1. Sampling of a Sinusoidal Waveform

1.1 Aim:

To sample an Analog signal waveform above its Nyquist sampling rate.

To obtain DFT of Analog waveform.

1.2 Matlab Function Used:

`linspace, abs, fft, subplot, strcat, annotation, stem`

1.3 Theory:

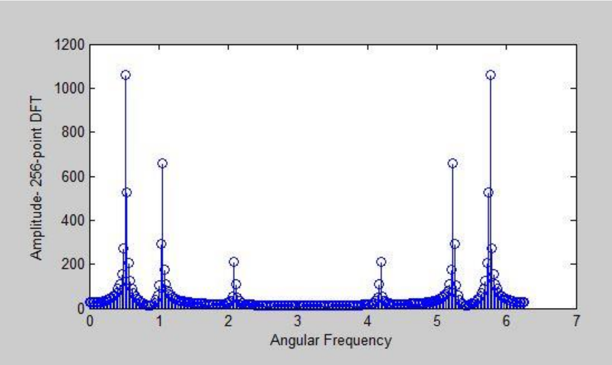
The Nyquist Theorem, also known as the sampling theorem, is a principle that engineers follow in the digitization of Analog signals. For analog-to-digital conversion (ADC) to result in a faithful reproduction of the signal, slices, called *samples*, of the analog waveform must be taken frequently. The number of samples per second is called the sampling rate or sampling frequency. Suppose the highest frequency component, in hertz, for a given analog signal is f_{max}. According to the Nyquist Theorem, the sampling rate must be at least $2f_{max}$, or twice the highest analog frequency component.

1.4 Source Code:

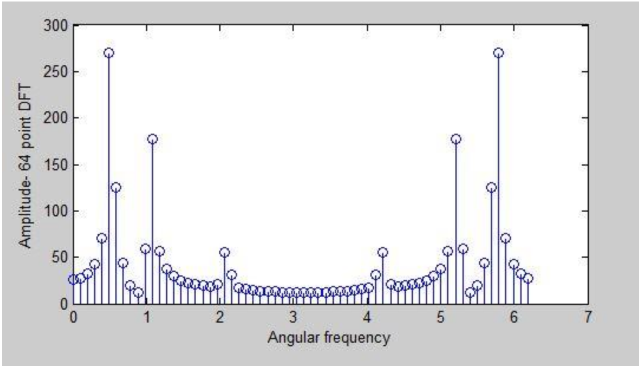
```
f=12000;
% Sampling with 12KHz
x= linspace(0,1,f);
% given function and samples
y= 10*cos(2*pi*1000*x)+6*cos(2*2*pi*1000*x)+2*cos(4*2*pi*1000*x);
% Computed 64,128,256 point DFT

a1=abs(fft(y(1:64)));
a2=abs(fft(y(1:128)));
a3=abs(fft(y(1:256)));
figure;
subplot(2,2,1);
p=strcat('sampling freq ',num2str(f));
annotation('textbox',[0.85 0.8 0.08 0.08],...
                        'String',strcat(num2str(f),' Hz'));
stem(0:2*pi/64:2*pi*63/64,a1);
xlabel('Angular frequency');
ylabel('Amplitude- 64 point DFT');
subplot(2,2,2);
stem(0:2*pi/128:2*pi*127/128,a2);
ylabel('Amplitude- 128 point DFT');
xlabel('Angular frequency');

subplot(2,2,3);
stem(0:2*pi/256:2*pi*255/256,a3);
ylabel('Amplitude- 256-point DFT');
xlabel('Angular Frequency');
```

1.5 Result:

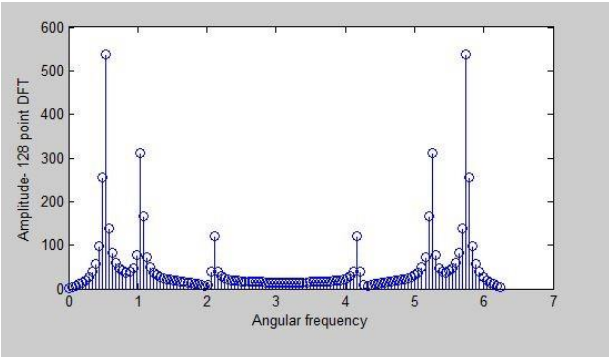
256 point DFT



64 Point DFT



128 Point DFT



1.6 Discussion:

- Since we took a sinusoidal waveform, the signal was bandlimited to 4 KHz
- Sampling was done at 12 KHz.
- Since we took a finite length waveform and obtained its DFT, the spectrum was different from the ideal.
- Other than sharp peaks at sinusoidal frequencies there was small noise throughout the spectrum.
- By increasing N in N-point DFT we were able to obtain higher resolution DFT spectrum.

2. Sampling at below Nyquist rate and effect of aliasing

2.1 AIM:

To Sample the signal at rate below the Nyquist sampling rate.
To observe the effect of aliasing.

2.2 MATLAB FUNCTION USED:

```
For,linspace,abs,fft, subplot,strcat, annotation, stem
```

2.3 THEORY:

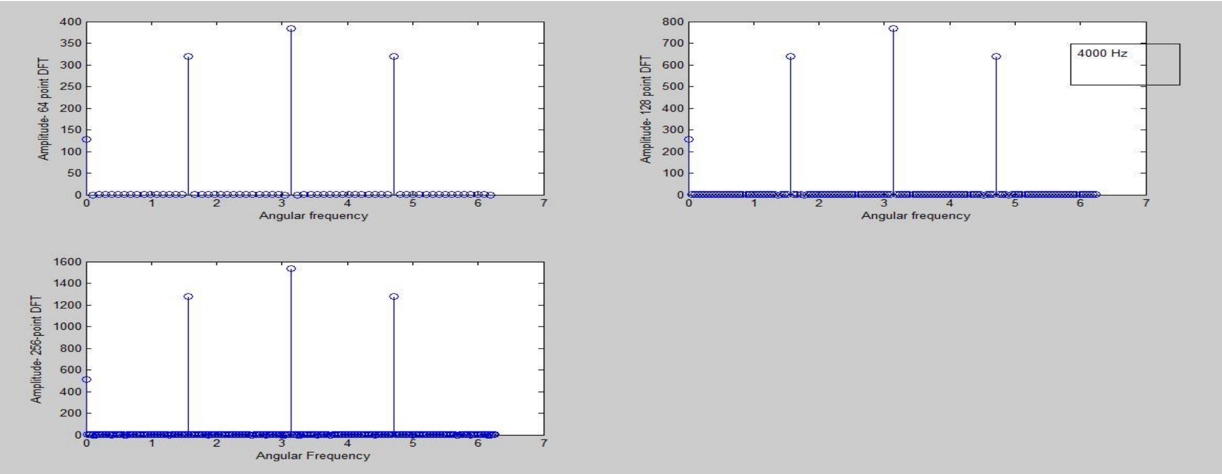
If the sampling rate is less than $2f_{max}$, some of the highest frequency components in the analog input signal will not be correctly represented in the digitized output. When such a digital signal is converted back to analog form by a digital-to-analog converter, false frequency components appear that were not in the original analog signal. This undesirable condition is a form of distortion called aliasing.

2.4 SOURCE CODE:

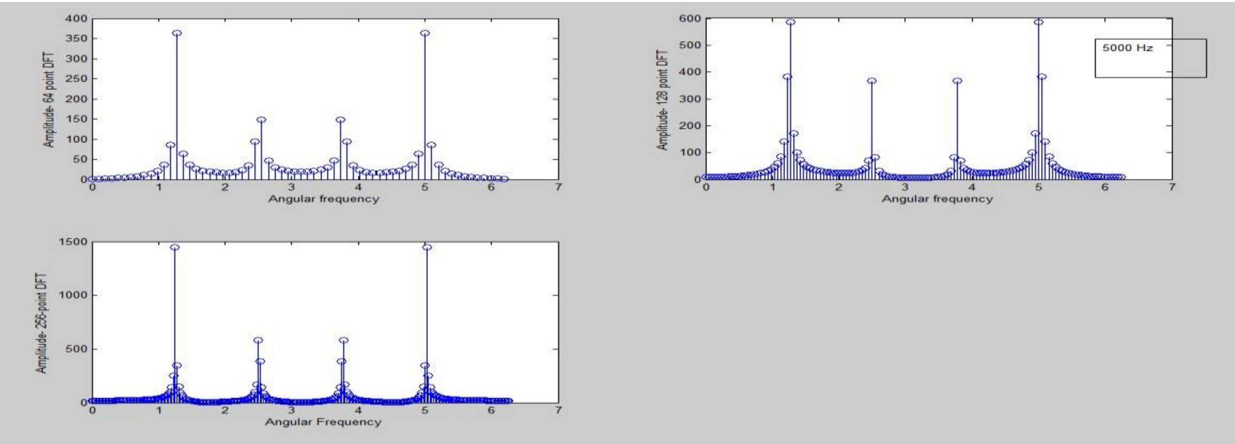
```
arr=[8001,5001,4001];
for i=1:3
% Sampling with 12KHz,8KHz, 5KHz, 4 KHz frequency
x= linspace(0,1,arr(i));
% given function and samples
y= 10*cos(2*pi*1000*x)+6*cos(2*2*pi*1000*x)+2*cos(4*2*pi*1000*x);
% Computed 64,128,256 point DFT

a1=abs(fft(y(1:64)));
a2=abs(fft(y(1:128)));
a3=abs(fft(y(1:256)));
figure;
subplot(2,2,1);
p=strcat('sampling freq ',num2str(arr(i)-1));
annotation('textbox',[0.85 0.8 0.08 0.08],...
                        'String',strcat(num2str(arr(i)-1),' Hz'));
stem(0:2*pi/64:2*pi*63/64,a1);
xlabel('Angular frequency');
ylabel('Amplitude- 64 point DFT');
subplot(2,2,2);
stem(0:2*pi/128:2*pi*127/128,a2);
ylabel('Amplitude- 128 point DFT');
xlabel('Angular frequency');

subplot(2,2,3);
stem(0:2*pi/256:2*pi*255/256,a3);
ylabel('Amplitude- 256-point DFT');
xlabel('Angular Frequency');

end
```

2.5 RESULT:

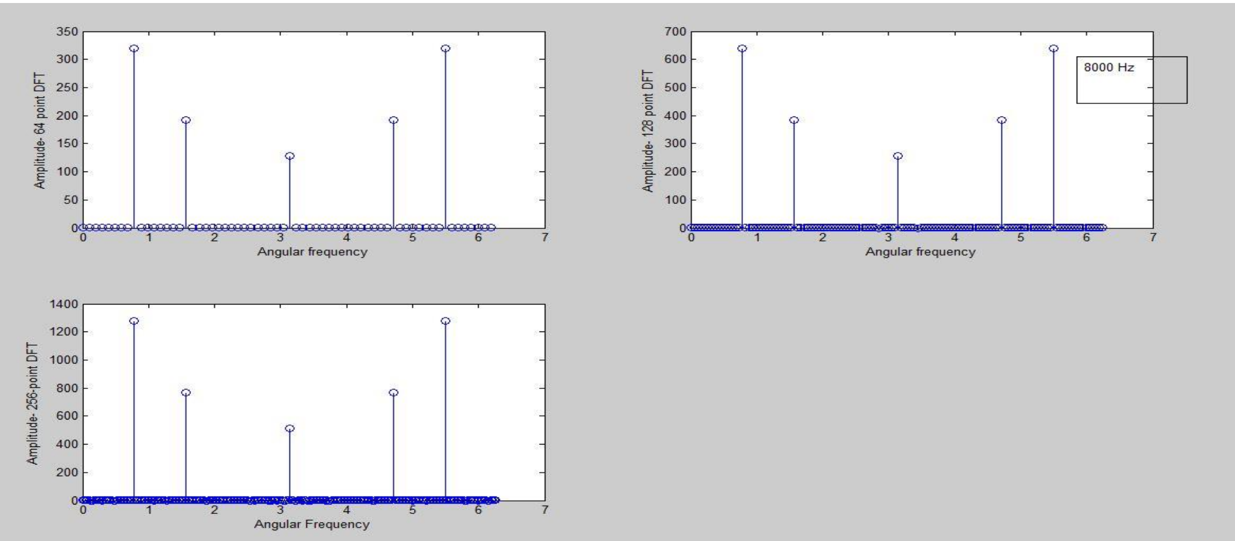
a) Frequency = 4KHZ



b) Frequency = 5KHZ



c) Frequency= 8 KHz



2.6 Discussion:

- At sampling frequency of 8 KHz (i.e. Nyquist Rate), the signal was samplesd without any loss of information.
- The Effect of aliasing occurs because the periodic copies of baseband spectrum around nF_s gets overlapped.
- The effect was clearly visible for 4 KHz and 5 KHz.

3. Spectrum of a Square Wave

3.1 Aim:

To observe the Spectrum of a square wave.

3.2 MATLAB function used:

`linspace, square, fftshift, abs, fft, plot, xlabel, ylabel`

3.3 Theory:

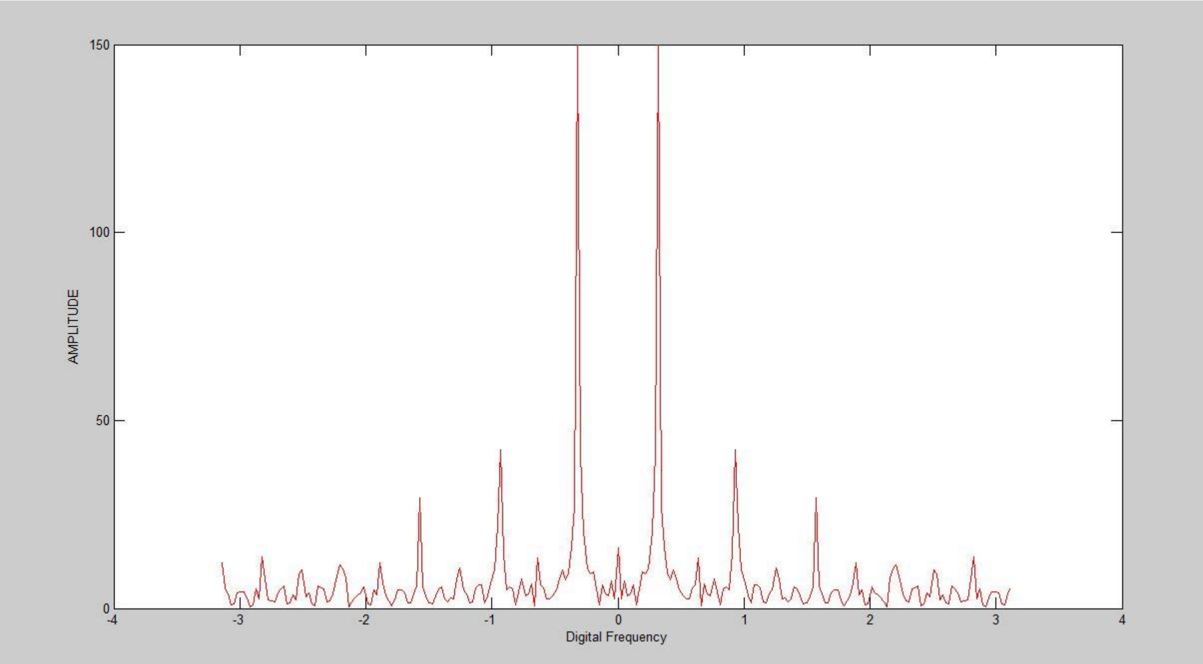
A square wave has theoretically has infinite bandwidth. For practical purposes, the spectrum beyond 10^{th} harmonic can be neglected.

3.4 Source Code:

```
% Sampling of square wave of T=1 ms with sampling frequency 20kHz
ys=linspace(0,1,20001);
xsq= square(2*pi*1000*ys);
xsqs= xsq(1,1:256);
z= fftshift(abs(fft(xsqs)));
figure;
plot(-128*2*pi/256:pi/128:127*pi/128,z,'r');
xlabel('Digital Frequency');
ylabel('AMPLITUDE');
```

3.5 Result:

Spectrum of a Square Wave.



3.6 Discussion:

- We have sampled the square wave signal with 20 KHz sampling rate. Since, its frequency domain representation contains infinite odd harmonics of 1 KHz as we have taken the odd symmetric square wave.

- We observe that higher frequency components are smaller in magnitude. But this smaller components give the mean square error when we construct the signal from its discrete Fourier transform.

4. Interpolation or Up sampling

4.1 AIM:

To interpolate an Analog Signal.

4.2 THEORY:

If an analog signal is sampled at a frequency higher than the Nyquist rate it is possible to interpolate the intermediate L-1 samples or in other words to obtain the samples at Fs2=LFs1 frequency. This can be simply done by passing the sampled signals through an ideal low pass filter of cut-off frequency F_{max} and sampling it again at a higher rate.

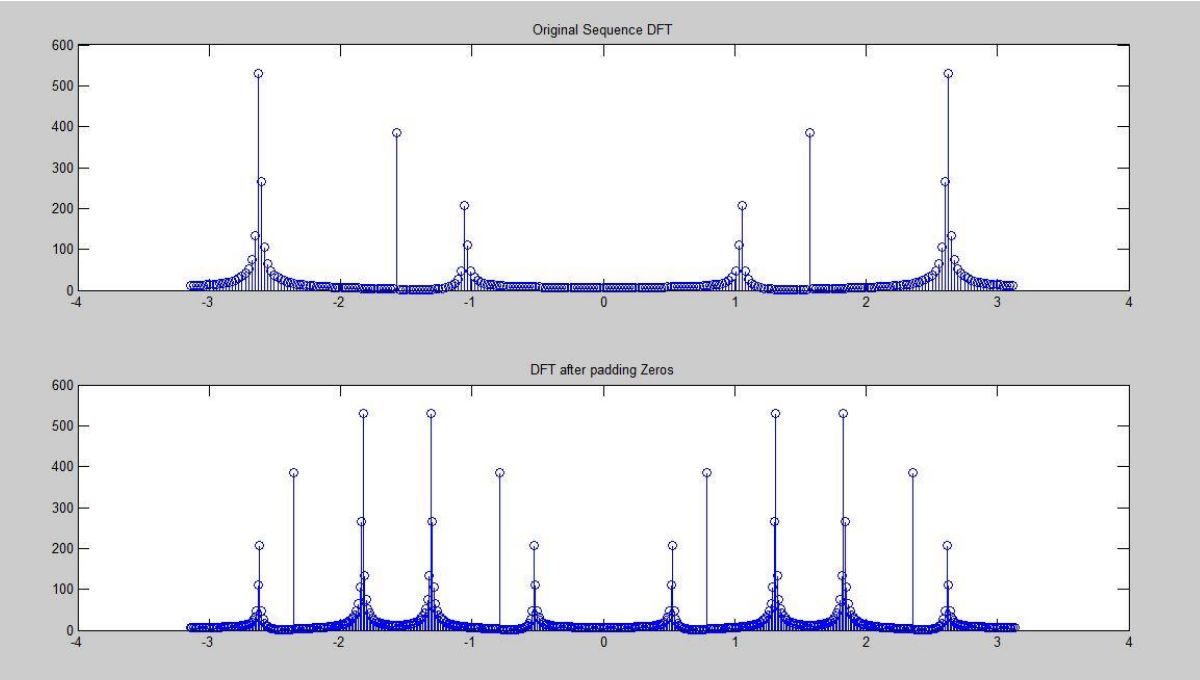
4.3 MATLAB FUNCTION USED:

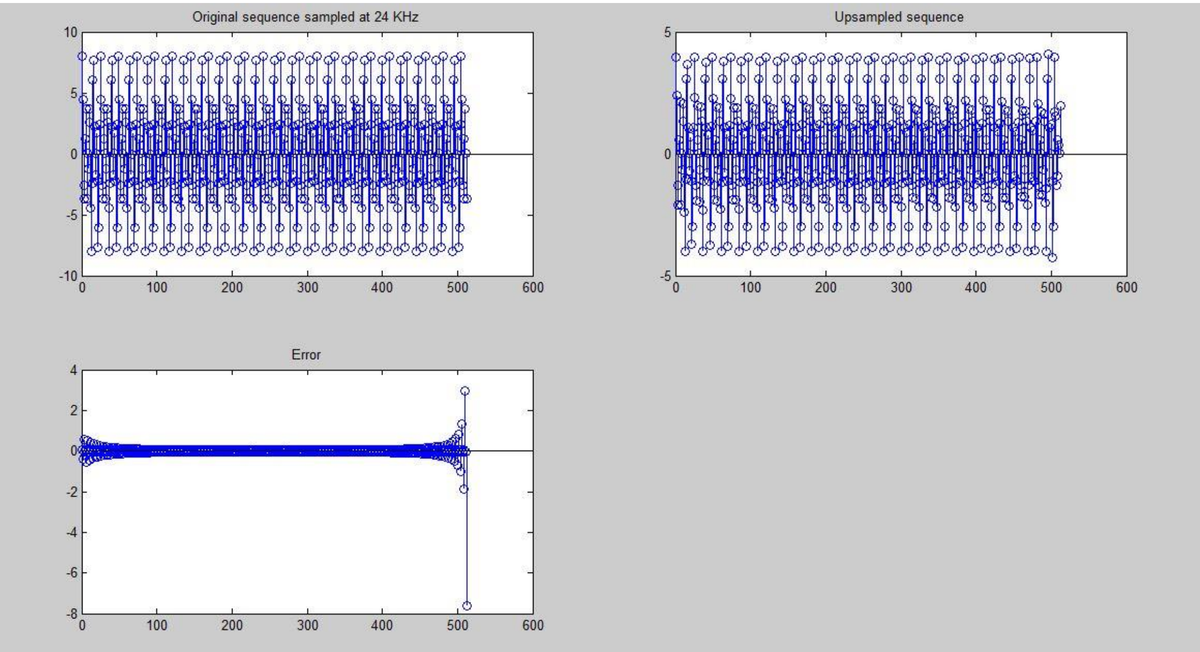
```
Linspace, fft, subplot, stem, title, zeros, for, ifft, figure
```

4.4 SOURCE CODE:

```
% first Sampling frequency 12 KHz and then upsampling factor 2
% find the error

x=linspace(0,1,12001);
y= 2*sin(2*pi*2000*x)+3*cos(2*pi*3000*x)+5*cos(2*pi*5000*x);
y1=fft(y,256);
subplot(2,1,1);
stem(-pi:2*pi/256:127*2*pi/256,fftshift(abs(y1)));
title('Original Sequence DFT');
y2=y(1:257);
y3=zeros(1,513);
for i=1:257
    y3(1,2*i-1)=y2(i);
end
% notice that size of y3 is 513, not 512

subplot(2,1,2);
stem(-pi:2*pi/512:255*2*pi/512,fftshift(abs(fft(y3,512))));
title('DFT after padding Zeros');
y4=fft(y3,512);
y4(1,129:385)=zeros(1,257);
yi4= ifft(y4);
x=0:1/24000:1/24;
z1= 2*sin(2*pi*2000*x)+ 3*cos(2*pi*3000*x)+ 5*cos(2*pi*5000*x);
z1s= z1(1:512);
figure;
subplot(2,2,1);
stem(1:512,z1s); title('Original sequence sampled at 24 KHz');
subplot(2,2,2);
stem(1:512,yi4); title('Upsampled sequence');
subplot(2,2,3);
 stem(1:512,z1s-2*yi4);title('Error');
```

4.5 Result:





4.6 Discussion:

- During up-sampling, we insert zero in between samples which causes duplication of lower frequencies in higher band.

- We need filter to filter out higher frequency range to have our original signal up-sampled. Here, in the case we have used brick wall filter for digital filtering.

Additional Experiment Question:

1. To plot DFT of Square Wave and Reconstruct Square wave using inverse Fourier transform. Find Error between original square wave and recovered square wave.

2. To reconstruct the square wave using inverse Fourier transform but one of the frequency component is missing.

Source Code

```
xlabel('Time ------>');ylabel('Magnitude ------>');t=0:1/20000:10;
s=square(2*pi*1000*t);
S=fft(s,256);
sz=S;
sz(1,36:39)=[0 0 0 0];
sz(1,217:220)=[0 0 0 0];
szr= ifft(sz);
s_abs=abs(S);
s1=ifft(S);

error=s(1:256)-s1;

subplot(2,2,1);
plot(t(1:256),s(1:256));
xlabel('Time ------>');ylabel('Amplitude ------>');
title('Original Signal: Square Wave Sampled at 20KhZ');
axis([0 .01 -2 2]);

subplot(2,2,2);

plot((-128:127)*10/128,fftshift(s_abs));
xlabel('Frequency (in KHz)------>');ylabel('Magnitude ------>');
title('DFT of Square Wave');

subplot(2,2,3);
plot(t(1:256),s1);axis([0 .01 -2 2]);
xlabel('Time ------>');ylabel('Amplitude ------>');
title('Recovered Signal');


subplot(2,2,4);
plot(1:256,error);
xlabel('Time ------>');ylabel('Magnitude ------>');
title('Error between Original Signal and Recovered Signal');

figure;
plot(t(1:256),szr);
axis([0 0.01 -2 2]);
```

```
title('Square wave after removing 3KHz component');
xlabel('time------>');
ylabel('Amplitude------>');
ms=rms(error);
```

Result: