"In The Name Of God"

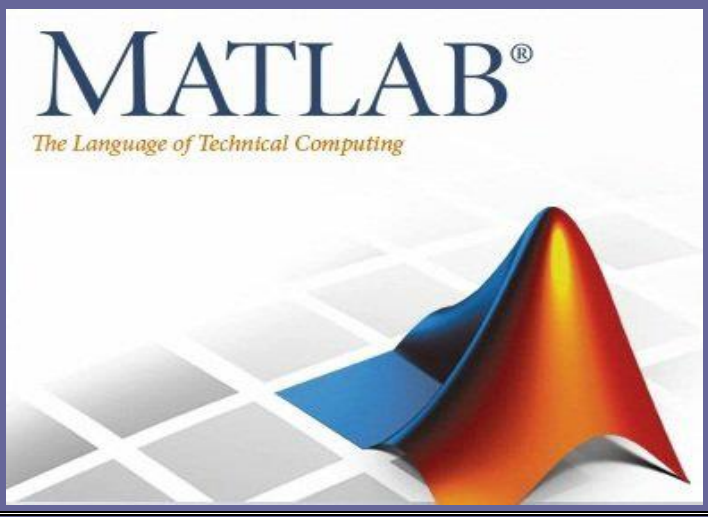
MATLAB project

Design by: MohammadReza Arani 810196405

Related Authority: Mr. Ali Ranjbar & Miss Hoda Barkhordarpoor

Chief TA:Dr.Jamal Kazazi

Sources: My own hardwork , Internet sites such as codesara.ir & good friends.



Sections:

Section1:Design and modification of Spectrum

Part1:Fast Fourier Transform(fft):

`Y = fft(X)` computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.

`Y = fft(X,n)` returns the n-point DFT. If no value is specified, Y is the same size as X.

If the input a matris ,output turns to be fourier trasnfor of each column ;otherwise, if it is a vector as an input the oiutput will be its transform.

$\int_{-\infty}^{\infty} x(t) * e^{-j2\pi ft} dt$ is the integral form and for the discrete form we use

The Discrete Fourier Transform can be expressed as

$$\mathbf{F}(n) = \sum_{k=0}^{N-1} f(k)\, e^{-j2\pi nk/N} \qquad (n = 0..1 : N-1)$$

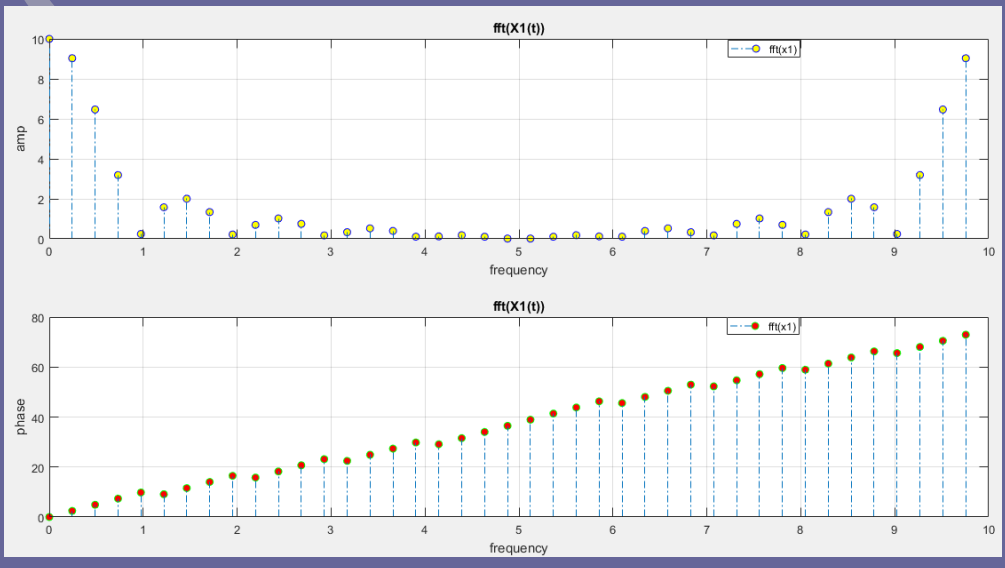
The relevant inverse Fourier Transform can be expressed as

$$f(k) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbf{F}(n)\, e^{j2\pi nk/N} \qquad (k = 0, 1, 2..., N-1)$$

Part2:

```
In[5]:= Integrate[Exp[-I * 2 * Pi * f * t], {t, 0, 1}]

Out[5]= -(-1 + e^(-2 f I π)) / (2 f I π)
```

Part3:

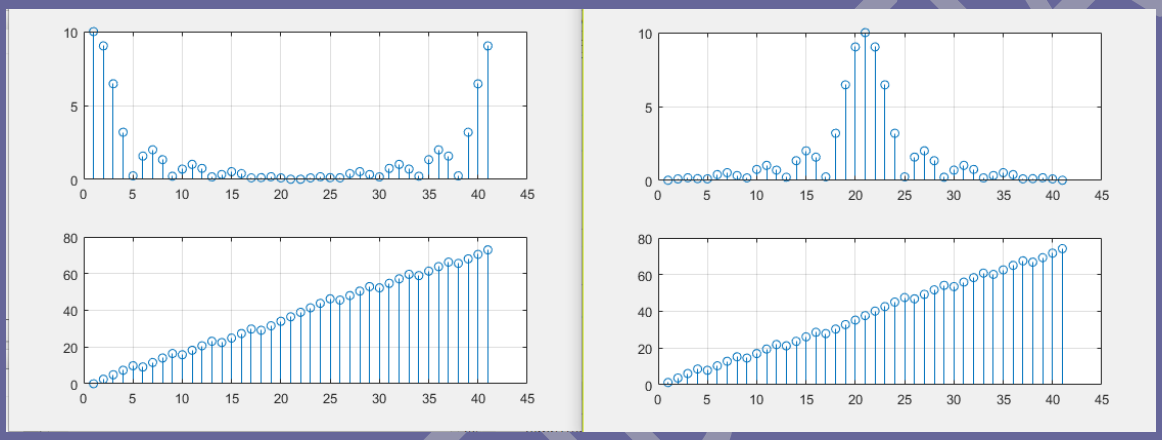


And there it is the code:

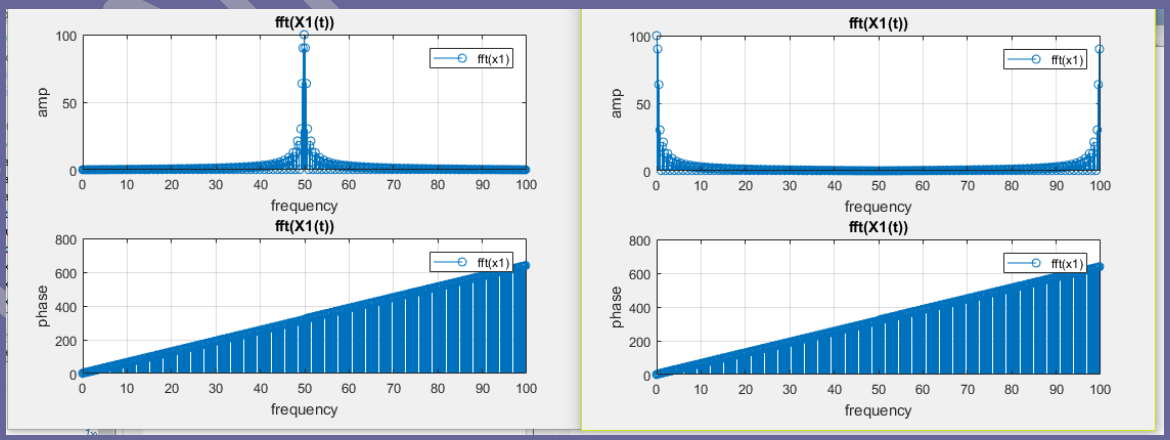
```
clear;
clc;
T=0.1;
Fs=1/T;
t= -2:1/Fs:2;
X1=heaviside(t)-heaviside(t-1);
Y1=fft(X1);
P1=abs(Y1);
L=length(X1);
f=(0:L-1)*(Fs/L); %frequency range
A1=unwrap(angle(Y1));
figure
subplot(2,1,1)
stem(f,P1,'LineStyle','-.', 'MarkerFaceColor','yellow','MarkerEdgeColor','blue');
xlabel('frequency');
ylabel('amp');
title('fft(X1(t))');
legend('fft(x1)');
grid on
subplot(2,1,2)
stem(f,A1,'LineStyle','-.', 'MarkerFaceColor','red','MarkerEdgeColor','green');
xlabel('frequency');
ylabel('phase');
title('fft(X1(t))');
legend('fft(x1)');
grid on
```

For adding numbers of sampling frequency we turn T into 0.01 ;

Before and after shift:



Turning T into 0.01:



Part4:Using fftshift let us have a better view of what we've

Made because it shifts the frequency to be started from 0

Which be its centre;

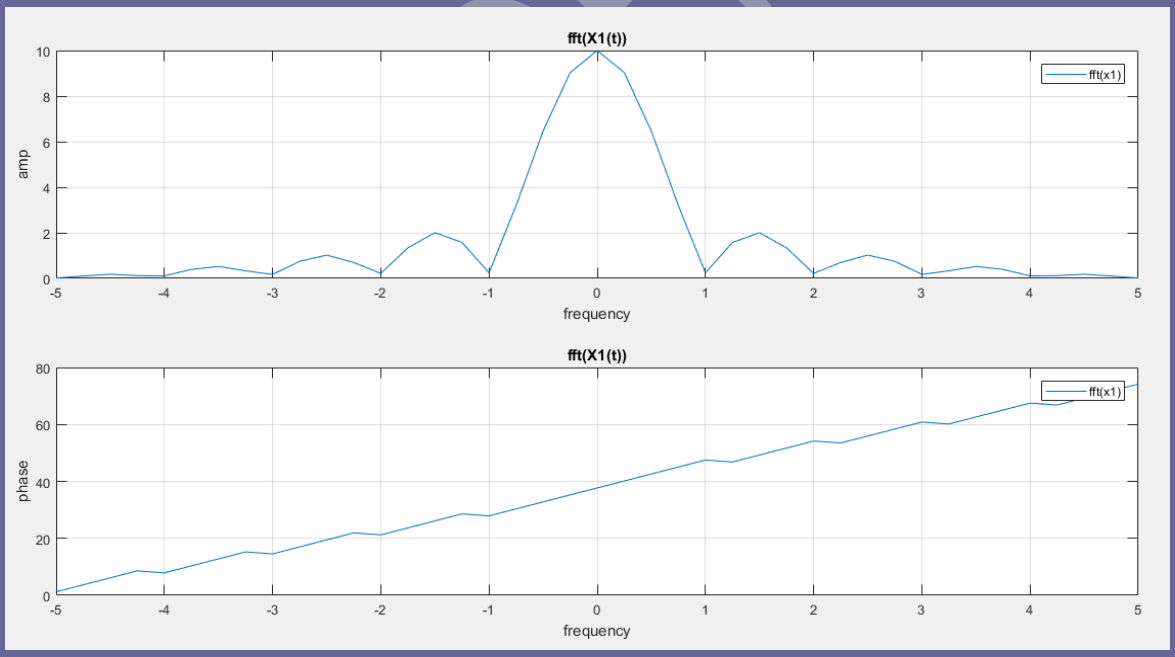
```
Y2=fftshift(Y1);
A2=unwrap(angle(Y2));
P2=abs(Y2);
figure
subplot(2,1,1)
stem(f,P2);
xlabel('frequency');
ylabel('amp');
title('fft(X1(t))');
legend('fft(x1)');
grid on
subplot(2,1,2)
stem(f,A2);
xlabel('frequency');
ylabel('phase');
title('fft(X1(t))');
legend('fft(x1)');
grid on
```

Part5:In this part for having the axel of frequency done well we decided to have our frequence between –Fs/2 & +Fs/2 because Fs is greater than the 2*f of signal.

What kind of information does the fourier transform phase represent?

When you perform any operation on a signal (say a filter) that will affect the amplitude and the phase of all frequency components of your signal. You understand what the amplitude change is. The phase simply reflects the (group) delay for each of the frequency components. The (group) delay is -d(phase)/d(w), where w=2*pi*f. An interesting kind of system is that for which the phase is linear. If that is the case then the delay (its derivative) is flat. If your operation manages to do that then the delay of all frequencies you are interested in is the same and there is no shape distortion of your signal (when you add them all up to make up your signal they are delayed by the same amount and so they align perfectly in the same way they were before the filter). This is important for diagnostics (say you are dealing with an electrocardiogram, for instance).
In terms of filters, the family of filters that has the flattest group delay (the most linear phase) is the Bessel filter, while the family that has the narrowest transition band is also the one that has the poorest phase response (less linear), especially around the cut-off frequency - that is the elliptic or Cauer filter.
For a single frequency the phase helps determine causality (which occurred first) or tracking the path of the signal (in EEG signals for example).

Sampling frequency for this pic is 10.



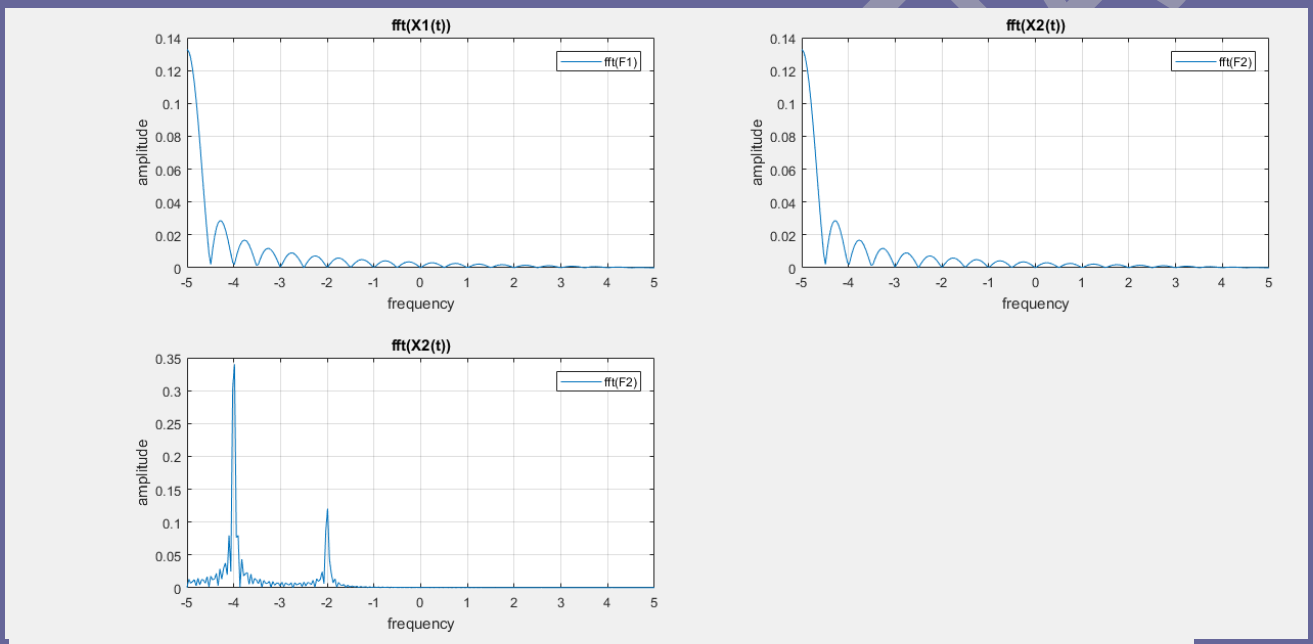
Part6: In order to have your written function(my_fft) we shall say that it gets 2 input ,first y which is the one we need the fourier transform and Fs which is the sampling frequency.
Next line is needed to have the length of the vector and then to find essential points we consider them to be lower than 2^p
And the 2^p?2+1 is the num of points because it is better to be like this, maybe much more better in performance or whatever….
In the line we get the fft we divide it by L to have it normalized.
Next line is to have our frequency range which contains half of the spectrum and to complete it we replace 0 with -1 to have it full described.
Last line is about the amplitude of what we need.

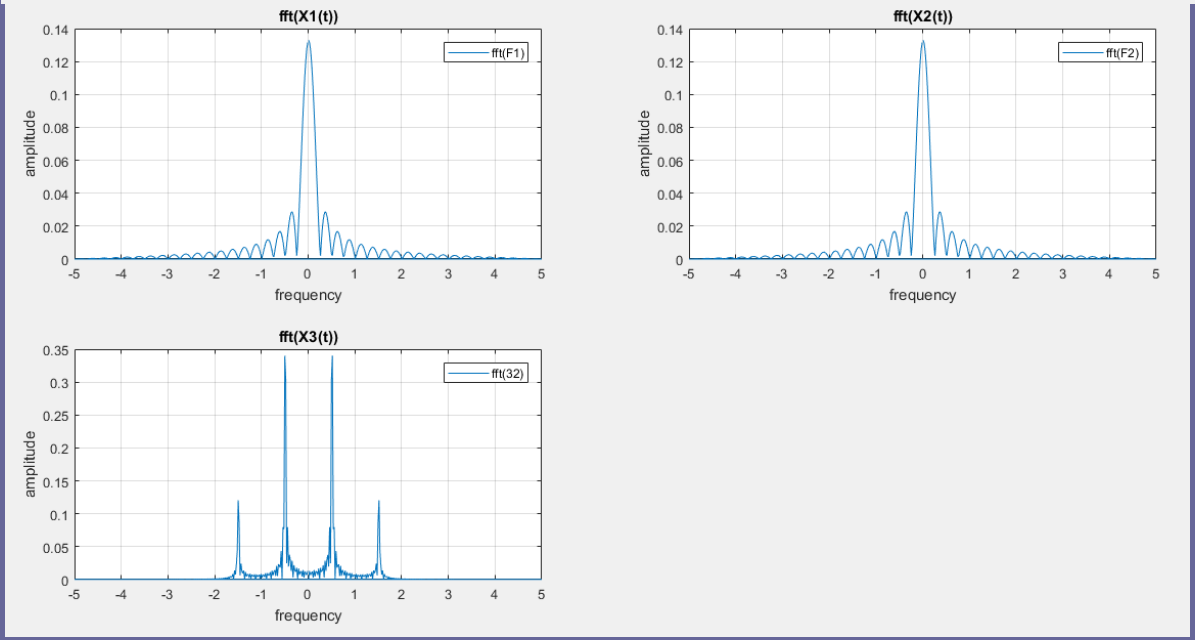
Part7:

We consider T=0.1 like last part.

Time range would be -15 to +15 as I like it☺)

```
clear;
clc;
T=0.1;
Fs=1/T;
t=-15:T:15;
F1=heaviside(t)-heaviside(t-4);
F2=heaviside(t-3)-heaviside(t-7);
F3=(cos(t*pi-pi/3)).^3;
[A1,f1]=my_fft(F1, Fs);
[f1]=fftshift(f1);

figure

subplot(2,2,1);
plot(f1,A1,'DisplayName','fft_F1');
xlabel('frequency');
ylabel('amplitude');
title('fft(X1(t))');
legend('fft(F1)');
grid on

[A2,f2]=my_fft(F2, Fs);
[f2]=fftshift(f2);

subplot(2,2,2);
plot(f2,A2,'DisplayName','fft_F2');
xlabel('frequency');
ylabel('amplitude');
title('fft(X2(t))');
legend('fft(F2)');
grid on

[A3,f3]=my_fft(F3, Fs);
[f3]=fftshift(f3);

subplot(2,2,3);
plot(f3,A3,'DisplayName','fft_F3');
xlabel('frequency');
ylabel('amplitude');
title('fft(X3(t))');
legend('fft(32)');
grid on
```





Those pics are for comparison after and before correction of my_fft function.

Corrected form of the function:

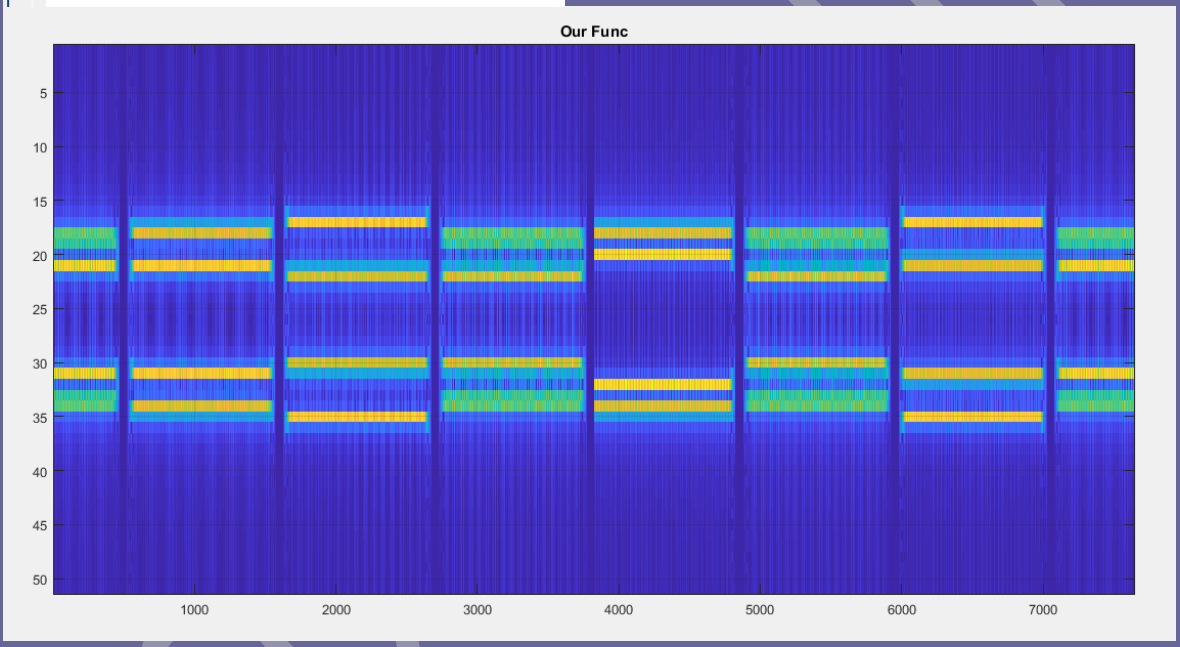
```
function [amp, f] = my_fft(y, Fs)
    L = length(y);
    NFFT = 2^nextpow2(L);
    Y = fft(y, NFFT)/L;
    f = Fs/2 * linspace(-1, 1, NFFT);
    amp = abs(Y(1 : NFFT ));
end
```

Section2:
Spectrogram

In this part finally we did sth exciting!!!
Here we got y named data and fs from audioread and then sectioning that into 50*1 windows and adding some fft and fftshift to get what we want and then abs for its amp & last but not the least plotting.Challenging part was how to separate it into windows which we learned at stiganography 1year ago from M.r Ranjbar .

```
clc;
clear;
[y,Fs]=audioread('tel.wav');
for j=1:7700-50
    Xt(:,j)=y(j:j+50);
end
HX=(fft(Xt));
P=fftshift(HX);
P2=abs(P);
figure
imagesc(P2);
title('Our Func');
grid on
```

Actually I used fftshift to match this part with the next part



In next part we need to let spectrogram function do the affairs but we don't know how it does which is questionable!!!
Reading its help really didn't help, by searching the net and receiving some help , I learned the way to put correct and sufficient numbers in its inputs.

X^ also contains lots of information or I say extra
Because it's just the transform of X which as you
Say it contains more info than x.

audio to spectrogram using Matlab

I am trying to convert multiple audio files into spectrogram using this code

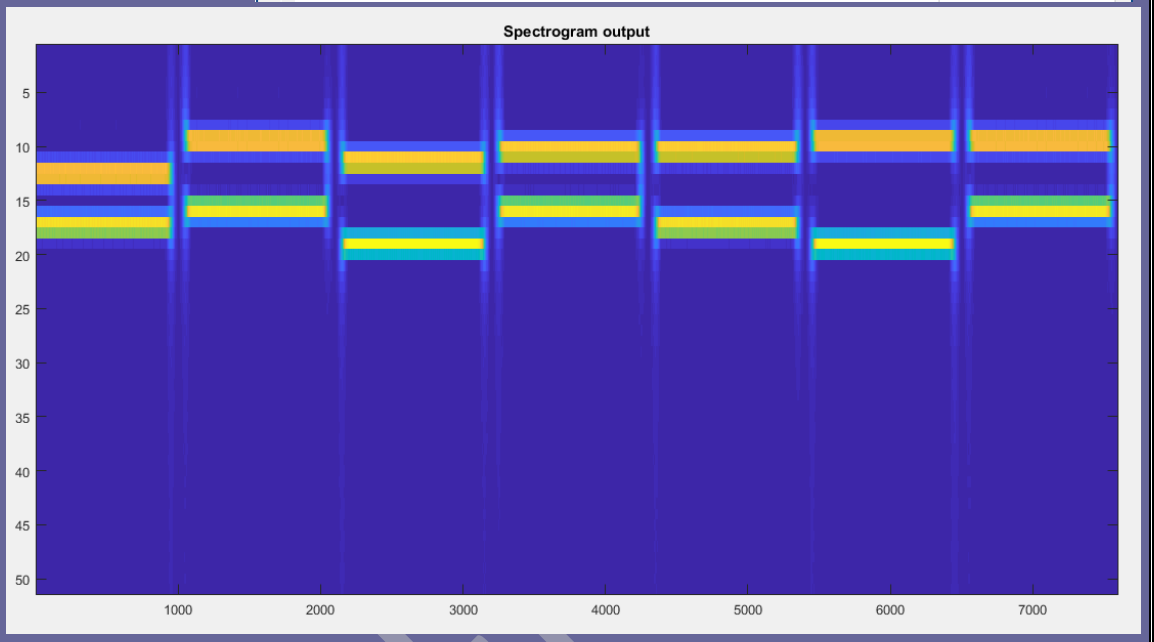
```
[y,fs]=audioread('audio (10).wav');
TotalTime = length(y)./fs;
window=hamming(512);
noverlap=256;
nfft=1024;
[S,F,T,P]=spectrogram(y,window,noverlap,nfft,fs,'yaxis');
ab= surf(T,F,10*log10(P),'edgecolor','none'); axis tight;view(0,90);
```

I receive an output of some audio files as spectrogram and I got an error on some audio file. an error I got is

```
Error using spectrogram>chkinput (line 252)
X must be a vector (either row or column).

Error in spectrogram (line 166)
chkinput(x);

Error in Untitled (line 8)
[S,F,T,P]=spectrogram(y,window,noverlap,nfft,fs,'yaxis');
```

when I examine the audio files than I find that audio with these properties of Y are not converted to spectrogram

```
clc;
clear;
[y,Fs]=audioread('tel.wav');
window=hamming(100);
noverlap=99;
nfft=100;
HX2=spectrogram(y,window,noverlap,nfft,Fs);
figure
imagesc(abs(HX2));
title('Spectrogram output');
```



To be honest I didn't find correct numbers for this part I asked someone;

We have 2 pics with differences and the reason maybe the algorithm or the built in func of matlab.

For telephone problem we can simply use the same algorithm preferably the first (understandable) To distinguish numbers of input by their sound and its special amp for each number if they are different in sound.

Also read: **continuous wavelet transform, Spectrogram from Wikipedia Matlab functioning in google and mathworks.**

Done!