DIGITAL SIGNAL PROCESSING LAB (BECE301P)

DUE DATE: 15-2-25 LAB SLOT: L35+L36

NAME: ISHAAN KEDAR REG. NO: 22BEC0153

OBJECTIVE:

To use MATLAB for spectral analysis of signals

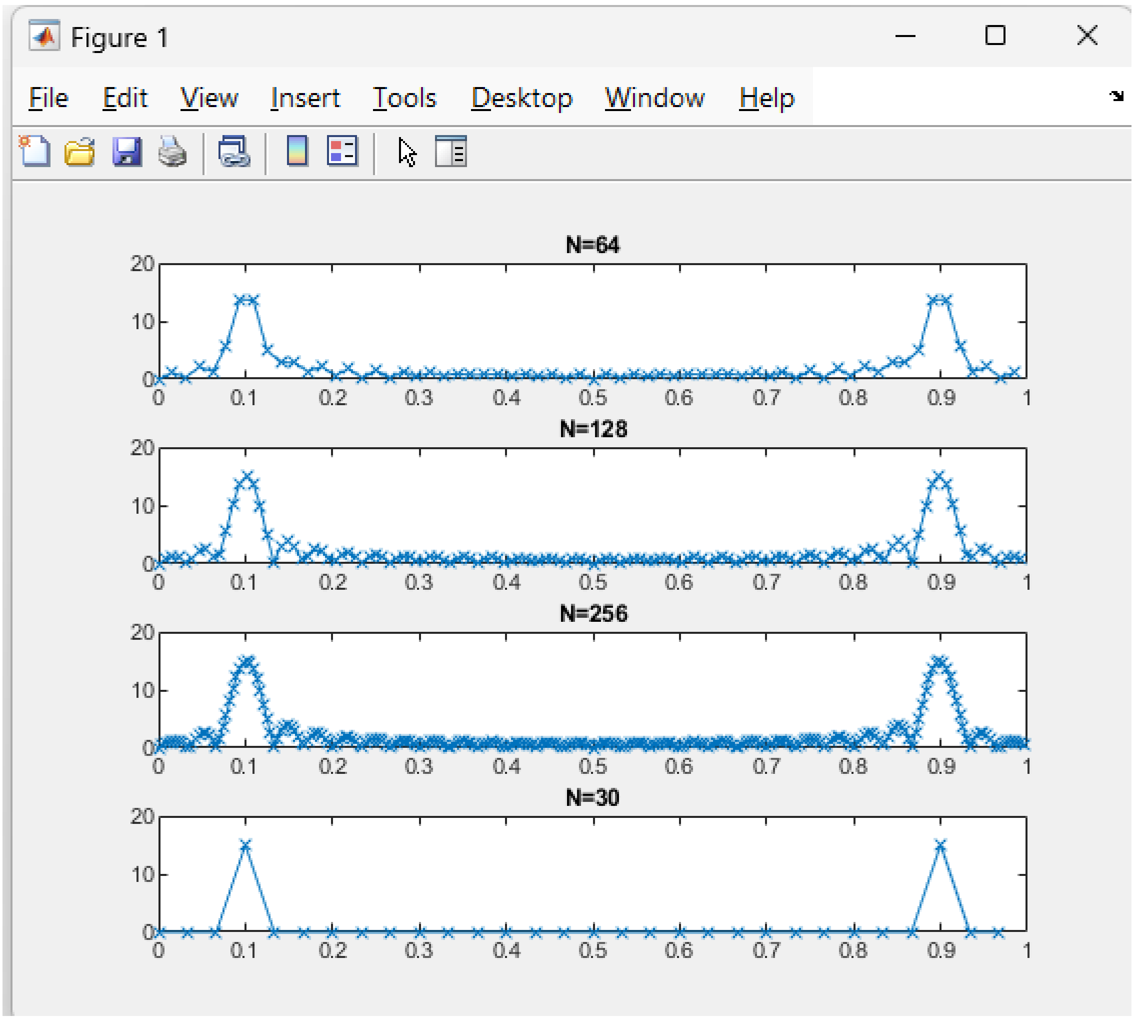
Example 1: The typical syntax for computing the FFT of a signal is FFT(x,N) where x is the signal, x[n], you wish to transform, and N is the number of points in the FFT. N must be at least as large as the number of samples in x[n].

To demonstrate the effect of changing the value of N, sythesize a cosine with 30 samples at 10 samples per period.

CODE:

```
% fft usage example

n = [0:29];
x = cos(2*pi*n/10);

N1 = 64;
N2 = 128;
N3 = 256;
N4 = 30;

x1 = abs(fft(x,N1));
x2 = abs(fft(x,N2));
x3 = abs(fft(x,N3));
x4 = abs(fft(x,N4));

%noramlize

f1 = [0 : N1-1]/N1;
f2 = [0 : N2-1]/N2;
f3 = [0 : N3-1]/N3;
f4 = [0 : N4-1]/N4;
figure(1);

subplot(4,1,1);
plot(f1,x1,'-x'),title('N=64'),axis([0 1 0 20]);
subplot(4,1,2);
plot(f2,x2,'-x'),title('N=128'),axis([0 1 0 20]);
subplot(4,1,3);
plot(f3,x3,'-x'),title('N=256'),axis([0 1 0 20]);
subplot(4,1,4);
plot(f4,x4,'-x'),title('N=30'),axis([0 1 0 20]);
```

OUTPUT:



- What happens if N is the same as the number of samples in x[n]?
- To find out, set N1 = 30. What does the resulting plot look like?
- Why does it look like this?

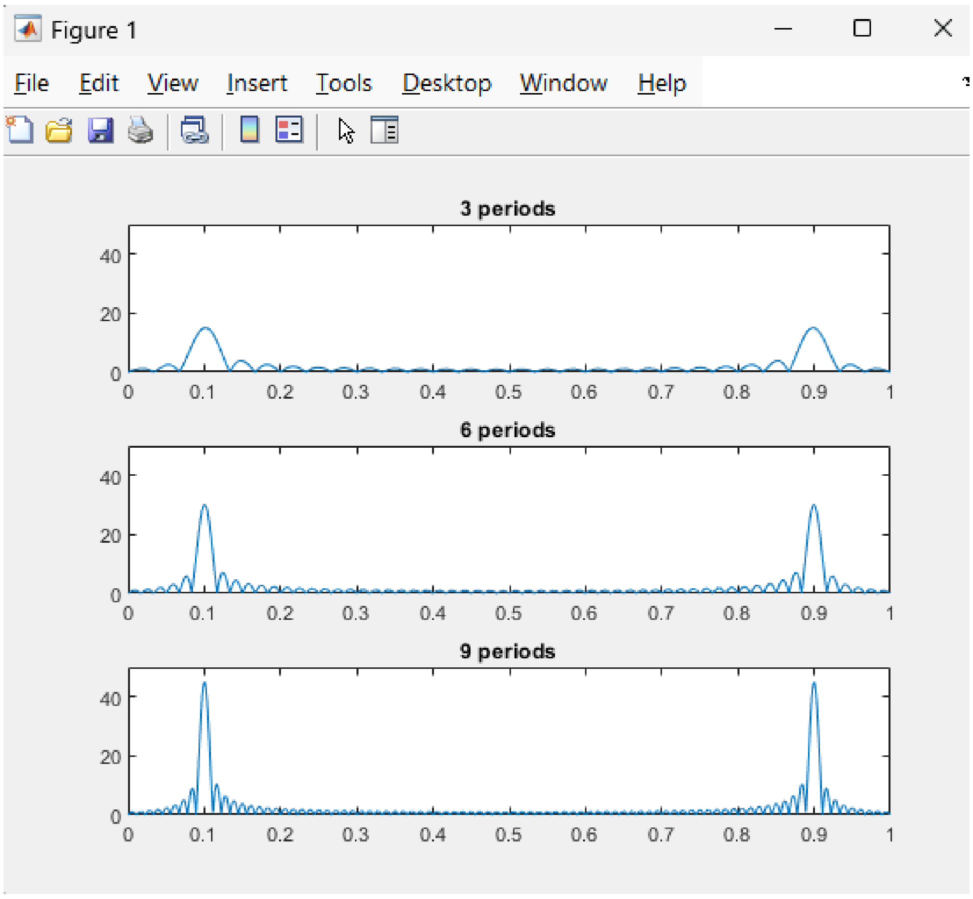
When N is equal is same as the number of samples the FFT output closely resembles that of true spectrum. If N becomes less than number of samples aliasing occurs.

Example 2: In the last example the length of x[n] was limited to 3 periods in length. Now, let's choose a large value for N (for a transform with many points), and vary the number of repetitions of the fundamental period.

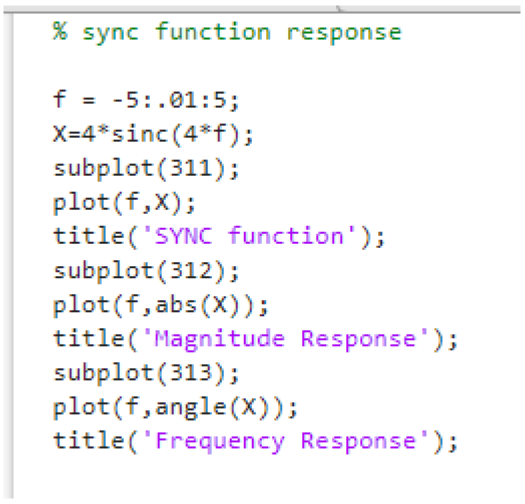
CODE:

```
n = [0:29];
x1 = cos(2*pi*n/10);
x2 = [x1 x1];
x3 = [x1 x1 x1];

N = 2048;
X1 = abs(fft(x1,N));
X2 = abs(fft(x2,N));
X3 = abs(fft(x3,N));
F = [0:N-1]/N;

figure(1);

subplot(3,1,1);
plot(F,X1),title('3 periods'),axis([0 1 0 50])
subplot(3,1,2)
plot(F,X2),title('6 periods'),axis([0 1 0 50])
subplot(3,1,3)
plot(F,X3),title('9 periods'),axis([0 1 0 50])
```

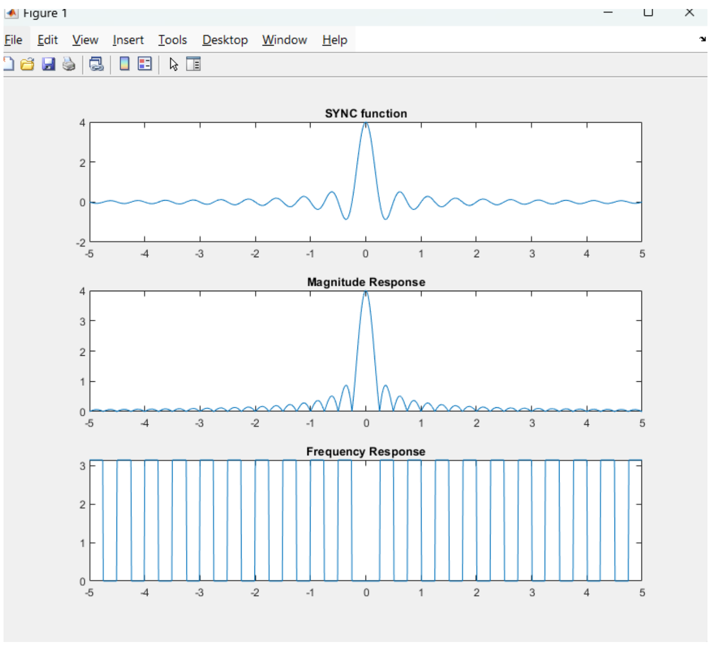
OUTPUT:



1. SYNC FUNCTION RESPONSE



```
% sync function response

f = -5:.01:5;
X=4*sinc(4*f);
subplot(311);
plot(f,X);
title('SYNC function');
subplot(312);
plot(f,abs(X));
title('Magnitude Response');
subplot(313);
plot(f,angle(X));
title('Frequency Response');
```



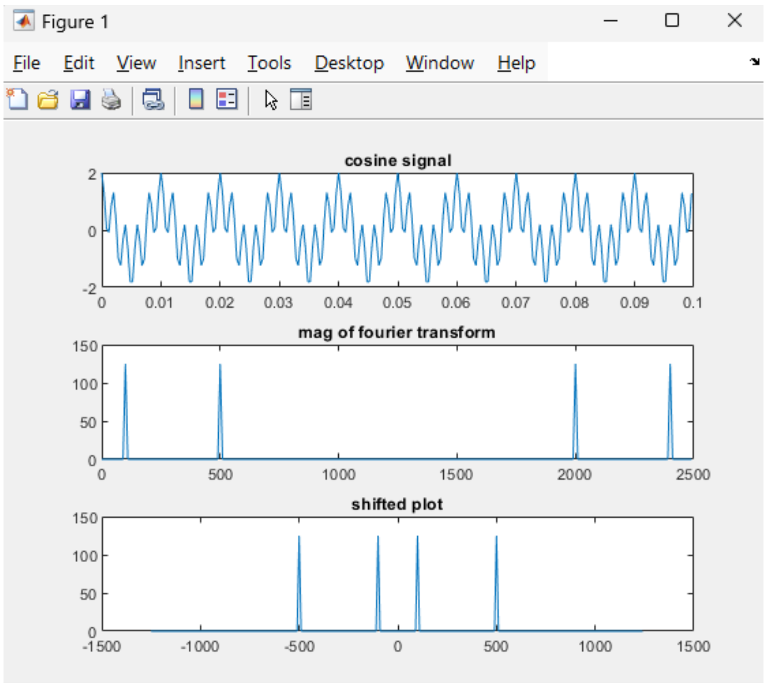
Inference: The magnitude response confirms low-pass filter behavior, with energy concentrated at lower frequencies.
The frequency response phase plot shows phase shifts that correspond to the oscillatory nature of the sinc function.

2. Fast Fourier transform in spectral analysis

CODE:

```
% fast fourier transform for spectral analysis
N = 250;
ts = .0004;
deltaf = 1/(N*ts);
t = [0:N-1]*ts;
x = cos(2*pi*100*t)+cos(2*pi*500*t);
subplot(311);
plot(t,x);
title('cosine signal');
xf = fft(x);
f=[0:N-1]*deltaf;
subplot(312);
plot(f,abs(xf));
title('mag of fourier transform')
xf_shift = fftshift(xf);
subplot(313);
plot([-N/2:N/2-1]*deltaf, abs(xf_shift));
title('shifted plot');
```

OUTPUT:



INFERENCE:

The magnitude spectrum reveals two prominent peaks corresponding to the signals frequency components

Peaks appear at 100 and 500hz

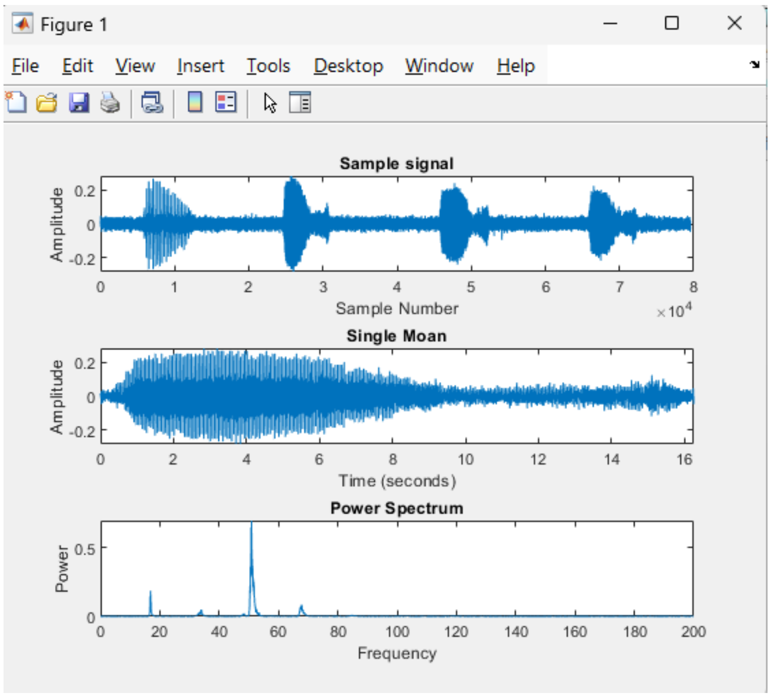
Shifted plot centers the plot showing peaks at +- 100 and +- 500 hz.

3. AUDIO SIGNAL SPECTRAL ANALYSIS

CODE:

```
%audio spectral analysis

whaleFile = "C:\Users\ishaa\Downloads\sem6\dsplabcodes\da2\bluewhale.au.au";
[x,fs] = audioread(whaleFile);
subplot(311);
plot(x);
xlabel('Sample Number');
ylabel('Amplitude');
title('Sample signal');
moan = x(2.45e4:3.10e4);
t = 10*(0:1/fs:(length(moan)-1)/fs);
subplot(312);
plot(t,moan)
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Single Moan');
xlim([0 t(end)]);
m = length(moan); % original sample length
n = pow2(nextpow2(m)); % transform length
y = fft(moan,n); % DFT of signal
f = (0:n-1)*(fs/n)/10;
power = abs(y).^2/n;
subplot(313);
plot(f(1:floor(n/2)),power(1:floor(n/2)));
xlabel('Frequency');
ylabel('Power');
title('Power Spectrum');
```

OUTPUT:



INFERENCE:

-Time-Domain Representation (Waveform)

- Provides insights into the duration and amplitude variations of the sound.
- Useful for visualizing how the sound changes over time.

-Frequency-Domain Representation (Power Spectrum)

- Reveals the dominant frequencies present in the audio signal.
- Shows harmonic components, which are critical for sound analysis.

TASK 1:

Display the signal, Amplitude spectrum and Phase Spectrum of the following signals;

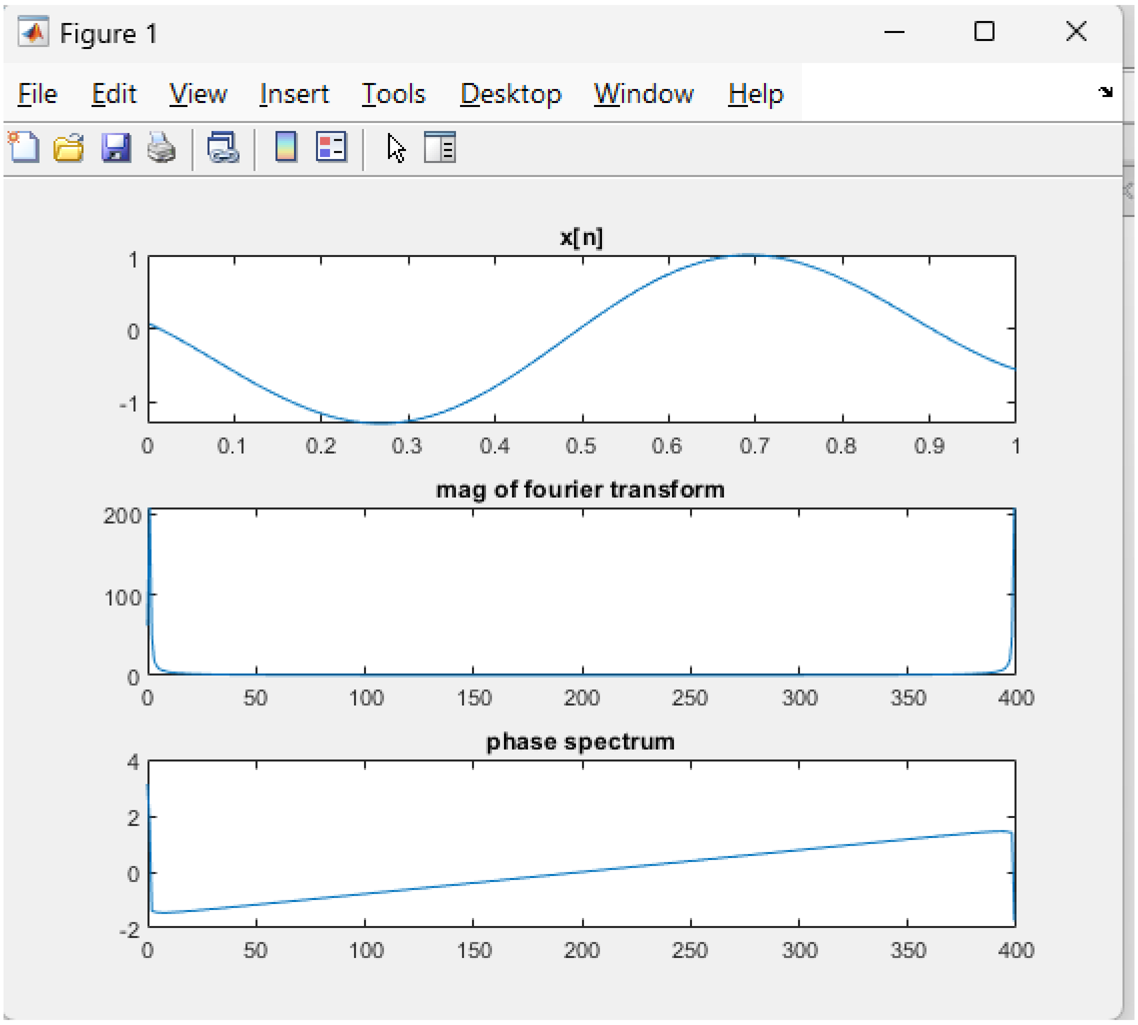
1. $$x[n] = \sin\left[\frac{50\,\pi n}{f_s}\right] + \cos\left[\frac{50\,\pi n}{f_s} + \frac{4\pi}{3}\right] + \sin\left[\frac{1000\,\pi n}{f_s} + \frac{4\pi}{5}\right]$$

   Where f_s is the sampling rate, $n = 1 : f_s$ and $f_s = 400$

CODE:

```
% fast fourier transform for spectral analysis
clc;
clear;
close;


fs=400;


N = fs;
ts = 1/fs;
deltaf = 1/(N*ts);
n = [1:N]*ts;
x =
sin((50*pi*n)/fs)+cos(((50*pi*n)/fs)+((4*pi)/3))+sin(((1000*pi*n)/fs)+(4*pi)/5);
subplot(311);
plot(n,x);
title('x[n]');
xf = fft(x);
f=[0:N-1]*deltaf;
subplot(312);
plot(f,abs(xf));
title('mag of fourier transform')

subplot(313);
plot(f,angle(xf));
title('phase spectrum');
```

OUTPUT:

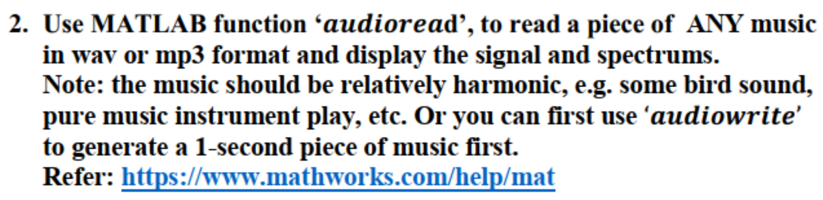


INFERENCE:

-Magnitude of Fourier Transform (mag of fourier transform - Second Plot)

- The FFT plot shows two prominent peaks at 50 Hz and 100 Hz, corresponding to the frequencies of the components in the signal.
- The presence of sharp peaks indicates the dominant frequency components.
- The frequency axis spans from 0 Hz to 400 Hz (sampling frequency).

TASK 2:

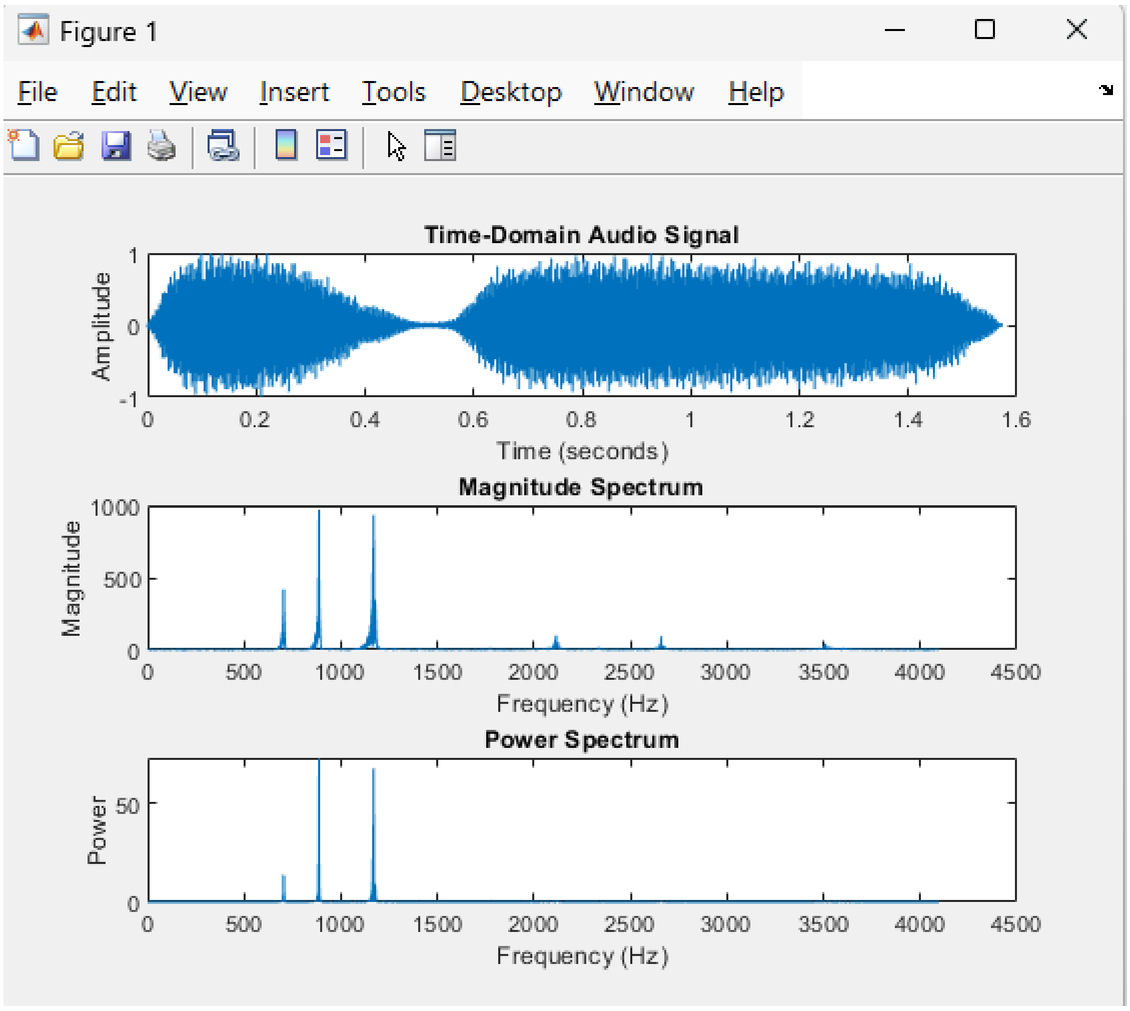


**2. Use MATLAB function ‘*audioread*’, to read a piece of ANY music in wav or mp3 format and display the signal and spectrums.
Note: the music should be relatively harmonic, e.g. some bird sound, pure music instrument play, etc. Or you can first use ‘*audiowrite*’ to generate a 1-second piece of music first.
Refer: https://www.mathworks.com/help/mat**

CODE:

```
clc;
clear;
close all;


chirpdat = load('train.mat');
audio_signal = chirpdat.y
fs = chirpdat.Fs;
% Extract a single channel if stereo
if size(audio_signal, 2) > 1
    audio_signal = mean(audio_signal, 2); % Convert to mono if stereo
end

% Plot time-domain signal
t = (0:length(audio_signal)-1) / fs;
subplot(3,1,1);
plot(t, audio_signal);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Time-Domain Audio Signal');

% Perform FFT
N = length(audio_signal);
nfft = 2^nextpow2(N); % Zero-padding to nearest power of 2
fft_audio = fft(audio_signal, nfft);
frequencies = (0:nfft-1) * (fs / nfft); % Frequency axis

% Plot Magnitude Spectrum
subplot(3,1,2);
plot(frequencies(1:nfft/2), abs(fft_audio(1:nfft/2)));
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('Magnitude Spectrum');

% Plot Power Spectrum (Magnitude Squared)
power_spectrum = abs(fft_audio).^2 / N;
subplot(3,1,3);
plot(frequencies(1:nfft/2), power_spectrum(1:nfft/2));
xlabel('Frequency (Hz)');
ylabel('Power');
title('Power Spectrum');

sound(audio_signal, fs);
```

OUTPUT:



INFERENCE:
Magnitude spectrum shows peaks at prominent frequencies indicating harmonics in the sound signal

The power spectrum indicated powerful periodic components.