Lab #9

EE 224

1. *Introduction:* In this lab, we learnt about the practical application of sinusoidal signals, and design and implement a bandpass FIR filters. We used a touch-tone dialer to encode and decode the transition of sinusoidal information.

2. *PreLab*

2.1.2. On running through the following code:

ftable = [1;2;3;4;5]\*[80,110]

 fs = 8000;

 xx = [ ];

 %disp(?--- Here we go through the Loop ---?)

 keys = rem(3:12,10) + 1;

 for ii = 1:length(keys)

   kk = keys(ii);

   xx = [xx,zeros(1,400)];

   krow = ceil(kk/2);

   kcol = rem(kk-1,2) + 1;

   xx = [xx, cos(2\*pi\*ftable(krow,kcol)\*(0:1199)/fs) ];

 end

 soundsc(xx,fs)

The value of the ftable obtained is:

ftable =

80 110

160 220

240 330

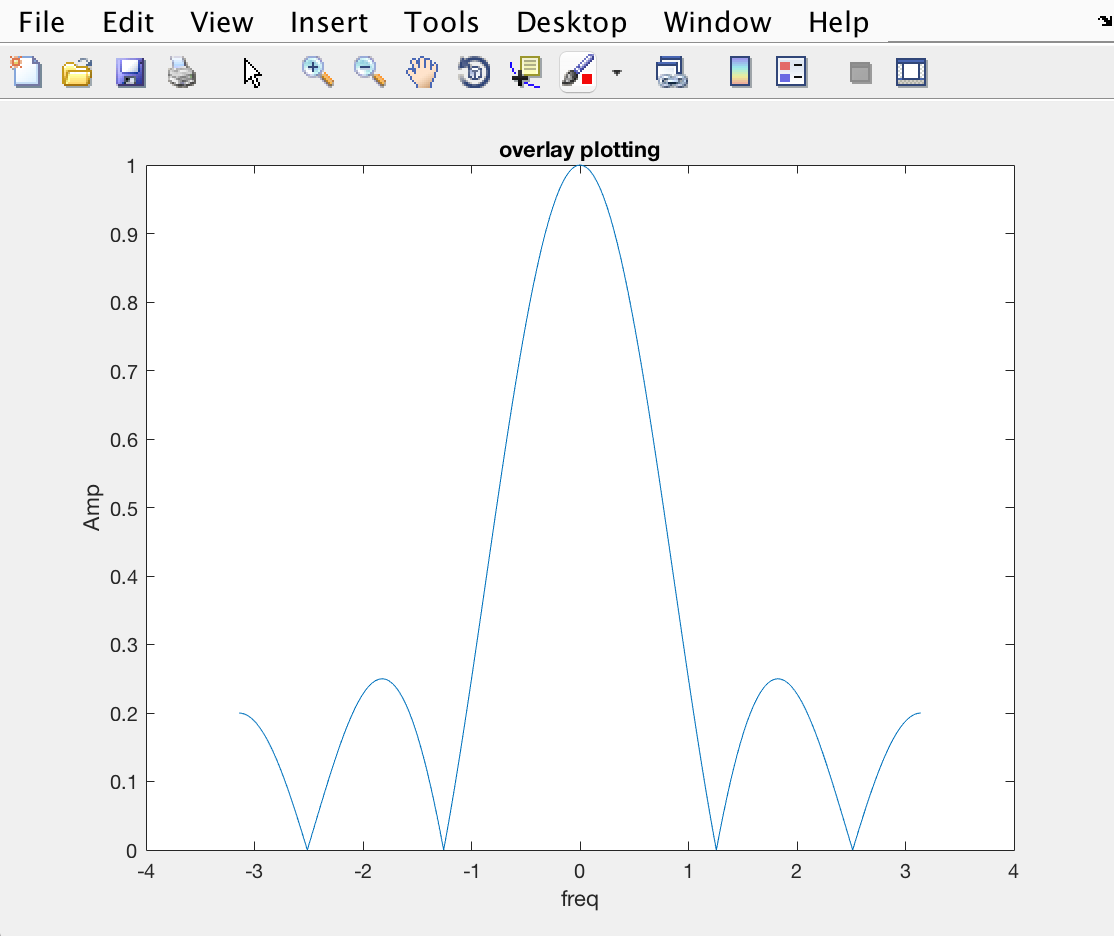
320 440

400 550

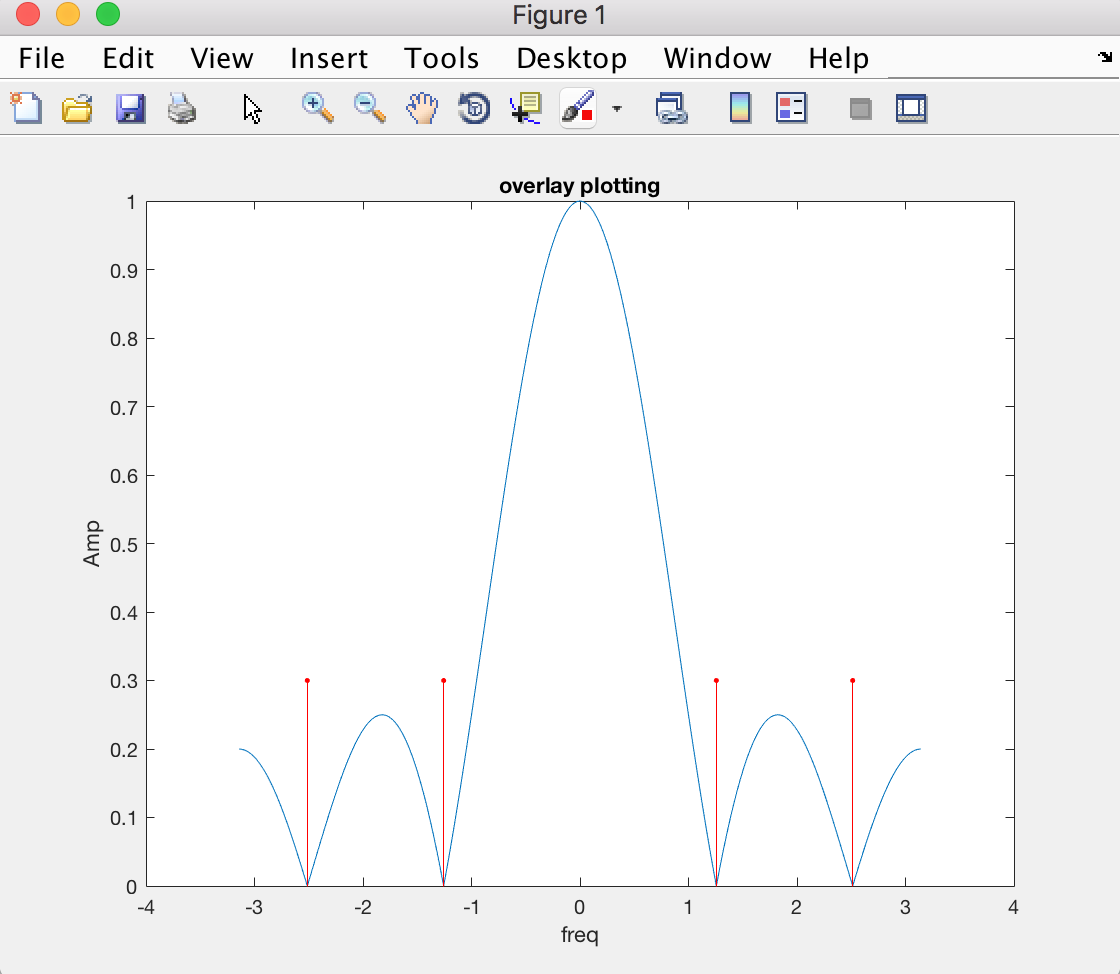
This program is somewhat inefficient even though the length of xx is small, and mostly unnoticeable, but due to allocation of large memory space.

2.2

a.



b.



3. *DTMF Synthesis*

3.1

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.

%

% Example function call

%  keyNames = ['5','1','5','2','3','B','\*'];

%  fs  = 8000;

%  xx = dtmfdial(keyNames,fs);

%  To verify performance after running this routine, use:

%  spectrogram(xx,'yaxis',256,128,256,fs)

%  This will show that the correct signals have been encoded

%dtmf.keys is a 4 by 4 array for the phone keys

% the column and row locations give frequency information

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 x= 1:length(keyNames)

    m=keyNames(x);

    [i,j]=find(m == dtmf.keys);

    xx=[xx,zeros(1,400)];

    row=dtmf.rowTones(i,j);

    col=dtmf.colTones(i,j);

    xx=[xx,cos(2\*pi\*row\*(0:1600)/fs)+cos(2\*pi\*col\*(0:1600)/fs)];

end

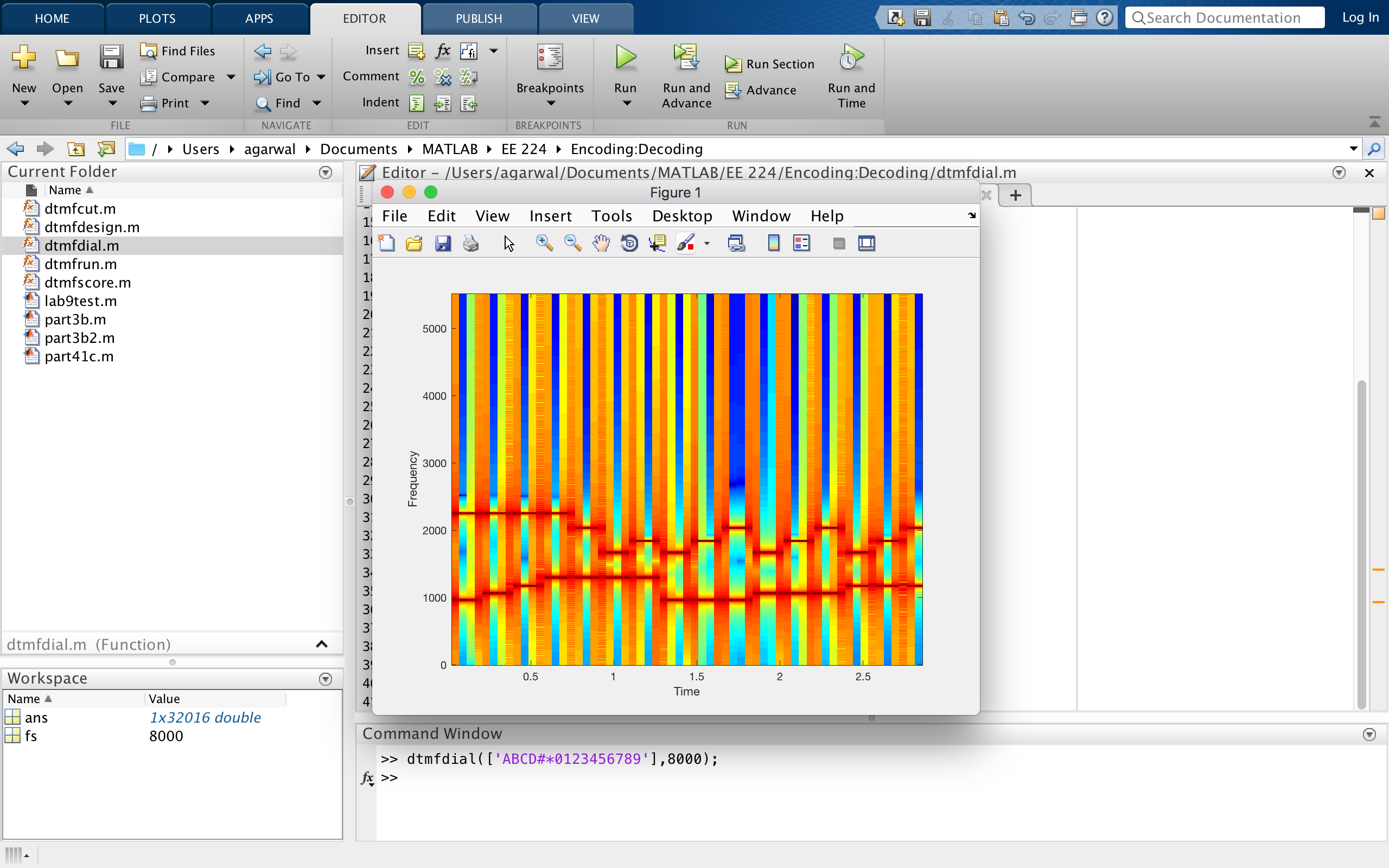
soundsc(xx,fs);

specgram(xx,1024,11025);

Using the command

>> dtmfdial ([‘ABCD#\*0123456789’],8000);

the following spectrogram is obtained:



3.2

1. .

x =0:(pi/1000):pi;

h=cos(0.2\*pi\*(0:50));

hh=freqz(h,1,x);

peak=max(abs(hh));

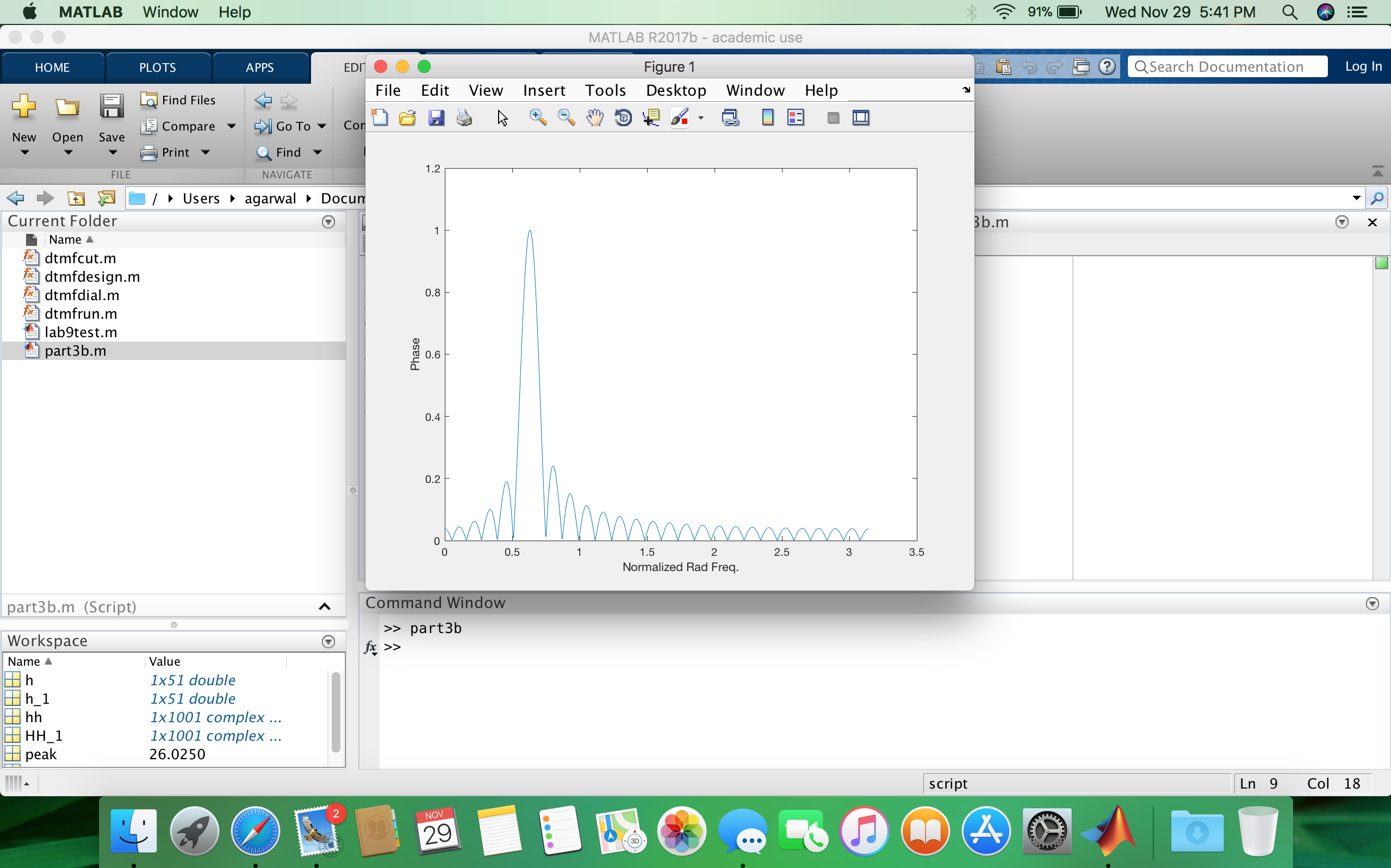
h\_1=1/peak\*cos(0.2\*pi\*(0:50));

HH\_1=freqz(h\_1,1,x);

plot(x,abs(HH\_1));

xlabel ('Normalized Rad Freq.');

ylabel ('Phase');



1. .

To find the width of the bandpass,

x =0:(pi/1000):pi;

h=cos(0.2\*pi\*(0:50));

hh=freqz(h,1,x);

peak=max(abs(hh));

h\_1=1/peak\*cos(0.2\*pi\*(0:50));

HH\_1=freqz(h\_1,1,x);

plot(x,abs(HH\_1));

xlabel ('Normalized Rad Freq.');

ylabel ('Phase');

bandpass=find(abs(HH\_1)>=0.707);

width1=x(bandpass(1));

width2=x(length(bandpass)+bandpass(1)-1);

width=width2-width1;

This gives us the width = 0.1037, with width1 = 0.5781, & width2 = 0.6817,

1. .

The range of the frequency component that would pass from this filter will be from,

0.5781\*8000 = 4624.8 rad/sec, to

0.6817\*8000 = 5453.6 rad/sec.

4. *DTMF Decoding*

4.1

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.

hh=[];

 for i = 1:length(fb)

     h=cos((2\*pi\*fb(i)\*(0:L-1))/fs);

     [H,W]=freqz(h);

     B=1/max(abs(H));

     h=B\*cos((2\*pi\*fb(i)\*(0:L-1))/fs);

        plot(W,abs(H),'color',[1-i/10 i/10 1]);

     hold on;

     hh=[hh,h'];

 end

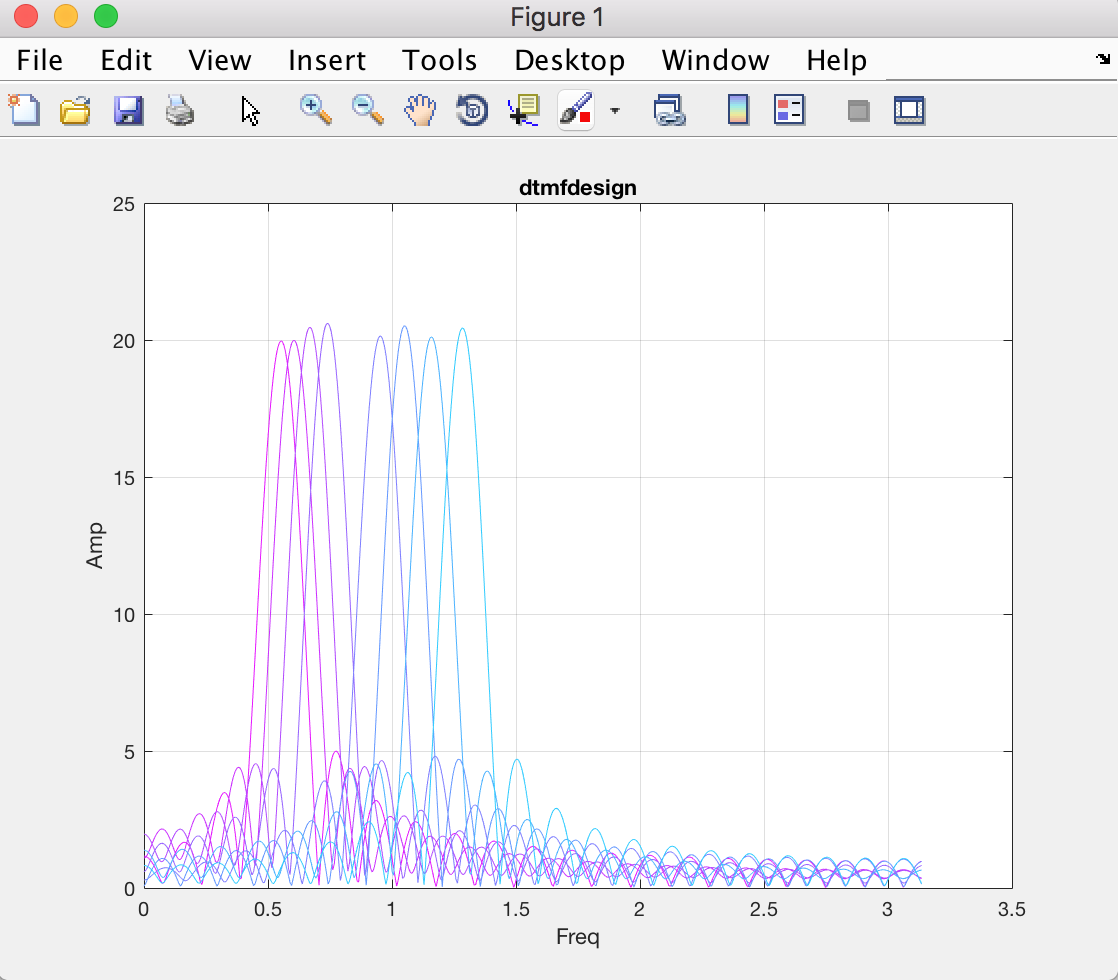
 title('dtmfdesign')

 xlabel('Freq');

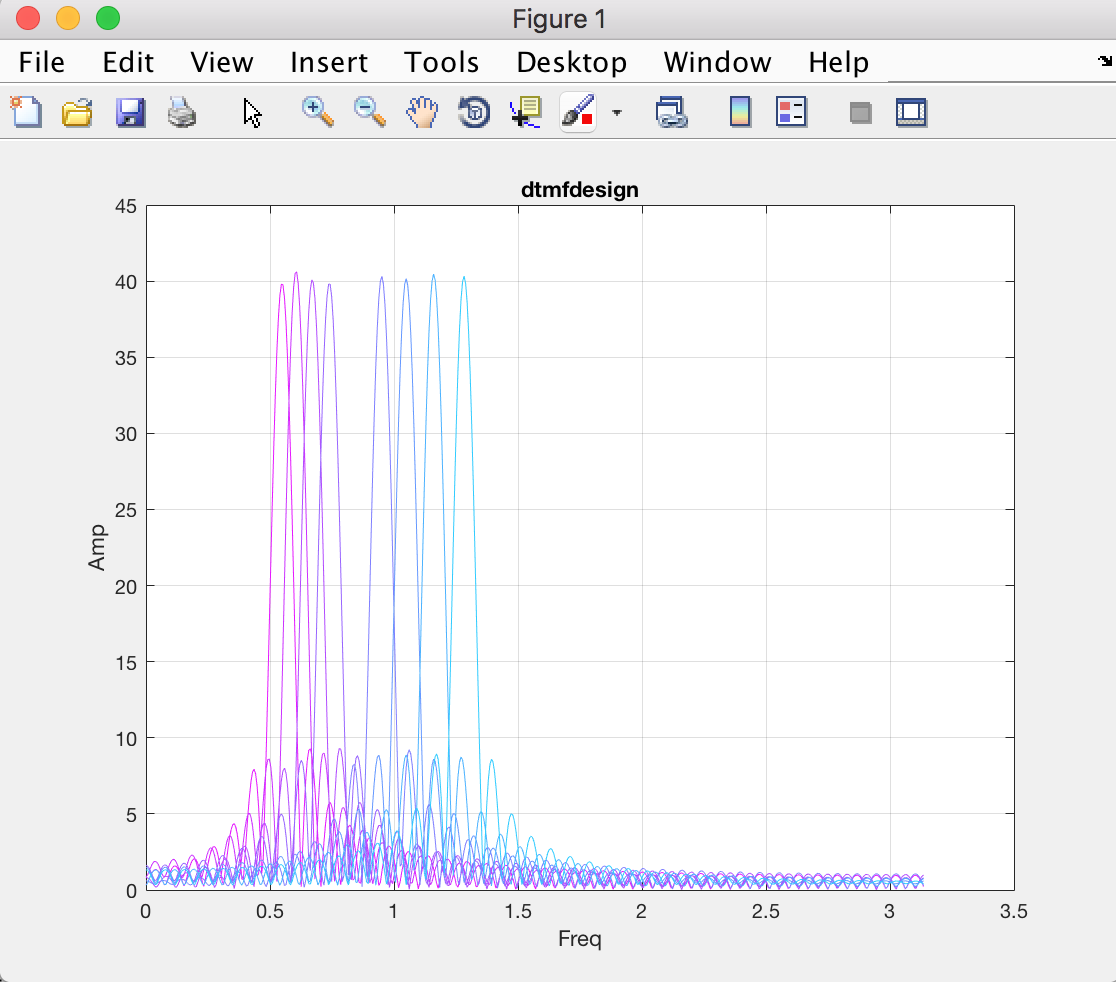
 ylabel('Amp')

 grid on

For L=40:



For L=80:



No, not each passband is narrow enough for only one frequency component to lie in.

The nearest corner frequency of 697 and 770 are hardest to meet the specs.

4.2.

function sc=dtmfscore(xx,hh)

xx = xx\*(2/max(abs(xx)));

Y = conv(xx,hh);

plot(abs(Y));

if (max(abs(Y)))>=0.59

    sc=1;

else

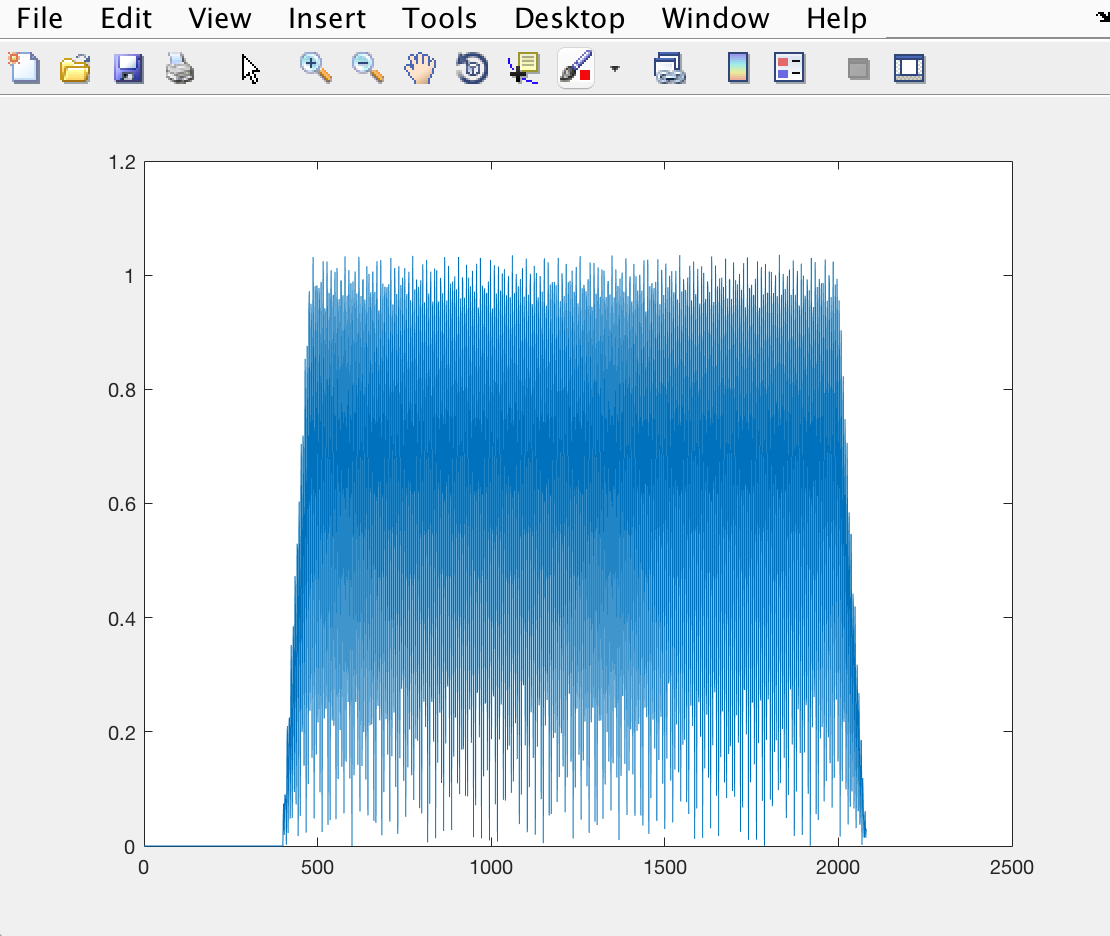
    sc=0;

end

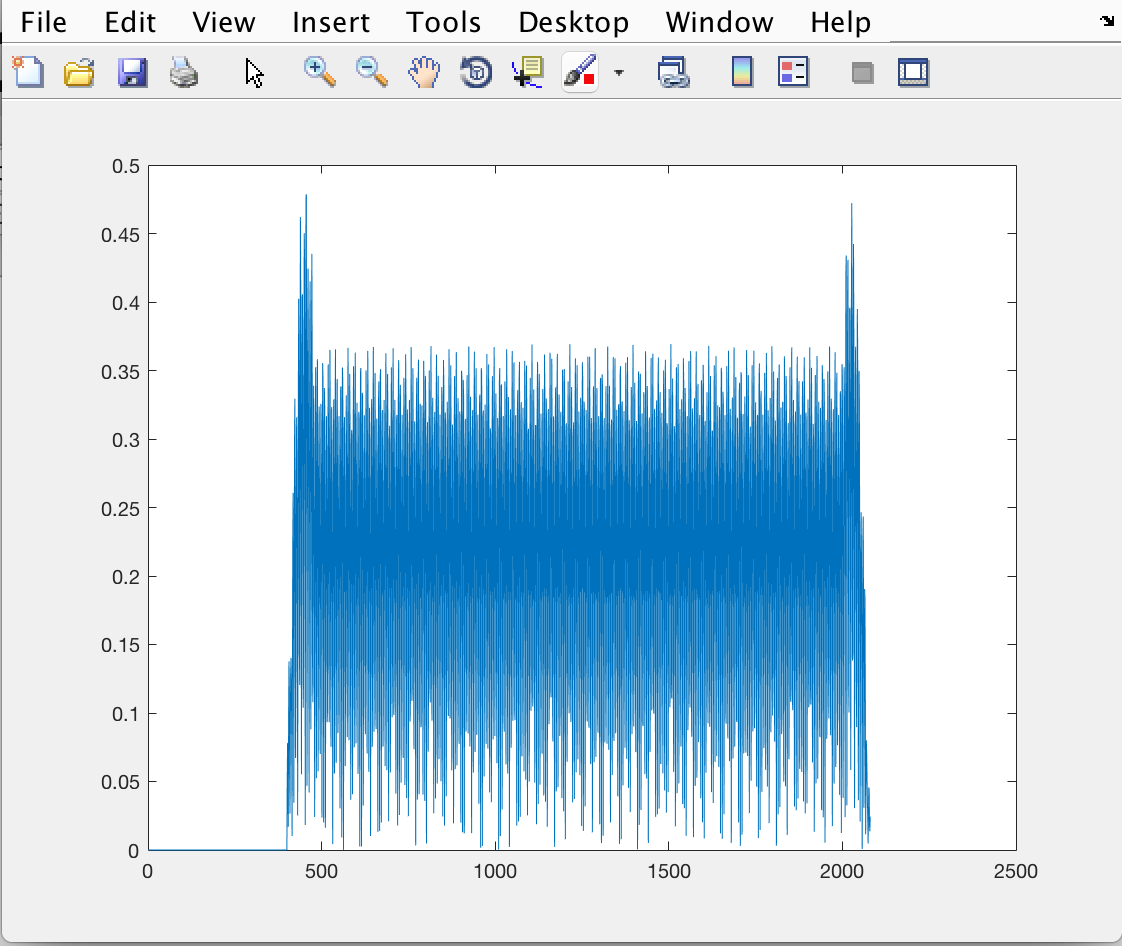
The maximum value for H(ejωˆ) should be 1, because to achieve a scoring threshold of 0.59.

After debugging through my score program, the following matched and mismatched graphs were obtained:

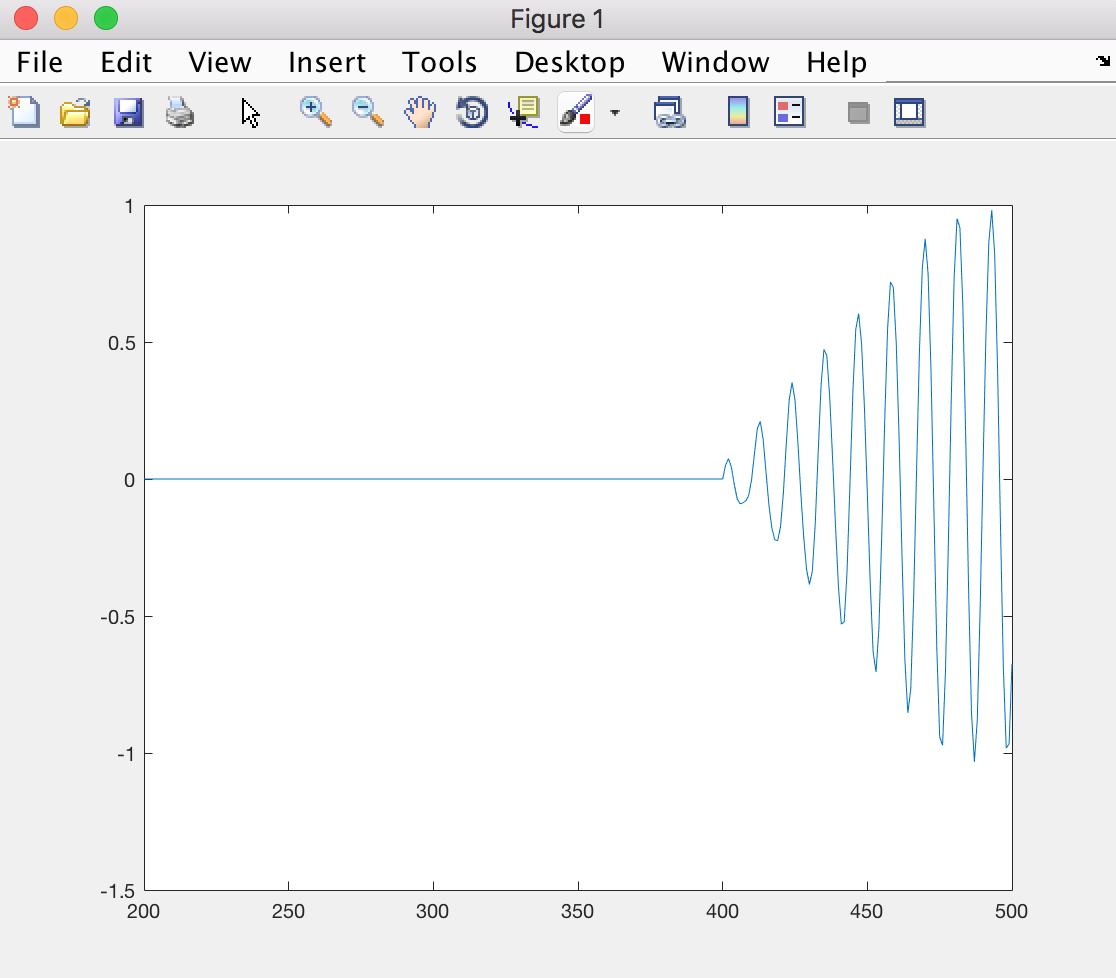
When ans = 1:



When ans = 0:



On 200-500 points scale:



4.3

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

%

% Inputs:

%   xx = DTMF waveform

%   L = filter length

%   fs = sampling freq

%

% Initialize key arrays and frequencies

dtmf.keys = ...

['1','2','3','A';

 '4','5','6','B';

 '7','8','9','C';

 '\*','0','#','D'];

center\_freqs = [697,770,852,941,1209,1336,1477,1633];

hh = dtmfdesign(center\_freqs,L,fs );

[nstart,nstop] = dtmfcut(xx,fs); %<--Find the beginning and end of tone bursts

keys = [];

M=zeros(1,8);

for kk=1:length(nstart)

    x\_seg = xx(nstart(kk):nstop(kk));

    for i=1:8

        M(i)=dtmfscore(x\_seg, hh(:,i));

    end

    ind=find(M==1);

    if (length(ind)>2)

       keys = [keys,'-1'];

    else

        row\_ind = find(M(1:4)==1);

        col\_ind = find(M(5:8)==1);

        keys = [keys,dtmf.keys(row\_ind,col\_ind)];

    end;

end

end

On the running the following command:

>> fs=8000;

>> tk=['4','0','7','\*','8','9','1','3','2','#','B','A','D','C'];

>> xx=dtmfdial(tk,fs);

>> soundsc(xx,fs)

>> L=80;

>> dtmfrun(xx,L,fs)

The following result was obtained:

ans =

'407\*89132#BADC'

The spectrogram of the signal from dtmfdial:

