ENCODING AND DECODING OF TOUCH-TONE SIGNALS

Ade-Bello, Abdul-Jelili

Department of Electrical and Computer Engineering

University of New-Mexico.

ECE539 (DIGITAL SIGNAL PROCESSING) PROJECT REPORT

ABSTRACT

This lab introduces a practical application where sinusoidal signals are used to transmit information (a touch-tone dialler) and a band pass FIR filters is used to extract the information encoded in the waveforms. MATLAB is use as a programming tool for this particular project; it's used to study the response of FIR filters to inputs such as complex exponentials and sinusoids. In addition we will use FIR filters to study properties such as linearity and time-invariance in relation to filters window sharpness and signal resolution. The purpose of this lab is not only to use sinusoidal signal to transmit information (in this case dial tones) but to learn how to characterize a filter by knowing how it reacts to different frequency components in the input.

2 INTRODUCTION

Telephone touch-tone pads generate dual tone multiple frequency (DTMF) signals to dial a telephone. When any key is pressed, the sinusoids of the corresponding row and column frequencies, shown in Fig. 1, are generated and summed. As an example, pressing the 5 key generates a signal containing the sum of the two tones, one at 770 Hz and the other at 1336 Hz. The frequencies were chosen to avoid harmonics. No frequency is an integer multiple of another, the difference between any two frequencies does not equal any of the frequencies, and the sum of any two frequencies does not equal any of the frequencies. This makes it easier to detect exactly which tones are present in the dialled signal in the presence of non-linear line distortions.

FREQS	1209 Hz	1336 Hz	1477 Hz	1633 Hz
697 Hz	1	2	3	A
770 Hz	4	5	6	B
852 Hz	7	8	9	C
941 Hz	*	0	#	D

Figure 1. Extended DTMF encoding table for Touch Tone dialing.

3 EXPERIMENT

3.1 ENCODING THE DIGITS

In the first part of this lab, two signals were concatenated then a DTMF dialler was produced. The MATLAB function *'ncode.m'* takes in a vector *'phone_No'* (the vector containing the integers and alphabets (0-9,A-D)) and a sampling frequency *'fs'* then returns the appropriate tone sequence variable *'x'* corresponding to a dialled phone number. Below is the MATLAb code for *'ncone.m'* and the result of the spectrogram of each keypads , the resulting keypad sound was listened to.

<u>Code</u> :

```
%*********************************************************************
% ncode Create a signal vector of tones which will dial
% DTMF(Touch Tone) telephone system.
% usage: x = ncode (phone_No,fs)
% phone_No = vector of characters containing valid phone numbers
% fs = sampling frequency
% x = signal vector that is the concatenation of DTMF tones.
%********************************************************************
function x = ncode(phone_No,fs)
keypads=...              % list of valid digits on phone
['1','2','3','A';
'4','5','6','B';
'7','8','9','C';
'*','0','#','D'];
cfreqs = ones(4,1)*[1209,1336,1477,1633];% columns coresponding frequecies
rfreqs = [697;770;852;941]*ones(1,4);    % rows of coresponding frequecies
x=[];
td1 = 0.2;                               % tone duration in seconds
td2 = 0.05;                              % silence duration in seconds
t=0:fs*td1;
    for i = 1:length(phone_No)
       if (find(phone_No(i)==keypads));
       [a,b]=find(phone_No(i)==keypads);
       sig = cos(2*pi*rfreqs(a,b)*t/fs)+cos(2*pi* cfreqs (a,b)*t/fs);
       slt = zeros(1,fs*td2);
       x = [x,slt,sig];
       else
       disp('Invalid Input.')
       return;
       end
    end
sound(x,fs)
T=1/fs; %sample time
k=0:(fs-1);
y=fft(x);
w=2*pi*k*T;%frequency in radian
figure(1)
plot (abs(y))
title('FFT magnitude spectrum');xlabel('frequency');ylabel('magnitude')
figure(2)
plot(x)
title('Dialed signal');xlabel('time');ylabel('magnitude')
figure(3)
[b, f, T]=specgram(y);
mesh(T, f, abs(b));axis([1 fs 1 T ]);
view(150, 50);
```

```
title('SPECTROGRAM MAGNITUDE OF DIALED KEYS');
ylabel('frequency');
xlabel('time');
zlabel('spectrogram magnitude');
spectrogram(y)

end
```

Result:

Input : >> k = ['5','0','5','D','#','*','6'];

>> fs=8000;

>> x = ncode(k,fs)

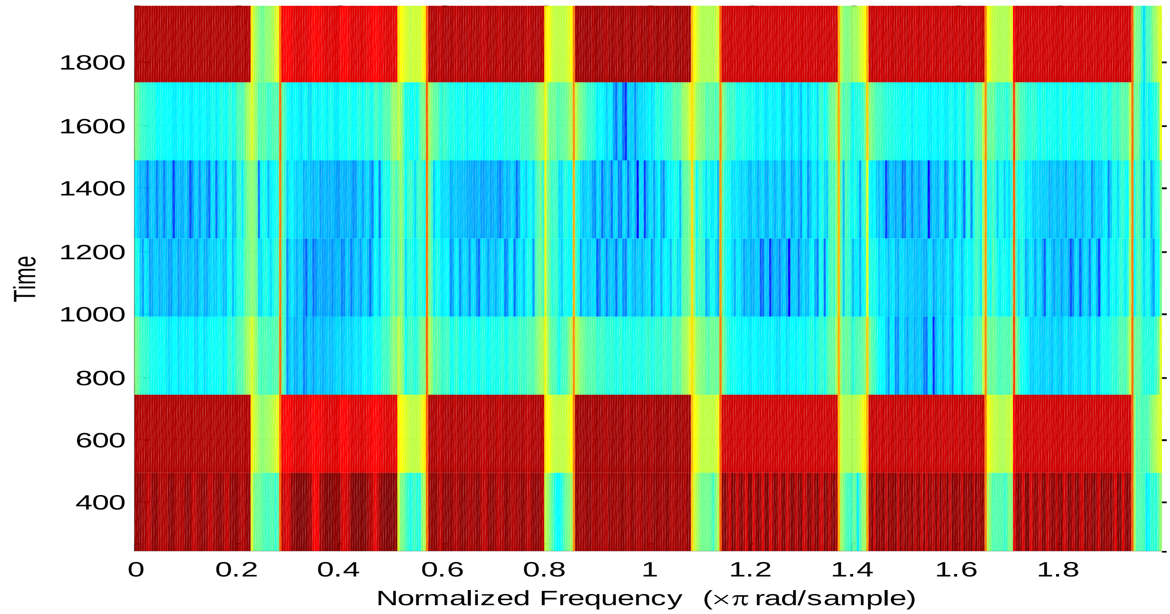


Figure 2.1 Spectrogram for input phone number 505D#*6 with sampling frequency of 8000Hz

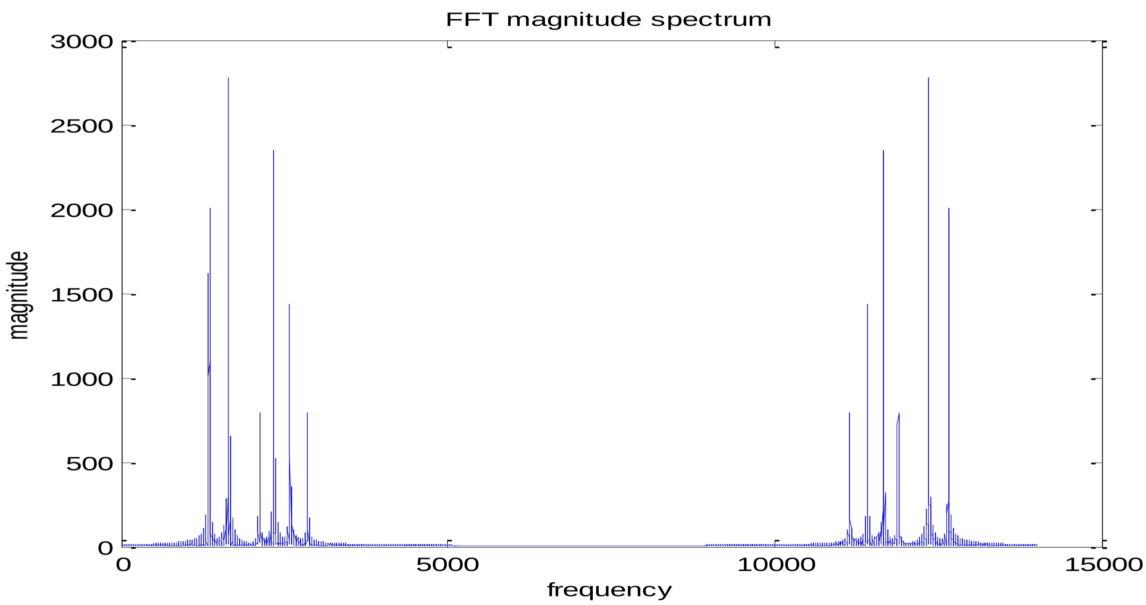


Figure 2.2 FFT of dialled number

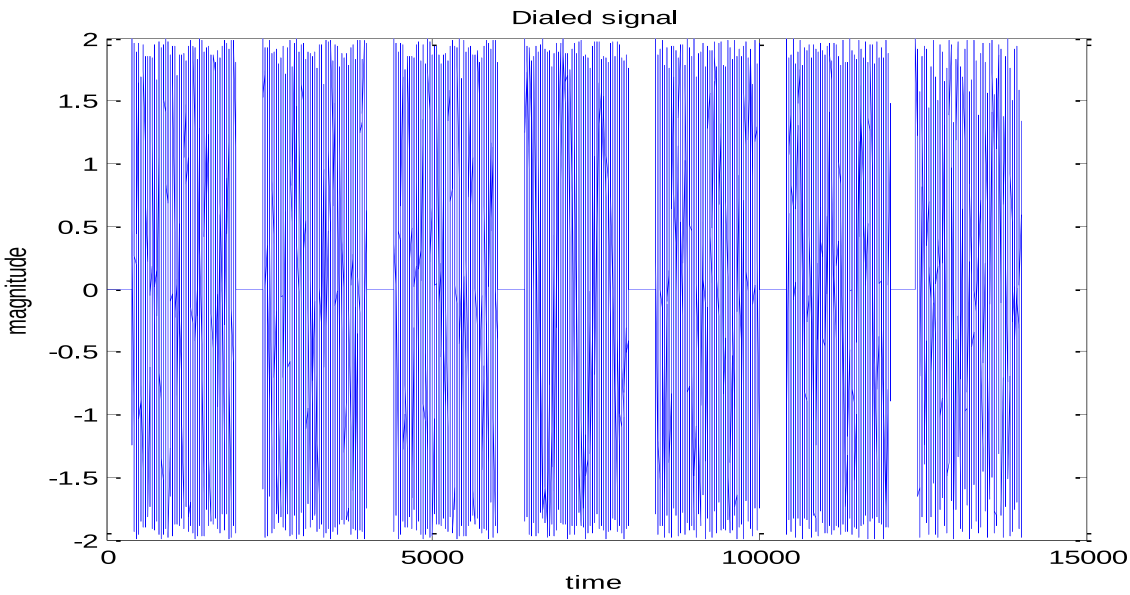


Figure 2.3 Dialled Digits

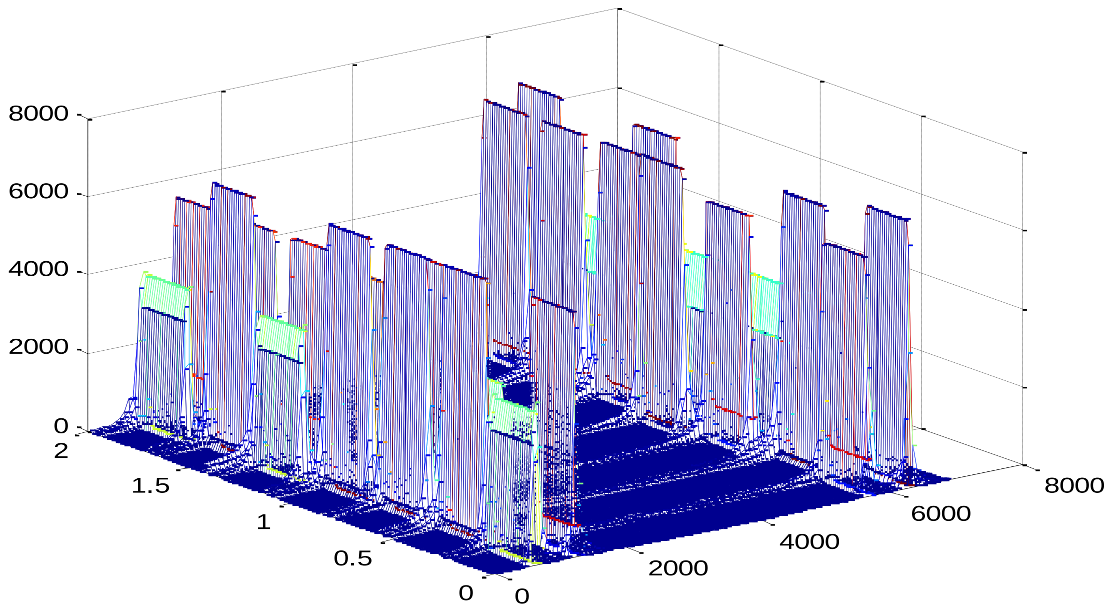


Figure 2.4 Spectrogram of Dialled Digits 505D#*6

3.2 BUILDING THE BAND-PASS FILTER

The next part of the experiment is to build a band-pass filter with a centre frequency of 'w' and a length '$\mathcal{L}$' to determine the window of the filter and frequency location of the pass band. The constant '$\mathcal{A}$' gives flexibility for scaling the filter's gain to meet a constraint such as making the maximum value of the frequency response equal to one, and *'CentreF'* defines the frequency location of the pass band. The bandwidth of the band pass filter is controlled by L; the larger the value of L, the narrower the bandwidth. This also determines the width of the main lobe which is equivalent to the resolution of the signal.

Code:

```
%*********************************************************************
% Impulse-Response of filter
% ww = ImpRes(CenterF, L, fs)
% returns a matrix (L by length(CenterF)) where each column contains
% the impulse response of a BandPass Filter
% CenterF = vector of center frequencies
% L = length of FIR bandpass filters
% fs = sampling frequecies
% Each BPF must be scaled so that its frequency response has a
% maximum magnitude equal to one.
%********************************************************************
function ww = ImpRes(CenterF, L, fs)
ww=ones(L,1)*ones(1,8);              %Size: L x 8.
n=0:L-1;
w=-pi:pi/2000:pi;                    % Step: pi/2000.
h1=cos(0.2*pi*n);
H1=freqz(h1,1,w);
A=1/max(abs(H1));
 for i=1:length(CenterF);
     sg=A*cos(2*pi*CenterF(i)*n/fs);
     ww(:,i)=sg(:);
 end

 for i=1:length(ww(1,:))
     HH=freqz(ww(:,i),1,w);
     plot(w,abs(HH));title('Set of Bandpass Filters'),grid on
     xlabel('Normalized Radian Frequency'),ylabel('Magnitude'),...
     axis([0,pi,0,max(abs(HH))+0.05]) %Adjust Figure Size.
     hold on
end
```

Result :

A plot of the frequency response magnitude and phase is shown below for different values of L(25,50,100,200) respectively. The fig3.1 plot shows the band pass filters with length-25. We can see that the frequency responses overlap too much, and multiple frequencies show up in one pass band. But with L=50 we can see the gradual separation of the pass band filters from each other. By setting L = 100, there is enough separation so that no two frequencies are in one pass band. However you can see that adjacent frequencies are not always in the stop band. With L=100 adjacent frequencies are not all completely in the stop band. Because L varies inversely with pass band width, by doubling L we can cut the pass band width in half, which in this case should be more than enough to also keep adjacent frequencies in the stop band. A plot for a L=200 filter is shown in fig 3.4, and all adjacent frequencies are within the stop band.

Figure 3.1 Bandpass with L=25

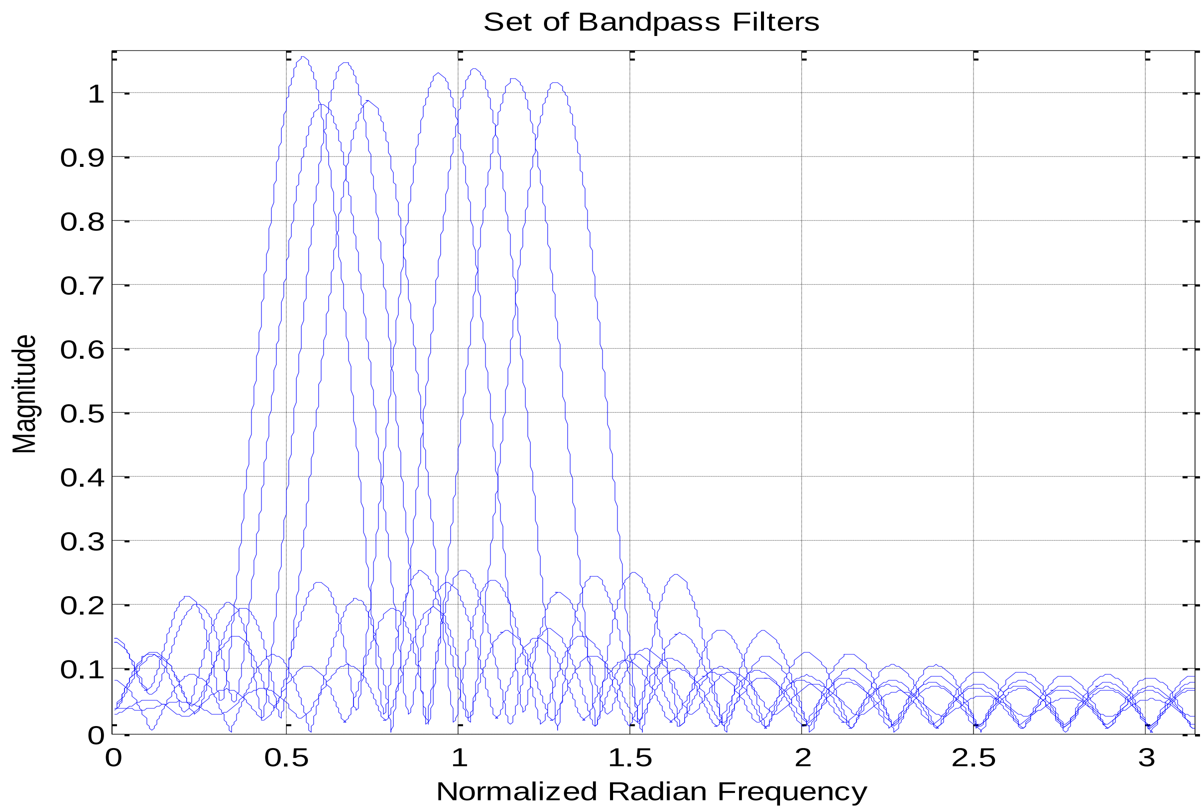


Figure 3.2 Bandpass with L=50

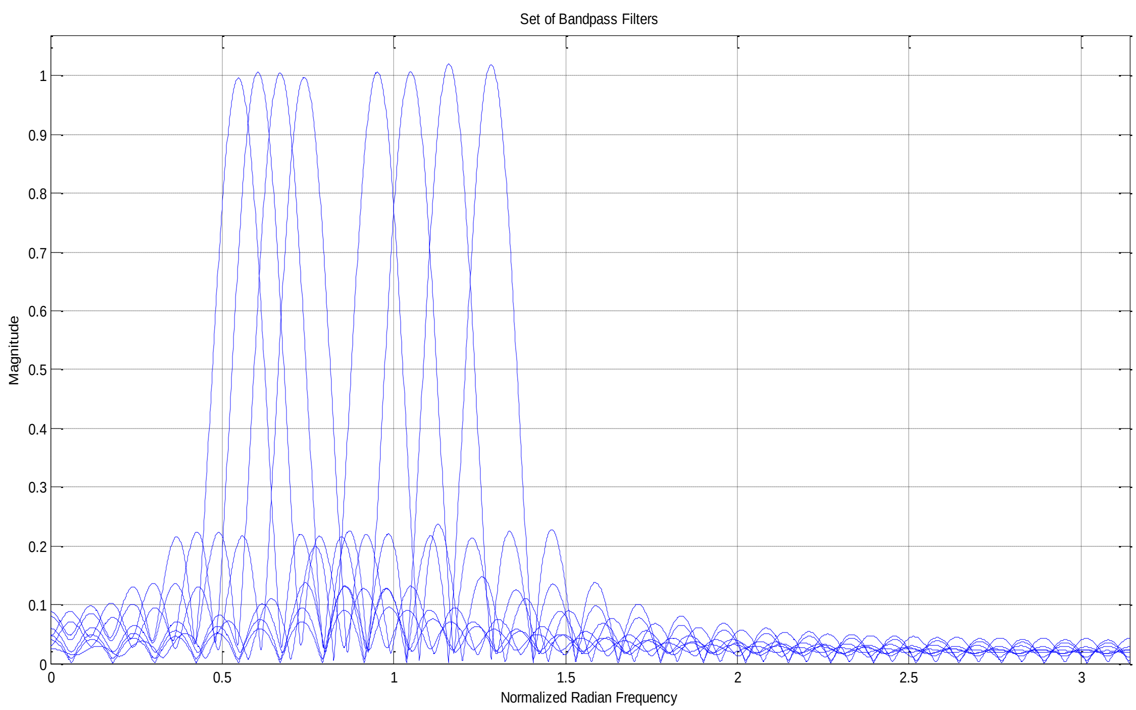


Figure 3.3 Bandpass with L=100

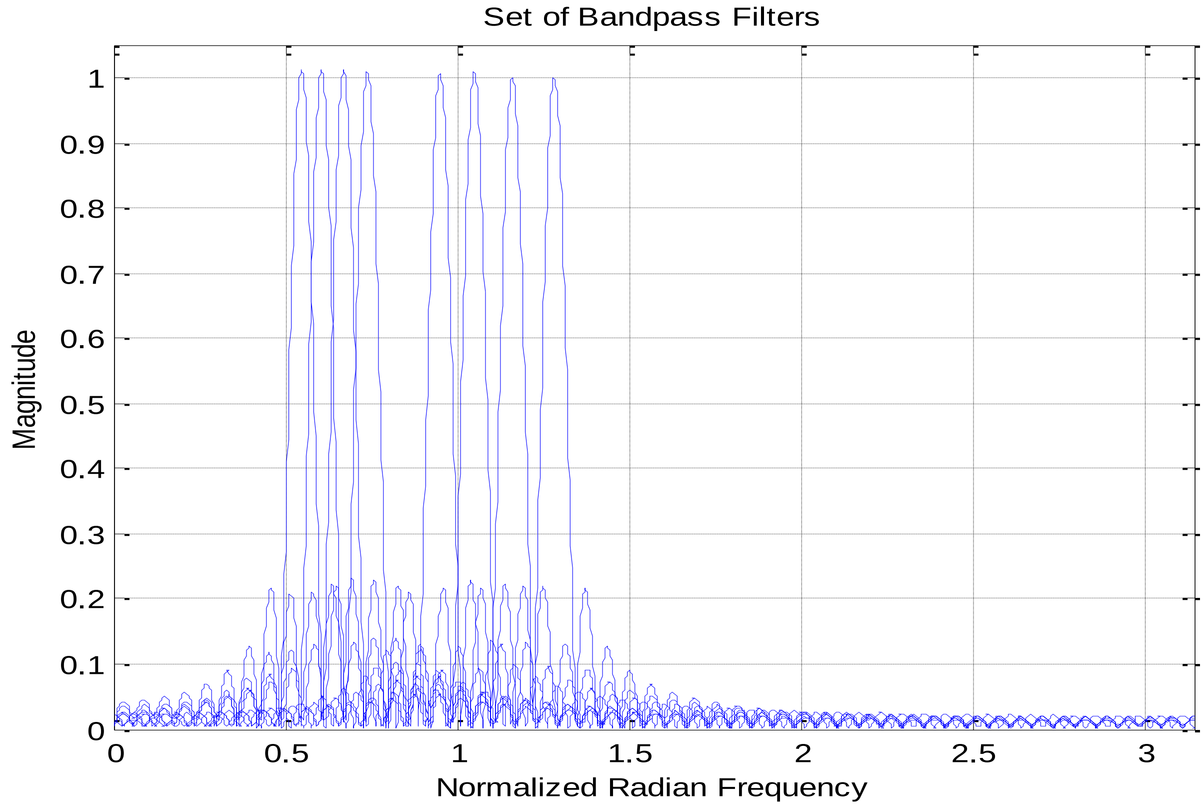
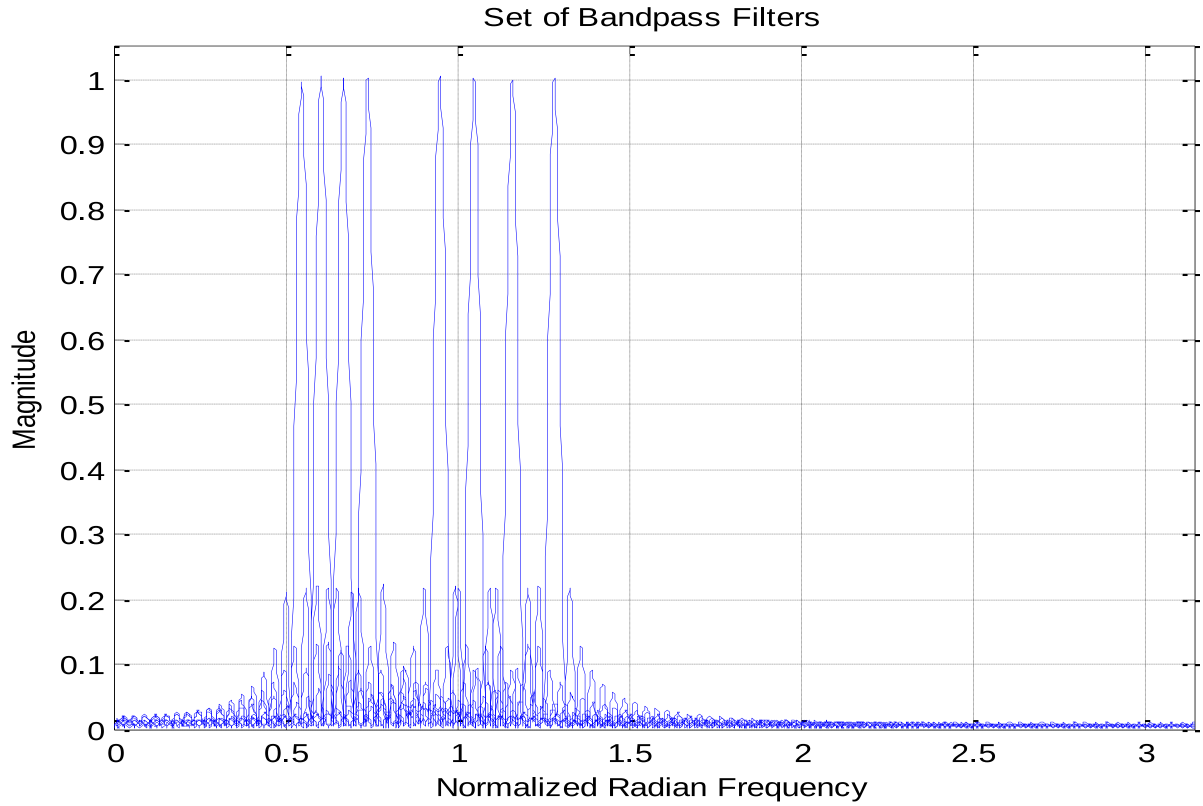


Figure 3.4 Bandpass with L=200



3.3 ISOLATION AND DETECTION OF THE DIGIT

In the next part of this project, we. It is possible to decode DTMF signals using a simple FIR filter bank. The filter bank in Fig. 4 consists of eight band pass filters, where each filter passes only one of the eight possible DTMF frequencies. The input signal for all the filters is the same signal. This section begins with the DTMF Decoding. A decoding system needs two pieces: a set of band pass filters (BPF) to isolate individual frequency components, and a detector to determine whether or not a given component is present. The detector will allocate a binary of 1s and 0s to output to each BPF output and decide which of the frequencies are most likely to be contained in the DTMF tone. The input to the filter bank is a DTMF signal, the outputs from two of the band pass filters (BPFs) will be larger than the rest. we detect which two outputs are the large ones, then we know the two corresponding frequencies. These frequencies are then used as row and column pointers to determine the keypad from the DTMF code. The measure of the output levels is the peak value at the filter outputs, because only one sinusoidal signal and the peak value would be the amplitude of the sinusoid passed by the filter.

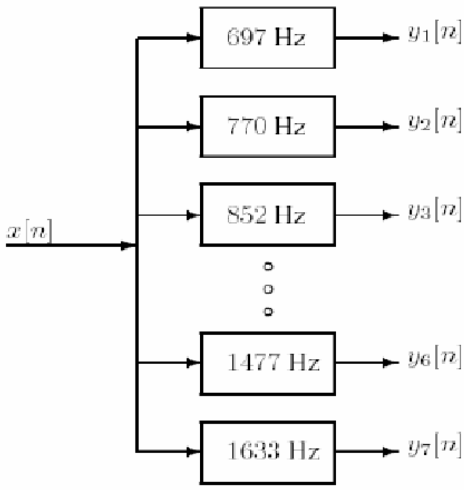


Figure 4: Filter bank consisting of bandpass filters (BPFs) which that frequencies corresponding to the eight DTMF component frequencies listed in Fig. 1. The number is each box is the center frequency of the BPF.

Code:

```
%*********************************************************************
% MaxAmp
% usage: peak = maxAmp(x, ww)
% returns maximum amplitude of the filtered output
% x = input DTMF tone
% ww = impulse response of ONE bandpass filter
% The signal detection is done by filtering x with a length-L
% BPF and then finding the maximum amplitude of the output.
% The peak is either 1 or 0.
%      peak = 1 if max(|y[n]|) is greater than, or equal to, 0.59
%      peak = 0 if max(|y[n]|) is less than 0.59
%*********************************************************************
function peak = maxAmp(x, ww)

x = x*(2/max(abs(x)));     % Scale the input x[n] to the range [-2,+2]
yy = conv(x,ww);           % convolution of signal with BPF impulse response
n = max(abs(yy));          % binary output of signal presence in waveform
if(n >= .59)
    peak = 1;
```

```
else
    peak = 0;
end
  plot(abs(yy));title('Check for maximum amplitude function'),grid on
  xlabel('n'),ylabel('Magnitude')
```

Result :

The maximum value of the magnitude for YY(exp(jω)) is equal to one for each filter: The reason is that we don't want the amplitude change after the signal passes through the filters. The gain of all filters that is equal to 1 can be used to adjust the magnitude of output if we control both frequency response and amplitude of input.

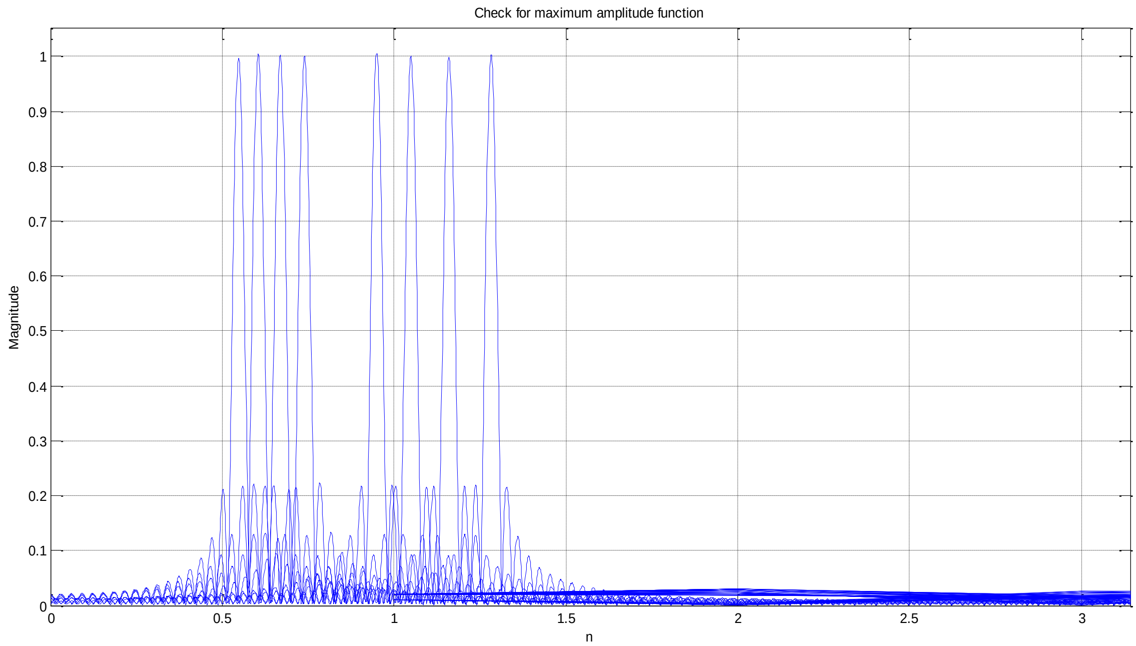


Figure 5. plot of available frequencies from BPF output

3.4 DECODING THE DIGITS

The DTMF decoding function ,'$decode.m$' make use of the binary information from '$max\mathcal{A}mp.m$' to determine which key was pressed based on an input DTMF tone. First, it designs the eight band pass filters that are needed. It then breaks the input signal down into individual segments. For each segment, it will have to call the '$max\mathcal{A}mp.m$' function to determine the value of the peak for different BPF outputs and then determine the key for that segment. The final output is the list of decoded keys. The task of breaking up the signal so that each segment corresponds to one key is done with the '$n\mathcal{T}ones.m$' function.

Codes:

```
%*********************************************************************
% DECODE
% keys = decode(x,L,fs)
% returns the list of key names found in x[n]
% keys = decoded keypads
```

```
% x = DTMF waveform
% L = filter length
% fs = sampling frequency
%******************************************************************
function keys = decode(x,L,fs)
keypads = ['1','2','3','A';...
           '4','5','6','B';
           '7','8','9','C';
           '*','0','#','D'];

cfreqs = ones(4,1)*[1209,1336,1477,1633];  % defines column matrix
rfreqs = [697;770;852;941]*ones(1,4);      % defines row matrix
CenterF = [rfreqs(:,1)' , cfreqs(1,:)];    % defines 1X8 vector of freqs

 % call ImpRes to form matrix of filters
ww = ImpRes( CenterF,L,fs );      % ww = L by 8 MATRIX of all the filters
 % call nTones to find the start and end tone indices
[numbeg,numend] = nTones(x,fs);
 keys = [];                            % initializes keys
for kk=1:length(numbeg)                % cycle through each tone
    n = [];
    x_1 = x(numbeg(kk):numend(kk));    % Extract one DTMF tone

    for i=1:length(CenterF)            % cycle through each filter
        zz = maxAmp(x_1,ww(:,i));
         n = [n,zz];      % creates a vector of ones and zeros representing
                          % where the frequency components lie.
    end
    aa = find(n==1);      % creates a vector of indicies where ones occur

% checks for impossible scores and skips if they are found
    if length(aa) ~= 2 || aa(1) > 4 || aa(2) < 5
        keys = [keys,'error'];
        continue
    end
    row = aa(1);                       % decodes row position from aa
    col = aa(2)-4;                     %decodes col position from aa
    keys = [keys, keypads(row,col)];   %sets keys equal to the current keys
                                       %and the key found in this iteration
end
```

Other code used

```
%*************************************************************************************
% nTones: find the DTMF tones within x[n]
% usage: [numbeg,numend] = nTones(x,fs)
% length of numbeg = number of tones found
% numbeg is the set of Beginning indices
% numend is the set of Ending indices
% x = input signal vector
% fs = sampling frequency
%*************************************************************************************
function [numbeg,numend] = nTones(x,fs)
x = x(:)'/max(abs(x));               % normalize x[n]
XLen = length(x);
Rlen = round(0.01*fs);
setP = 0.02;                         % make everything below 2% zero
```

```
x = filter( ones(1,Rlen)/Rlen, 1, abs(x) );
x = diff(x>setP);   % find the differnece of adjacent vectors of x[n]
x_1 = find(x~=0)';
  if x(x_1(1))<0,
      x_1 = [1;x_1];
  end
  if x(x_1(end))>0,
      x_1 = [x_1;XLen];
  end
val_x = [];
 while length(x_1)>1
    if x_1(2)>(x_1(1)+10*Rlen)
       val_x = [val_x, x_1(1:2)];
    end
       x_1(1:2) = [];
 end
numbeg = val_x(1,:); numend = val_x(2,:);
*************************************************************************************
```

Results

Input : >> k = ['5','0','5','D','#','*','6'];

>> fs=8000;

>> x = ncode(k,fs)

Output:>>keys = decode(x,L,fs)

keys = 505D#*6

5. DECODED PHONE NUMBERS

The following code shows the input and output when the program is tested with the input

```
>> P=['*','#','5','0','5','6','3','2','9','1','8','7','A','D'];

>> L=100;

>> fs=8000;

>> x = ncode(P,fs)

>> keys = decode(x,L,fs)

   keys  = *#5056329187AD
```

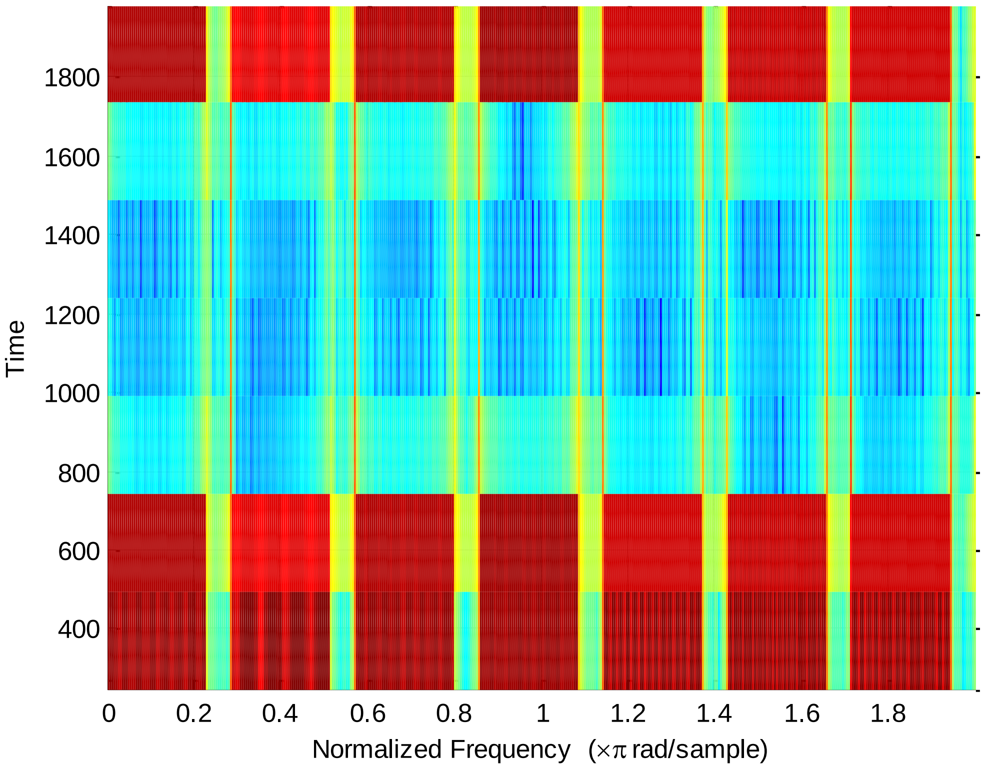


Figure 6. spectrogram of phone number ['*','#','5','0','5','6','3','2','9','1','8','7','A','D']

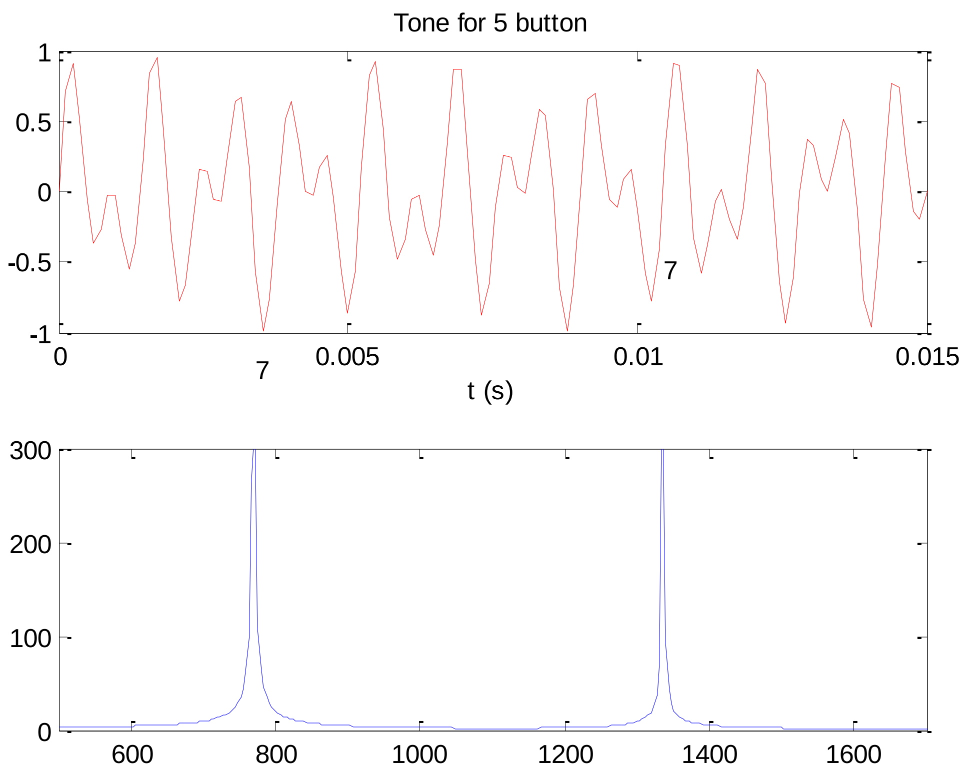


Figure7. Graph for button 5

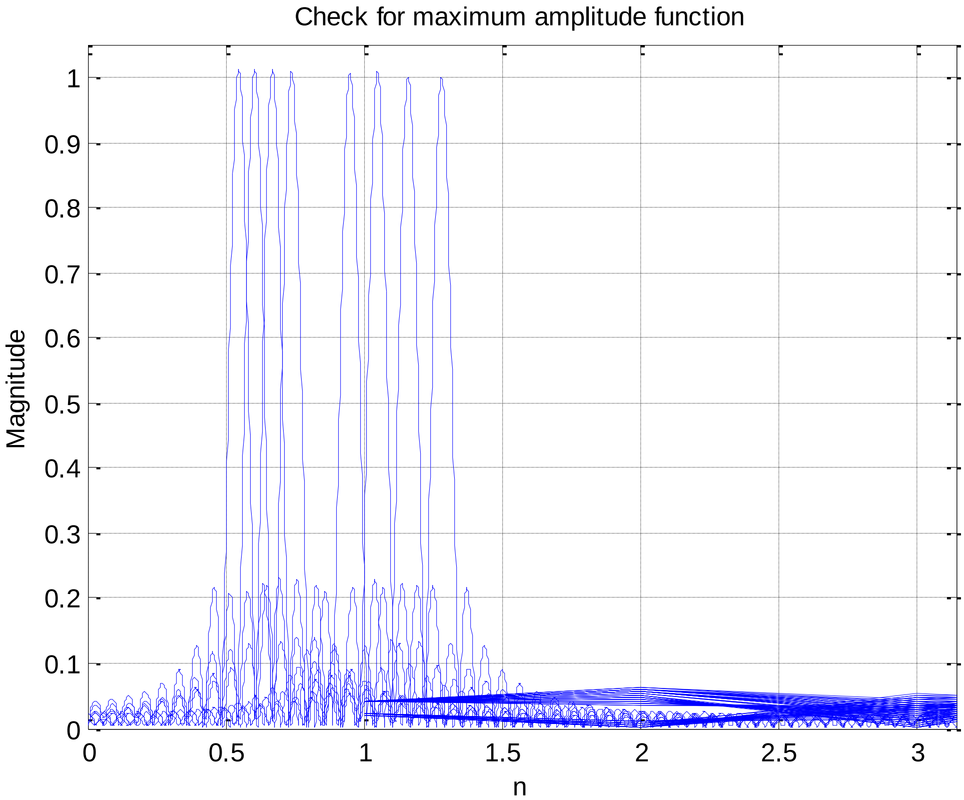


Figure 8. maximum amplitude function for L=100 and fs=8000

Using values of L greater than or equal to 66 provide sufficient selectivity in determining the corresponding numbers (keys) of the DTMF tones as seen above. Therefore, the decoding accuracy is limited to values of L greater than or equal to 66.

Verification

>> L=64 ;>> keys = decode(x,L,fs)

keys = *#error0errorerror329error87AD

>> L=65;>> keys = decode(x,L,fs)

keys = *#5056329error87AD

>> L=66;>> keys = decode(x,L,fs)

keys = *#5056329187AD

6. CONCLUSION

In general, from the application of Fourier transform and in relation with the use of touch tone. Assigning waveforms to digits with signature addresses can be made used of in several ways like voice recognition, speech security and advance signal processing. Although the use of DFT is limited to sequences that have zeros at the end which also limit it resolution ability and stationarity since it cannot accommodate signals with time-varying frequency intent. But from the above experiment it can be conclude that resolution of signals can be controlled by varying the values of the length of the BPF which signals passes through.