Table of Contents


```
%{
Yonatan Carver
ECES 352 - Lab 9

%}
clear; clc; close all

keynames =
 ['8','9','1','#','4','0','7','8','9','1','3','2','*','D','A','B','C'];
fs = 10000;
xx = dtmfdial(keynames, fs);
% soundsc(xx)

%{
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
% OUTPUT
%  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);

dtmf.Tones = dtmf.rowTones + dtmf.colTones;

% make placeholder that is length of keynames with padded zeros in
 between
length_sound = 2000; % duration of each tone pair: 0.20 seconds (2000
 zeros)
length_silence = 500; % duration of silence: 0.05 seconds (500)
% xx = zeros(1, length(keynames)*length_sound +
 length(keynames)*length_silence);
```

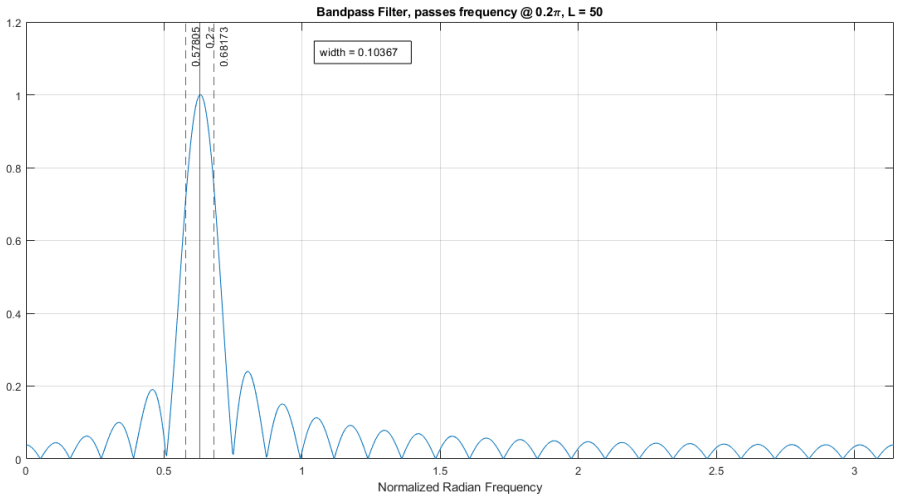
```
xx=[];

for i = 1:length(keynames)
 kk = keynames(i);
 xx = [xx, zeros(1,length_silence)];
 [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:length_sound)/fs) +
 cos(2*pi*col_f*(0:length_sound)/fs)];
end

% specgram(xx, 1024, 11025);

end

%}
```

(3.2) Simple Bandpass Filter Design

(a)

```
n = 0 : (pi/1000) : pi;
wc = 0.2*pi;    % center frequency
hh = cos(wc .* (0:50));  % bandpass equation
HH = freqz(hh,1,n);   % frequency response of digital filter
peak_v = max(abs(HH));  % peak value
h_1 = 1/peak_v * cos(wc .* (0:50));  % frequency response @ peak value
HH_1 = freqz(h_1,1,n);     % peak value = 1


figure('Name', 'Bandpass
 filter', 'units', 'normalized', 'outerposition', [0 0.04 1 0.96])
plot(n, abs(HH_1));
grid on
xlabel('Normalized Radian Frequency')
title('Bandpass Filter, passes frequency @ 0.2\pi, L = 50')

xline(0.2*pi, 'Label', '0.2\pi'); % line to show center frequency
xlim([0 pi])

% (b)
passband = find(abs(HH_1) >= 0.707);
upper_3db = n(passband(1)); % upper 3dB frequency
lower_3db = n(length(passband) + passband(1) - 1); % lower 3dB
 frequency
width = lower_3db - upper_3db;  % width of bandpass filter

% lines to show the two 3dB frequencies
xline(upper_3db, 'Label', num2str(upper_3db), 'LineStyle', '--');
xline(lower_3db, 'Label', num2str(lower_3db), 'LineStyle', '--');

annotation('textbox',...
```

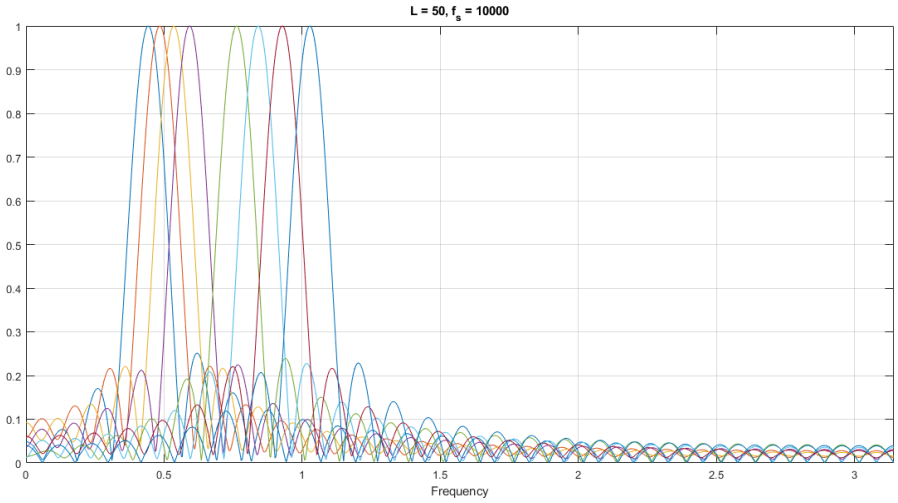
```
[0.387666666666667 0.813664596273292 0.104925925925926
0.0760869565217391],...
'String',{['width = ', num2str(width)]},...
'FitBoxToText','on');
```

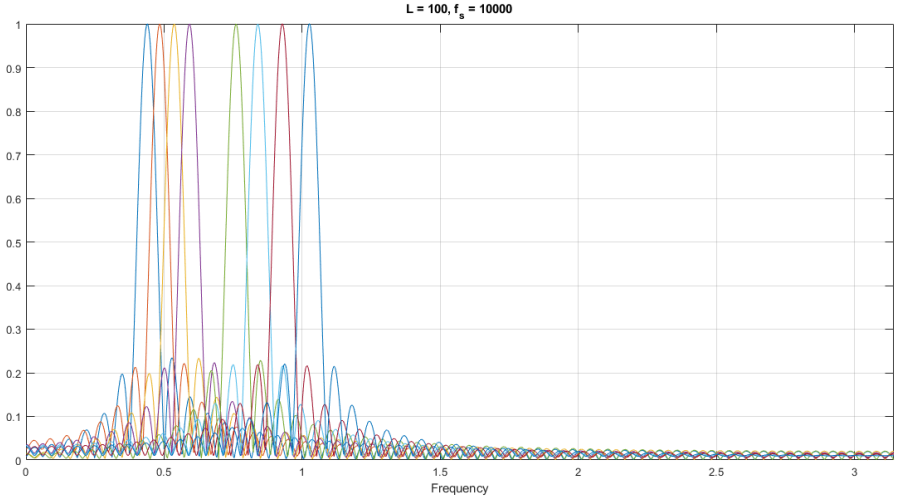


(4) Lab: DTMF Decoding

```
% 4.1 (c)

%{
function hh = dtmfdesign(fcent, L, fs)
% DTMFDESIGN
% hh = dtmfdesign(fcent, L, fs)
% returns a matrix (L by length(fcent)) where each column contains the
 impulse
% response of a BPF, one for each frequency in fcent
% fcent = vector of center frequencyes
% L = length of FIR bandpass filters
% fs = sampling frequency

% Each BPF must be scaled so that its frequency response has a maximum
 magnitude
% to one

% n = 1:L-1;
% hh = (2 ./ fcent) .* cos((2 .* pi .* fcent .* n) );
% ./ fs

n = 0 : pi/1000 : pi;
hh = zeros(L, length(fcent));
beta = zeros(1, length(fcent));

for i = 1:length(fcent)
 h = cos(2*pi*fcent(i) .* (0:L-1)/fs); % bandpass equation
 HH = freqz(h, 1, n);  % freq response
```

```
 beta(i) = 1/max(abs(HH)); % get beta value
 hh_1 = h .* beta(i);  % freq response
 hh(:,i) = hh_1;    % set column to hh_1
end

%}

ww_d = 0 : pi/1000 : pi;
for L = 40:100
 hh = dtmfdesign([697, 770], L, fs);
 h_1 = hh(:, 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

% 4.1 (d)
figure('Name', 'L = 50, f_s =
 10000', 'units', 'normalized', 'outerposition', [0 0.04 1 0.96])
L = 50;
for i = 1:8
 hh = dtmfdesign([697, 770, 852, 941, 1209, 1336, 1477, 1633], L, fs);
 h_d = hh(:,i);     % get each column of hh
 HH_d = freqz(h_d, 1, ww_d);  % frequency response
 plot(n, abs(HH_d));
 title('L = 50, f_s = 10000')
 xlabel('Frequency')
 ylim([0 1])
 xlim([0 pi])
 hold on; grid on
end

% 4.1 (e)
figure('Name', 'L = 100, f_s =
 10000', 'units', 'normalized', 'outerposition', [0 0.04 1 0.96])
L = 100;
for i = 1:8
 hh = dtmfdesign([697, 770, 852, 941, 1209, 1336, 1477, 1633], L, fs);
 h_e = hh(:,i);     % get each column of hh
 HH_e = freqz(h_e, 1, ww_d);  % frequency response
 plot(n, abs(HH_e))
 title('L = 100, f_s = 10000')
 xlabel('Frequency')
 xlim([0 pi])
 hold on; grid on
end
```





4.2 A Scoring Function: dtmfscore.m

```
%{
function sc = dtmfscore(xx, hh)
 xx = xx * (2/max(abs(xx)));
 yy = conv(xx, hh);
 if (max(abs(yy))) >= 0.71
  sc = 1;
 else
  sc = 0;
 end

%  ww = 200:500;
%  figure
%  plot(ww, yy(200:500));
end

%}
```

```
hh_sc = dtmfdesign(697, 50, fs);
xx = dtmfdial('3', 10000);
sc = dtmfscore(xx, hh_sc);

%{
function keys = dtmfrun(xx, L, fs)

 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);
 [start_loc, stop_loc] = dtmfcut(xx, fs); % beginning & end of sounds

 keys = [];
 sc = zeros(1,8);  % placeholder for score values
 for i = 1:length(start_loc)
  x_seg = xx(start_loc(i):stop_loc(i)); % segment of sound
  for j = 1:8
   sc(j) = dtmfscore(x_seg, hh(:,j));
  end
  num_freq = find(sc == 1);
  if length(num_freq) > 2
   keys = [keys, '-1'];
  else
   row_index = find(sc(1:4) == 1);  % if positive match
   col_index = find(sc(5:8) == 1);  % if positive match
   keys = [keys, dtmf.keys(row_index, col_index)];
  end
 end
end

%}

% fs = 10000;
keynames =
 ['8','9','1','#','4','0','7','8','9','1','3','2','*','D','A','B','C'];
xx = dtmfdial(keynames, fs);
L = 100;
message_sent = keynames
decoded = dtmfrun(xx, L, fs) % decoded sequence

disp('Sent message == decoded')
disp(message_sent == decoded)


message_sent =

    '891#40789132*DABC'
```

```
decoded =

    '891#40789132*DABC'

Sent message == decoded
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

Published with MATLAB® R2018b