Kalman Filtering in a Mass-Spring System

Andrea Arnold and Franz Hamilton

Department of Mathematics
Center for Quantitative Sciences in Biomedicine

North Carolina State University

July 30, 2016

Practical Overview

Goal: Implement Kalman filter for linear system

1. Solve a linear system
2. Generate noisy data
3. Implement Kalman filter for state estimation
4. Explore filter parameters

Example: Mass-Spring System

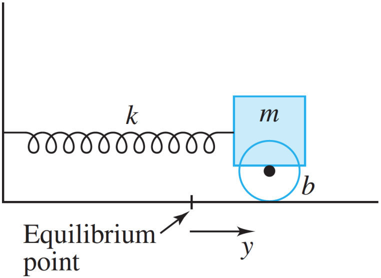

Figure 4.1 Damped mass–spring oscillator

[Figure from Nagle, Saff and Snider (2011)]

Example: Mass-Spring System

Consider the damped mass-spring oscillator

$$mp''(t) + bp'(t) + kp(t) = 0$$

where

- $p(t)$ denotes the position of mass at time t
- $m > 0$ is the mass
- $b \geq 1$ is the damping coefficient
- $k > 0$ is the spring constant

Example: Mass-Spring System

This can be written as the first-order linear system

$$\begin{aligned} \frac{dp}{dt} &= v \\ \frac{dv}{dt} &= -\frac{k}{m}p - \frac{b}{m}v \end{aligned}$$

where $v(t) = p'(t)$ denotes the velocity of the mass at time t, and letting

$$x = \begin{bmatrix} p \\ v \end{bmatrix} \in \mathbb{R}^2$$

yields the matrix equation

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} x = Ax.$$

Step 1: Define Model Values

Define: System parameters

- $m = 10$
- $k = 5$
- $b = 3$

Define: Coefficient matrix

- $A = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix}$

Step 1: Define Model Values

```
% Assign model parameters
m = 10; % mass > 0
k = 5;  % spring constant > 0
b = 3;  % damping coefficient ≥ 1
A = [0 1; -k/m -b/m];  % coefficient matrix for ...
    continuous system
```

Define: Time interval of solution

- $t_0 = 0$
- $t_F = 30$
- $h = 0.2$

Define: Right-hand side of your differential equation (use MATLAB inline function!)

Define: System initial condtion

Solve: Use ode45 to solve system

Step 2: Solve Mass-Spring System

```
rhs = @(t,x) A*x;   % rhs of function
% Simulate system
xinit = [1;0];   % initial value
h     = 0.2;     % time step
T      = 30;
time   = 0:h:T;
[¬,trueTrajectory] = ode45(rhs,time,xinit);
```

Step 3: Generate Data

Assume: We can measure p (i.e. first column of our solution from ode45)

Assume: Observations affected by Gaussian noise with standard deviation of 0.3

Step 3: Generate Data

```
obsNoise = 0.3; %Define observation noise level
obs = trueTrajectory(:,1); %First state is observable
obs = obs+obsNoise*randn(size(obs)); %Add noise
```

Discretization Needed!

Note: Our system is continuous

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{b}{m} \end{bmatrix} x = \mathrm{A}x$$

but Kalman filter as presented is for discrete systems (for continuous see Kalman-Bucy)

Solution: Discretize

$$\begin{aligned} X_{j+1} &= X_j + h\mathrm{A}X_j + V_{j+1} \\ &= (\mathrm{I} + h\mathrm{A})X_j + V_{j+1} \end{aligned}$$

Discretization Needed!

State evolution equation:

$$X_{j+1} = (\mathrm{I} + h\mathrm{A})X_j + V_{j+1}$$

Observation equation:

$$Y_j = \begin{bmatrix} 1 & 0 \end{bmatrix} X_j + W_{j+1}$$

Step 4: Initialize Filter Parameters

Define: State transition and observation matrices

$$
\begin{aligned}
F &= \mathrm{I} + h\mathrm{A} \\
G &= \begin{bmatrix} 1 & 0 \end{bmatrix}
\end{aligned}
$$

Define: Initial state and covariance

$$
\begin{aligned}
\bar{x}_{0|0} &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
\Gamma_{0|0} &= \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}
\end{aligned}
$$

Step 4: Initialize Filter Parameters

NC STATE UNIVERSITY

```
xbarEstimate = [1;0]; %Initial state
Gamma = .1*eye(length(xbar)); %Initial covariance
varEstimate = diag(Gamma); %Initial state variance
F = eye(length(xbar))+h*A; %State transition matrix
G = [1 0]; %Observation function
```

Step 4: Initialize Filter Parameters

Define: Innovation and observation noise matrices...**VERY IMPORTANT!!!**

$$C = \begin{bmatrix} 0.0001 & 0 \\ 0 & 0.0001 \end{bmatrix}$$

$$D = obsNoise^2$$

Step 4: Initialize Filter Parameters


```
C = 0.0001*eye(2); %Innovation or system error
D = obsNoise^2; %Observation error
```

Goal: Estimate p and v using noisy observations of p

1. **Prediction step**: Update the prior mean and covariance using the formulas

$$\overline{x}_{j+1|j} = \mathsf{F}\overline{x}_{j|j}$$

and

$$\Gamma_{j+1|j} = \mathsf{F}\Gamma_{j|j}\mathsf{F}^{\mathsf{T}} + \mathsf{C}.$$

2. **Observation update**: After observing y_{j+1}, update the posterior mean and error covariance via the formulas

$$\overline{x}_{j+1|j+1} = \overline{x}_{j+1|j} + \mathsf{K}_{j+1}(y_{j+1} - \mathsf{G}\overline{x}_{j+1|j})$$

and

$$\Gamma_{j+1|j+1} = (\mathsf{I} - \mathsf{K}_{j+1}\mathsf{G})\Gamma_{j+1|j},$$

where

$$\mathsf{K}_{j+1} = \Gamma_{j+1|j}\mathsf{G}^{\mathsf{T}}(\mathsf{G}\Gamma_{j+1|j}\mathsf{G}^{\mathsf{T}} + \mathsf{D})^{-1}.$$

3. Process all data!

Step 5: Implement Kalman Filter

```
for i = 2:length(obs)
    % Prediction step
    xbar = F*xbar;
    Gamma = F*Gamma*F' + C;
    % Observation update
    K = (Gamma*G')/(G*Gamma*G'+D);  % aka K = ...
        Gamma*G'*inv(G*Gamma*G'+D);
    xbar = xbar + K*(obs(i) - G*xbar);
    Gamma = Gamma - K*G*Gamma;
    xbarEstimate(:,i) = xbar;
    varEstimate(:,i) = diag(Gamma);
end
```

Goal: Plot your results!

Step 6: Analyze Your Results


```
figure;subplot(2,1,1);
plot(time,trueTrajectory(:,1),'k','linewidth',3);
hold on;
plot(time,obs,'bo','markerfacecolor','b','markersize',6)
plot(time,xbarEstimate(1,:),'r','linewidth',3);
plot(time,xbarEstimate(1,:)+2*sqrt(varEstimate(1,:)),'-.r')
plot(time,xbarEstimate(1,:)-2*sqrt(varEstimate(1,:)),'-.r')
axis([0 30 -1 1.5])
set(gca,'fontsize',30)
ylabel('p','fontsize',30)
subplot(2,1,2);
plot(time,trueTrajectory(:,2),'k','linewidth',3);
hold on;
plot(time,xbarEstimate(2,:),'r','linewidth',3);
plot(time,xbarEstimate(2,:)+2*sqrt(varEstimate(2,:)),'-.r')
plot(time,xbarEstimate(2,:)-2*sqrt(varEstimate(2,:)),'-.r')
axis([0 30 -1 1.5])
set(gca,'fontsize',30)
xlabel('Time','fontsize',30)
ylabel('v','fontsize',30)
```

Further Investigation

1. What happens if you make C smaller? Larger?
2. What happens if you make D smaller? Larger?
3. What happens if you add more noise?