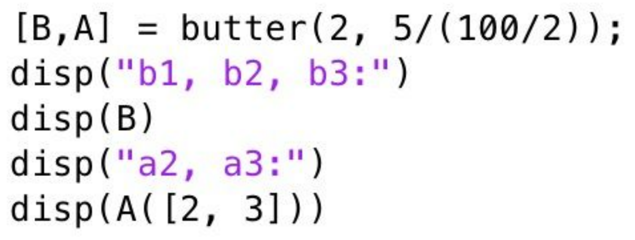
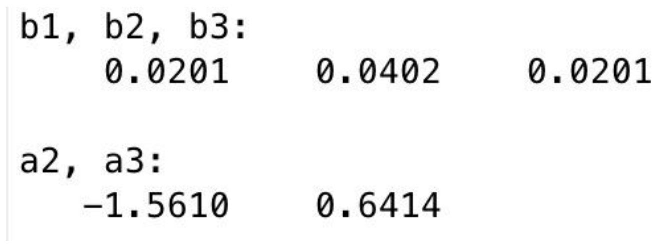
Zhanpeng He

1.
My codes for calculating parameters:



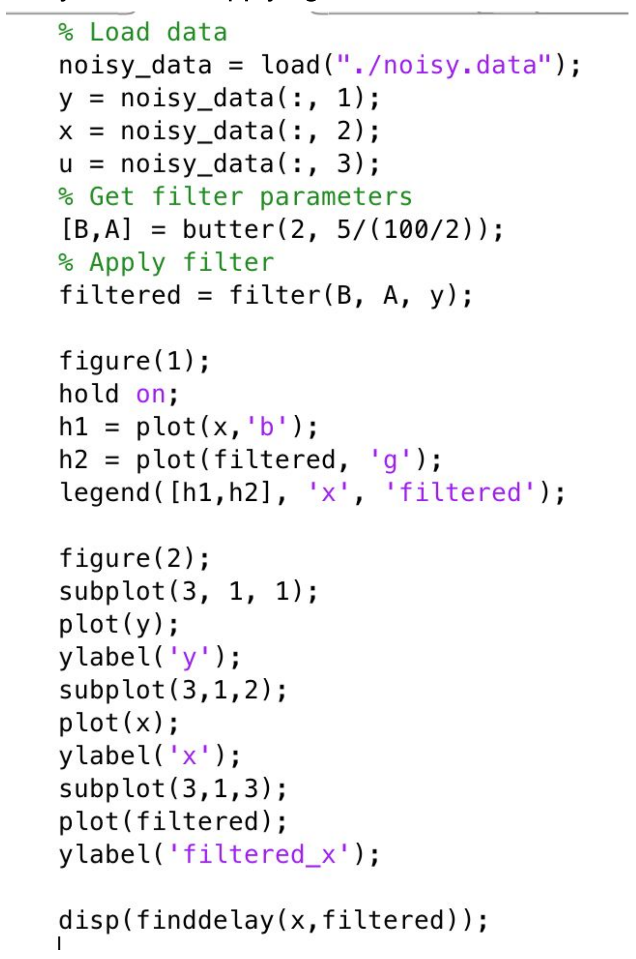
```
[B,A] = butter(2, 5/(100/2));
disp("b1, b2, b3:")
disp(B)
disp("a2, a3:")
disp(A([2, 3]))
```

Values:



```
b1, b2, b3:
    0.0201    0.0402    0.0201

a2, a3:
   -1.5610    0.6414
```

2. My codes for applying the filter:

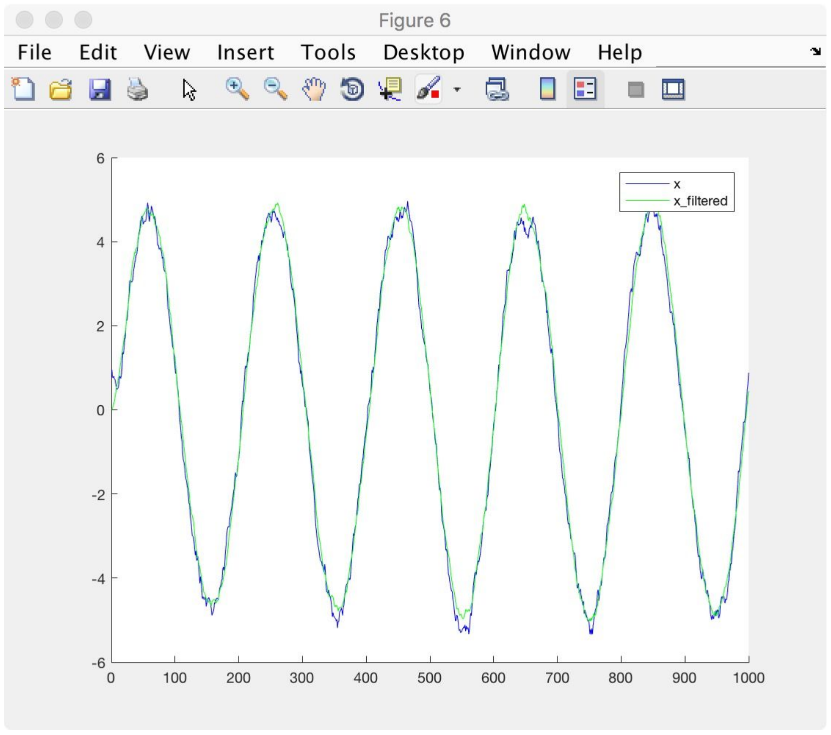


```
% Load data
noisy_data = load("./noisy.data");
y = noisy_data(:, 1);
x = noisy_data(:, 2);
u = noisy_data(:, 3);
% Get filter parameters
[B,A] = butter(2, 5/(100/2));
% Apply filter
filtered = filter(B, A, y);

figure(1);
hold on;
h1 = plot(x,'b');
h2 = plot(filtered, 'g');
legend([h1,h2], 'x', 'filtered');

figure(2);
subplot(3, 1, 1);
plot(y);
ylabel('y');
subplot(3,1,2);
plot(x);
ylabel('x');
subplot(3,1,3);
plot(filtered);
ylabel('filtered_x');

disp(finddelay(x,filtered));
```

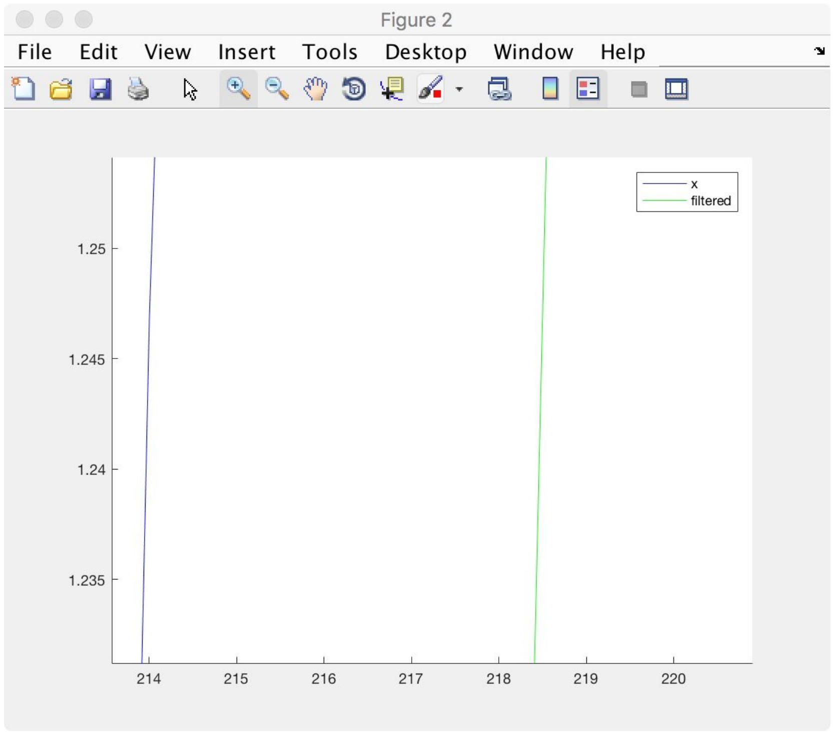
Zhanpeng He

Results:

The plot of x and filtered x data:

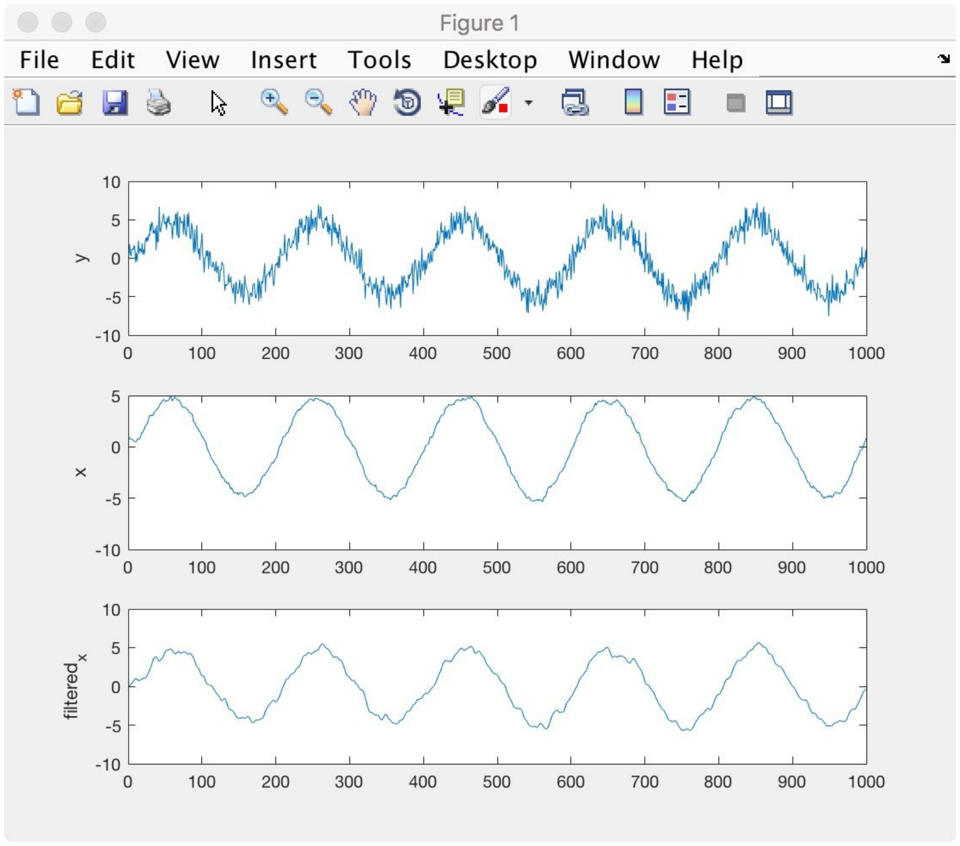


To estimate the delay, I zoomed in the plot:



Here we can see that the delay is about 5 steps.

Zhanpeng He

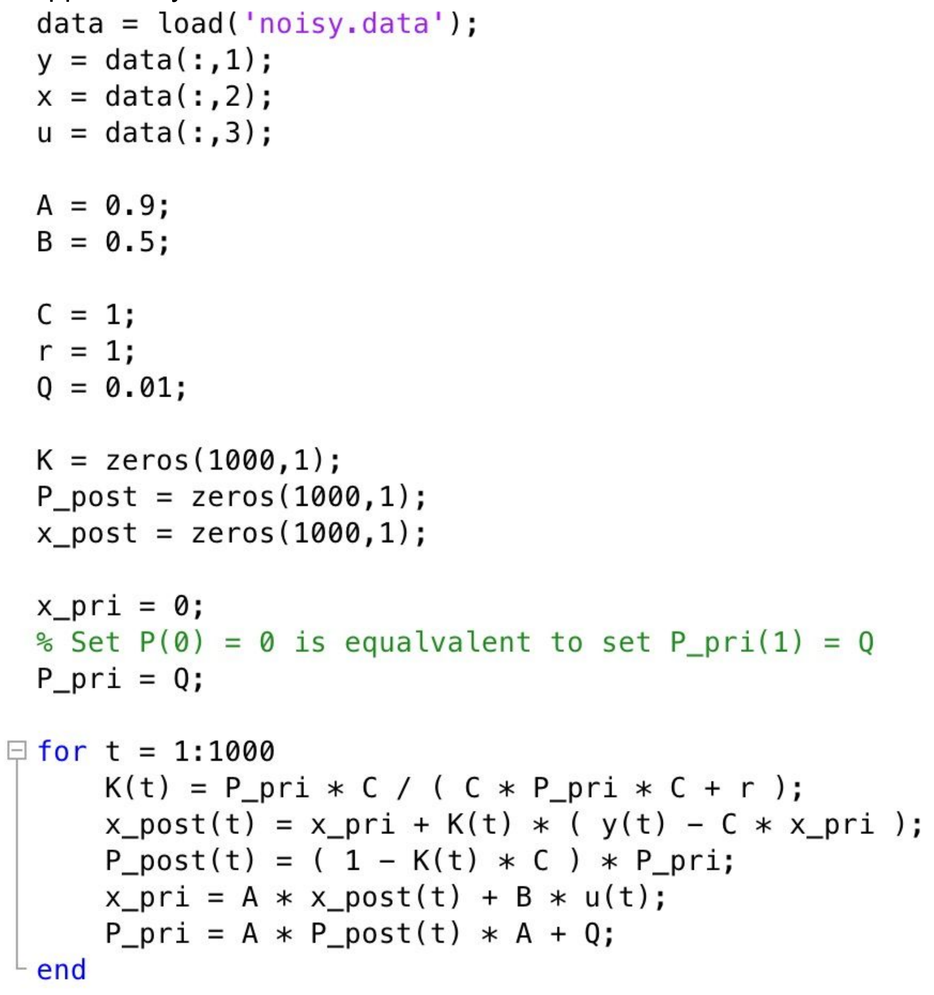


From the plots, the quality of the filter is ok and it filters out most of the noisy signal. However, it also brings some delay to the data. From the second plot, I zoom in the plot to estimate the delay. We can see that the delay is about 5 steps. I use the finddelay function to estimate the delay directly and result is 6 steps, which is 0.01*6=0.06 s.

Zhanpeng He

3.

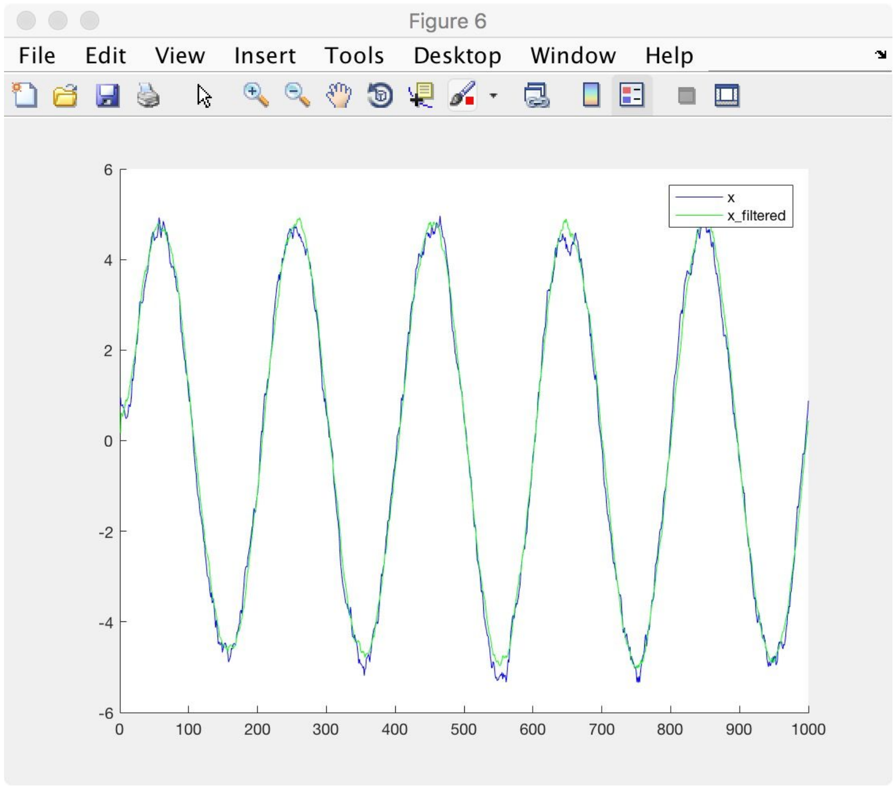
A snippet of my codes

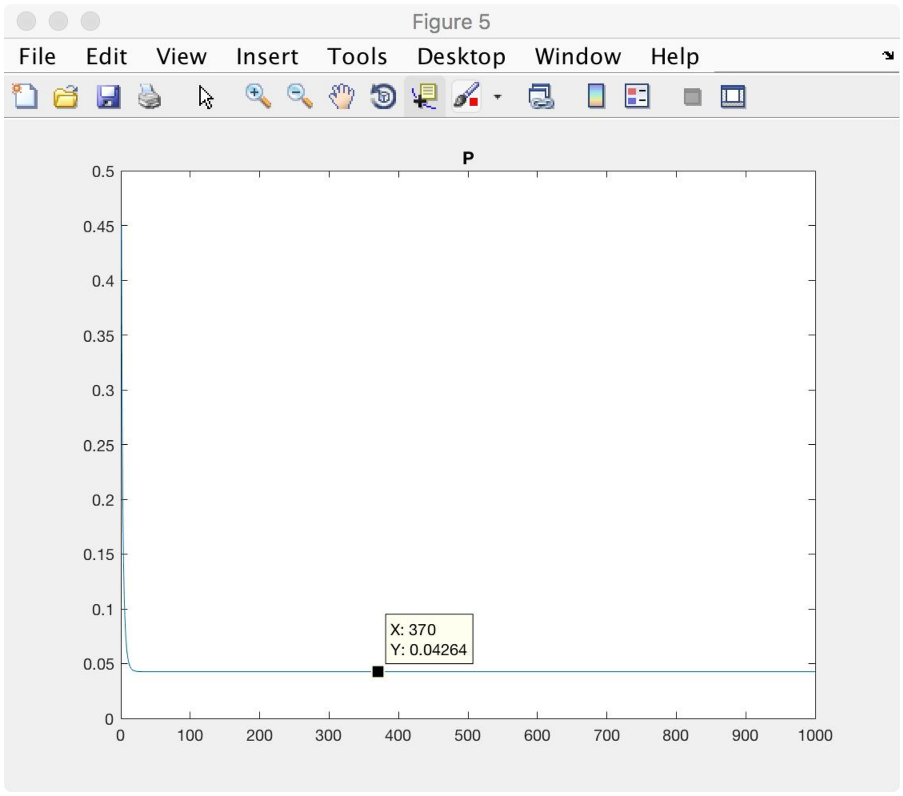


```
data = load('noisy.data');
y = data(:,1);
x = data(:,2);
u = data(:,3);

A = 0.9;
B = 0.5;

C = 1;
r = 1;
Q = 0.01;

K = zeros(1000,1);
P_post = zeros(1000,1);
x_post = zeros(1000,1);

x_pri = 0;
% Set P(0) = 0 is equalvalent to set P_pri(1) = Q
P_pri = Q;

for t = 1:1000
    K(t) = P_pri * C / ( C * P_pri * C + r );
    x_post(t) = x_pri + K(t) * ( y(t) - C * x_pri );
    P_post(t) = ( 1 - K(t) * C ) * P_pri;
    x_pri = A * x_post(t) + B * u(t);
    P_pri = A * P_post(t) * A + Q;
end
```

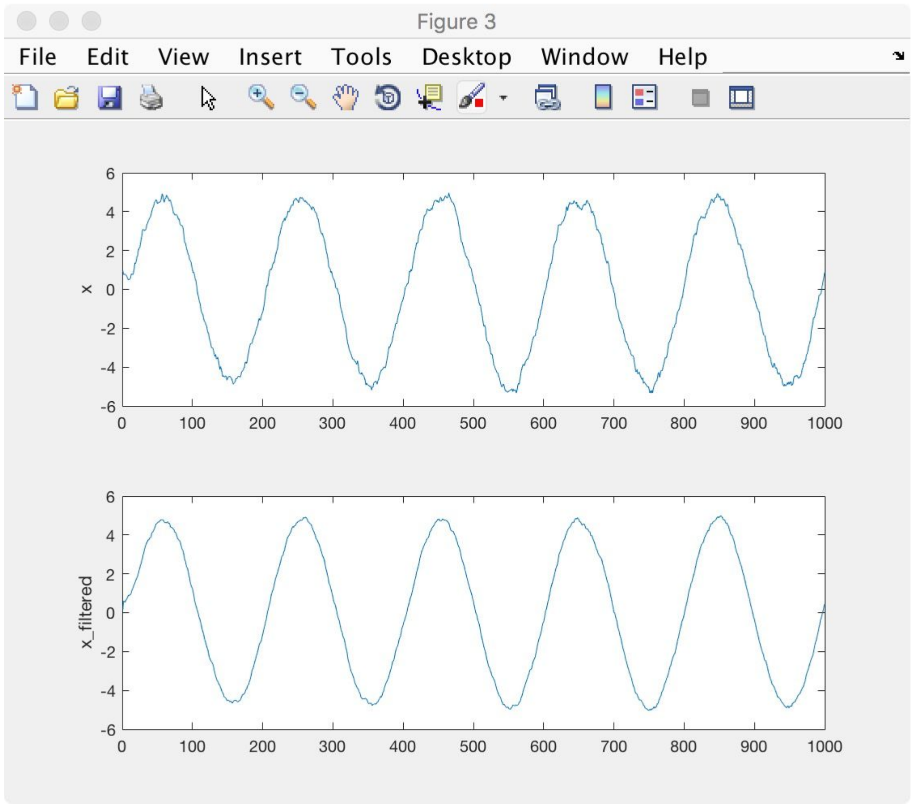
Zhanpeng He

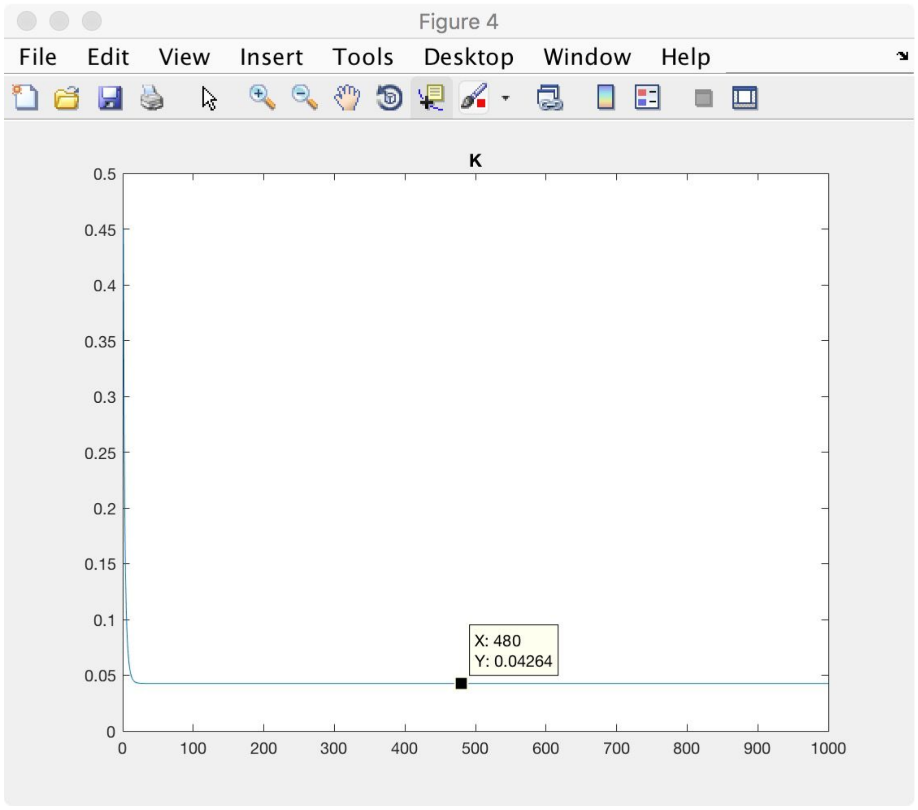
Here are some of the plots from my codes:





Zhanpeng He





I initialize P_post as A*1*A+Q, which means that I initialize P as 1. Actually, I tried different initialization of P and they all converge very fast to the same value. I used finddelay function to estimate the delay the it is only 1 step, which is 0.01s. Also compared to b), kalman filter actually filter out the noisy data better and has less delay.