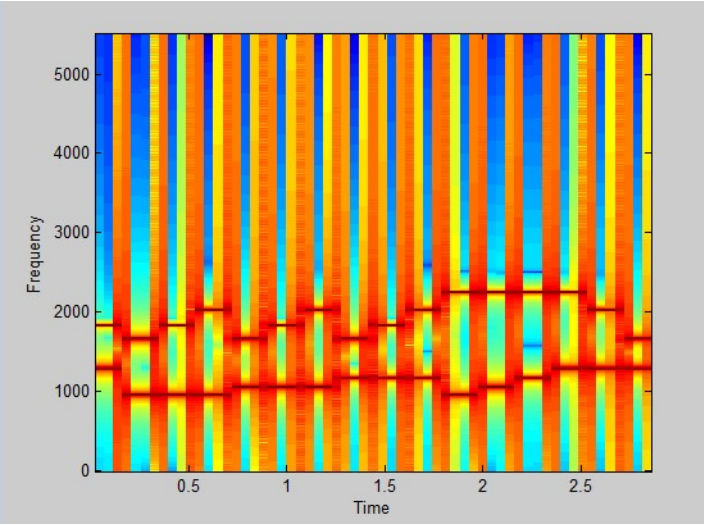
Warm-Up

3.1 DTMF Synthesis

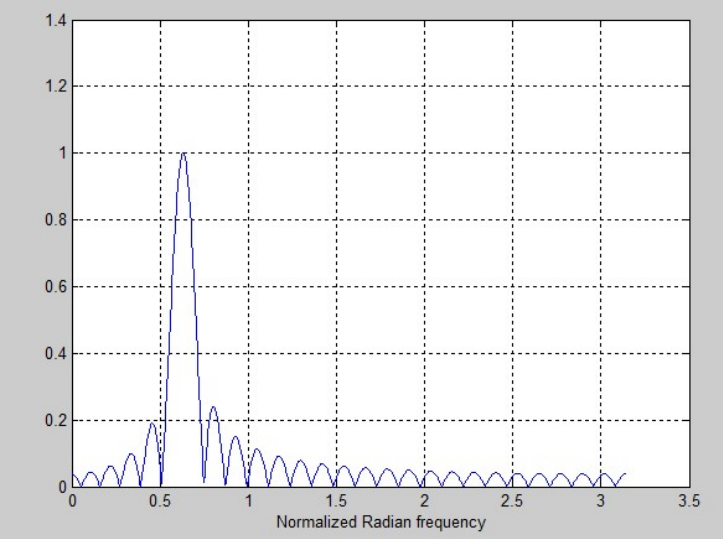
```
function xx=dtmfdial(keyNames,fs)

%DTMFDIAL Create a signal vector of tones which will dial
%         a DTMF (Touch Tone) telephone system.
%
% usage:  xx=dtmfdial(keyNames,fs)
%  keyNames=vector of characters containing valid key names
%        fs=sampling frequency
%   xx=signal vector that is the concatenation of DTMF tones.
%
dtmf.keys = ['1','2','3','A';
             '4','5','6','B';
             '7','8','9','C';
             '*','0','#','D'];
dtmf.colTones = ones(4,1)*[1209,1336,1477,1633];
dtmf.rowTones = [697;770;852;941]*ones(1,4);
xx=[];
for zz=1:length(keyNames)
    kk=keyNames(zz);
    xx=[xx,zeros(1,400)];
    [ii,jj]=find(kk==dtmf.keys);
    row_f=dtmf.rowTones(ii,jj);
    col_f=dtmf.colTones(ii,jj);
    xx=[xx,cos(2*pi*row_f*(0:1600)/fs)+cos(2*pi*col_f*(0:1600)/fs)];
end
soundsc(xx,fs);
specgram(xx,1024,11025);
```

```
>> dtmfdial(['0123456789ABCD#*'],8000);
>>
```



3.2 Simple Bandpass Filter Design

```
% 3.2 simple bandpass filter design
% part (a)
ww=0:(pi/1000):pi;
h=cos(0.2*pi*(0:50));    % assume the belta value is 1
HH=freqz(h,1,ww);        % get the frequency response
peak_v=max(abs(HH));     % find the peak value
h_1=1/peak_v*cos(0.2*pi*(0:50));  % get the frequency response whose
peak
                                  % value is 1
HH_1=freqz(h_1,1,ww);             % get the frequency response
figure (1)
plot(ww,abs(HH_1));
xlabel(' Normalized Radian frequency')

% part (b)
pb=find(abs(HH_1)>=0.707);
BW1=ww(pb(1)); % find the two 3db frequencies of the bandpass filter
BW2=ww(length(pb)+pb(1)-1);
delta_BW=BW2-BW1;   %find the bandpass width
```



Part (a)

β=0.0385

part (b)

The two 3-db frequency is 0.5781 rad and 0.6871 rad respectively.

Part (c)

The analog frequency components that will be passed by the band pass filter is from

f_1 =0.5781*8000=4624.8 rad/s to f2=0.6871*8000=5496.8 rad/s .

Lab Exercises: DTMF Decoding

4.1 simple Bandpass Filter Design

Part (a) & (b)

```
function hh = dtmfdesign(fb,L,fs)
%DTMFDESIGN
% hh = dtmfdesign(fb, L, fs)
% returns a matrix (L by length(fb)) where each column contains
% the impulse response of a BPF, one for each frequency in fb
% fb = vector of center frequencies
% L = length of FIR bandpass filters
% fs = sampling freq
%
% Each BPF must be scaled so that its frequency response has a
% maximum magnitude equal to one.
ww = 0:pi/1000:pi;
hh = zeros(L,length(fb));  % create a L*length(fb) matrix
beta = zeros(1, length(fb));  % create a vector ,length is length of fb

for k = 1:length(fb)
    h = cos(2*pi*fb(k)*(0:L-1)/fs);
    HH = freqz(h,1,ww);
    beta(k) = 1/max(abs(HH));    %get the beta value
    hh_1 = h*beta(k);       %get the frequency response whose peak
                             % value is 1
    hh(:,k) = hh_1;
end
```

```
% part 4.1
% part (c)
for L = 40:100
 t = dtmfdesign([697,770],L,8000);
 h_1 = t(:,1);
 HH_1 = freqz(h_1,1,ww_d);
 stb_index = find(abs(HH_1)>=0.25);
 stb_right = ww_d(stb_index(1)+length(stb_index)-1);
 if (2*pi*770/8000>=stb_right)
     L_min = L;
     break
 end
end
```

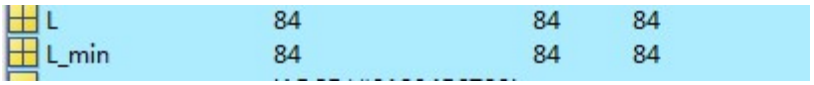
```
% part (d)
ww_d=0:pi/1000:pi;
for kk = 1:8
   t = dtmfdesign([697,770,852,941,1209,1336,1477,1633],40,8000);
   h_d = t(:,kk);
   HH_d = freqz(h_d,1,ww_d);
   figure (1)
   plot(ww,abs(HH_d));
   xlabel(' Normalized Radian frequency')
   hold on
end

%part (e)
for kk = 1:8
   t = dtmfdesign([697,770,852,941,1209,1336,1477,1633],80,8000);
   h_e = t(:,kk);
   HH_e = freqz(h_e,1,ww_d);
   figure (2)
   plot(ww,abs(HH_e));
   xlabel(' Normalized Radian frequency')
   hold on
end
```

Part c)

As the code shown above, I can find the minimum length L so that the frequency response will satisfy the specifications on pass-band width and stop-band rejection.

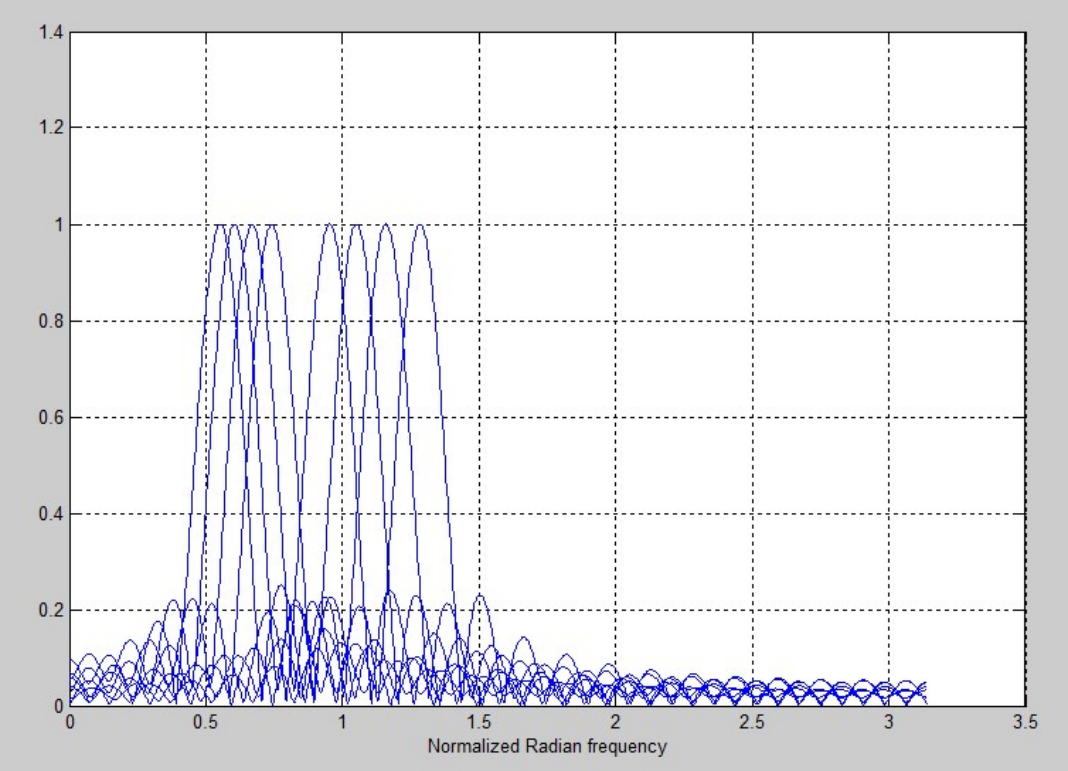
For the eight DTMF frequencies, we only need to check the nearest two frequencies. That is 697Hz and 770Hz. If the 770Hz is in the region of the stop-band of the filter which isolates the 697Hz component (i.e. the filter is centered at 2*pi*697/8000), then all the others will satisfy the specifications on pass-band width and stop-band rejection.

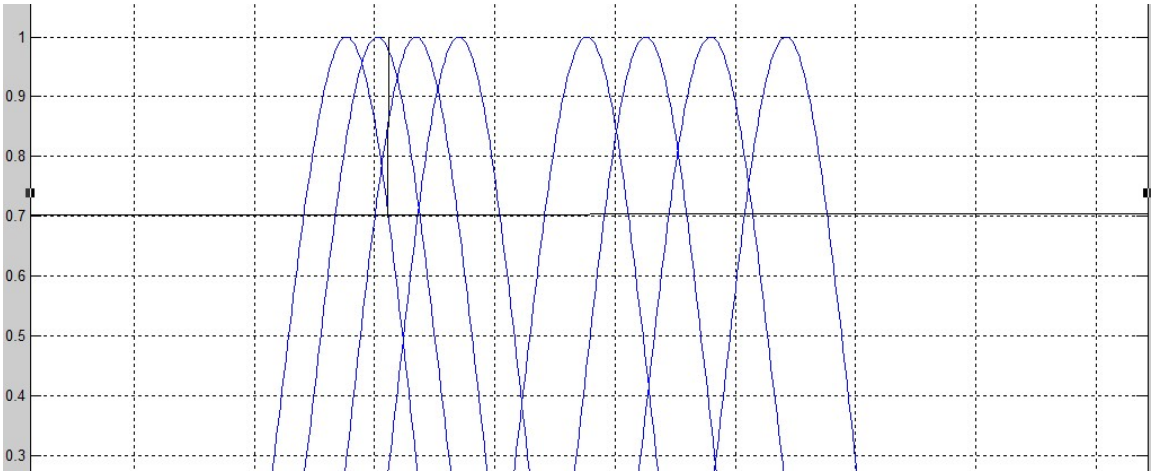
The Minimum Length L=84



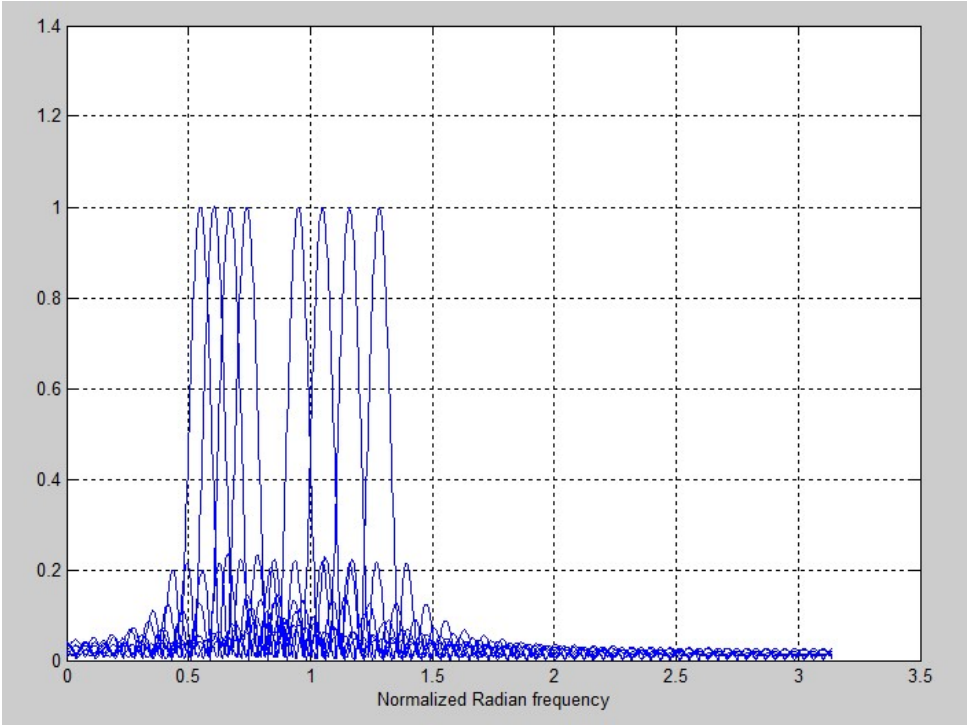
Part d) & e)

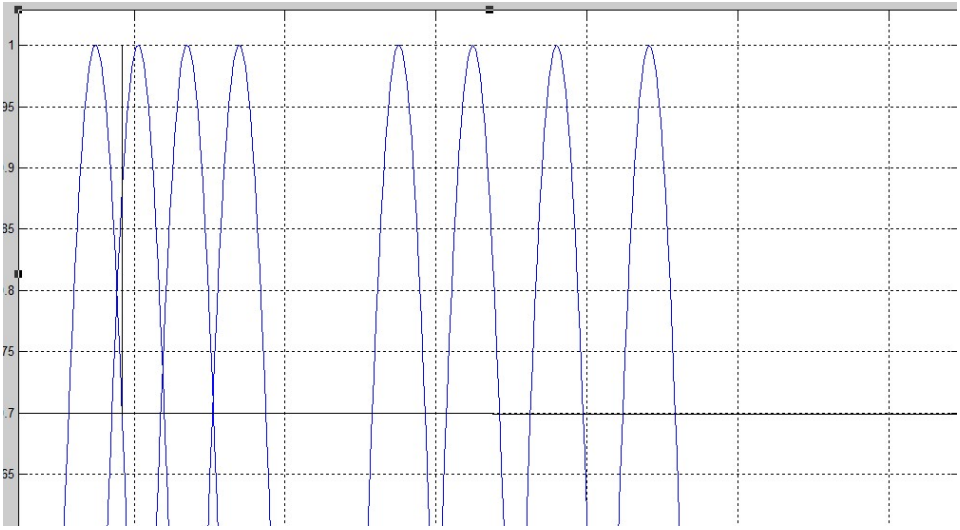
L=40

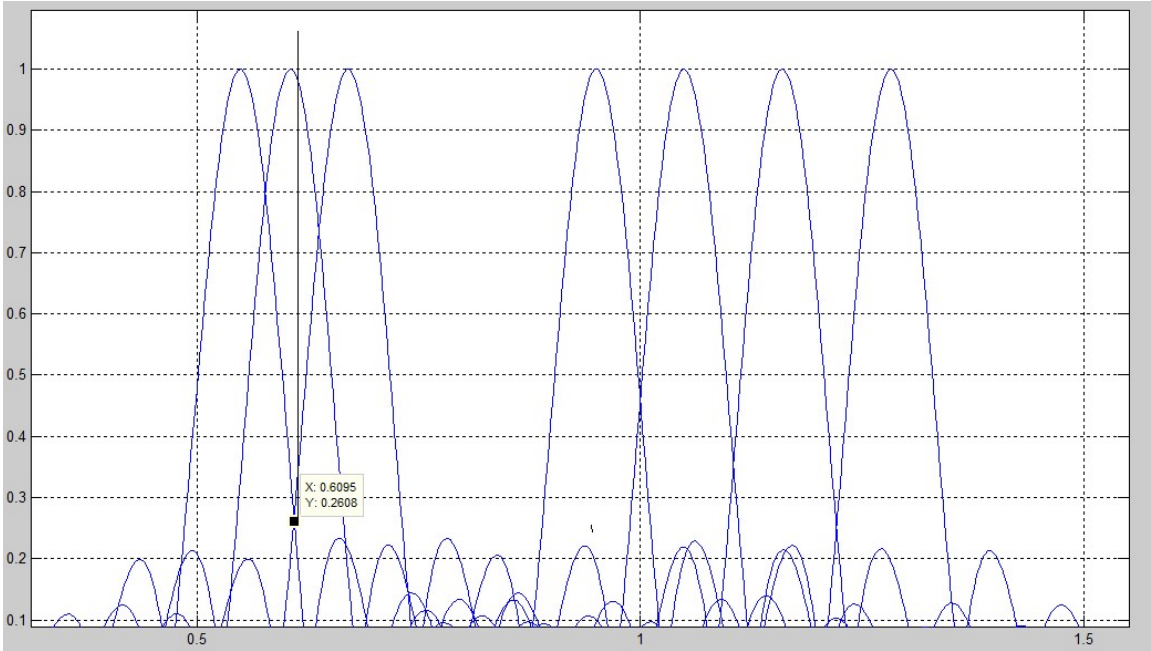

1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3
3.5
Normalized Radian frequency


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3

L=80


1.4
1.2
1
0.8
0.6
0.4
0.2
0
0
0.5
1
1.5
2
2.5
3
3.5
Normalized Radian frequency





For both L=40 and L=80, f_s =8000Hz, the pass-bands are <u>not</u> narrow enough to separate the DTMF frequency components.

We only need to choose f_b=697Hz and check whether the 770Hz is in the filter's stop-band.

When L=40, the 770 Hz in the pass-band of the filter. It is not narrow enough to separate the DTMF frequency components.

When L=80, due to the larger L, the pass-band become narrower. So the 770 Hz is not in the pass-band of the filter. However, since the 770Hz is not in the stop-band of the filter, the L=80 still cannot ensure that the pass-bands are narrow enough to separate all the DTMF frequency components.

Part f)

If one component is in the pass-band, then the filter can pass it. If one component is in the stop-band, then the filter will reject it.

When L>=84, it can ensure that each filter's pass-band is narrow enough so that only one frequency component lies in the pass-band and the others are in the stop-band.

Among the eight DTMF frequencies, since the 697Hz and 770 Hz are the nearest two frequencies, the filters which is centered at 697Hz and 770Hz are hardest to meet the specifications for the chosen value of L.

4.2 A scoring Function: dtmfscore.m

Part (a) & (b)

```
function sc = dtmfscore(xx, hh)
%DTMFSCORE
% usage: sc = dtmfscore(xx, hh)
% returns a score based on the max amplitude of the filtered output
% xx = input DTMF tone
% hh = impulse response of ONE bandpass filter
%
% The signal detection is done by filtering xx with a length-L
% BPF, hh, and then finding the maximum amplitude of the output.
% The score is either 1 or 0.
% sc = 1 if max(|y[n]|) is greater than, or equal to, 0.59
% sc = 0 if max(|y[n]|) is less than 0.59
%
xx = xx*(2/max(abs(xx))); %--Scale the input x[n] to the range [-2,+2]
yy = conv(xx,hh);
if (max(abs(yy)))>=0.59
    sc=1;
else
    sc=0;
end
ww=200:500;
plot(ww,yy(200:500));
```

Part d)

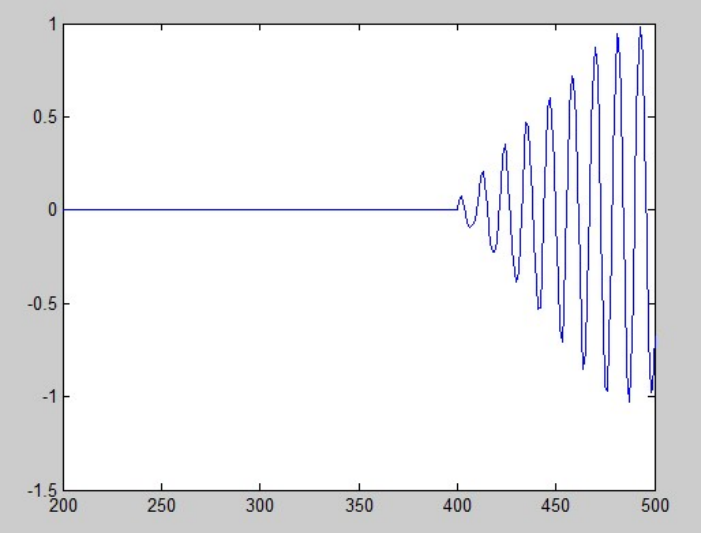
The maximum value of the magnitude for the frequency response must be equal to one because when the sinusoids in the DTMF pass the band-pass filter, it will experience a known gain. In order to be convenient to get the scoring threshold (59% level), we set the max value to be one. Thus, the scoring threshold will be 0.59.

Part e) no question need to answer

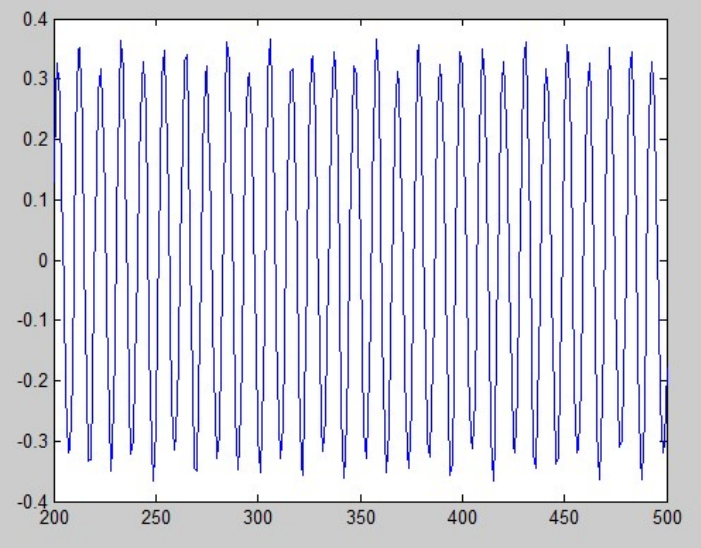
Matlab code

```
hh_c = dtmfdesign(697,80,8000);
xx = dtmfdial('3',8000);
p = dtmfscore(xx,hh_c);
```

```
hh_c = dtmfdesign(697,80,8000);
xx=dtmfdial('4',8000);
p=dtmfscore(xx,hh_c);
```



Matched situation (figure 1)



Mismatched situation (figure 2)

The key "3" is pressed the tones of two DTMF frequency components, 697Hz and 1477Hz respectively. I use the 697Hz as the frequency location of the pass-band to design a filter. The p=1 means that the DTMF tone includes the BPF. (See figure 1)

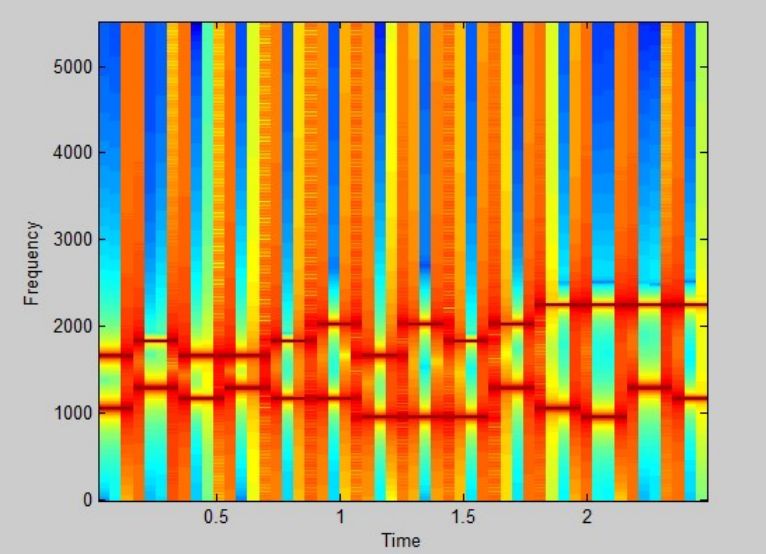
The key "4" is pressed the tones of two DTMF frequency components, 770Hz and 1209Hz respectively. I use the 697Hz as the frequency location of the pass-band to design a filter. The p=0 means that the DTMF tone does not include the BPF. (See figure 2)

4.3 DTMF Decode Function: dtmfrun.m

```
function keys = dtmfrun(xx,L,fs)
%DTMFRUN keys = dtmfrun(xx,L,fs)
% returns the list of key names found in xx.
% keys = array of characters, i.e., the decoded key names
% xx = DTMF waveform
% L = filter length
% fs = sampling freq
%
center_freqs = [697,770,852,941,1209,1336,1477,1633];
dtmf.keys = ['1','2','3','A';
             '4','5','6','B';
             '7','8','9','C';
             '*','0','#','D'];

hh = dtmfdesign( center_freqs,L,fs );
% hh = L by 8 MATRIX of all the filters. Each column contains the
% impulse response of one BPF (bandpass filter)
%
[nstart,nstop] = dtmfcut(xx,fs); %<--Find the beginning and end of tone
bursts
keys = [];
p = zeros(1,8);   % store the score values
for kk=1:length(nstart)
x_seg = xx(nstart(kk):nstop(kk)); %<--Extract one DTMF tone
  for ii = 1:8
      p(ii) = dtmfscore(x_seg,hh(:,ii));  %determine the socre value
  end
  num_freq = find(p==1);
  if length(num_freq)>2          % check whether there is any error in
score
      keys = [keys,'-1'];
  else
        row_index = find(p(1:4)==1);  % get the row index
        col_index = find(p(5:8)==1);  % get the column index
        keys = [keys,dtmf.keys(row_index,col_index)];
  end
end
end
```

4.4 Telephone Number

```
>> fs = 8000; %<--use this sampling rate in all functions
tk = '407*89132#BADC';
xx = dtmfdial( tk, fs );
L =100; %<--use your value of L
dtmfrun(xx, L, fs)

ans =

407*89132#BADC
```



4.5 Demo

If needed, I can show my work in the lab class.