Data Assimilation Practical

Tyrus Berry
Dept. of Mathematical Sciences
George Mason University

July 28, 2018

Practical Overview

Goal: Implement Ensemble Kalman filter (EnKF)

1. Solve a linear system
2. Generate synthetic noisy data
3. Implement Kalman filter for state estimation
4. Explore filter parameters
5. Solve a nonlinear system
6. Implement an Ensemble Kalman Filter
7. Introduce parameter estimation

EXAMPLE: MASS-SPRING SYSTEM

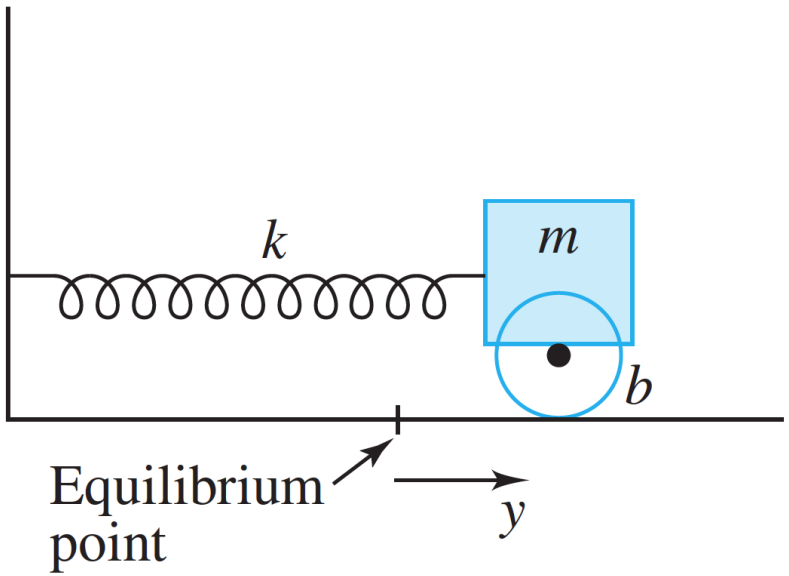

Figure 4.1 Damped mass–spring oscillator

[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

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

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
```

Step 2: Solve Mass-Spring 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

```
1  rhs = @(t,x) A*x;   % rhs of function
2  % Simulate system
3  xinit = [1;0];   % initial value
4  h     = 0.2;     % time step
5  T      = 30;
6  time   = 0:h:T;
7  [¬,trueTrajectory] = ode45(rhs,time,xinit);
8  figure(1);plot(time,trueTrajectory,'k');
```

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 ...
    Gaussian noise with variance (obsNoise)^2
figure(1);hold on;
plot(time,obs,'ro');
```

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 (with step size h). Linear ODE has a linear solution given by the matrix exponential, $F = e^{hA}$.

$$x_{j+1} = Fx_j + w_{j+1}$$

Discretization Needed!

State evolution equation:

$$x_{j+1} = Fx_j + w_{j+1}$$

Observation equation:

$$y_j = \begin{bmatrix} 1 & 0 \end{bmatrix} x_j + v_j$$

Step 4: Initialize Filter Parameters

Define: State transition and observation matrices

$$\begin{aligned} F &= e^{hA} \\ H &= \begin{bmatrix} 1 & 0 \end{bmatrix} \end{aligned}$$

Define: Initial state and covariance

$$\begin{aligned} x_0^+ &= \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ P_0^+ &= \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \end{aligned}$$

Step 4: Initialize Filter Parameters

```
x_posterior = [1;0]; %Initial state
P_posterior = .1*eye(length(x_posterior)); ...
    %Initial covariance
stateEstimate = x_posterior; %Initial state estimate
varEstimate = diag(P_posterior); %Initial state ...
    variance
F = expm(h*A); %State transition matrix
G = [1 0]; %Observation function
```

Step 4: Initialize Filter Parameters

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

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

$$R = (obsNoise)^2$$

STEP 4: INITIALIZE FILTER PARAMETERS

```
1 Q = 0.0001*eye(2); %System noise covariance
2 R = obsNoise^2; %Observation noise covariance
```

Step 5: Implement Kalman Filter

Goal: Estimate p and v using noisy observations of p

1. **Prediction step**: Predict the state and covariance

$$x_j^- = \mathsf{F} x_{j-1}^+$$

and

$$\mathsf{P}_j^- = \mathsf{F}\mathsf{P}_{j-1}^+\mathsf{F}^\mathsf{T} + \mathsf{Q}.$$

Step 5: Implement Kalman Filter

2. **Observation update**: After observing y_j, update the posterior mean and error covariance via the formulas

$$x_j^+ = x_j^- + K_j(y_j - Hx_j^-)$$

and

$$P_j^+ = (I - K_j H)P_j^-,$$

where

$$K_j = P_j^- H^T (HP_j^- H^T + R)^{-1}.$$

3. Process all data!

STEP 5: IMPLEMENT KALMAN FILTER

```
for j = 2:length(obs)
    % Prediction step
    x_prior = F*x_posterior;
    P_prior = F*P_posterior*F' + Q;
    % Observation update
    K = (P_prior*H')/(H*P_prior*H'+R);  % i.e. K ...
        = Gamma*H'*inv(H*Gamma*H'+R);
    x_posterior = x_prior + K*(obs(j) - H*x_prior);
    P_posterior = P_prior - K*H*P_prior;
    stateEstimate(:,j) = x_posterior;
    varEstimate(:,j) = diag(P_posterior);
end
```

Step 6: Analyze Your Results

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','r','markersize',6)
plot(time,xbarEstimate(1,:),'b','linewidth',3);
plot(time,xbarEstimate(1,:)+2*sqrt(varEstimate(1,:)),'-.b')
plot(time,xbarEstimate(1,:)-2*sqrt(varEstimate(1,:)),'-.b')
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,:),'b','linewidth',3);
plot(time,xbarEstimate(2,:)+2*sqrt(varEstimate(2,:)),'-.b')
plot(time,xbarEstimate(2,:)-2*sqrt(varEstimate(2,:)),'-.b')
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 Q smaller? Larger?
2. What happens if you make R smaller? Larger?
3. What happens if you add more noise?

Define the Vector Field (RHS of ODE)

The Lorenz-63 ODE:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = x(\rho - z) - y$$
$$\dot{z} = xy - \beta z$$

```
1 function dx = L63(t,x)
2       rho = 28; sigma = 10; beta = 8/3;
3     dx = zeros(3,size(x,2));
4     dx(1,:) = sigma*(x(2,:) - x(1,:));
5     dx(2,:) = x(1,:).*(rho - x(3,:)) - x(2,:);
6     dx(3,:) = x(1,:).*x(2,:) - beta*x(3,:);
7 end
```

GENERATE DATA

```
xinit = randn(3,1);  % initial value
h     = 0.01;     % time step
T     = 100;
time  = 0:h:T;
[¬,truth] = ode45(@(t,x) L63(t,x),time,xinit);
obs = truth(:,1) + randn(size(truth(:,1)))/4;
figure(1);plot(time,truth,'k');
figure(2);plot3(truth(:,1),truth(:,2),truth(:,3));
```

Simple Solver

The Lorenz-63 ODE:

$$\dot{x} = f(x)$$
$$x(t+h) \approx x(t) + hf(x(t))$$

```
1  function x = EulerL63(x,t)
2        h = 0.001;
3        n = t/h;
4        for i = 1:n
5           x = x + h*L63(x);
6        end
7  end
```

The EnKF

```
%%% Define the dynamics and obs
f = @(x) EulerL63(x,0.01);
h = @(x) x(1,:);
Q = 0.03*eye(3);
R = 1/16;
```

```
function [x,P] = EnKF(x,P,f,h,Q,R,obs)
    n = length(x);
    [U,S,¬] = svd(P);  % Don't use Cholesky!
    sqrtP = U*sqrt(S)*U';  %sigma points
    ens(:,1:n) = repmat(x,1,n) + sqrtP;
    ens(:,n+1:2*n) = repmat(x,1,n) - sqrtP;
    %%% Forecast Step
    f_ens = f(ens);
    x_prior = mean(f_ens,2);
    cf_ens = f_ens - repmat(x_prior,1,2*n);
    P_prior = cf_ens*cf_ens'/(2*n-1) + Q;
    %%% Observe
    h_ens = h(ens);
    y_prior = mean(h_ens,2);
    ch_ens = h_ens - repmat(y_prior,1,2*n);
    P_y = ch_ens*ch_ens'/(2*n-1) + R;
    P_xy = cf_ens*ch_ens'/(2*n-1);
    %%% Ensemble Kalman Update
    K = P_xy/P_y; %Gain matrix
    P = P_prior - K*P_y*K';
    x = x_prior + K*(obs - y_prior);
```

The EnKF

```
x = randn(3,1);
P = eye(3);
for i = 1:length(time)
    [x,P] = EnKF(x,P,f,h,Q,R,obs(i,:));
    xa(:,i) = x;
end
figure;plot(xa')
hold on;plot(truth)
```

Further Investigation

1. Parameter Estimation: Assume $\sigma = 10$ is unknown
2. Increase the dimension of x to be 4-dimensional (4th dimension is σ)
3. Update the L63 code to use x(4,:) as σ
4. Update the rest of the code to use a 4-dimensional state