a)

[B,A] = butter(N,Wn)

where N means Nth order lowpass digital Butterworth filter, and Wn means the cutoff frequency.

In this question, we use second order Butterworth filter, so N = 2.

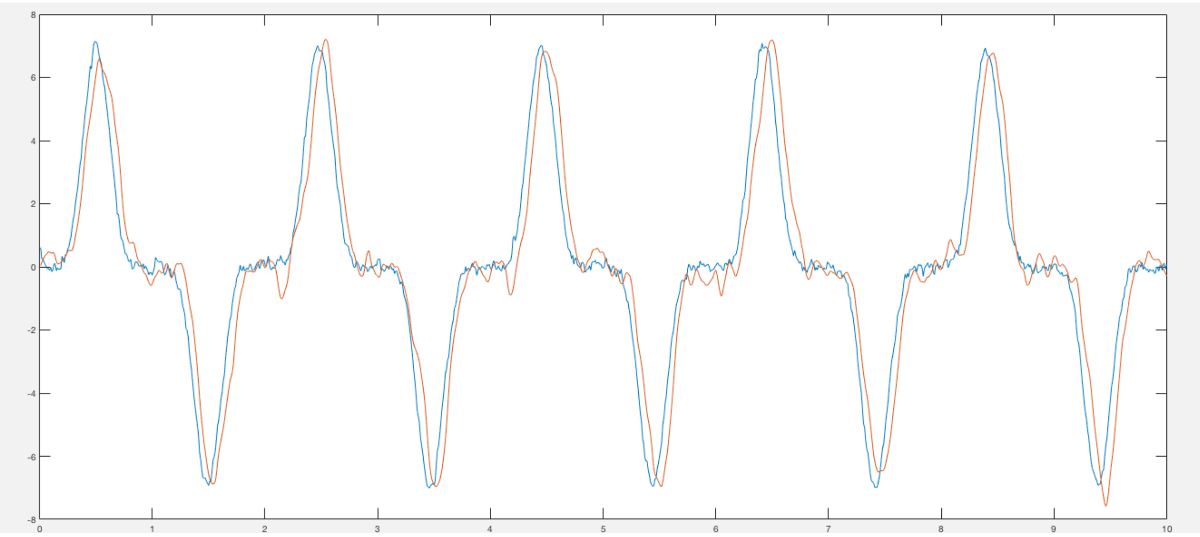
Sampling frequency is 100Hz, and cutoff frequency is 5Hz, so Wn = 5/(100/5) = 0.1.

Thus, [B,A] = butter(2, 0.1)

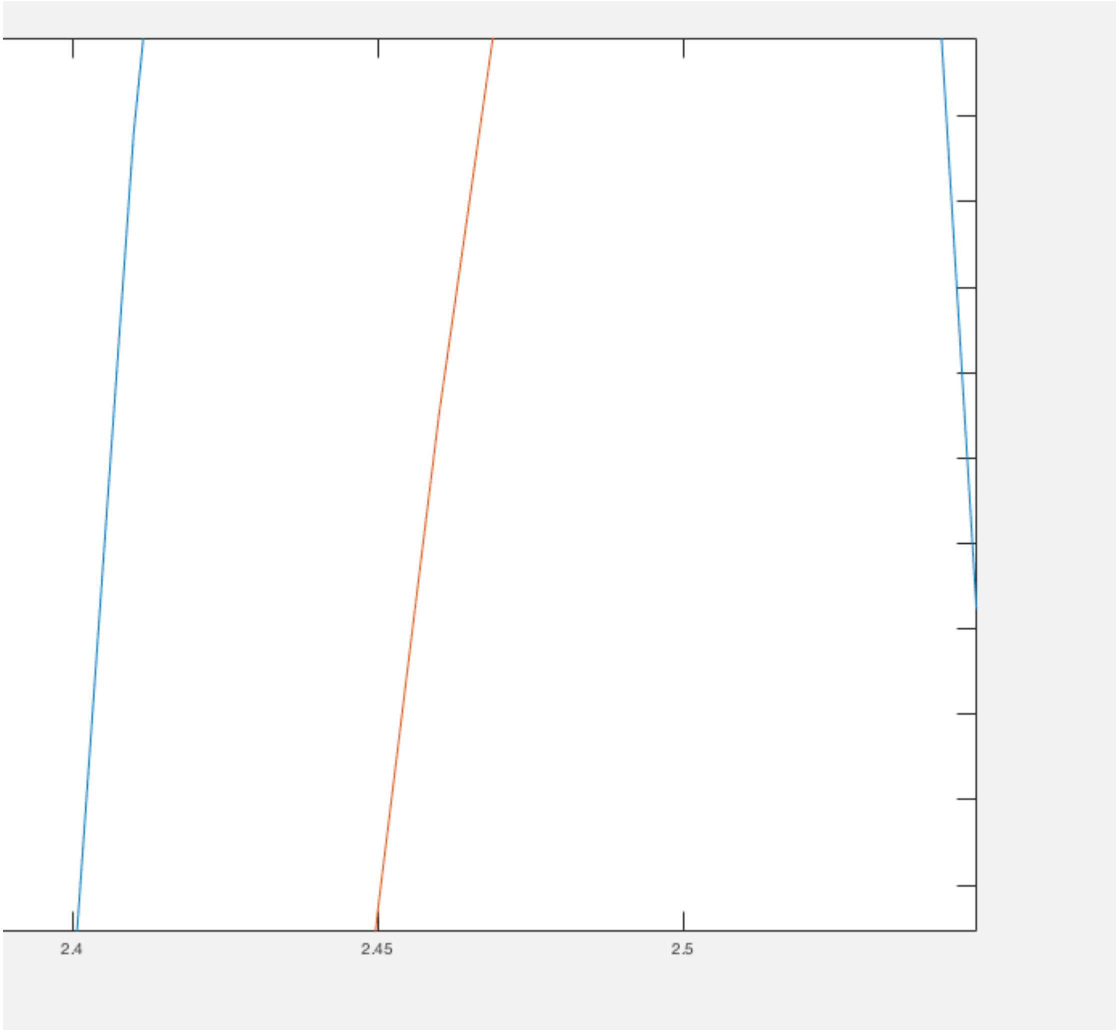
```
>>
>>
>> [B,A] = butter(2, 0.1)

B =

    0.0201    0.0402    0.0201


A =

    1.0000   -1.5610    0.6414
```

b_1 = 0.0201 b_2 = 0.0402 b_3 = 0.0201 a_2 = -1.5610 a_3= 0.6414 b)

```
input = importdata('noisy.data');
Y = input(:, 1);
X = input(:, 2);
[B,A] = butter(2, 0.1);
YY = filter(B, A, Y);
time=0.01:0.01:10;
res = [X YY];
delay = finddelay(X, YY) * 0.01
figure
plot(0.01:0.01:10, res)
```



The filter works pretty well, basically get the value we want, but has some delay, and shake a lot more when position near 0.



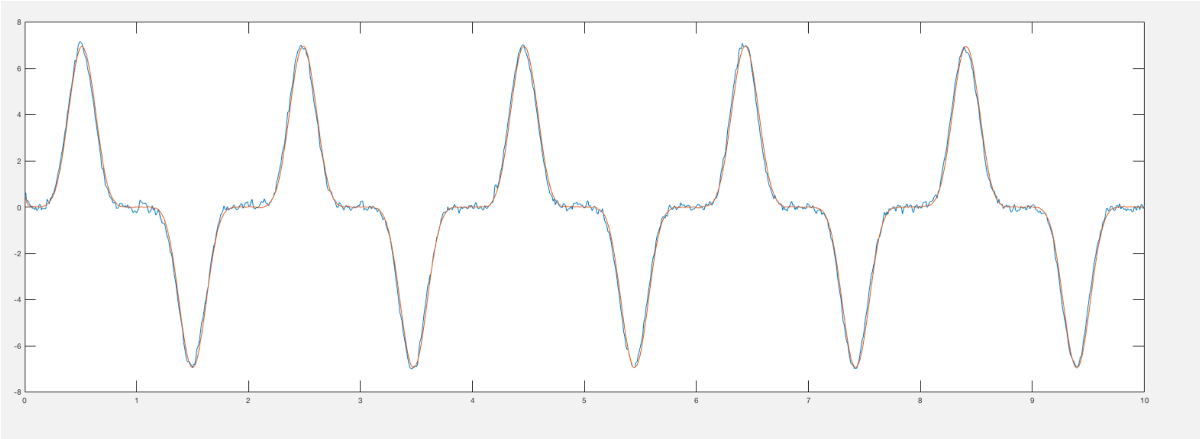
When we zoom in the plot, we can find the delay is about 0.05.

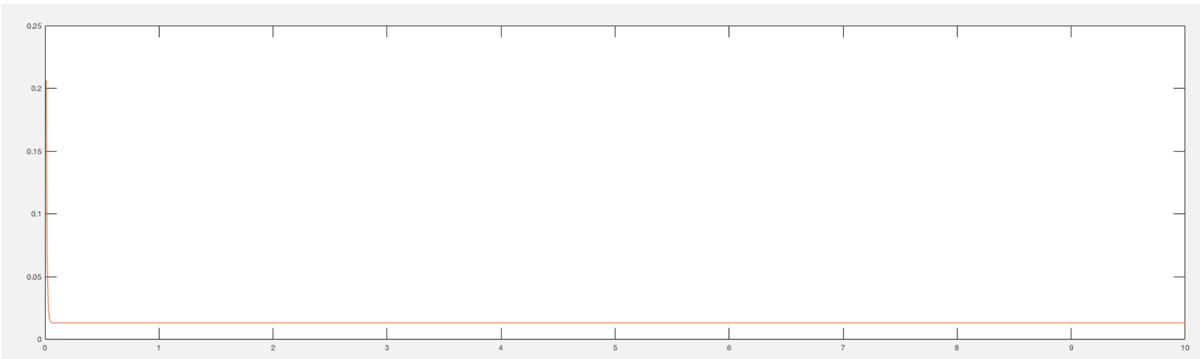
```
>> hw3b

delay =

    0.0500
```

Besides from the plot, we can also get delay from script by “finddelay” function. In conclusion, delay is 0.05

c)

```
input = importdata('noisy.data');
Y = input(:, 1);
X = input(:, 2);
U = input(:, 3);
len = length(X);

p = 1;
a = 0.5;
b = 3.5;
q = 0.01;
r = 1.0;
c = 1;
x_hat = 0;
x_tilde = 0;
K = [];
P = [];
X_Hat = [];
i = 1;
while i <= len
    p_tilde = a * p * a + q;
    k = (p_tilde * c) / (c * p_tilde * c + r);
    K = [K ; k];
    x_hat = x_tilde + k * (Y(i) - c * x_tilde);
    X_Hat = [X_Hat ; x_hat];
    p = (1 - k * c) * p_tilde;
    P = [P ; p];
    x_tilde = a * x_hat + b * U(i);
    i = i + 1;
end
delay = finddelay(X, X_Hat) * 0.01
figure
plot(0.01:0.01:10,[X X_Hat])
figure
plot(0.01:0.01:10,[K P])
```

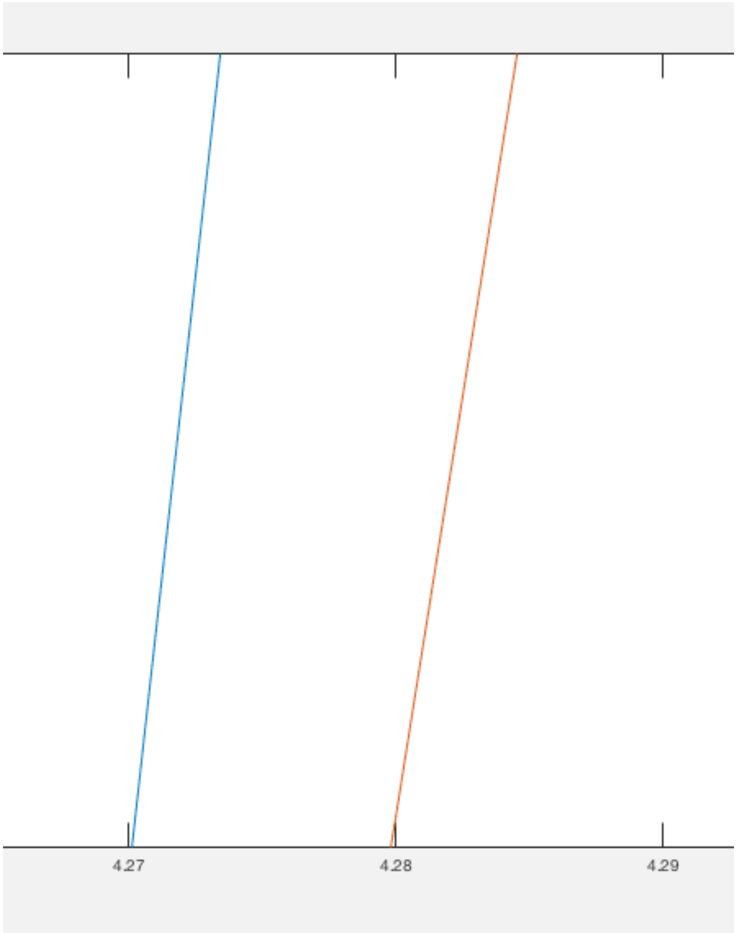




This filter works awesome, and much better than Butterworth filter. The line is almost coincident with line for X. This filter has less delay than Butterworth, and more smooth than Butterworth when position near 0. The line of this filter is even smooth than line for X.

$\mathrm{P} = \mathrm{E}\left\{e^n \, e^{n^T}\right\}$, and $e^n = \; x^n - \hat{x}^n$

Because at the beginning, we initialize $x_0 = 0$, but we don't know what exactly x_0 is, so we have some noisy at beginning, so we initialize $p_0 \; = 1$.



When we zoom in the plot, we can find the delay is about 0.01.

```
>> hw3c

delay =

    0.0100
```

Besides from the plot, we can also get delay from script by “finddelay” function. In conclusion, delay is 0.01.