$$\mu_{k|k-1} = A\mu_{k-1|k-1}$$

$$\Sigma_{k|k-1} = A\Sigma_{k-1|k-1}A^T + Q$$

Implementation

- Setup a parameter structure
- Generate data
- Inference
- Test and visualise

Setup

```
% Transition model;
A = [1 1;0 1]; % Transition matrix
Q = 0.1*eye(2); % Transition noise covariance
% Observation model
C = [1 0]; % Observation matrix
R = 0.2; % Observation noise covariance
% Prior p(x_1)
m = [0 1]';
P = eye(2);
% Dimensions
N = size(A,1); % State
M = size(C, 1); % Observation
%% Need to check consistency of params
```

Generate Data

```
% Number of time slices
K = 40;

data.x = zeros(M, K);
data.y = zeros(N, K);
% Square roots
sQ = sqrtm(Q); sR = sqrtm(R); sP = sqrtm(P);
for k=1:K,
    if k==1,
        data.x(:,k) = m + sP*randn(M, 1);
    else
        data.x(:,k) = A*data.x(:,k-1) + sQ*randn(M, 1);
    end;
    data.y(:,k) = C*data.x(:, k) + sR*randn(N,1);
end;
```

Implementation, Initialisation

```
alpha_pred.l = zeros(1, K);  
alpha_pred.mu = zeros(M, K);  
alpha_pred.Sig = zeros(M, M, K);
```

```
alpha.l = zeros(1, K);  
alpha.mu = zeros(M, K);  
alpha.Sig = zeros(M, M, K);
```

Implementation, Kalman Filter (potentially instable)

```
for k=1:K,
    % Predict
    if k==1,
        alpha_pred.l(k) = 0;
        alpha_pred.mu(:, k) = m;
        alpha_pred.Sig(:, :, k) = P;
    else
        alpha_pred.l(k) = alpha.l(k-1);
        alpha_pred.mu(:, k) = A*alpha.mu(:, k-1);
        alpha_pred.Sig(:, :, k) = A*alpha.Sig(:, :, k-1)*A' + Q;
    end;

    % Update
    e = data.y(:,k) - C*alpha_pred.mu(:,k);
    S = C*alpha_pred.Sig(:, :, k)*C' + R;
    G = alpha_pred.Sig(:, :, k)*C'*inv(S);
    alpha.l(k) = alpha_pred.l(k) - 0.5*e'*inv(S)*e - 0.5*log(det(2*pi*S));
    alpha.mu(:, k) = alpha_pred.mu(:, k) + G*e;
    alpha.Sig(:, :, k) = alpha_pred.Sig(:, :, k) - G*C*alpha_pred.Sig(:, :, k);
end;
```

Numerical Issues

- Avoid using `inv` in a serious implementation, use slash (or backslash) instead.

```
Sig*C'*inv(C*Sig*C' + R);
```

should be implemented as

```
(Sig*C')/(C*Sig*C' + R);
```

- Avoid using `det`, as it can likely blow up in magnitude

```
0.5*log(det(2*pi*S))
```

should be implemented using Cholesky factorisation

```
sum(log(diag(chol(2*pi*S))))
```

The Kalman Smoother (RTS)

- The aim is computing

$$p(x_k | y_{1:K})$$

- Known as Rauch-Tung-Striebel (RTS) Smoother
- A Correction Smoother for LDS
- Is different from forward-backward algorithm of hmm's,
 - No need to store the observations

The Kalman Smoother (RTS)

- At the last time slice, the forward pass computes the exact marginal $p(x_K|y_{1:K})$
- Recursively “correct” the filtered estimates for $k = K : -1 : 1$ as

$$\begin{aligned} p(x_k|y_{1:K}) &= \int dx_{k+1} p(x_k, x_{k+1}|y_{1:K}) \\ &= \int dx_{k+1} p(x_k|x_{k+1}, y_{1:K}) p(x_{k+1}|y_{1:K}) \\ &= \int dx_{k+1} p(x_k|x_{k+1}, y_{1:k}) p(x_{k+1}|y_{1:K}) \\ &= \langle p(x_k|x_{k+1}, y_{1:k}) \rangle_{p(x_{k+1}|y_{1:K})} \\ &= \int dx_{k+1} \frac{p(x_k, x_{k+1}|y_{1:k})}{p(x_{k+1}|y_{1:k})} p(x_{k+1}|y_{1:K}) \end{aligned}$$

- Divide by the “old” marginal, multiply in a “new” marginal

Factorisation of $p(x_k, x_{k+1} | y_{1:k})$

- Consider the joint density $p(x_k, x_{k+1} | y_{1:k})$

$$\mathcal{N} \left(\begin{pmatrix} x_k \\ x_{k+1} \end{pmatrix}; \begin{pmatrix} \mu_{k|k} \\ A\mu_{k|k} \end{pmatrix}, \begin{pmatrix} \Sigma_{k|k} & \Sigma_{k|k}A^\top \\ A\Sigma_{k|k} & A\Sigma_{k|k}A^\top + Q \end{pmatrix} \right)$$

- Factorise as $p(x_k, x_{k+1} | y_{1:k}) = p(x_{k+1} | y_{1:k})p(x_k | x_{k+1}, y_{1:k})$

- Result

$$p(x_k | x_{k+1}, y_{1:k}) = \mathcal{N}(x_k; \bar{\mu}, \bar{\Sigma}) \quad (1)$$

$$\bar{\mu} = \mu_{k|k} + \Sigma_{k|k}A^\top (A\Sigma_{k|k}A^\top + Q)^{-1} (x_{k+1} - A\mu_{k|k}) \quad (2)$$

$$\bar{\Sigma} = \Sigma_{k|k} - \Sigma_{k|k}A^\top (A\Sigma_{k|k}A^\top + Q)^{-1} A\Sigma_{k|k} \quad (3)$$

Using $\mathcal{N}(x_1; \mu_1 + \Sigma_{12}\Sigma_2^{-1}(x_2 - \mu_2), \Sigma_1 - \Sigma_{12}\Sigma_2^{-1}\Sigma_{12}^\top)\mathcal{N}(x_2; \mu_2, \Sigma_2)$

Write

$$\begin{aligned} J_k &= \Sigma_{k|k} A^\top (A \Sigma_{k|k} A^\top + Q)^{-1} \\ &= \Sigma_{k|k} A^\top \Sigma_{k+1|k}^{-1} \end{aligned}$$

From 2 and 3, we have

$$x_k = (I - J_k A) \mu_{k|k} + J_k x_{k+1} + \epsilon \quad (4)$$

where

$$\begin{aligned} \bar{\Sigma} &= \Sigma_{k|k} - J_k A \Sigma_{k|k} \\ \epsilon &\sim \mathcal{N}(\epsilon; 0, \bar{\Sigma}) \\ x_{k+1} | y_{1:K} &\sim \mathcal{N}(x_{k+1}; \mu_{k+1|K}, \Sigma_{k+1|K}) \end{aligned}$$

Taking the expectation and the covariance of 4, we obtain

$$\begin{aligned}
 x_k | y_{1:K} &\sim \mathcal{N}(x_k; \mu_{k|K}, \Sigma_{k|K}) \\
 \mu_{k|K} &= (I - J_k A) \mu_{k|k} + J_k \mu_{k+1|K} \\
 &= \mu_{k|k} + J_k (\mu_{k+1|K} - \mu_{k+1|k}) \\
 x_k - \mu_{k|K} &= J_k (x_{k+1} - \mu_{k+1|K}) + \epsilon \\
 \text{Cov}[x_k] &= \langle (x_k - \mu_{k|K})(x_k - \mu_{k|K})^\top \rangle \\
 &= J_k \langle (x_{k+1} - \mu_{k+1|K})(x_{k+1} - \mu_{k+1|K})^\top \rangle J_k^\top + \langle \epsilon \epsilon^\top \rangle \\
 \Sigma_{k|K} &= J_k \Sigma_{k+1|K} J_k^\top + \Sigma_{k|k} - J_k A \Sigma_{k|k} \\
 &= J_k \Sigma_{k+1|K} J_k^\top + \Sigma_{k|k} - J_k (A \Sigma_{k|k} A^\top + Q) J_k^\top \\
 &= \Sigma_{k|k} + J_k (\Sigma_{k+1|K} - \Sigma_{k+1|k}) J_k^\top
 \end{aligned}$$

Cross terms are needed for learning

$$\begin{aligned}\text{Cov}[x_k, x_{k+1}] &= \langle (x_k - \mu_{k|K})(x_{k+1} - \mu_{k+1|K})^\top \rangle \\ &= \langle (J_k(x_{k+1} - \mu_{k+1|K}) + \epsilon)(x_{k+1} - \mu_{k+1|K})^\top \rangle \\ &= \langle (J_k(x_{k+1} - \mu_{k+1|K}))(x_{k+1} - \mu_{k+1|K})^\top \rangle \\ &= J_k \Sigma_{k+1|K}\end{aligned}$$

RTS Smoother Equations

$$J_k = \Sigma_{k|k} A^\top \Sigma_{k+1|k}^{-1}$$

$$\mu_{k|K} = \mu_{k|k} + J_k (\mu_{k+1|K} - \mu_{k+1|k})$$

$$\Sigma_{k|K} = \Sigma_{k|k} + J_k (\Sigma_{k+1|K} - \Sigma_{k+1|k}) J_k^\top$$

RTS Smoother Implementation

```
% RTS Smoother
gamma.mu = zeros(M, K);
gamma.Sig = zeros(M, M, K);
for k=K:-1:1,
    if k==K,
        gamma.mu(:, k) = alpha.mu(:, k);
        gamma.Sig(:, :, k) = alpha.Sig(:, :, k);
    else
        J = alpha.Sig(:, :, k)*A'*inv(alpha_pred.Sig(:, :, k+1));
        gamma.mu(:, k) = alpha.mu(:, k) + ...
            J*(gamma.mu(:, k+1)-alpha_pred.mu(:, k+1));
        gamma.Sig(:, :, k) = alpha.Sig(:, :, k) + ...
            J*(gamma.Sig(:, :, k+1) - alpha_pred.Sig(:, :, k+1) )*J';
    end;
end;
```

Filtered vs. Smoothed

