Jordan Ditzler
11/27/2017

ECE 201-203
Philip Snehav Chakram, Negar Etemadyrad

Lab Report Project

Jordan Ditzler - G00967092

Jordan Ditzler
11/27/2017

ECE 201-203
Philip Snehav Chakram, Negar Etemadyrad

Introduction:

In this lab, we will be going into heavy detail about DTMF signals and how they relate to the key presses on a phone dial. We will be able, by the end of this lab, to encode a number into a tone from the keypad and decode by using filters to get the number that was dialed.

Main Body:

Part1:

3.1)

```
ftable = [1;2;3;4] * [100,175,230];
fs = 11025;
xx = [ ];
disp('--- Here we go through the loop ---')
keys = ceil(11.9*rand(1,10)+0.1); %--- Make a random test sequence
for ii = 1:length(keys)
kk = keys(ii);
xx = [xx,zeros(1,400)];
krow = ceil(kk/3);
kcol = rem(kk-1,3)+1;
disp(['kk=', num2str(kk),' krow=',num2str(krow),...
' kcol=', num2str(kcol), ' freq=', num2str(ftable(krow,kcol))])
xx = [xx, cos(2*pi*ftable(krow,kcol)/fs*(0:2499))];
end
soundsc(xx,fs)
```

b) 4x3 table

100	175	230
200	350	460
300	525	690
400	700	920

c) 29000 samples, 2.63 seconds

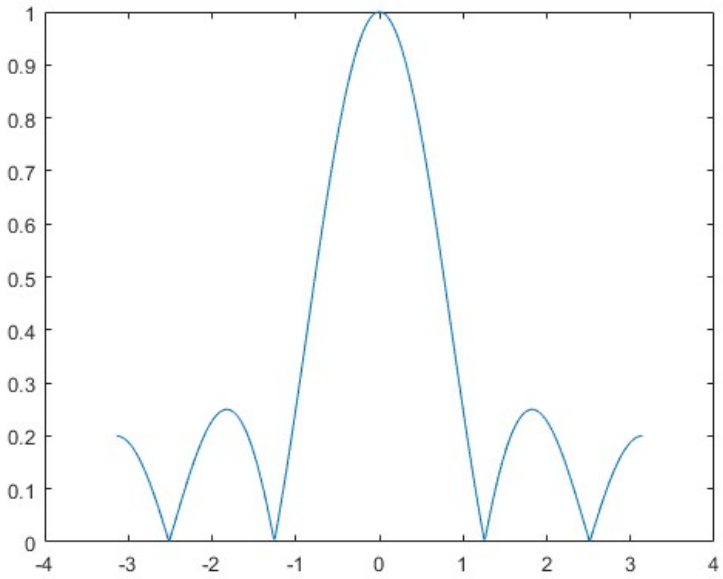
Jordan Ditzler
11/27/2017

ECE 201-203
Philip Snehav Chakram, Negar Etemadyrad

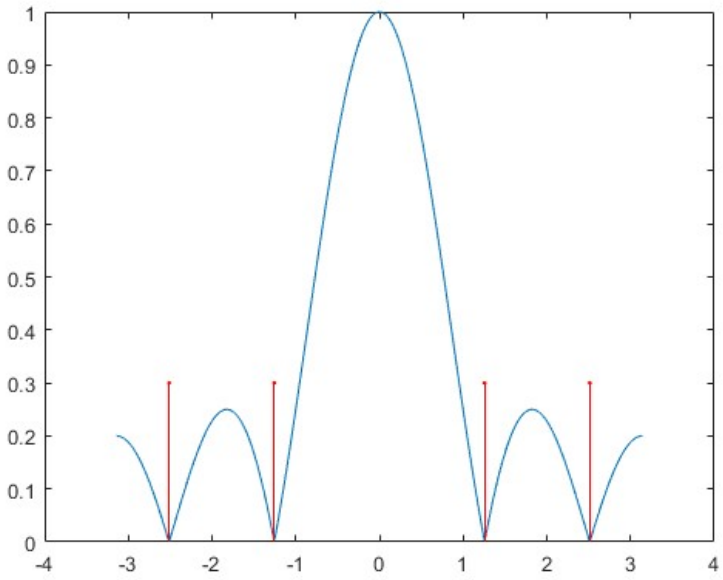
3.2)

```
ww = linspace(-pi,pi,400);
HH = freqz(ones(1,5)/5,1,ww);
plot(ww,abs(HH));
hold on, stem(2*pi/5*[-2,-1,1,2],0.3*ones(1,4),'r.'), hold off
```

a)



b)



3.3)

```
function dtmfsig = dtdial(nums,fs)
%DTMFDIAL Create a vector of tones which will dial
% a DTMF (Touch Tone) telephone system.
%
% usage: dtmfsig = dtmfdial(nums,fs)
% nums = vector of numbers ranging from 1 to 12
% fs = sampling frequency
% dtmfsig = vector containing the corresponding tones.
%
tone_cols = ones(4,1)*[1209,1336,1477];
tone_rows = [697;770;852;941]*ones(1,3);
% dur_tone = 0.15; % duration of tone
% dur_pause = 0.05; % duration of pause in between tones
% tt = 0:1/fs:dur_tone; % time vector
dtmfsig = [ ];
for ii = 1:length(nums) % go through each key
    kk = nums(ii); % gets num from input
    dtmfsig = [dtmfsig,zeros(1,501)];
    krow = ceil(kk/3);
    kcol = rem(kk-1,3)+1; % gets row/col position
    f1 = tone_rows(krow,kcol);
    f2 = tone_cols(krow,kcol); % gets tone from row/col position
    dtmfsig = [dtmfsig, cos(2*pi*f1/fs*(0:2499))+cos(2*pi*f2/fs*(0:2499))]; %
Combines tones
end
```

Jordan Ditzler
11/27/2017

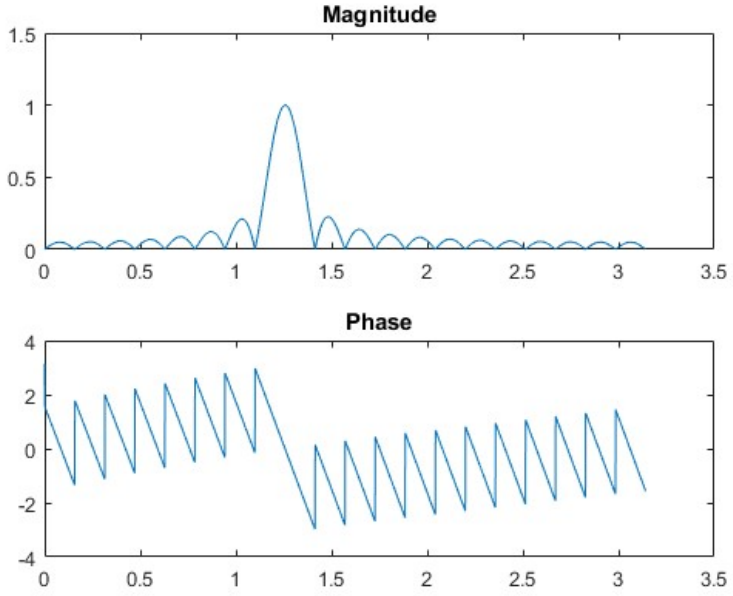
ECE 201-203
Philip Snehav Chakram, Negar Etemadyrad

Part2:

3.1)

a)

```
L = 40;
ll = 0:L-1;
ww = 0:1/1000:pi;
W = 0.4;
bb = (2/L)*cos(pi*W*ll);
H = freqz(bb,1,ww);
subplot(2,1,1);
plot(ww,abs(H));
title('Magnitude');
subplot(2,1,2);
plot(ww,angle(H));
title('Phase');
```



b)

k-values	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0
Magnitude	0	0	0	0	0	0	1	0	0	0	0
Phase	1.552	1.795	2.042	2.295	2.558	2.838	0	0.3151	0.6913	1.12	1.563
k-values	1	2	3	4	5	6	7	8	9	10	
Magnitude	0	0	0	1	0	0	0	0	0	0	
Phase	2.015	2.435	2.792	0	0.288	0.5676	0.8312	1.084	1.312	-1.567	

c)

L = 40 passband width = 0.1390.

L = 20 passband width = 0.2800.

L = 80 passband width = 0.0700.

When L is doubled or halved, the passband width is doubled or halved inversely.

d) In the graph, the filter pass components at $\omega = 0.4\pi$ because that's when the freqz() reaches the maximum magnitude of 1. At the other $\omega = 0.1\pi k$ for $k \neq \pm 4$, the magnitude is 0, therefore is rejected.

e)

$x[n] = 7 + 7\cos(0.4\pi n + \pi/2) + 7\cos(0.5\pi n + \pi/4)$

$y[n] = 7 + 7\cos(0.5\pi n + \pi/4)$

Jordan Ditzler
11/27/2017

ECE 201-203
Philip Snehav Chakram, Negar Etemadyrad

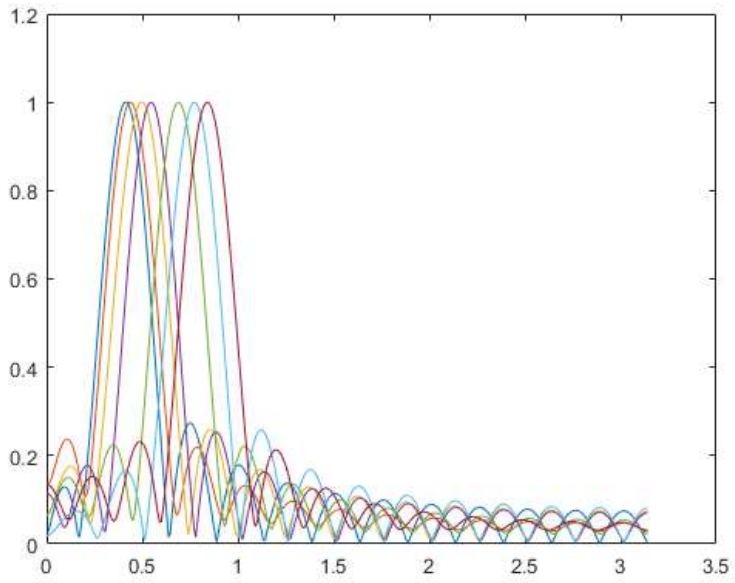
3.2.1)

```
function hh = dtdesign(fcent, L, fs)
%DTDESIGN
% hh = dtdesign(fcent, L, fs)
% returns a matrix (L by length(fcent)) where each column is the
% impulse response of a BPF, one for each frequency in fcent
% fcent = vector of center frequencies
% L = length of FIR bandpass filters
% fs = sampling freq
%
% The BPFs must be scaled so that the maximum magnitude
% of the frequency response is equal to one.
ll = 0:L-1; % length vector
ww = 0:pi/300:pi; % omega vector
for ii = 1:length(fcent)
    h1(ii,:) = cos(2*pi*fcent(ii)*ll/fs); % band pass filter
    bb(ii,:) = abs(1/(max(freqz(h1(ii,:),1,ww)))); % get bb scaler
coefficients
    h2(ii,:) = h1(ii,:)*bb(ii,:); % concatenate vectors
end
hh = h2'; % turn filters into matrix
```

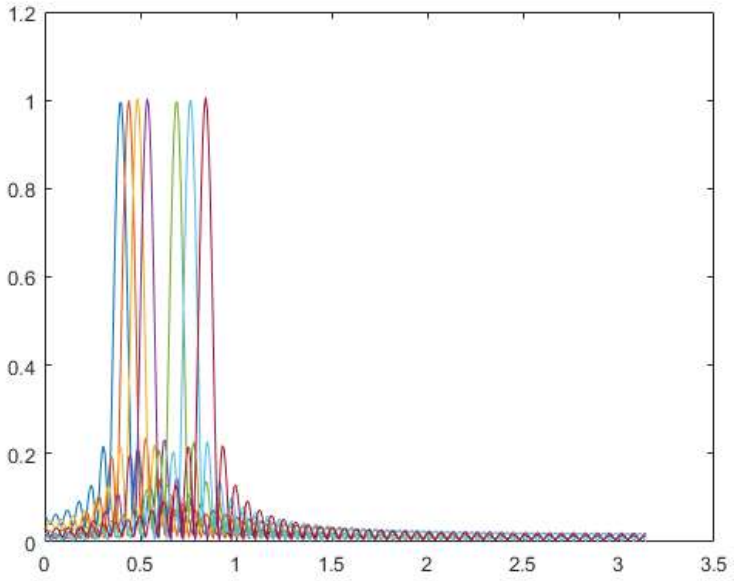
a) Since the max return of freqz() is 1, I will use that to get the constant β.

c)

```
hh = dtdesign([697 770 852 941 1209 1336 1477], 25, 11025);
ww = linspace(0,pi,400);
for ii = 1:length(hh)
    plot(ww,abs(freqz(hh(:,ii),1,ww)));
    hold on;
end
```



d)



The width of the passband varying inversely with the filter length L is true as demonstrated by the two plots. Plot c has a wider passband with a smaller L, while plot d has a smaller passband with a larger L.

e) The smallest value of L is 45.

3.2.2)

```
function sc = dtscore(xx, hh)
%DTSCORE
% usage: sc = dtscore(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 0.95
% sc = 0 if max(|y[n]|) is less than 0.95
%
xx = xx*(2/max(abs(xx))); %---Scale x[n] to the range [-2,+2]
yy = conv(xx,hh); % convolution of xx and hh
if max(yy(100:length(yy)-100)) >= 0.95
    sc = 1; % scores 1 if >= .95
else
    sc = 0; % or else it = 0
end
```

c) The maximum magnitude is 1 because of the same reason we chose the maximum value of β to be 1 in dtdesign().

3.3)

```
function keys = dtrun(xx,L,fs)
%DTRUN keys = dtrun(xx,L,fs)
% returns the list of key numbers found in xx.
% xx = DTMF waveform
% L = filter length
% fs = sampling freq
%
freqs = [697,770,852,941,1209,1336,1477];
hh = dtdesign( freqs,L,fs );
% hh = L by 7 MATRIX of all the filters. Each column contains the
% impulse response of one BPF (bandpass filter)
%
[nstart,nstop] = dtcut(xx,fs); %<--Find the tone bursts
keys = [ ];
temp_loc = [ ];
key_table = [1 2 3;4 5 6;7 8 9;10 11 12]; % table of keypad
for kk = 1:length(nstart) % for each tone
    x_seg = xx(nstart(kk):nstop(kk)); %<--Extract one DTMF tone
    temp_loc = [ ]; % resets location variable
    for jj = 1:length(freqs) % goes through each column
        temp_loc = [temp_loc,dtscore(x_seg,hh(:,jj))]; % tests for frequency
components
    end
    oo = find(temp_loc == 1); % finds where ones occur
    if length(oo) ~= 2 || oo(1) > 4 || oo(2) < 5 % Skips bad scores
        continue
    end
    row = oo(1);
    col = oo(2)-4; % gets row/col position
    key_num = key_table(row,col);
    keys = [keys,key_num]; % sets current key to a key from the key_table
end
```

Conclusion:

We successfully encoded and decodes a DTMF signal from a generic phone using MATLAB and its functions. We learned about how to use and implement a bandpass filter as well as gaining knowledge in MATLAB in-depth coding.