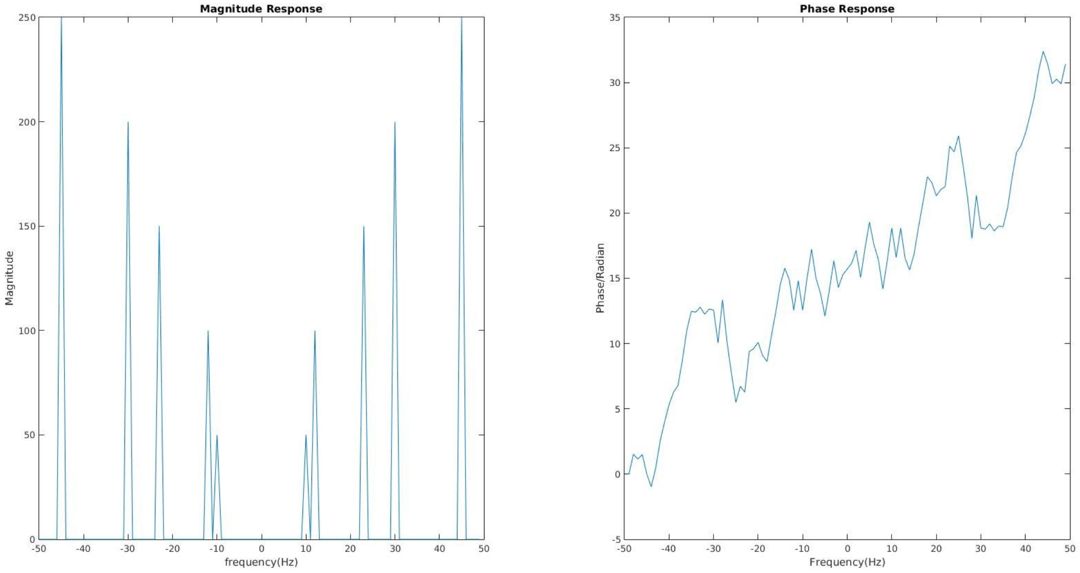
Hanfei Geng
hgeng4
Dec 10th, 2016

Q1
1)
code:

```
 load('signal.mat')
signal = x
size = length(signal)
S = fft(signal,size);
S = fftshift(S)

w = fftshift([0:size-1]/size*2*pi)
w(1:size/2)=w(1:size/2)-2*pi
fs = 100

%plotting
freq = fs*w/2/pi;
subplot(121)
plot(freq,abs(S))
title('Magnitude Response')
xlabel('frequency(Hz)')
ylabel('Magnitude')
subplot(122)
plot(freq,phase(S))
title('Phase Response')
xlabel('Frequency(Hz)')
ylabel('Phase/Radian')
```



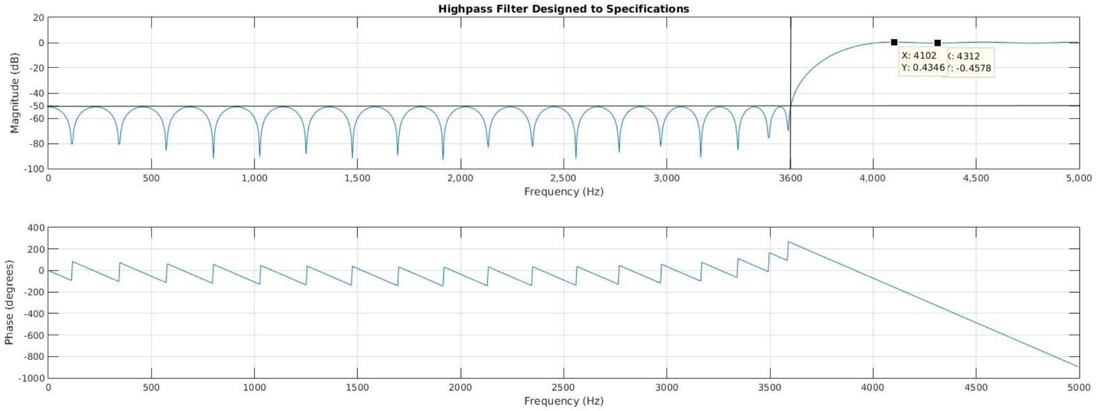
There are 5 tones. They are at 45Hz,30Hz,,23Hz,12Hz,10Hz.

Q2:

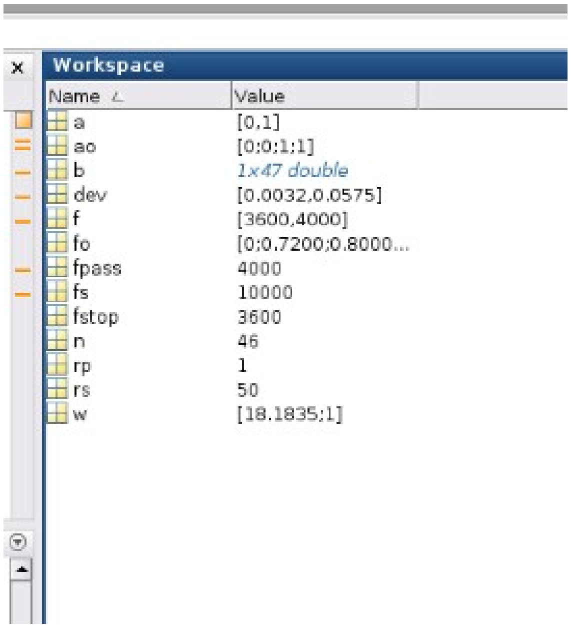
1)

code:

```
fs = 10000
fstop = 3600
fpass = 4000
rs = 50
rp = 1

f = [fstop fpass]
a = [0 1]

dev = [10^(-rs/20) (10^(rp/20)-1)/(10^(rp/20)+1)];
[n,fo,ao,w] = firpmord(f,a,dev,fs);
b = firpm(n,fo,ao,w);
freqz(b,1,1024,fs)
title('Highpass Filter Designed to Specifications')
```



The length of the filter:47

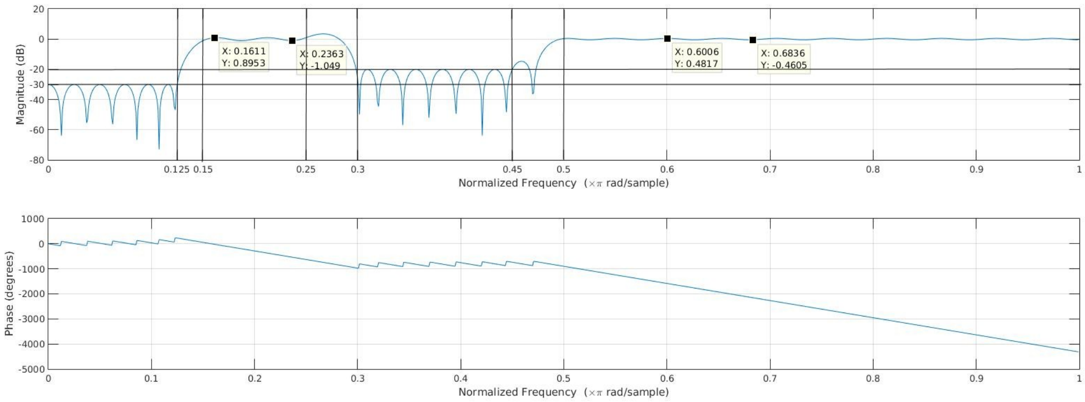


Workspace

Name	Value
a	[0,1]
ao	[0;0;1;1]
b	*1x47 double*
dev	[0.0032,0.0575]
f	[3600,4000]
fo	[0;0.7200;0.8000...
fpass	4000
fs	10000
fstop	3600
n	46
rp	1
rs	50
w	[18.1835;1]

2)
code:

```
rs1 = 30
rs2 = 20
rp1 = 2
rp2 = 1

fs = 1
f = [0.125/2,0.15/2,0.25/2,0.3/2,0.45/2,0.5/2]
dev = [10^(-rs1/20),(10^(rp1/20)-1)/(10^(rp1/20)+1),10^(-rs2/20),(10^(rp2/20)-1)/
(10^(rp2/20)+1)]
a = [0,1,0,1]

[n,fo,ao,w] = firpmord(f,a,dev,fs);
b = firpm(n,fo,ao,w);
freqz(b,1,1024)
```

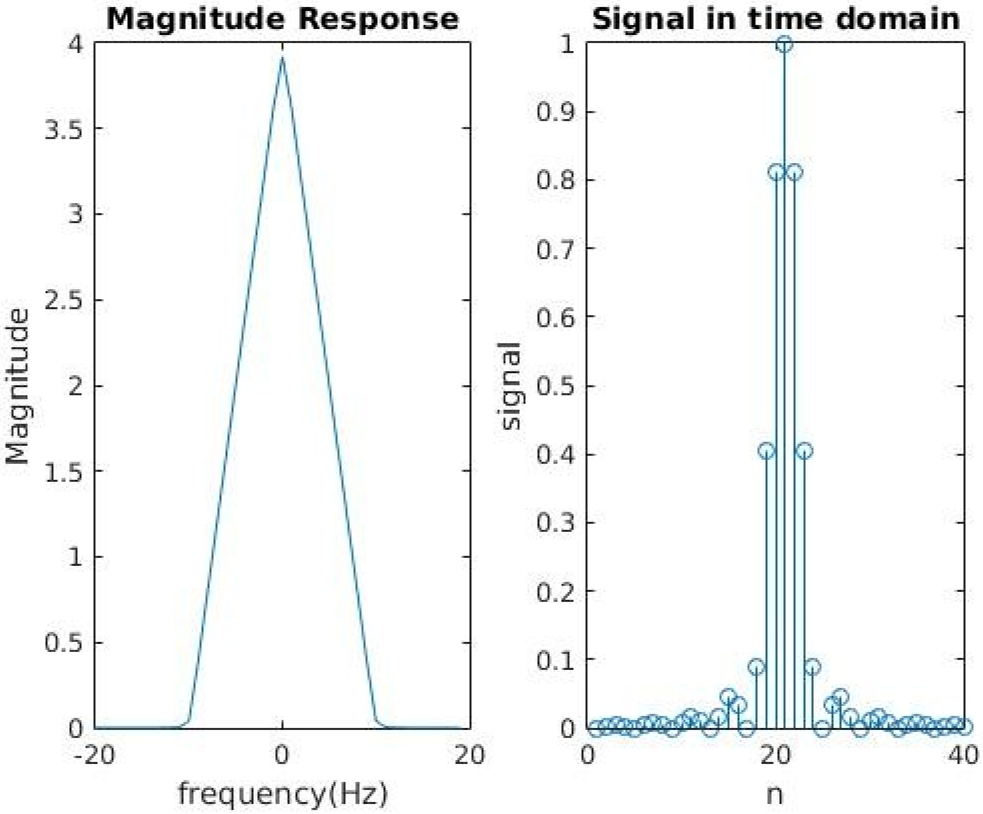


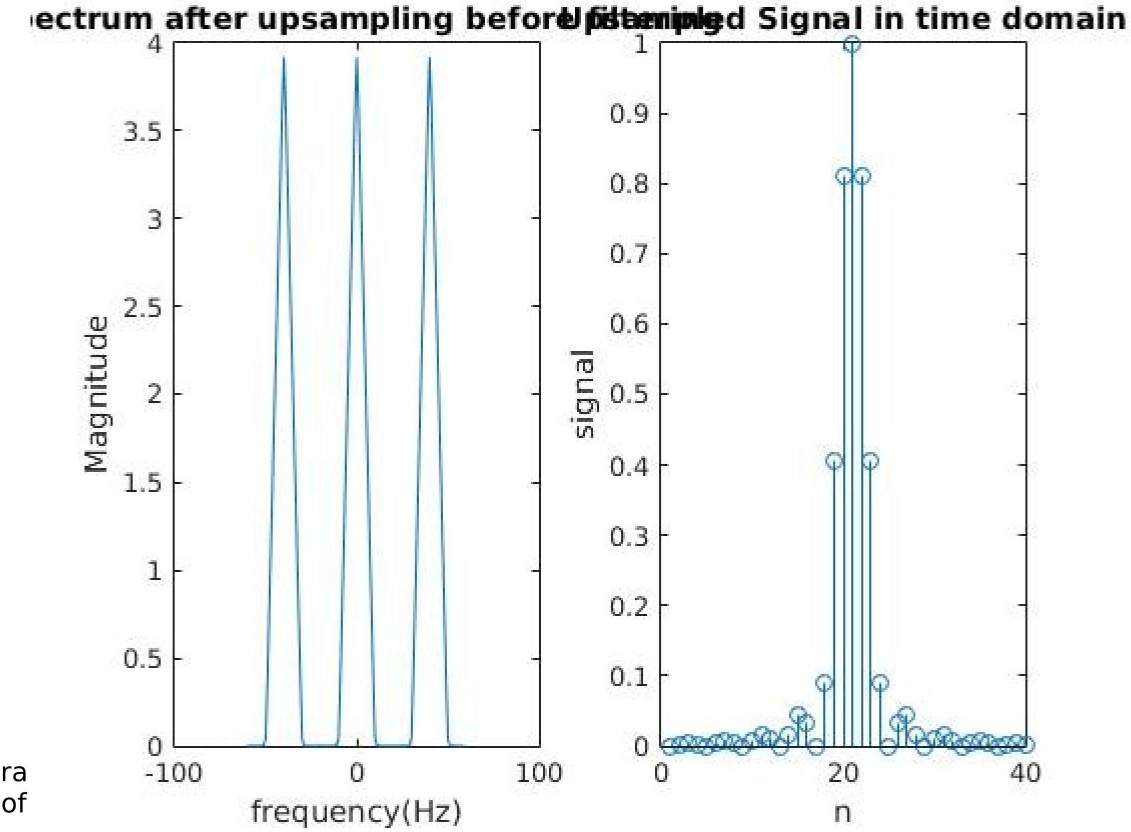
Q3
code:

```
load('samplerate.mat')
signal = x

%part1
size = length(signal)
S = fft(signal,size);
S = fftshift(S)

w = fftshift([0:size-1]/size*2*pi)
w(1:size/2)=w(1:size/2)-2*pi
fs = 40

%plotting
freq = fs*w/2/pi;
figure(1)
subplot(121)
plot(freq,abs(S))
title('Magnitude Response')
xlabel('frequency(Hz)')
ylabel('Magnitude')
subplot(122)
stem(signal)
title('Signal in time domain')
xlabel('n')
ylabel('signal')

%part2
U = 3
D = 5
signal_up = upsample(signal,U);
sizeU = length(signal_up);
SU = fftshift(fft(signal_up));
wU = fftshift([0:sizeU-1]/sizeU*2*pi);
wU(1:sizeU/2) = wU(1:sizeU/2) - 2*pi;
fsU = U*fs
freqU = fsU*wU/2/pi;

figure(2)
subplot(121)
plot(freqU,(abs(SU)))
title('Spectrum after upsampling before filtering')
xlabel('frequency(Hz)')
ylabel('Magnitude')
subplot(122)
stem(signal)
title('Upsampled Signal in time domain')
xlabel('n')
ylabel('signal')

%part3
filtered = zeros(length(SU),2);
filtered(:,1) = freqU;
filtered(:,2) = (SU(1,:))';
filtered(1:30,2) = 0;
filtered(92:120,2) = 0;
figure(3)
subplot(121)

filtered(:,2) = U * filtered(:,2);
plot(filtered(:,1),(abs(filtered(:,2))))
title('Upsampled spectrum after filtering')
xlabel('frequency(Hz)')
ylabel('Magnitude')
subplot(122)
signal_up_filter = ifft(ifftshift((filtered(:,2))))
stem(signal_up_filter)
title('Upsampled filtered signal in time domain')
xlabel('n')
ylabel('signal')



%part4
signal_down = downsample(real(signal_up_filter),D);



sizeD = length(signal_down);
SD = fftshift(fft(signal_down));
wD = fftshift([0:sizeD-1]/sizeD*2*pi);
wD(1:sizeD/2) = wD(1:sizeD/2) - 2*pi;
fsD = fsU/D
freqD = fsD*wD/2/pi;
figure(4)
subplot(121)
plot(freqD,(abs(SD)))
title('Final sample after downsampling')
xlabel('frequency(Hz)')
ylabel('Magnitude')
subplot(122)
stem(signal_down)
title('After Downsample')
xlabel('n')
ylabel('signal')
```

Part1:

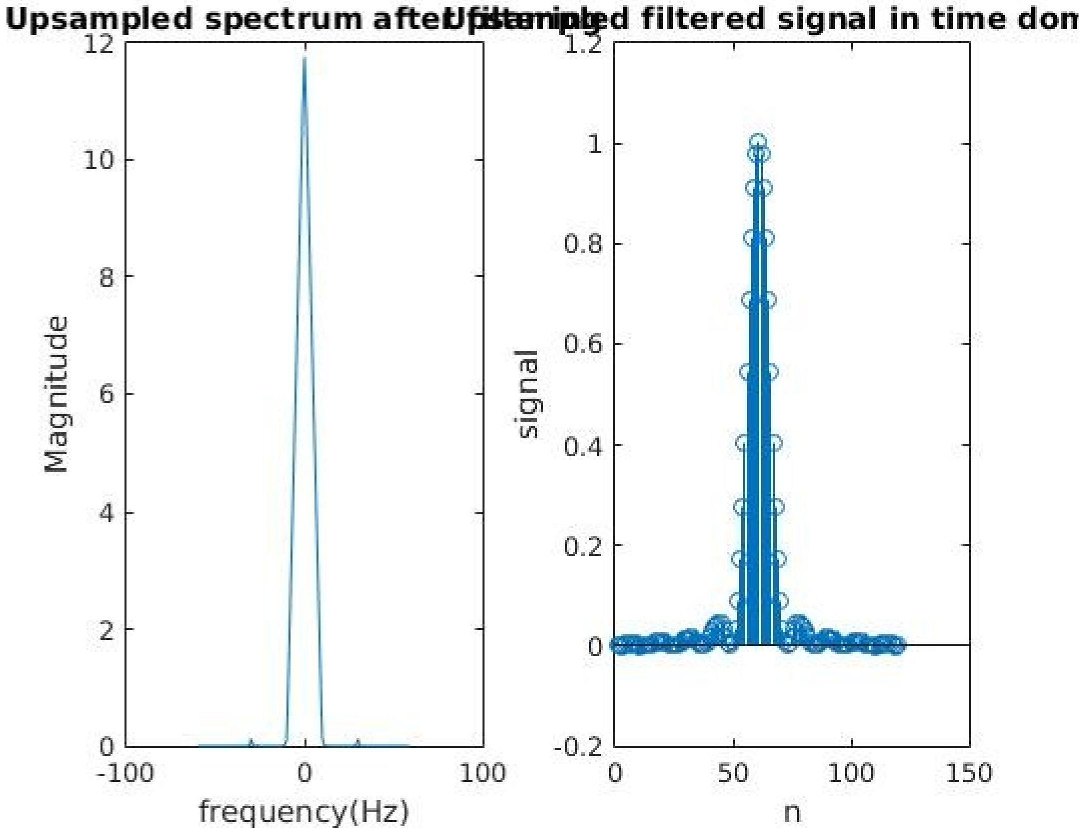
Part2:


Magnitude Response
Magnitude
4
3.5
3
2.5
2
1.5
1
0.5
0
-20
0
20
frequency(Hz)
Signal in time domain
signal
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
20
40
n



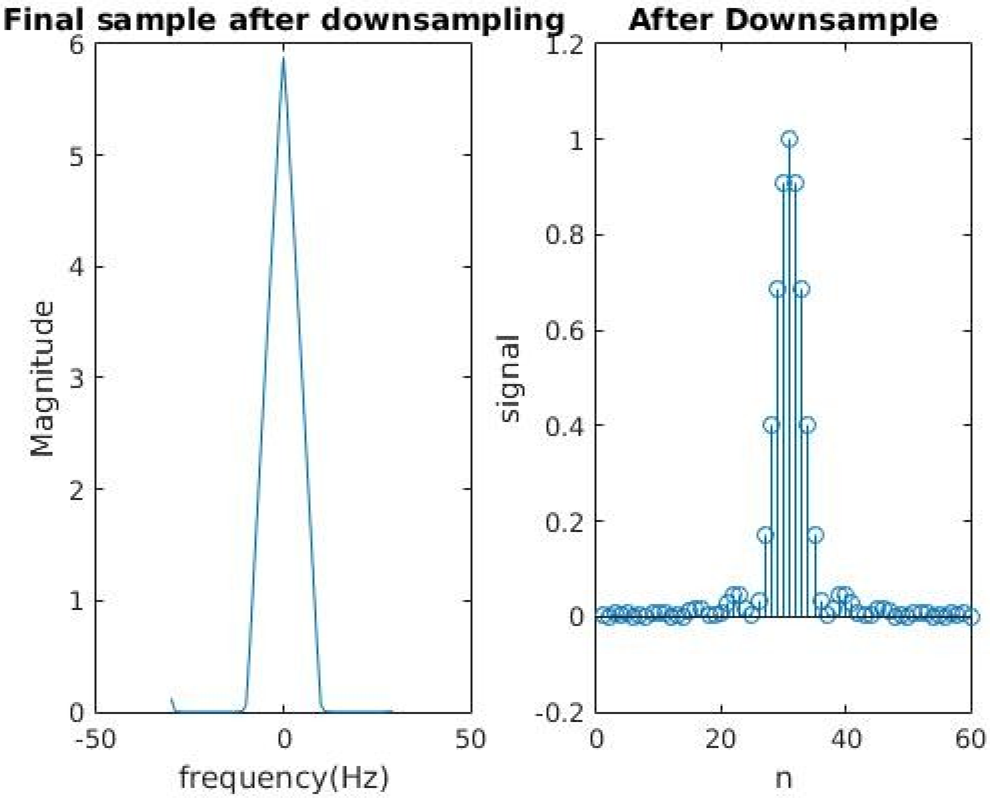
There are extra copies of the spectrum that the original spectrum does not have, because upsampling essentially shrinks the entire spectrum, causing extra copies to enter -pi to pi range, due to the scaling property of fourier transform.

Part3


Upsampled spectrum afte
Upsampled filtered signal in time dom
12
10
8
6
4
2
0
Magnitude
-100
0
100
frequency(Hz)
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
signal
0
50
100
150
n

Part4

Final

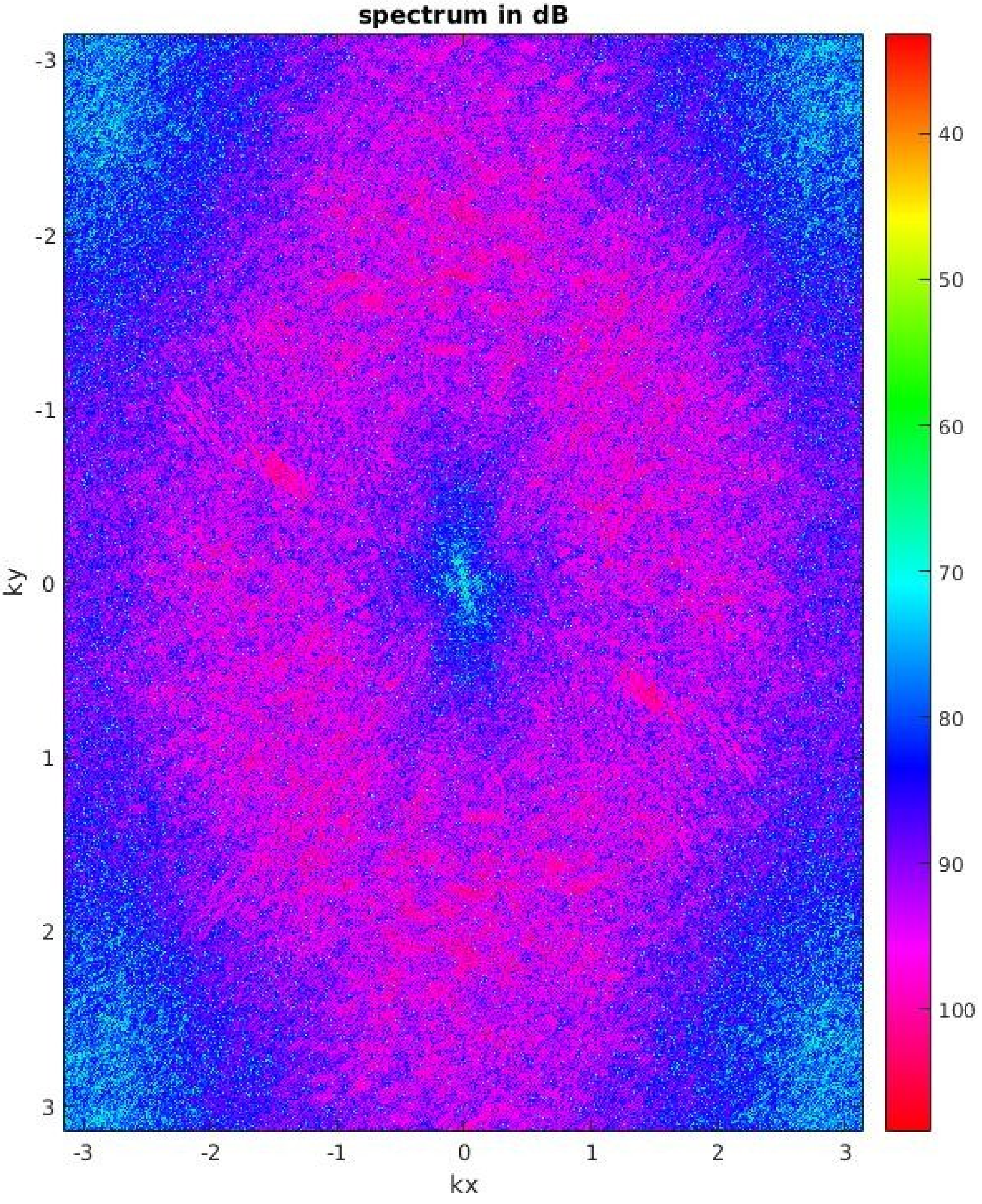


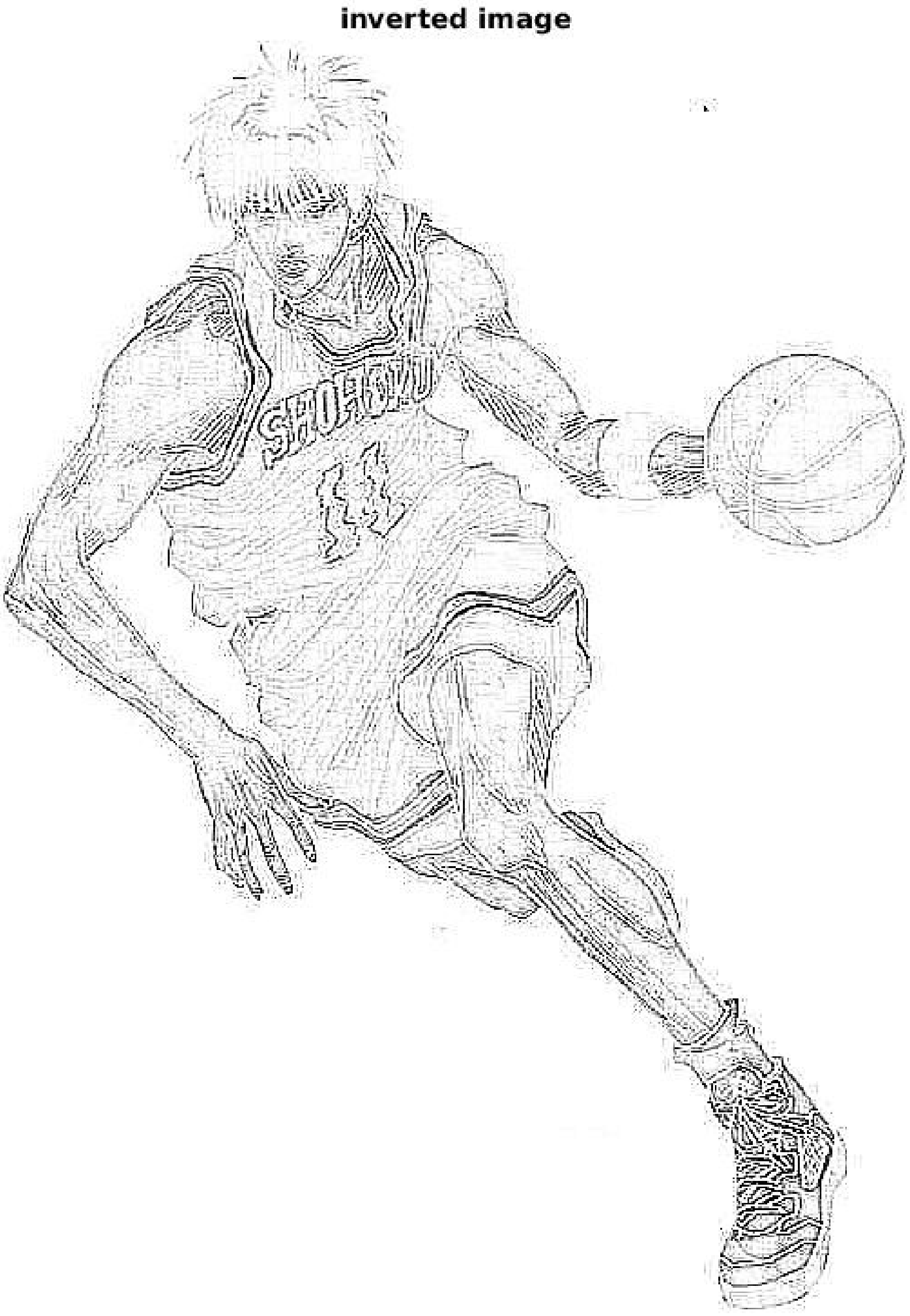
sample frequency: 60Hz
Maximum D:5

Q4:
Part1:
code:

```
image = imread('image1.jpg')
filter = [-1,-1,-1;
          -1,8,-1;
          -1,-1,-1]
filter_image = conv2(double(image),filter,'same');
imshow(uint8(filter_image));
spectrum = fft2(filter_image);
spectrum = fftshift(spectrum)

M = length(image)
N = length(image(1,:))

kX = fftshift([0:N-1]/N*2*pi)
kX(1:N/2) = kX(1:N/2) - 2*pi
kY = fftshift(([0:M-1]/M*2*pi))
kY(1:M/2) = kY(1:M/2) - 2*pi



figure(1)
```

```
colormap hsv
imagesc(kX,kY,mag2db(abs(spectrum)))
colorbar('Direction','reverse')
title('spectrum in dB')
xlabel('kx')
ylabel('ky')

figure(2)
inverted = ones(M,N)*256-filter_image;
imshow(uint8(inverted))
title('inverted image')
```


spectrum in dB
ky
-3
-2
-1
0
1
2
3
kx
-3
-2
-1
0
1
2
3
40
50
60
70
80
90
100

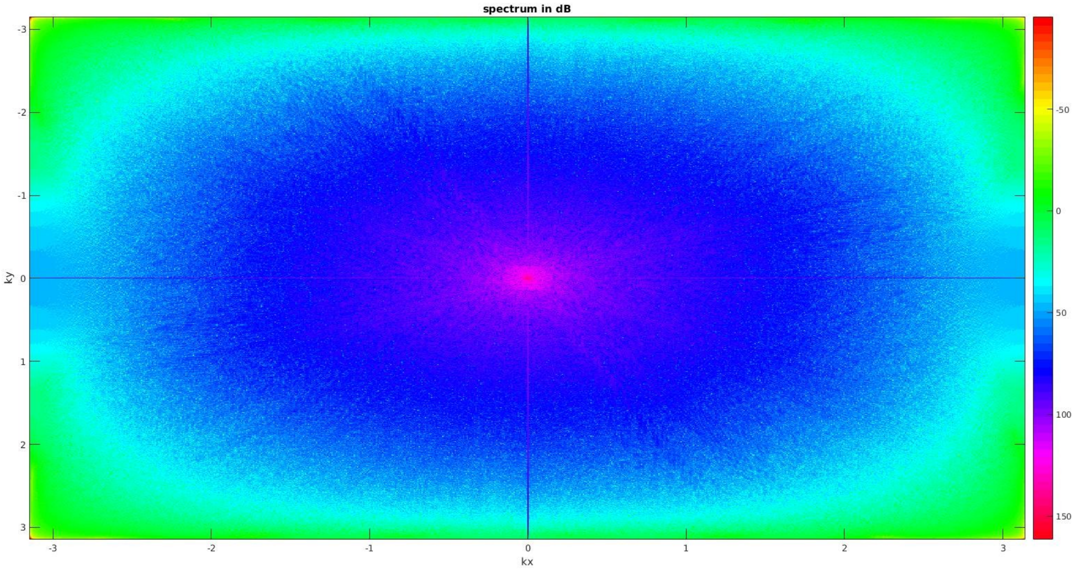


type: high pass filter

Part2:

code:

```
image = imread('image2.jpg')
filter = [1/16,1/8,1/16;
          1/8,1/4,1/8;
          1/16,1/8,1/16]
filter_image = conv2(double(image),filter,'same');
spectrum = fft2(filter_image);
spectrum = fftshift(spectrum)

M = length(image)
N = length(image(1,:))

kX = fftshift([0:N-1]/N*2*pi)
kX(1:N/2) = kX(1:N/2) - 2*pi
kY = fftshift(([0:M-1]/M*2*pi))
kY(1:M/2) = kY(1:M/2) - 2*pi

figure(2)
colormap hsv
imagesc(kX,kY,mag2db(abs(spectrum)))
colorbar('Direction','reverse')
title('spectrum in dB')
xlabel('kx')
ylabel('ky')
```



type:lowpass filter