Digital Signal Processing (B) Lab

Motrort Vin

Introduction

The Digital Signal Processing Laboratory is an integral and important component of the course. The laboratory has two basic objectives:
(1) Reinforce concepts from the lecture.
(2) Strengthen your ability to processing signals by using computers.

Lab 1 Block Processing Method and Sample Processing Method

Let x = [1, 1, 1, 1, 3, 3, 3, 3, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1] be an input to the filter described by the I/O equation: y(n)=x(n)+x(n-2)-2x(n-3)

1.1 Compute the corresponding output signal using convolution function of MATLAB.

%DSP_Lab_1.1.m

```
clc,clear
x = [1 1 1 1 3 3 3 3 1 1 1 2 2 2 2 1 1 1 1];
h = [1 0 1 -2];
fprintf('The Output:');
y = conv(x, h)
```

The Output:

```
y =[1 1 2 0 2 2 4 0 -2 -2 -4  1  1  2  0 -1 -1 -2  0 -1 -1 -2]
```

Discussion:

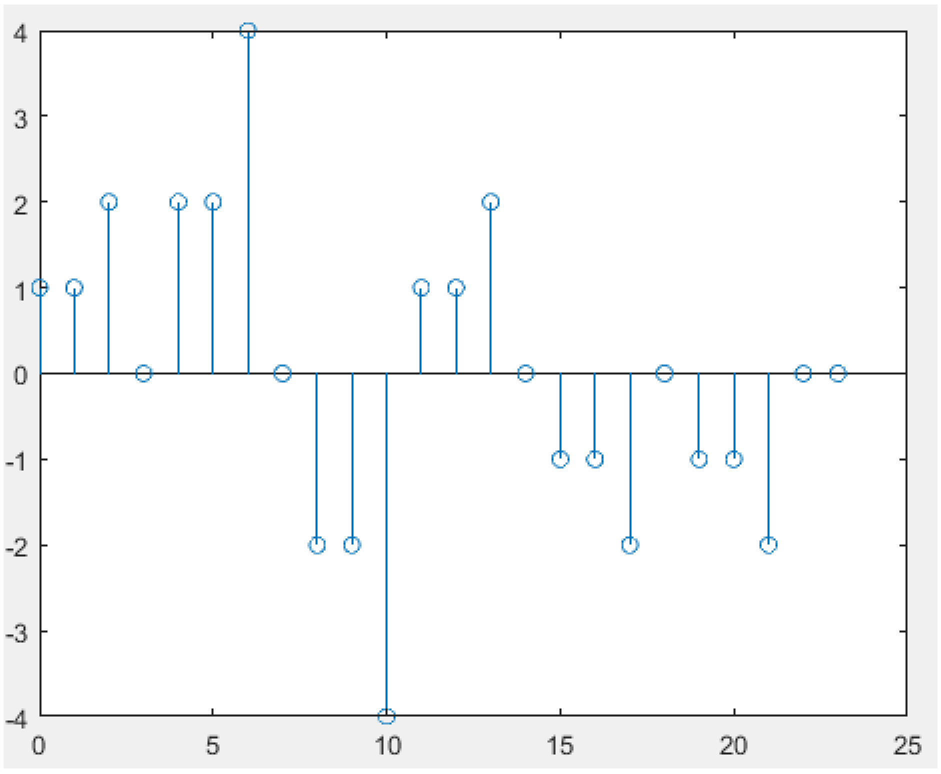
So, we can see the convolution function of MATLAB is:y=conv(x, h).

1.2 Compute the same output using the overlap-add method of block convolution by partitioning the input into length-5 blocks.

%DSP_Lab_1.2.m

```
clc,clear
L = 5 ;
x = [1 1 1 1 3; 3 3 3 1 1; 1 2 2 2 2; 1 1 1 1 0];
y = zeros(1,24);
for i =0 : 3
    t = 0 : 4;
    h = dirac(t) + dirac(t-2);
    h = 1 * sign(h)-2 * sign(dirac(t-3));
    mid = [zeros(1,i*L) conv(x(i+1,:),h) zeros(1,15-i*L)];
    y = y + mid;
end
j = 0 : 23;
fprintf('The Output:');
stem(j, y)
```

The Output:



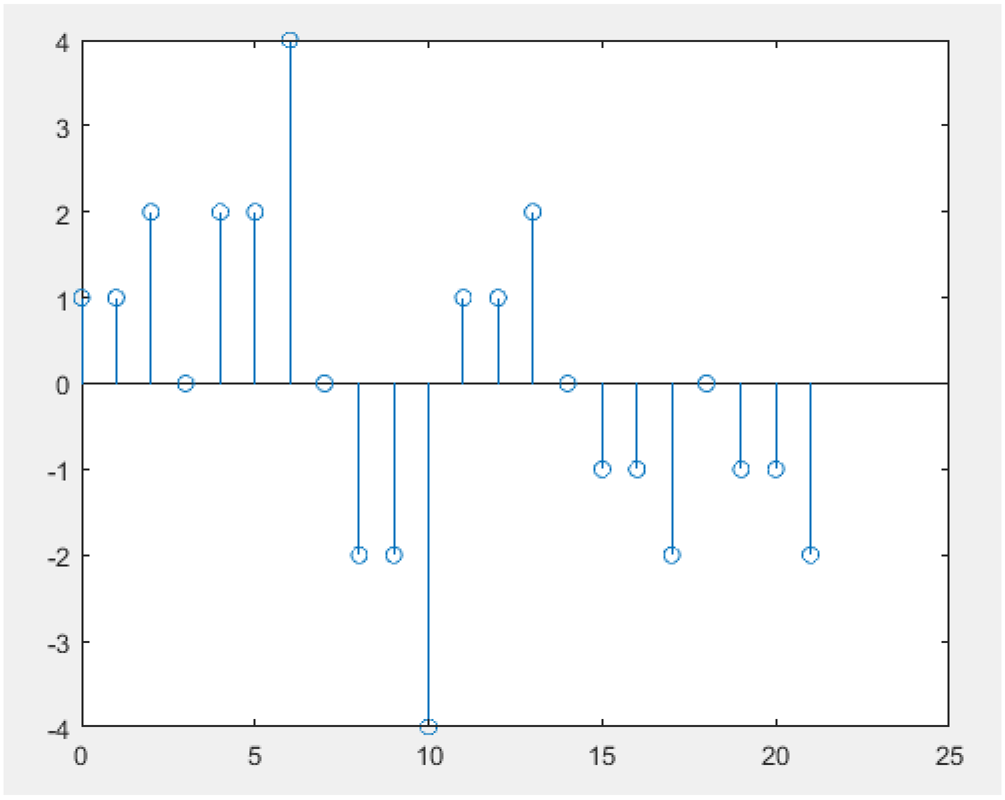
Discussion:
Because we need to using the overlap-add method of block convolution by partitioning the input into length-5 blocks and there are only 19 numbers in x, we need to put a '0' in the end as the twentieth number of x.
When forming the figure, we need to make sure that 'j' is matched with y, which means if there are N numbers in y, then the maximum of j must be N-1.

1.3 Introduce appropriate internal states, write the corresponding sample processing algorithm and present the results.

%DSP_Lab_1.3.m

```
clc,clear
x = [1 1 1 1 3 3 3 3 1 1 1 2 2 2 2 1 1 1 1];
y = zeros(1,22);
w_0 = [x 0 0 0];
w_1 = zeros(1,22);
w_2 = zeros(1,22);
w_3 = zeros(1,22);
for n= 1:22
    if (n-1)<1
        w_1(n) = 0;
    else
        w_1(n) = w_0(n-1);
    end
    if (n-2) <1
        w_2(n) = 0;
    else
        w_2(n) = w_1(n-1);
    end
    if (n-3) <1
        w_3(n) = 0;
    else
        w_3(n) = w_2(n-1);
    end
    y(n) = w_0(n) + w_2(n) - 2*w_3(n);
end
j = 0 : 21;
stem(j, y)
```

The Output:



Discussion:
In this part, we need to notice that the N of "x(N)" must be a positive number and it can't bigger than the number of the numbers in x.
When forming the figure, we need to make sure that 'j' is matched with y, which means if there are N numbers in y, then the maximum of j must be N-1.
Besides, we need to find out the length of x and h, so that we can get the length of y. In this

part, we need to make sure that the length of w_0, w_1,w_2 and w_3 is the same with which of y.

Lab 2 Pole-zero Designs

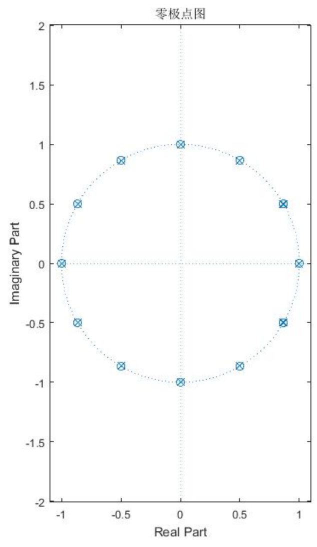
Assume a current signal of an 50Hz electrical system is sampled by the frequency of 600Hz. Use pole-zero design method to design a filter to remove all the DC and AC harmonics. Draw the magnitude response and pole-zero plot of the filter. Set the amplitude of the frequency components in the original signal, then draw the time domain curve and spectrum of the signal before and after filtering.

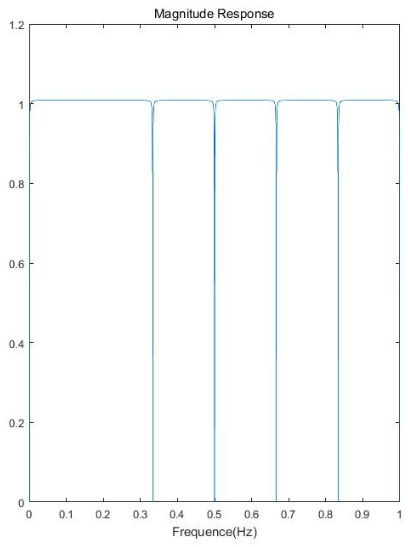
%DSP_Lab_2.m

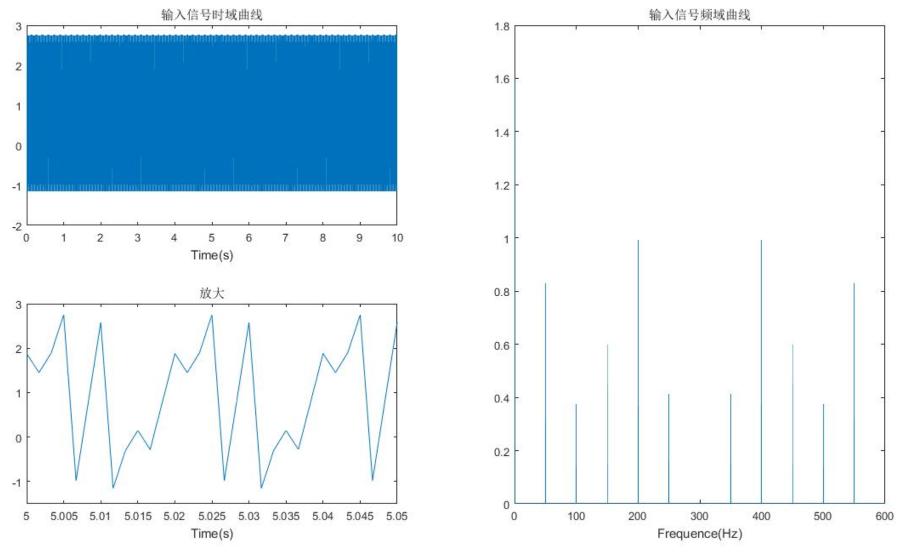
```
clc,clear
figure(1);
r = 0.98;
p = r^(1/12);
num = [1,-(exp((pi/6)*1i)+exp((-pi/6)*1i))*p,p*p,0,0,0,0,0,0,0,0,0,-
1,(exp((pi/6)*1i)+exp((-pi/6)*1i))*p,-p*p];
den = [1,-(exp((pi/6)*1i)+exp((-pi/6)*1i)),1,0,0,0,0,0,0,0,0,0,-
r,(exp((pi/6)*1i)+exp((-pi/6)*1i))*r,-r];
subplot(1,2,1)
zplane(num,den);
title('零极点图');
w = 0:pi;
[h,w] = freqz(num,den,1024*1024);
H = abs(h);
subplot(1,2,2);
plot(w/pi,H);
xlabel('Frequence(Hz)')
title('Magnitude Response');
fs = 600;
N = 1024*1024;
n = 1:N;
t = n/fs;
x = 1*sin(2*pi*50*t)+0.45*sin(4*pi*50*t)-
0.6*sin(6*pi*50*t)+1.2*sin(8*pi*50*t)-0.5*sin(10*pi*50*t)+0.8;
T = 0:1/fs:6000/fs;
figure(2);
subplot(2,2,1);
plot(T,x(1:6001));
xlabel('Time(s)');
title('输入信号时域曲线');
subplot(2,2,3);
plot(T,x(1:6001));
xlabel('Time(s)');
title(' ·放大');
```

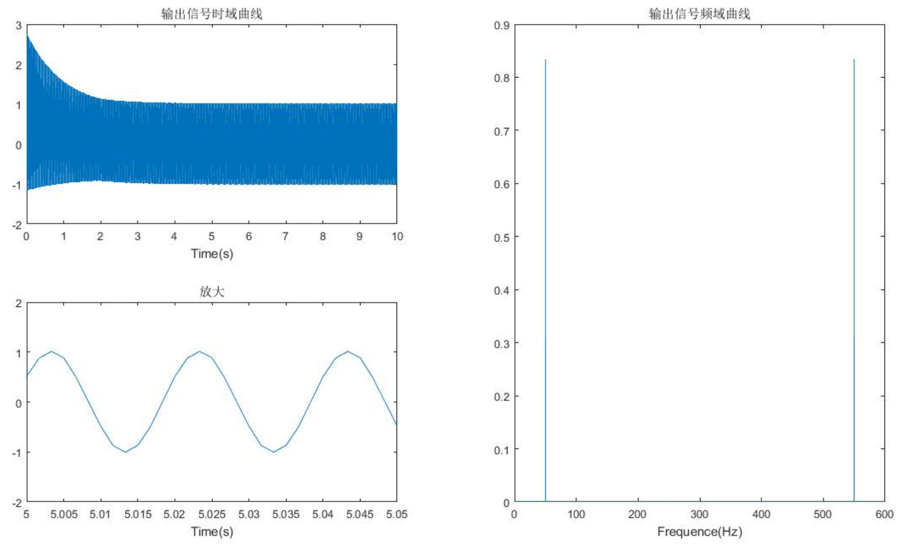
```
axis([5.0,5.05,-1.5,3])
Xtmp = fft(x,N);
X = abs(Xtmp)*2/N;
f = n*fs/N;
subplot(2,2,[2 4]);
plot(f,X);
xlabel('Frequence(Hz)')
title('输入信号频域曲线')
y=filter(num,den,x);
figure(3);
subplot(2,2,1);
plot(T,y(1:6001));
xlabel('Time(s)');
title('输出信号是与曲线');
subplot(2,2,3);
plot(T,y(1:6001));
xlabel('Time(s)');
title('放大');
axis([5.0,5.05,-2,2])
Ytmp=fft(y,N);
Y = abs(Ytmp)*2/N;
subplot(2,2,[2 4]);
plot(f,Y);
xlabel('Frequence(Hz)')
title('输出信号频域曲线')
```

The Output:









Discussion:

In this Lab, the most important thing is that we need to calculate the pole-zero plot of the filter according to the wave we want to filter.

According to the figures above, we can see that this filter works very well.

Lab 3 Frequency Resolution

Use the data in Example 9.1.3 and 9.1.4, draw Fig. 9.1.5-9.1.9, compare the results and discuss the effect of windowing.

%DSP_Lab_3.m

```
clc,clear
L0=100;
L1=200;
n0=0:L0-1;
n1=0:L1-1;
```

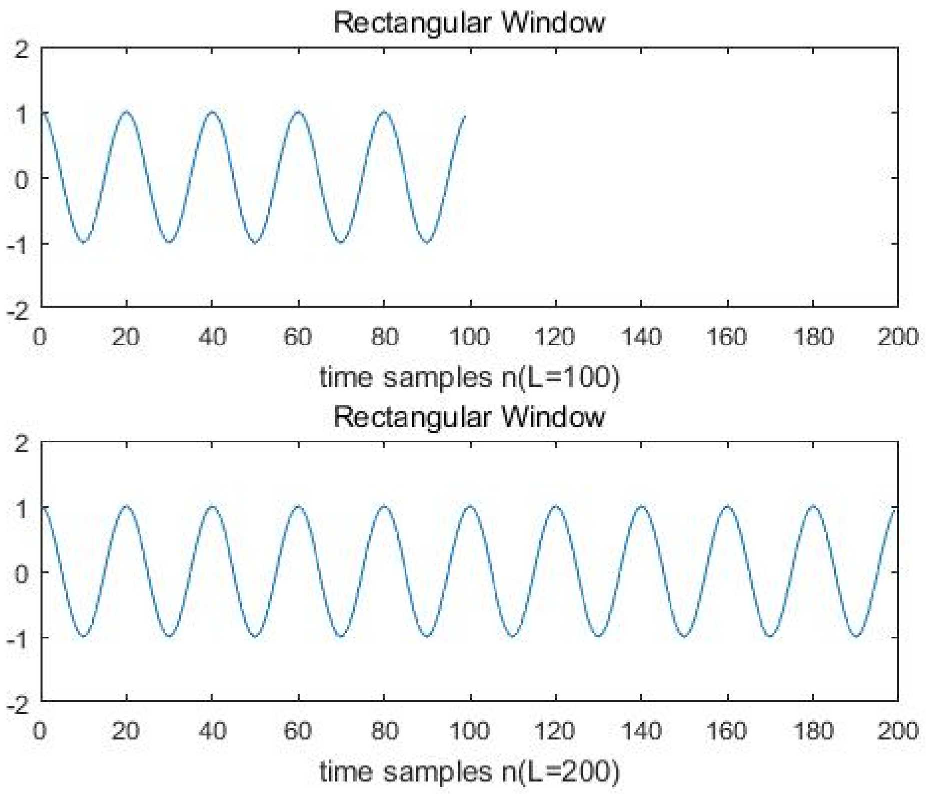
```
f0=50;
fs=1000;
x_Rect1=cos(2*pi*f0/fs*n0);
x_Rect2=cos(2*pi*f0/fs*n1);
x_Hamm1=(0.54-0.46*cos(2*pi*n0/(L0-1))).*cos(2*pi*f0/fs*n0);
x_Hamm2=(0.54-0.46*cos(2*pi*n1/(L1-1))).*cos(2*pi*f0/fs*n1);
% Fig.9.1.5
figure(1);
subplot(2,1,1);
plot(n0,x_Rect1);
axis([0 200 -2 2]);
xlabel('time samples n(L=100)');
title('Rectangular Window');
subplot(2,1,2);
plot(n1,x_Rect2);
axis([0 200 -2 2]);
xlabel('time samples n(L=200)');
title('Rectangular Window');

%Fig.9.1.6
figure(2);
subplot(2,1,1);
plot(n0,x_Hamm1);
axis([0 200 -2 2]);
xlabel('time samples n(L=100)');
title('Hamming Window');
subplot(2,1,2);
plot(n1,x_Hamm2);
axis([0 200 -2 2]);
xlabel('time samples n(L=200)');
title('Hamming Window');

%Fig.9.1.7
N=2000;
k=0:1999;
x_Rect1=fft(x_Rect1,N);
x_Hamm1=fft(x_Hamm1,N);
w=k/N;
figure(3);
subplot(1,2,1);
plot(w*2,abs(x_Rect1));
axis([0 0.2 0 100]);
xlabel('w in units of pi');
title('Magnitude Spectra, L=100');
```

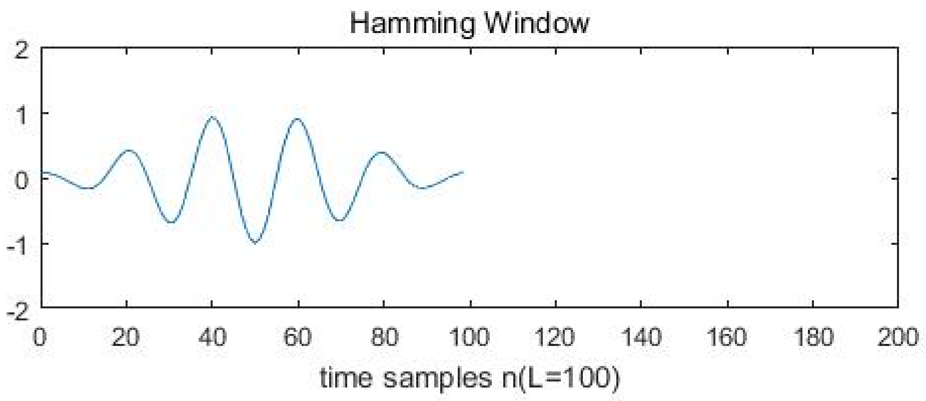
```
hold on;
plot(w*2,abs(x_Hamm1));
axis([0 0.2 0 100]);

x_Rect2=fft(x_Rect2,N);
x_Hamm2=fft(x_Hamm2,N);
subplot(1,2,2);
plot(w*2,abs(x_Rect2));
axis([0 0.2 0 100]);
xlabel('w in units of pi');
title('Magnitude Spectra, L=200');
hold on;
plot(w*2,abs(x_Hamm2));
axis([0 0.2 0 100]);

f1=2000;
f2=2500;
f3=3000;
fs=10000;
L_10=10;
L_20=20;
L_40=40;
L_100=100;
a0=0:L_10-1;
a1=0:L_20-1;
a2=0:L_40-1;
a3=0:L_100-1;

%Fig.9.1.8
x_R10=cos(2*pi*f1/fs*a0)+cos(2*pi*f2/fs*a0)+cos(2*pi*f3/fs*a0);
X_R10=fft(x_R10,N);
figure(4);
subplot(2,2,1);
plot(w,abs(X_R10));
axis([0 1 0 50]);
xlabel('f/fs');
title('Rectangular Window, L=10');

x_H10=(cos(2*pi*f1/fs*a0)+cos(2*pi*f2/fs*a0)+cos(2*pi*f3/fs*a0)).*(0.
54-0.46*cos(2*pi*a0/(L_10-1)));
X_H10=fft(x_H10,N);
subplot(2,2,2);
plot(w,abs(X_H10));
axis([0 1 0 50]);
```

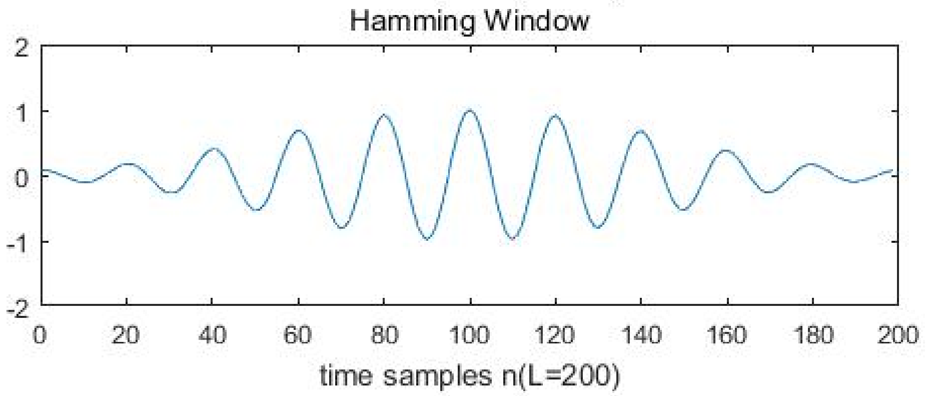
```
xlabel('f/fs');
title('Hamming Window, L=10');

x_R20=cos(2*pi*f1/fs*a1)+cos(2*pi*f2/fs*a1)+cos(2*pi*f3/fs*a1);
X_R20=fft(x_R20,N);
subplot(2,2,3);
plot(w,abs(X_R20));
axis([0 1 0 50]);
xlabel('f/fs');
title('Rectangular Window, L=20');

x_H20=(cos(2*pi*f1/fs*a1)+cos(2*pi*f2/fs*a1)+cos(2*pi*f3/fs*a1)).*(0.
54-0.46*cos(2*pi*a1/(L_20-1)));
X_H20=fft(x_H20,N);
subplot(2,2,4);
plot(w,abs(X_H20));
axis([0 1 0 50]);
xlabel('f/fs');
title('Hamming Window, L=20');

%Fig.9.1.9
x_R40=cos(2*pi*f1/fs*a2)+cos(2*pi*f2/fs*a2)+cos(2*pi*f3/fs*a2);
X_R40=fft(x_R40,N);
figure(5);
subplot(2,2,1);
plot(w,abs(X_R40));
axis([0 1 0 50]);
xlabel('f/fs');
title('Rectangular Window, L=40');

x_H40=(cos(2*pi*f1/fs*a2)+cos(2*pi*f2/fs*a2)+cos(2*pi*f3/fs*a2)).*(0.
54-0.46*cos(2*pi*a2/(L_40-1)));
X_H40=fft(x_H40,N);
subplot(2,2,2);
plot(w,abs(X_H40));
axis([0 1 0 50]);
xlabel('f/fs');
title('Hamming Window, L=40');

x_R100=cos(2*pi*f1/fs*a3)+cos(2*pi*f2/fs*a3)+cos(2*pi*f3/fs*a3);
X_R100=fft(x_R100,N);
subplot(2,2,3);
plot(w,abs(X_R100));
axis([0 1 0 50]);
```

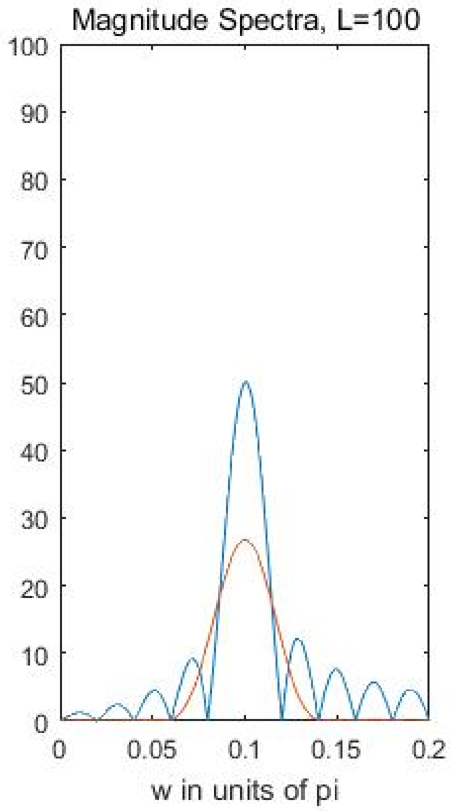
```
xlabel('f/fs');
title('Rectangular Window, L=100');

x_H100=(cos(2*pi*f1/fs*a3)+cos(2*pi*f2/fs*a3)+cos(2*pi*f3/fs*a3)).*(0
.54-0.46*cos(2*pi*a3/(L_100-1)));
X_H100=fft(x_H100,N);
subplot(2,2,4);
plot(w,abs(X_H100));
axis([0 1 0 50]);
xlabel('f/fs');
title('Hamming Window, L=100');
```

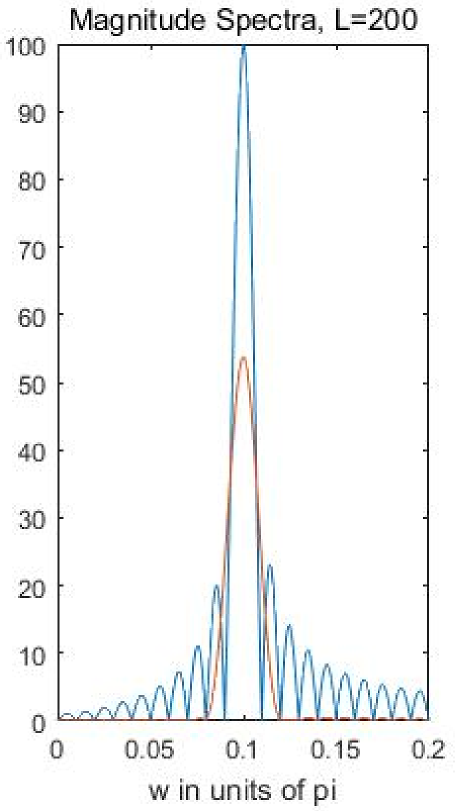
The Output:

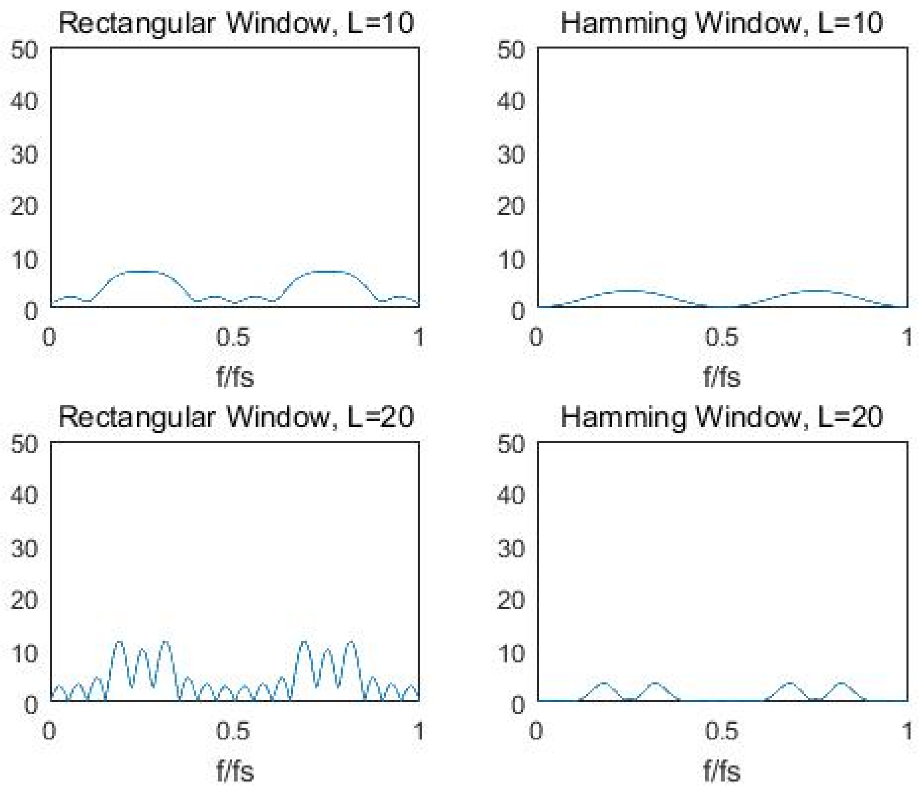


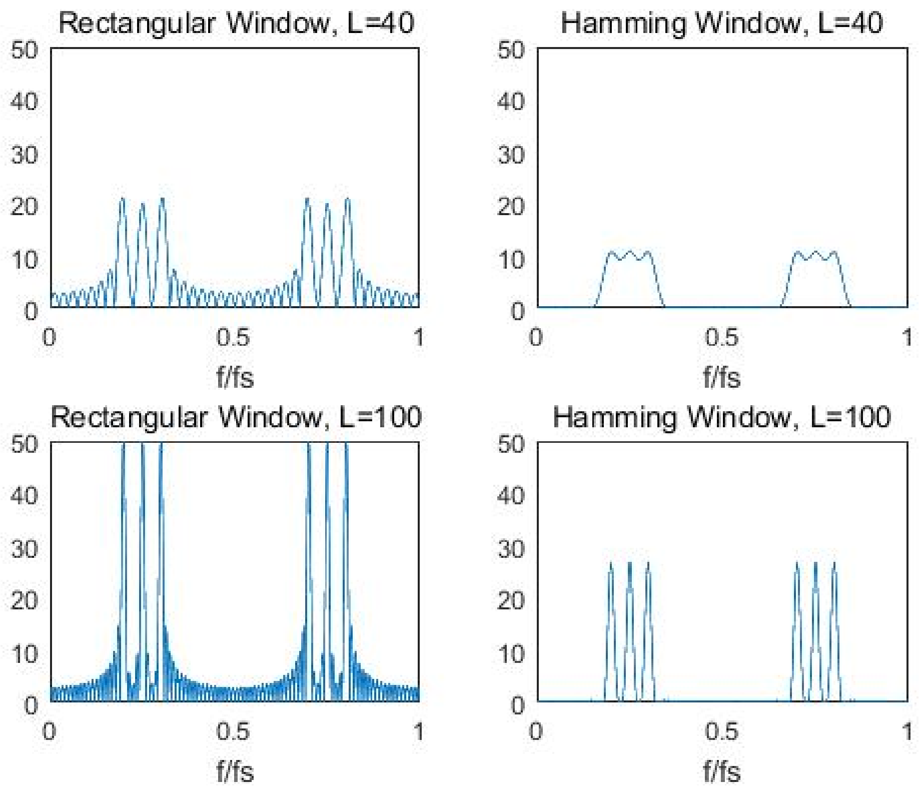

Hamming Window
time samples n(L=100)


Hamming Window
time samples n(L=200)


Magnitude Spectra, L=100
w in units of pi


Magnitude Spectra, L=200
w in units of pi


Rectangular Window, L=10
Hamming Window, L=10
Rectangular Window, L=20
Hamming Window, L=20
f/fs


Rectangular Window, L=40
Hamming Window, L=40
Rectangular Window, L=100
Hamming Window, L=100
f/fs

Discussion:

As you can see, the result of Lab3 is pretty successful. The figures above is basically the same as which in our picture.

Lab 4 Digital Filter Design

This lab involves the design of FIR and IIR filters. You may use the following Matlab functions: filter(), ones(), zeros(), zp2tf(), ceil(), buttap(), real(), poly(), impulse(), hanning(), hamming(), round(), sin(), freqz(), angle(), abs().

(1) The transfer function of a discrete-time system is

$$H(z)=\frac{1-az^{-1}+bz^{-2}-cz^{-3}+dz^{-4}}{1+0.22z^{-1}+0.037z^{-2}+0.142z^{-3}-0.107z^{-4}-0.013z^{-5}}$$

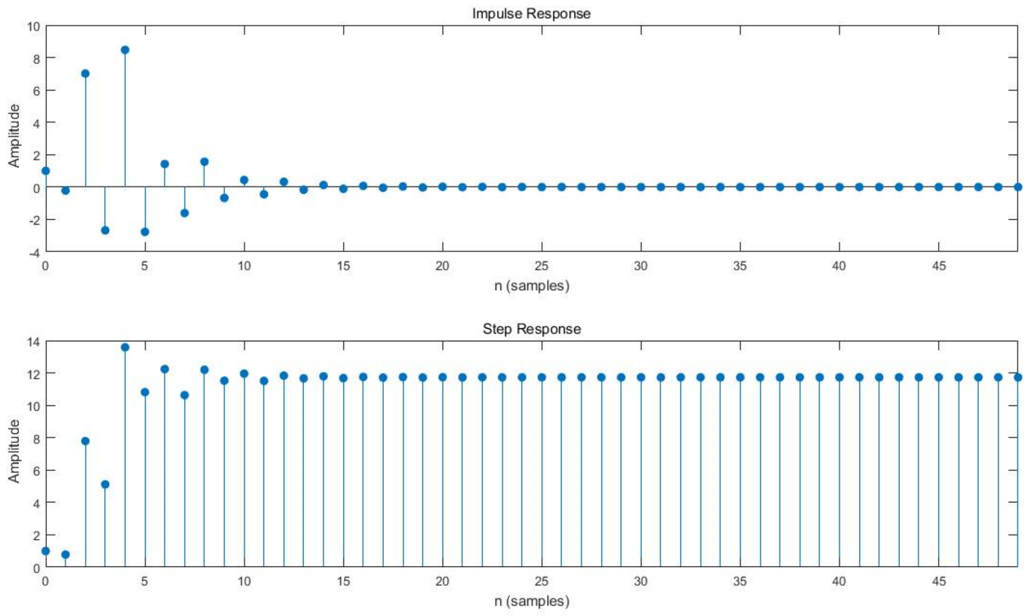
where abcd is equal to the last four digits of your student's ID number.

Determine the impulse response and step response of the system.

%DSP_Lab_4.1.m

```
clc,clear
num=[1,0,7,-1,8,0]; %The last four digits are 0,7,1 and 8.
mot=[1,0.22,0.037,0.142,-0.107,-0.013];
figure(1);
subplot(2,1,1);
impz(num,mot,50);
subplot(2,1,2);
stepz(num,mot,50);
```

The Output:



Discussion:

So, as you can see, this transfer function is build successfully, and the results that Matlab output are also able to prove it. We can see that the output signal is unstable at first, but as time passed, the signal become to be stable.

(2) Plot the magnitude squared frequency response of 5th-, 10th-, 20th-, and xth-order lowpass Butterworth filter, where

$$x = \left(c + \sum_i I_i \right) \bmod 20$$

c = your class number

I_i : the ith digit in your student ID number

If x happens to be 5 or 10, then let $x = 2$.

My class number is that 1603014.

My student's ID number is that 516021910718.

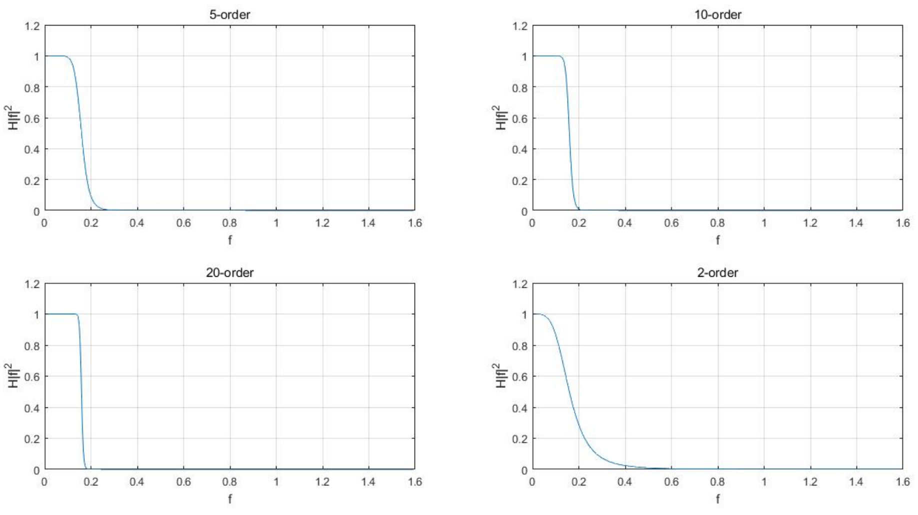
x= (1603104+5+1+⋯+7+1+8) mod 20 =5, so I let x=2.

%DSP_Lab_4.2.m

```
clc,clear
figure(1)
subplot(2,2,1);
%5th-order
[z,p,k] = buttap(5);          % Butterworth filter prototype
[num,den] = zp2tf(z,p,k);     % Convert to transfer function form
[H,w]=freqs(num,den);         % Frequency response of analog filter
Hf=abs(H);
Hf2=power(Hf,2);
f=w/(2*pi);
plot(f,Hf2);
grid on
ylabel('H|f|^2')
xlabel('f')
title('5-order');
%10th-order
subplot(2,2,2);
[z,p,k] = buttap(10);         % Butterworth filter prototype
[num,den] = zp2tf(z,p,k);     % Convert to transfer function form
[H,w]=freqs(num,den);         % Frequency response of analog filter
Hf=abs(H);
Hf2=power(Hf,2);
f=w/(2*pi);
plot(f,Hf2);
grid on
ylabel('H|f|^2')
xlabel('f')
title('10-order');
%20th-order
subplot(2,2,3);
[z,p,k] = buttap(20);         % Butterworth filter prototype
[num,den] = zp2tf(z,p,k);     % Convert to transfer function form
```

```
[H,w]=freqs(num,den);          % Frequency response of analog filter
Hf=abs(H);
Hf2=power(Hf,2);
f=w/(2*pi);
plot(f,Hf2);
grid on
ylabel('H|f|^2')
xlabel('f')
title('20-order');
%2th-order
subplot(2,2,4);
[z,p,k] = buttap(2);           % Butterworth filter prototype
[num,den] = zp2tf(z,p,k);       % Convert to transfer function form
[H,w]=freqs(num,den);          % Frequency response of analog filter
Hf=abs(H);
Hf2=power(Hf,2);
f=w/(2*pi);
plot(f,Hf2);
grid on
ylabel('H|f|^2')
xlabel('f')
title('2-order');
```

The Output:



Discussion:

So what we know from this part is that the greater the order of filter is, the faster the waveform decline. Another thing we get from this part is that butter filter is a useful filter. In the future, I think it’s a nice way to make a filter.

(3) Design an FIR bandstop filter with the following specifications:

Lower transition band: $\omega_{pa} = 0.4\pi, \omega_{sa} = 0.45\pi$

Upper transition band: $\omega_{sb} = 0.65\pi, \omega_{pb} = 0.7\pi$

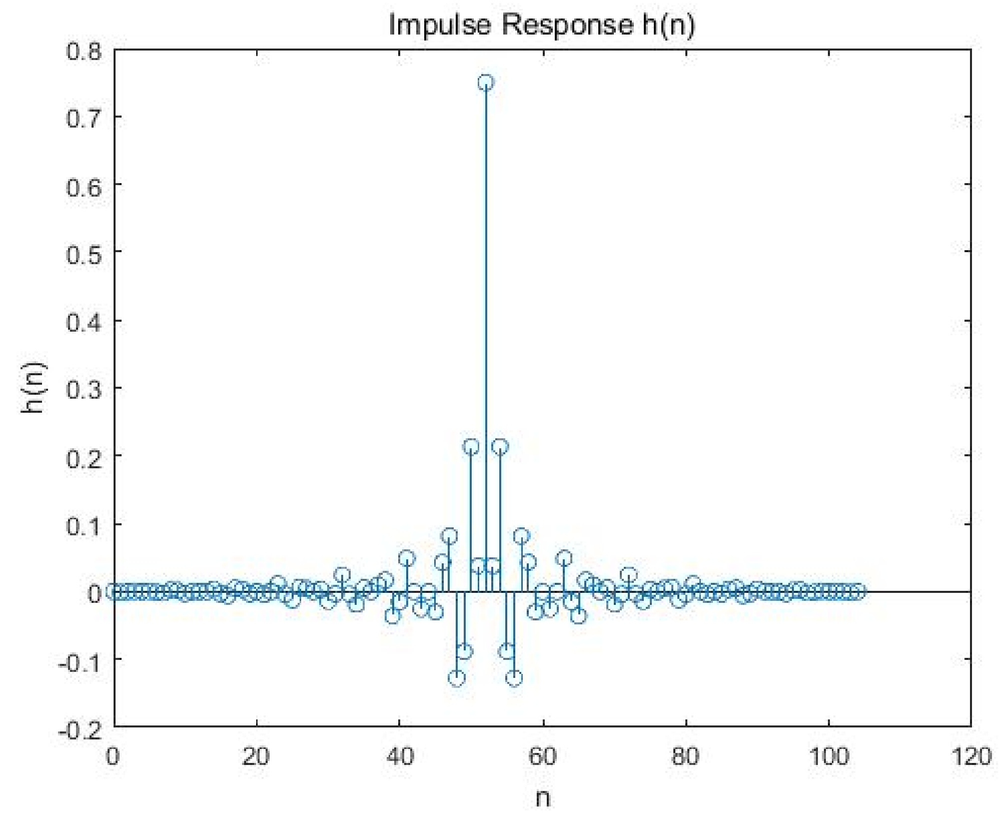
$A_{pass} = 0.5$ dB, $A_{stop} = 41$ dB

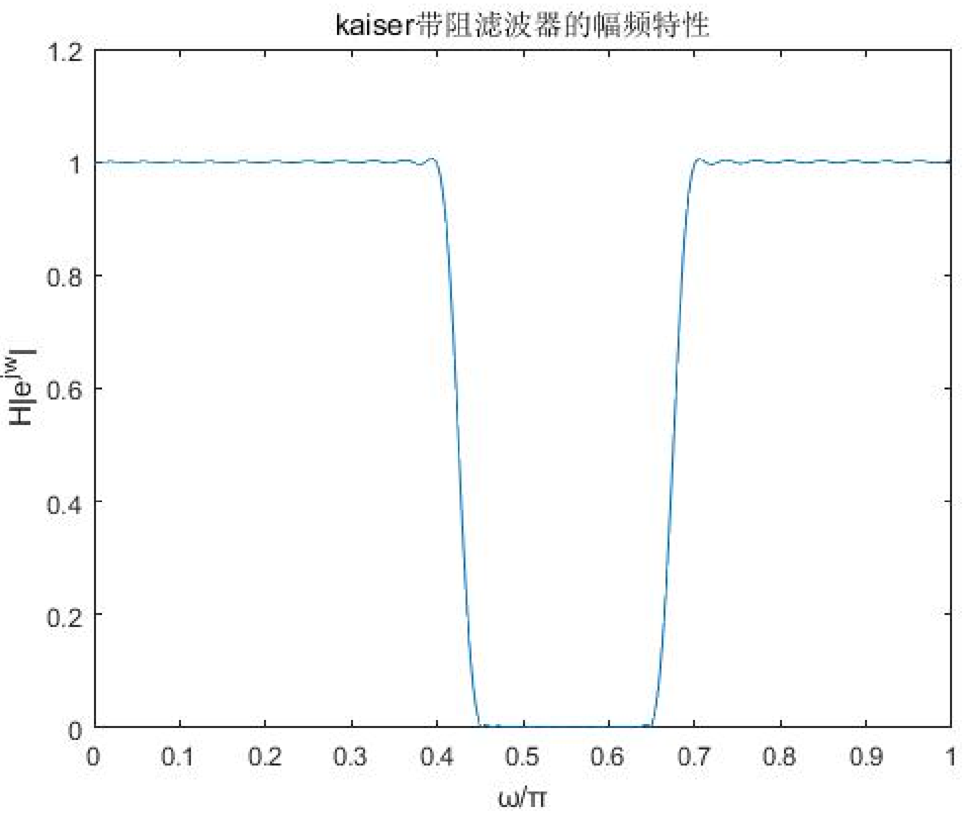
Determine and plot $h(n)$ and $|H(e^{j\omega})|$.

```
%DSP_Lab_4.3.m
clc;
close all;
clear all;

%Parameters
Ap_ = 0.1;     % Min Passband Ripple (db)
Aa_ = 41;       % Min Stopband Attenuation (db)
Wpa = 0.4*pi;      % Lower passband Freq (rad/s)
Wsa = 0.45*pi;      % Lower stopband Freq (rad/s)
Wsb = 0.65*pi;      % Higher stopband Freq (rad/s)
Wpb = 0.7*pi;      % Higher stopband Freq (rad/s)
Ws  = 1;     % Sampling Freq (rad/s)

Df = min(Wsa-Wpa, Wpb-Wsb)/2/pi;
W1 = (Wpa+Wsa)/2;
W2 = (Wsb+Wpb)/2;

% 2. Choose Delta

delta_a = 10^(-Aa_/20);
c = 10 ^ ( Ap_ / 20);
delta_p = (c-1)/(c+1);
delta = min(delta_a, delta_p);

% 3. Get Aa from delta
Aa = -20*log10(delta);          % Actual stopband attenuation

% 4. Calculate alpha
if (Aa <= 21)
   alpha = 0;
elseif ((21 < Aa) && (Aa <= 50))
   alpha = 0.5842*(Aa-21)^0.4 + 0.07886*(Aa-21);
else
   alpha = 0.1102*(Aa - 8.7);
end

% 5. Calculate D and N
```

```
if (Aa <= 21)
    D = 0.9222;
else
    D = (Aa - 7.95)/14.36;
end

N = ceil ( Ws * D / Df +1);

if (mod(N,2) == 0)
    N = N + 1;
end

% 6. Form Kaiser window

M = (N-1)/2;
n = 0:1:N-1;
wn = kaiser(N,alpha)';
alpha_1=(N-1)/2;
m=n-alpha_1+eps;
a1=sin(pi*m)./(pi*m);
a2=sin(W1*m)./(pi*m);
a3=sin(W2*m)./(pi*m);

dn = a1 + a2 -a3;
h = wn.*dn;
figure(1)
stem(n,h);
title('Impulse Response h(n)');
xlabel('n');
ylabel('h(n)');
[x,k] = freqz(h,1,512);
figure(2)
plot(k/pi,abs(x));
title('kaiser带阻滤波器的幅频特性');
xlabel('w/pi');
ylabel('H|e^j^w|');
```

The Output:





Discussion:

So we can see that, the result of this lab is very successful. In this part, I choose the Kaiser

Window. Although the calculation of this part is hard and confused, the FIR filter that I designed works very well. If you check the figure below, you may find out that the filter match the condition of the Lab very well. What we can tell ourselves is that it’s a successful Lab.