System Identification of a Mass-Spring-Damper System

Agus Hasan

1 System Modeling

Consider the following model of a mass-spring-damper system

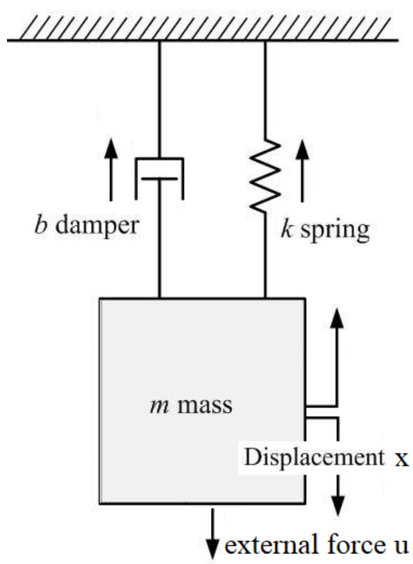

Figure 1: Spring-mass-damper system.

From Newton second law of motion, we have

$$\Sigma F(t) = m\ddot{x}(t) \tag{1}$$

or

$$u(t) - b\dot{x}(t) - kx(t) = m\ddot{x}(t) \tag{2}$$

Thus, the mass-spring-damper system can be described by the following second-order ODE:

$$u(t) = b\dot{x}(t) + kx(t) + m\ddot{x}(t). \tag{3}$$

System (3) can be transformed into a first-order system by letting

$$x_1(t) = x(t), \tag{4}$$
$$x_2(t) = \dot{x}(t), \tag{5}$$

as such, we have

$$\dot{x}_1(t) = x_2(t), \tag{6}$$
$$\dot{x}_2(t) = -\frac{k}{m}x_1(t) - \frac{b}{m}x_2(t) + \frac{1}{m}u(t). \tag{7}$$

Discretizing (6)-(7), we have

$$x_1(i+1) = x_1(i) + x_2(i)\Delta t, \tag{8}$$
$$x_2(i+1) = -\frac{k\Delta t}{m}x_1(i) + \left(1 - \frac{b\Delta t}{m}\right)x_2(i) + \frac{\Delta t}{m}u(i). \tag{9}$$

where $i = 1, 2, \cdots$

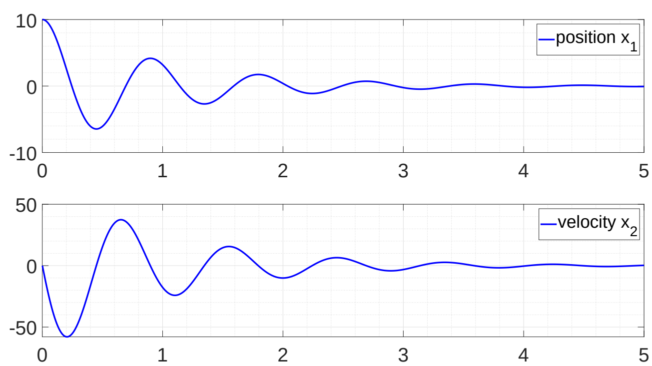

Figure 2: Solution to the mass-spring-damper system with $m = 10$, $b = 20$, $k = 500$, and initial condition $[10, 0]$.

2 System Identification

Let us assume that we can measure the position x_1 using an ultra-wideband sensor and the velocity x_2 using an accelerometer. We will use these measurements to estimate the unknown parameter b and k simultaneously. This problem is called a system identification problem. To this end, we will use an extended Kalman filter algorithm. First, we augment the unknow parameters b and k into the system, such that we have

$$\begin{aligned} x_1(i+1) &= x_1(i) + x_2(i)\Delta t, & (10)\\ x_2(i+1) &= -\frac{k(i)\Delta t}{m} x_1(i) + \left(1 - \frac{b(i)\Delta t}{m}\right) x_2(i) + \frac{\Delta t}{m} u(i), & (11)\\ b(i+1) &= b(i), & (12)\\ k(i+1) &= k(i). & (13) \end{aligned}$$

Remark that in the above system, the parameters b and k are considered as new state variables.

Let us denote

$$\boldsymbol{x}(i) = \begin{pmatrix} x_1(i) \\ x_2(i) \\ b(i) \\ k(i) \end{pmatrix}. \tag{14}$$

Thus, we can write (10)-(13) as

$$\boldsymbol{x}(i+1) = \boldsymbol{F}(\boldsymbol{x}(i)) + \boldsymbol{B} u(i), \tag{15}$$

where

$$\begin{aligned} \boldsymbol{F}(\boldsymbol{x}(i)) &= \begin{pmatrix} x_1(i) + x_2(i)\Delta t \\ -\frac{k\Delta t}{m} x_1(i) + \left(1 - \frac{b(i)\Delta t}{m}\right) x_2(i) + \frac{\Delta t}{m} u(i) \\ b(i) \\ k(i) \end{pmatrix}, & (16)\\ \boldsymbol{B} &= \begin{pmatrix} 0 \\ \frac{\Delta t}{m} \\ 0 \\ 0 \end{pmatrix}. & (17) \end{aligned}$$

The Jacobian of $\boldsymbol{F}$ is given by

$$\boldsymbol{J}_{\boldsymbol{F}(\boldsymbol{x}(i))} = \begin{pmatrix} 1 & \Delta t & 0 & 0 \\ -\frac{k(i)\Delta t}{m} & 1-\frac{b(i)\Delta t}{m} & -\frac{\Delta t x_2(i)}{m} & -\frac{x_1(i)\Delta t}{m} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{18}$$

The Kalman filter algorithm is given by

For $i = 1, 2, \cdots$

predict

$$\begin{aligned} \hat{\boldsymbol{x}}^-(i) &= \boldsymbol{J}_{\boldsymbol{F}}(i-1)\hat{\boldsymbol{x}}^+(i-1) + \boldsymbol{B}(i-1)\boldsymbol{u}(i-1) = \text{a priori estimate} \\ \boldsymbol{P}^-(0) &= \boldsymbol{J}_{\boldsymbol{F}}(i-1)\boldsymbol{P}^+(i-1)\boldsymbol{J}_{\boldsymbol{F}}^\intercal(i-1) + \boldsymbol{Q}(i-1) \end{aligned}$$

update

$$\begin{aligned} \boldsymbol{K}(i) &= \boldsymbol{P}^-(i)\boldsymbol{C}^\intercal(i)\left(\boldsymbol{C}(i)\boldsymbol{P}^-(i)\boldsymbol{C}^\intercal(i) + \boldsymbol{R}(i)\right)^{-1} \\ \hat{\boldsymbol{x}}^+(i) &= \hat{\boldsymbol{x}}^-(i) + \boldsymbol{K}(i)(\boldsymbol{y}(i) - \boldsymbol{C}(i)\hat{\boldsymbol{x}}^-(i)) = \text{a posteriori estimate} \\ \boldsymbol{P}^+(i) &= (\boldsymbol{I} - \boldsymbol{K}(i)\boldsymbol{C}(i))\,\boldsymbol{P}^-(i) \end{aligned}$$

```
%% Mass-Spring-Damper System
% By Agus Hasan

clear;
clc;

dt = 0.001;                    % time step
tf = 20;                       % simulation time

m = 10;                        % mass
k = 500;                       % spring coefficient
b = 5;                         % damper coefficient

% system description
B = [0;1/m]*dt;                % control matrix
H = [1 0 0 0;                  % measurement matrix digital twin
     0 1 0 0];
C = [1 0;                      % measurement matrix physical twin
     0 1];
u = 0;                         % initial input

% error covariance matrix
```

```
Q = 0.1*eye(4);               % model covariance matrix
R = 0.1;                      % measurement covariance matrix

% initial data
x     = [10 0]';              % initial condition
xhat  = [10 0 0 400]';        % estimated initial condition
Pplus = 10000*eye(4);         % initial matrix propagation

% for plotting
xArray    = [];
xhatArray = [];

for i=1:tf/dt

    if i>5000
        b = 6;
    end
    if i>10000
        u = 50000;
    end
    if i>10010
        u = 0;
    end
    xArray    = [xArray x];
    xhatArray = [xhatArray xhat];
    % Simulate the system
    x = [1 dt;-k*dt/m 1-(b*dt/m)]*x+B*u;
    y = C*x;
    % Prediction
    F = [1 dt 0 0;            % Jacobian matrix
         -xhat(4)*dt/m 1-(xhat(3)*dt/m) -dt*xhat(2)/m ...
             -xhat(1)*dt/m;
         0 0 1 0;
         0 0 0 1];
    xhat  = [xhat(1)+xhat(2)*dt;
            (-xhat(4)*dt/m)*xhat(1)+(1-(xhat(3)*dt/m))*xhat(2)+dt*u/m;
            xhat(3);
            xhat(4)];
    Pmin  = F*Pplus*F' + Q;
    % Update
    K     = Pmin*H'*inv(H*Pmin*H' + R);
    Pplus = (eye(4)-K*H)*Pmin;
    xhat  = xhat + K*(y-H*xhat);
end

```

```
figure(1)
subplot(2,1,1)
plot(dt:dt:tf,xArray(1,:),'-b','LineWidth',3)
hold on;
plot(dt:dt:tf,xhatArray(1,:),':r','LineWidth',3)
legend('physical twin position','digital twin position')
set(gca,'FontSize',24)
grid on;
grid minor;
subplot(2,1,2)
plot(dt:dt:tf,xArray(2,:),'-b','LineWidth',3)
hold on;
plot(dt:dt:tf,xhatArray(2,:),':r','LineWidth',3)
legend('physical twin velocity','digital twin velocity')
set(gca,'FontSize',24)
grid on;
grid minor;

figure(2)
subplot(2,1,1)
plot(dt:dt:tf,[5*ones(1,5000) ...
    6*ones(1,15000)],'-b','LineWidth',3)
hold on;
plot(dt:dt:tf,xhatArray(3,:),':r','LineWidth',3)
legend('physical twin damping coef.','digital twin damping ...
    coef.')
ylim([4.5 6.5])
set(gca,'FontSize',24)
grid on;
grid minor;
subplot(2,1,2)
plot(dt:dt:tf,[500*ones(1,5000) ...
    500*ones(1,15000)],'-b','LineWidth',3)
hold on;
plot(dt:dt:tf,xhatArray(4,:),':r','LineWidth',3)
ylim([405 550])
legend('physical twin spring coef.','digital twin spring ...
    coef.')
set(gca,'FontSize',24)
grid on;
grid minor;

figure(3)
curve1 = animatedline('Color','b','LineWidth',2);
curve2 = ...
```

```
    animatedline('Color','r','LineStyle',':','LineWidth',2);
set(gca,'XLim',[0 20],'YLim',[−15 15]);
legend('physical twin position','digital twin position')
ylabel('position (m)')
xlabel('time (s)')
grid on;
tm = dt:10*dt:tf;
for i = 1:length(tm)
    addpoints(curve1,tm(i),xArray(1,10*i));
    hold on
    addpoints(curve2,tm(i),xhatArray(1,10*i));
    drawnow
    G(i) = getframe(gcf);
end
video = VideoWriter('MSDvel.');
open(video)
writeVideo(video,G)
close(video)

figure(4)
curve2 = ...
    animatedline('Color','r','LineStyle',':','LineWidth',2);
set(gca,'XLim',[0 20],'YLim',[4.5 6.5]);
legend('digital twin damping coef.')
ylabel('damping coeff')
xlabel('time (s)')
grid on;
tm = dt:10*dt:tf;
for i = 1:length(tm)
    addpoints(curve2,tm(i),xhatArray(3,10*i));
    drawnow
    G(i) = getframe(gcf);
end
video = VideoWriter('MSDspring.');
open(video)
writeVideo(video,G)
close(video)
```

3 Results

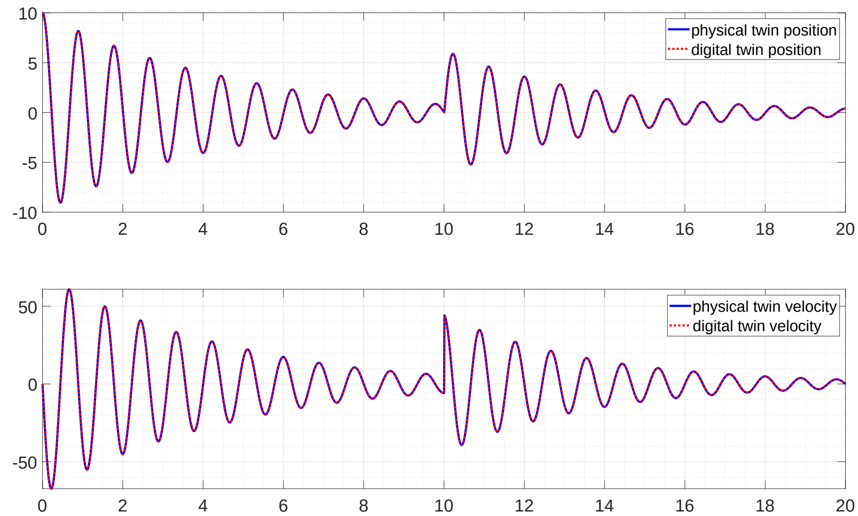

Figure 3: Position and velocity between the physical system and the digital system.

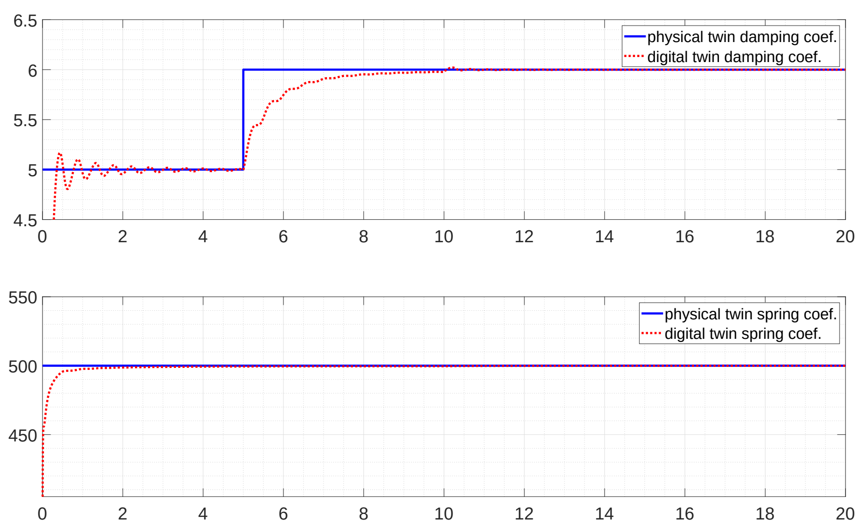

Figure 4: Parameter estimation from system identification using extended Kalman filter.