Lab 11 – Encoding and Decoding Touch-Tone (DTMF) Signals[*]
EE252L: Signals and Systems Lab

Dr. Basit Memon and Ms. Hira Mustafa

Name: ____

Objectives

By the end of this lab, you will be able to encode and decode the DTMF signals on MATLAB.

Instructions: After completion of each task get it marked by sharing your results and code with any of the RA, your code must be self-explanatory i.e. use proper comments. *Attach rubrics at the end of your report.*

1 Introduction

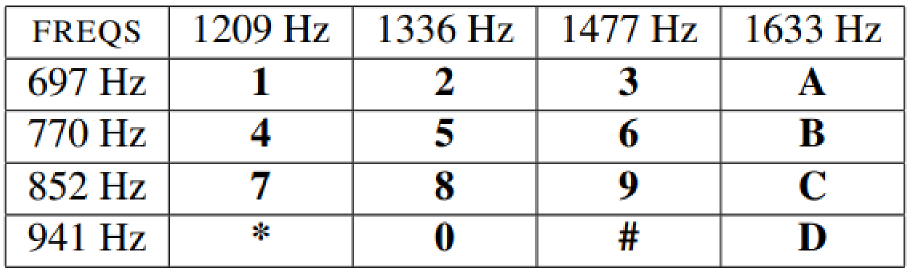
This lab introduces a practical application where sinusoidal signals are used to transmit information: a touch-tone dialer. Bandpass FIR filters can be used to extract the information encoded in the waveforms. The goal of this lab is to design and implement bandpass FIR filters in MATLAB, and to do the decoding automatically. In the experiments of this lab, you will use `firfilt()`, or `conv()`, to implement filters and `freqz()` to obtain the filter's frequency response.1 As a result, you should learn how to characterize a filter by knowing how it reacts to different frequency components in the input.

1.1 Background: Telephone Touch Tone Dialing

Telephone touch-tone[1] 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 (in Fig. 1) are generated and summed, hence dual tone. As an example, pressing the 5 key generates a signal containing the sum of the two tones at 770 Hz and 1336 Hz together.
The frequencies in Figure 1 were chosen (by the design engineers) to avoid harmonics. No

[*]Adapted from DSP First, 2e
[1]Touch Tone is a registered trademark

frequency is an integer multiple of another, the difference between any two frequencies does not equal any of the other frequencies, and the sum of any two frequencies does not equal any of the other frequencies. This makes it easier to detect exactly which tones are present in the dialed 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. When any key is pressed the tones of the corresponding column and row are generated and summed. Keys A-D (in the fourth column) are not implemented on commercial and household telephone sets, but are used in some military and other signaling applications.

1.2 DTMF Decoding

There are several steps to decoding a DTMF signal:

1. Divide the time signal into short time segments representing individual key presses.
2. Filter the individual segments to extract the possible frequency components. Bandpass filters can be used to isolate the sinusoidal components.
3. Determine which two frequency components are present in each time segment by measuring the size of the output signal from all of the bandpass filters.
4. Determine which key was pressed, 0–9, A–D, *, or # by converting frequency pairs back into key names according to Figure 1.

It is possible to decode DTMF signals using a simple FIR filter bank. The filter bank in Figure 2 consists of eight band-pass filters, such that each of them passes through only one of the eight possible DTMF frequencies. The input signal for all the filters is the same DTMF signal.

Here is how the system should work: When the input to the filter bank is a DTMF signal, the outputs from two of the bandpass filters (BPFs) should be larger than the rest. If we detect (or measure) 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 key from the DTMF code. A good measure of the output levels is the peak value at the filter outputs, because when the BPF is working properly it should pass only one sinusoidal signal and the peak value would be the amplitude of the sinusoid passed by the filter. More discussion of the detection problem can be found in the last task.

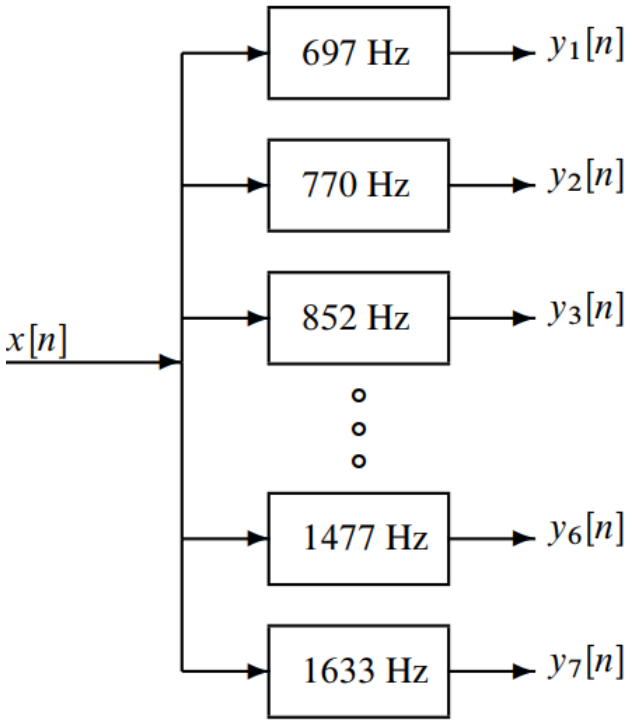


Figure 2: Filter bank consisting of bandpass filters (BPFs) which pass frequencies corresponding to the eight DTMF component frequencies listed in Figure 1. The number in each box is the center frequency of the BPF.

1.3 MATLAB Implementation

The DTMF system consists of functions to generate DTMF signals and functions to decode them. Some of the necessary functions have been implemented, while others will be constructed as part of this lab. A summary of the necessary files is as follows:

- `dtmfdial.m` – This function implements a DTMF dialer based on the frequency table defined in Figure 1. This function generates 0.05 seconds of silence followed by 0.2 seconds of tone for each key given in a text string argument. For instance, to generate the tones corresponding to the numbers 123, after writing your function store the data in the vector xx, and listen to the result, type:

  ```
  keyNames=['1' '2' '3'];
  [xx]=dtmfdial(keyNames, 8000);
  sound(xx, 8000);
  ```

  Any length text string can be given as the input.

- `dtmfcut.m` – This function parses a DTMF sound file, indicating the start and stop indexes of each individual set of tone bursts. The function is called as `[nstart,nstop]=dtmfcut(xx,fs)`, where `nstart` and `nstop` are vectors containing the start and stop indexes of each DTMF burst. For instance, to place all the DTMF segments in individual vectors, you can use:

```
[nstart,nstop] = dtmfcut(xx,fs); % Find the beginning and end of tone bursts
for kk=1:length(nstart)
indx = [(nstart(kk):nstop(kk)]; % seg. indices
x_seg(kk,[1:length(indx)])=xx(indx); %Extract one DTMF tone
end
```

This function is implemented and can be downloaded from LMS.

- `dtmfdesign.m` – This function computes the impulse responses for all of the bandpass filters shown in Figure 2. The impulse responses will be used to filter the DTMF signal and isolate individual sinusoids. This function must be constructed as part of this lab. More details on the arguments and structure of this function are given below.
- `dtmfscore.m` – This function in will be used to indicate if a specific sinusoid is present in a DTMF signal segment. Given a DTMF segment and the impulse responses for a specific bandpass filter, it will return a binary decision indicating if there is a sinusoid, with frequency equal to the center frequency of the bandpass filter defined by the impulse response, present in the DTMF segment. This function must be constructed as part of this lab. More details on the arguments and structure of this function are given below.
- `dtmfrun.m` – This final function returns a text string corresponding to the keys represented in the DTMF signal. It will utilize the other provided and written functions. This function must be constructed as part of this lab. As with the other functions you must construct, more details on the arguments and structure of the function are given below.

Task 1- Warm-up: Encoding from a Table

1. Explain how the following program uses frequency information stored in a table to generate a long signal via concatenation.
2. Determine the size of the table and all of its entries, and then state the playing order of the frequencies.
3. Determine the total length of the signal played by the soundsc function. How many samples and how many seconds?

```
ftable = [1;2;3;4;5]*[80,110]
fs = 8000; xx = [ ];
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;
dur=0.15;
t=0:1/fs:dur;
xx = [xx, cos(2*pi*ftable(krow,kcol)*t)];
end
soundsc(xx,fs);
```

Tip: For better understanding run the code line by line and see the results on command window.

2 Creating Function in MATLAB

To learn how to write funtion in MATLAB, refer 'Creating funtion' document on LMS.

3 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);
%This is an incomplete function
%----Write your code here-----
```

Given above is the skeleton of dtmfdial.m, a DTMF phone dialer.

Task 2-DTMF Dial Function

Write a function, `dtmfdial.m`, to implement a DTMF dialer based on the frequency table defined in Figure 1. A skeleton of `dtmfdial.m` is given above. In this warm-up, you must complete the dialing code with additional lines of code, so that it implements the following:

1. The input to the function is a vector of characters, each one being equal to one of the key names on the telephone. The MATLAB structure called dtmf contains the key names in the field `dtmf.keys` which is a 4x4 array that corresponds exactly to the keyboard layout in Fig. 1.

2. The output should be a vector of samples with sampling rate `fs=` 8000 Hz containing the DTMF tones, one tone pair per key. Remember that each DTMF signal is the sum of a pair of (equal amplitude) sinusoidal signals. The duration of each tone pair should be exactly 0.20 sec., and a silence, about 0.05 sec. long, should separate the DTMF tone pairs. These times can be declared as fixed code in dtmfdial. (You do not need to make them variable in your function.)

3. The frequency information is given as two 4x4 matrices (`dtmf.colTones and dtmf.rowTones`): one contains the column frequencies, the other has the row frequencies. You can translate a key such as the 6 key into the correct location in these 4x4 matrices by using MATLAB's find function. For example, the key 6 is in row 2 and column 3, so we would generate sinusoids with frequencies equal to `dtmf.colTones(2,3) and dtmf.rowTones(2,3)`. To convert an key name to its corresponding row-column indices, consider the following example:

   ```
   [ii,jj] = find('3'==dtmf.keys).
   ```

   Also, consult the MATLAB code in Task 1 above and modify it for the 4x4 tables in dtmfdial.m.

4 Band-pass Filter

A band pass filter allows frequencies within a specific frequency range and rejects (attenuates) frequencies outside that range.Its passband width is controlled by L, being inversely proportional to L. It is also possible to create a filter whose passband is centered around some frequency other than zero. One simple way to do this is to define the impulse response of an L-point FIR as:

$$h[n] = \beta cos(\omega_c n), 0 \leq n < L$$

where L is the length of the filter and ω_c is the center frequency for the pass-band. For example, we pick $\omega_c = 0.2\pi$ if we want the peak of the filter's passband to be centered at 0.2π Also, it is possible to choose β so that the maximum value of the frequency response magnitude will be one. The bandwidth of the bandpass filter is controlled by L; the larger the value of L, the narrower the bandwidth. You are familiar with its designing as done previously.

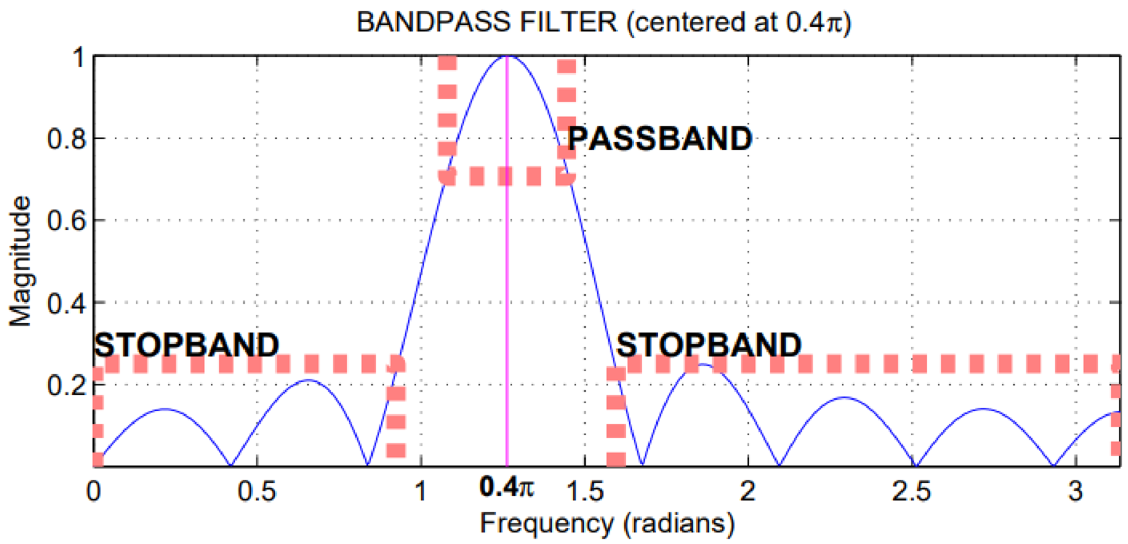


Figure 3: The frequency response of an FIR bandpass is shown with its passband and stopband regions.

Task 3: Band pass filter

A simple bandpass filter design method is presented in which the impulse response of the FIR filter is simply a finite length cosine of the form:

$$h[n] = \beta cos\left(\frac{2\pi f_b n}{f_s}\right), 0 \leq n \leq L-1$$

where L is the filter length, and f_s is the sampling frequency. The constant β 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. The parameter fb defines the frequency location of the passband, e.g., we pick $f_b = 852$ if we want to isolate the 852 Hz component. The bandwidth of the bandpass filter is controlled by L; the larger the value of L, the narrower the bandwidth.

1. Write a function `B=maxbeta(L,fb,fs)` in which you will devise a MATLAB strategy for picking the constant β so that the maximum value of the **frequency response** will be equal to one. Write the one or two lines of MATLAB code that will do this scaling operation in general.
   Hint: Let MATLAB measure the peak value of the unscaled frequency response (without β), and then have MATLAB compute β to scale the peak to be one.
2. Test your function for $f_b = 852$, $L = 51$ and $f_s = 8000$.

5 DTMF Decoding

A DTMF decoding system needs two pieces: a set of bandpass filters (BPF) to isolate individual frequency components, and a detector to determine whether or not a given component is present. The detector must "score" each BPF output and determine which two frequencies are most likely to be contained in the DTMF tone. In a practical system where noise and interference are also present, this scoring process is a crucial part of the system design, but we will only work with noise-free signals to understand the basic functionality in the decoding system.

To make the whole system work, you will have to write three M-files: `dtmfrun`, `dtmfscore` and `dtmfdesign`. An additional M-file called `dtmfcut` is contained in the ZIP archive Lab11. The main M-file should be named `dtmfrun.m`. It will call `dtmfdesign.m`, `dtmfcut.m`, and `dtmfscore.m`.

Following is the skeleton of `dtmfdesign.m` function.

```
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.

%This is an incomplete function
%----Write your code here-----
```

Task 4: Simple Bandpass Filter Design: dtmfdesign.m

1. Using the skeleton of the `dtmfdesign.m` function provided above, complete this function with additional lines of code. This function should produce all eight(8) bandpass filters needed for the DTMF filter bank system. Store the filters in the columns of the matrix hh whose size is Lx8. Here you will use the `maxbeta` function already created in task 3, which gives the maximum value of the frequency response of a filter to be equal to one. Implementation is to be done for all the 8 frequencies.

2. The rest of this section describes how you can exhibit that you have designed a correct set of BPFs. In particular, you should justify how to choose L, the length of the filters. When you have completed your filter design function, you should run the $L = 40$ and $L = 80$ cases, and then you should determine empirically the minimum length L so that the frequency response will satisfy the specifications on passband width and stopband rejection. Such that the frequencies do not overlap one another.

Task 4: continues

3. Generate the eight (scaled) bandpass filters with $L = 40$ and $f_s = 8000Hz$. Plot the magnitude of the frequency responses all together on one plot (the range $0 \leq \omega \leq \pi$) is sufficient because $H(e^{j\omega})$ is symmetric).

   Indicate the locations of each of the eight DTMF frequencies (697, 770, 852, 941, 1209, 1336, 1477, and 1633 Hz) on this plot to illustrate whether or not the passbands are narrow enough to separate the DTMF frequency components.

   The expected graph is to have all the 8 frequencies in it just like an example plot shown in Figure 4.

   Hint: For instance in the case of a simple 50-point averaging filter these commands can be used as follows to indicate the frequencies of the zeros of the filter:

```
L=50;
ww=0:pi/L:pi
bb=ones(1,50)/50; %a 5 point average filter
HH=freqz(bb,1,ww)
plot(ww,abs(HH))
%This is just an example this is not the exact code
```

4. Repeat the previous part with L = 80 and fs=8000Hz. The width of the passband is supposed to vary inversely with the filter length L. Explain whether or not that is true by comparing the length 80 and length 40 cases.

5. Comment on the selectivity of the bandpass filters, i.e., use the frequency response to explain how the filter passes one component while rejecting the others. Is each filter's passband narrow enough so that only one frequency component lies in the passband and the others are in the stopband? Since the same value of L is used for all the filters, which filter drives the problem?
   In other words, for which center frequency is it hardest to meet the specifications for the chosen value of L?

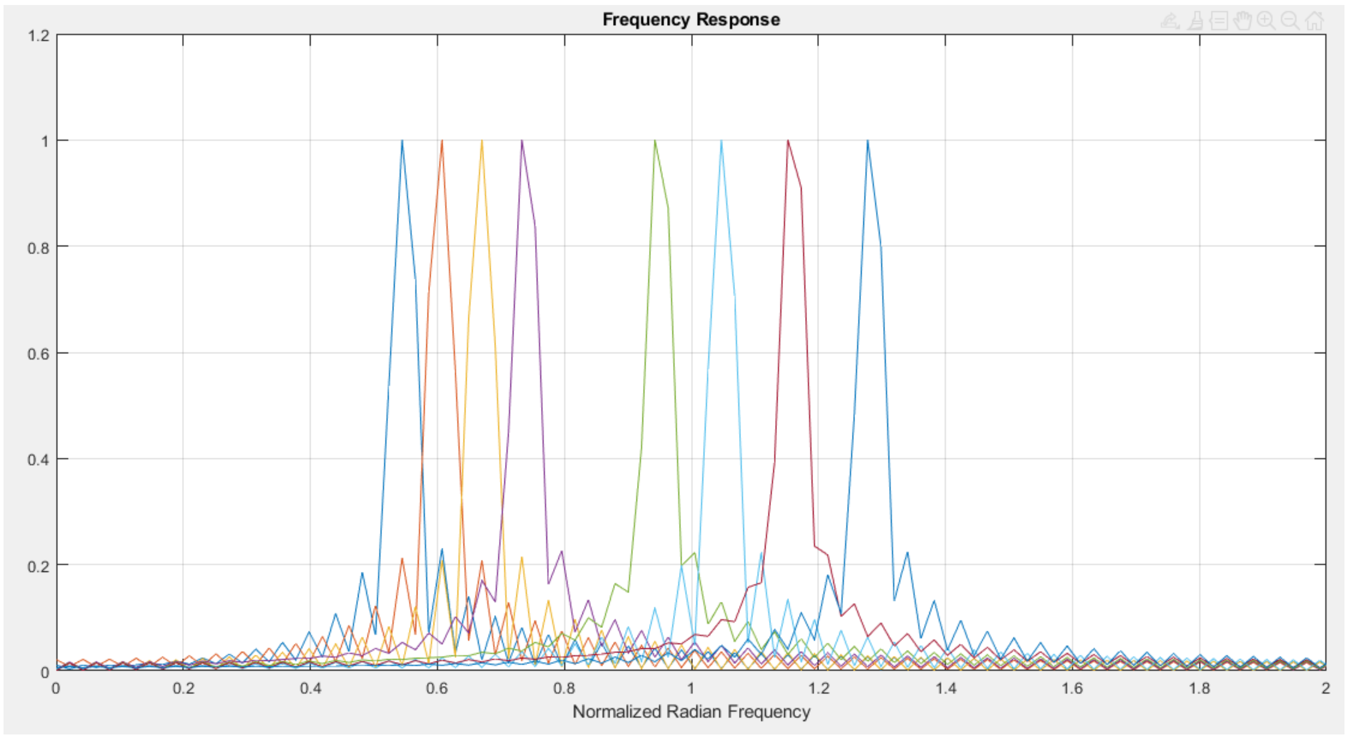


Figure 4: The frequency response of set of bandpass filters is shown.