Joseph Veal

Section B

4/12/17

Lab 10: Encoding and Decoding Tone Touch Signals

Introduction

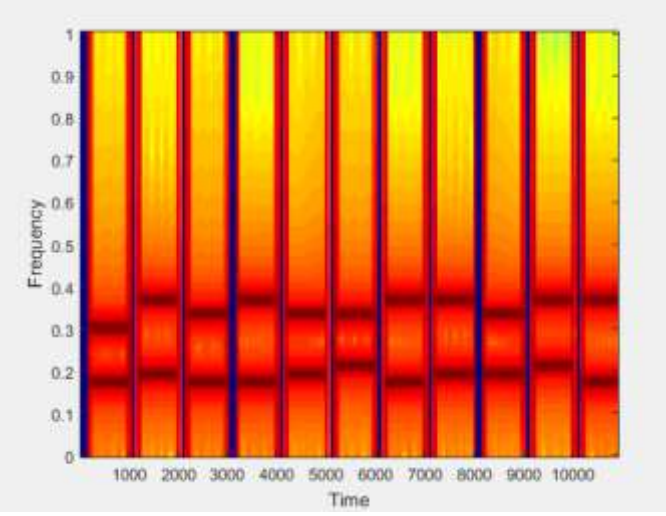
In this lab we learned about a DTMF system and how to encode and decode the signals. We learned how to use band pass filters to decode a signal. Our task was to put in a series of characters and output the same series of characters. We had to generate our output based on signals that corresponded to the input keys. We had to use several different functions to make the system work. They will be described in the following paragraphs.

Tasks

1. Create dtmfdial which encodes the signal based on certain key values
2. Create dtmfdesign which creates the bandpass filters needed to output the correct values
3. Create dtmfscore which is used to assign values of 0 or 1 to represent pass.
4. Use dtmfcut to segment the signal into the different tones
5. Create dtmfrun to input a signal and output the values that correspond to that signal
6. Run the code to prove that it works

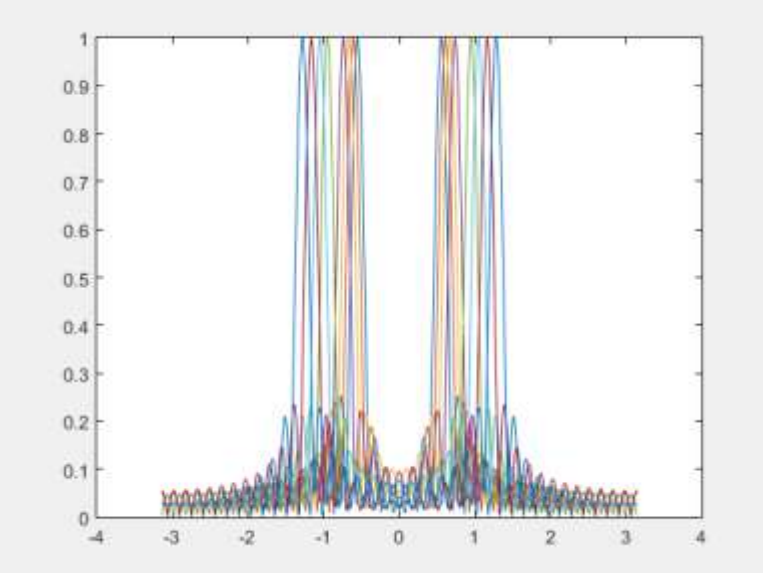
3.1

In this part of the lab we constructed the dtmfdial function. We did this by using the find function and matrix multiplication to select the correct row and column frequencies. In this process we selected values that were equal to some value on the keypad. To check to make sure our function had the correct output we used the sprecgram function. Our output from an arbitrary phone number is shown in this example.



4.1

Our next step was to create the function dtmfdesign. We used this function to create bandpass filters. We needed the band pass filters so that we could detect which frequency components were present when we were decoding the signal. To create these filters we used cosine functions and centered them at specific frequencies we wanted to be in the pass band. The frequencies corresponded to the rows and columns of the keypad matrix. We had to multiply our filter by a certain value in order to normalize it. We called this value B. The value we wanted was the peak value of the matrix we were using. This helped keep our magnitude in the desired range. Below we have plotted the magnitude response of all of our band pass filters.



We were then asked to consider what would happen if we were to change the length of the filter. We tried many values including 40 and 80. We noticed that the value of L determined the width of the passband. We decided to go with a value of 80 because it gave us a narrower pass band, and hopefully more accurate results.

4.2

Our next task was to create dtmfscore. This function calculated if a component of the signal was in the pass band or the stop band. It did this by comparing how much of the signal was allowed through by the filter. We needed the function to return a 1 if the component was in the pass band, otherwise it needed to return a zero. We also normalized the magnitude of the input signal, to match that of the response. We used convolution to create the output from the filter response and the signal.

4.3

In this part of the lab we put all of the functions from previous parts together to create dtmfrun. This function took in a signal and outputted the keys that the signal represented. To do this we used a for loop and the return of the score function to find which two frequency components were in that signal. In order to make this work, we called dtmfdesign and dtmfscore inside this function. They allowed us to do the filtering and the allocating all in the same place. We also used a given function dtmfcut so that

we could break the signal up into segments. We did this part as many times as there were segments. There would be a new segment every time the tone changed.

4.4

This part of the lab called for us to demonstrate our code using a telephone number. We have the output below, at the end of the lab we have the function that we called in order to calculate it.

```
2 -    L = 80;
3 -    fs = 8000;
4 -    keys = ['4','0','7','*','8','9','1','3','2','#','B','A','D','C'];
5      %hh = dtmfdesign(fb,L,fs);
```

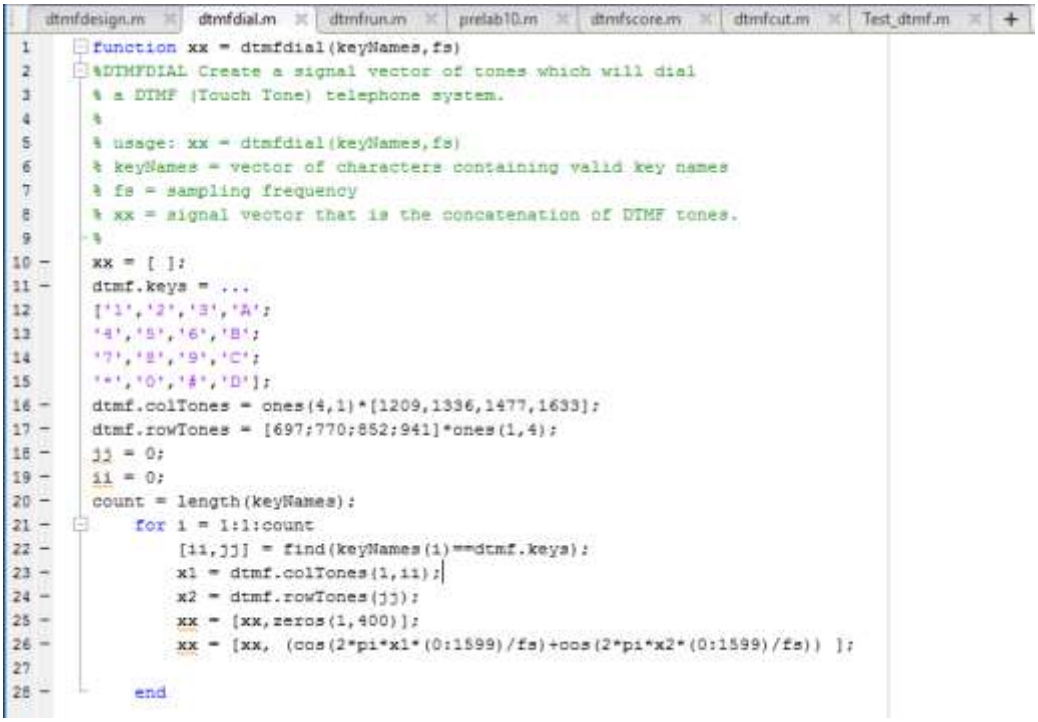
```
Command Window
  >> Test_dtmf
  4
  0
  7
  *
  8
  9
  1
  3
  2
  #
  B
  A
  D
  C
fx >>
```

Conclusion

While this was a challenging lab, we can take away a lot from this experience. We learned to implement filters at a high level of precision. We also used new concepts of encoding and decoding. The work we did constantly called on us to use things we had learned early in the course. We needed things from convolution to the concepts from the frequency domain. We learned that a signal can have various components and we took advantage of that to decode our signal.

Code

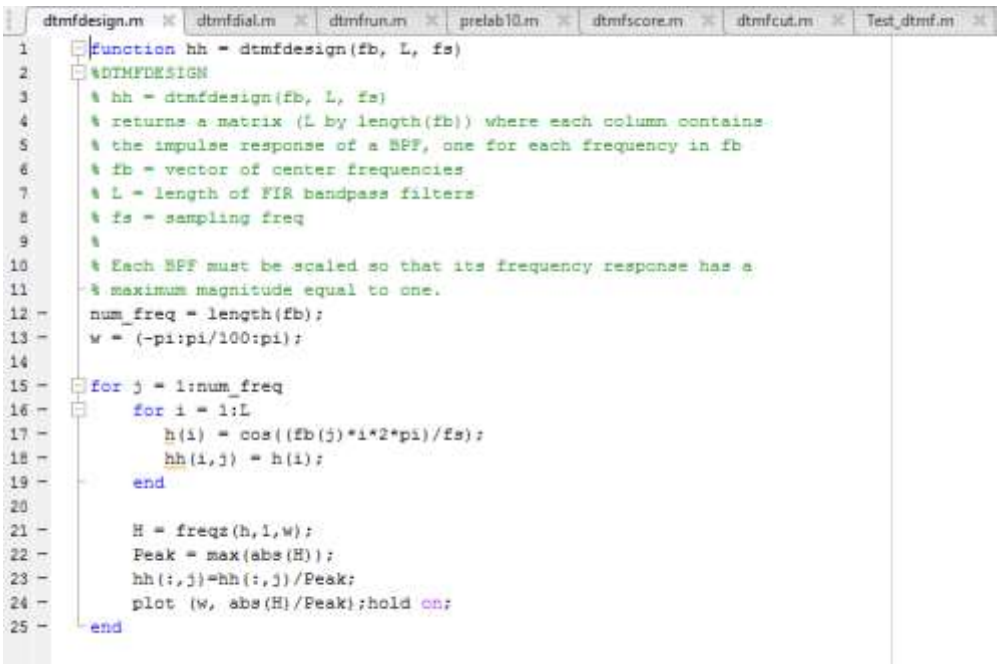
Dtmfdial



```
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.
%
xx = [ ];
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);
jj = 0;
ii = 0;
count = length(keyNames);
    for i = 1:1:count
        [ii,jj] = find(keyNames(i)==dtmf.keys);
        x1 = dtmf.colTones(1,ii);
        x2 = dtmf.rowTones(jj);
        xx = [xx,zeros(1,400)];
        xx = [xx, (cos(2*pi*x1*(0:1599)/fs)+cos(2*pi*x2*(0:1599)/fs)) ];

    end
```

Dtmfdesign

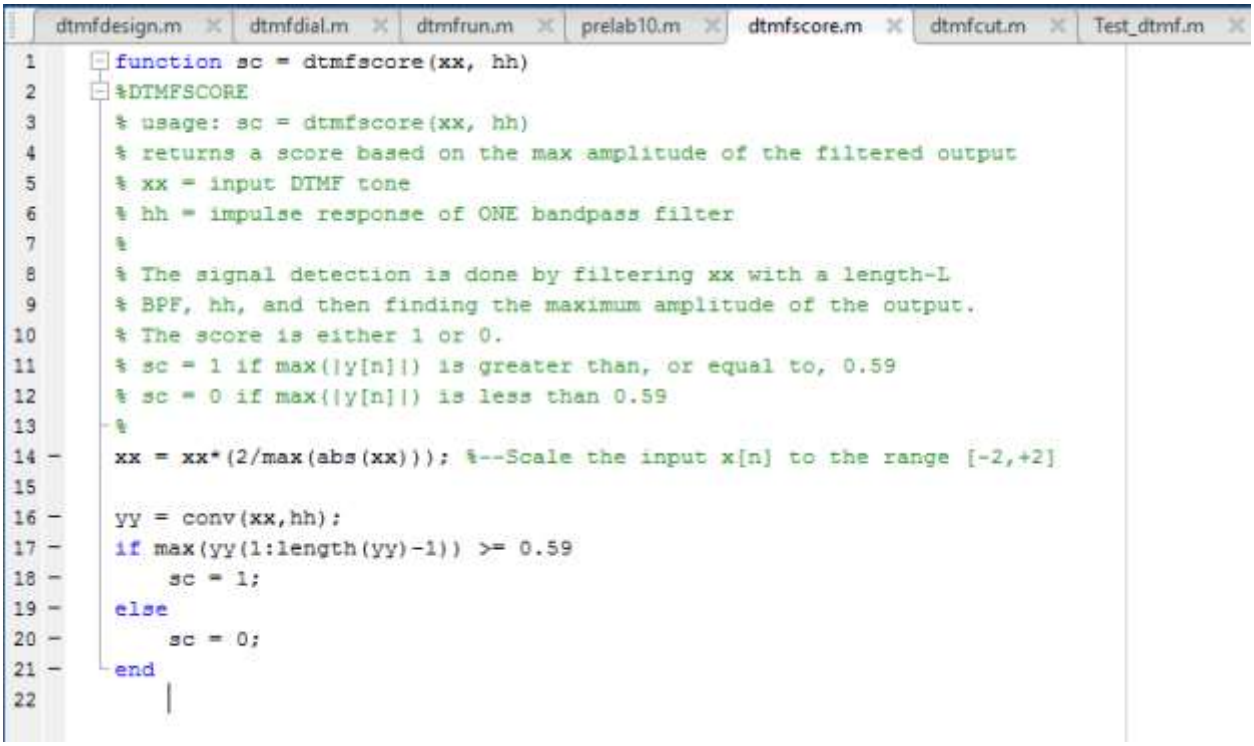


```
function hh = dtmfdesign(fb, L, fs)
%DTMFDESIGN
% hh = dtmfdesign(fb, L, fs)
% returns a matrix (L by length(fb)) where each column contains
% the impulse response of a BPF, one for each frequency in fb
% fb = vector of center frequencies
% L = length of FIR bandpass filters
% fs = sampling freq
%
% Each BPF must be scaled so that its frequency response has a
% maximum magnitude equal to one.
num_freq = length(fb);
w = (-pi:pi/100:pi);

for j = 1:num_freq
    for i = 1:L
       h(i) = cos((fb(j)*i*2*pi)/fs);
       hh(i,j) = h(i);
    end

    H = freqz(h,1,w);
    Peak = max(abs(H));
    hh(:,j)=hh(:,j)/Peak;
    plot (w, abs(H)/Peak);hold on;
end
```

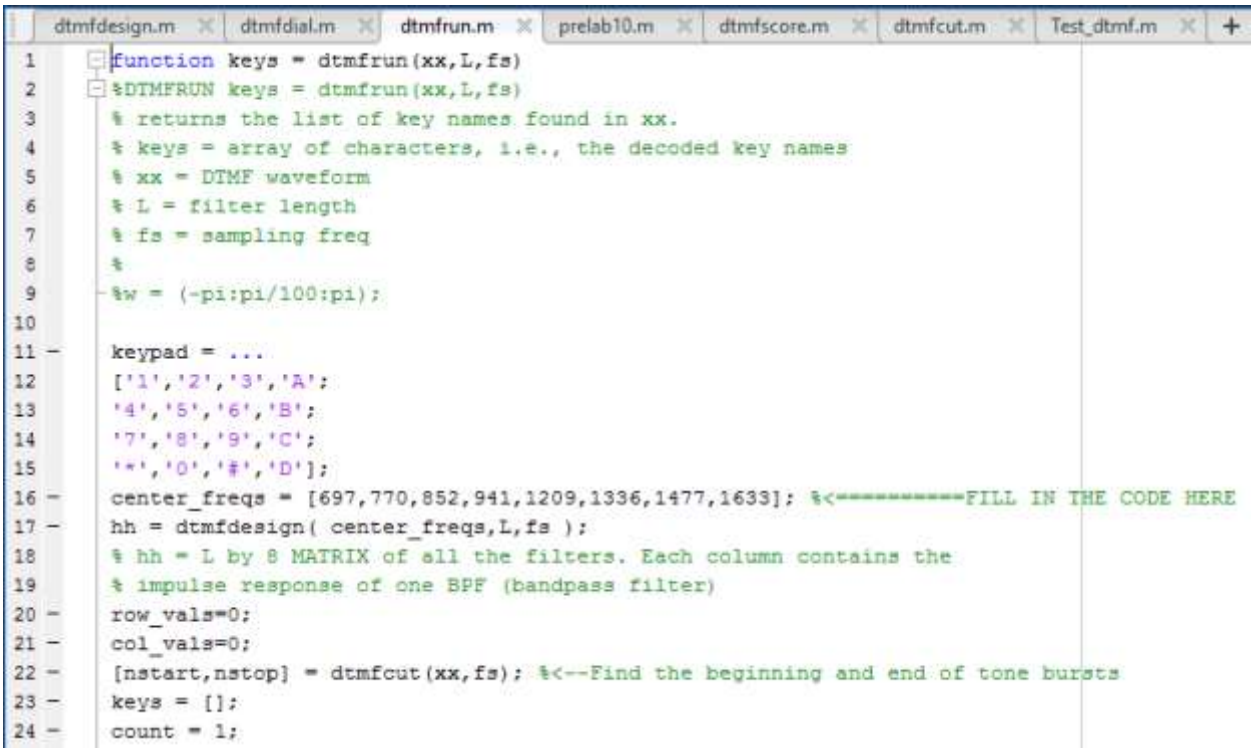
Dtmfscore



```
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(yy(1:length(yy)-1)) >= 0.59
    sc = 1;
else
    sc = 0;
end
```

Dtmfrun



```
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
%
%w = (-pi:pi/100:pi);

keypad = ...
['1','2','3','A';
'4','5','6','B';
'7','8','9','C';
'*','0','#','D'];
center_freqs = [697,770,852,941,1209,1336,1477,1633]; %<==========FILL IN THE CODE HERE
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)
row_vals=0;
col_vals=0;
[nstart,nstop] = dtmfcut(xx,fs); %<--Find the beginning and end of tone bursts
keys = [];
count = 1;
```

Dtmfrun continued

```
    for i = 1:8

             sc = dtmfscore(x_seg,hh(:,i));

            if i <= 4 && sc == 1
               row_vals = i;
            end
            if i > 4 && sc == 1

                col_vals = i-4;
            end

        end
    if row_vals ~= 0 && col_vals ~= 0
            keys(count) = keypad(col_vals,row_vals);
            display(keypad(col_vals,row_vals));
            count = count+1;
    end

end

```

Function Calls

dtmfdesign.m | dtmfdial.m | dtmfrun.m | prelab10.m | dtmfscore.m | dtmfcut.m | Test_dtmf.m

```
%fb = [697,770,852,941,1209,1336,1477,1633];
L = 80;
fs = 8000;
keys = ['4','0','7','*','8','5','1','3','2','#','B','A','D','C'];
%hh = dtmfdesign(fb,L,fs);

xx = dtmfdial(keys,fs);

%[nstart,nstop] = dtmfcut(xx,fs);

%sc = dtmfscore(xx,hh)




output = dtmfrun(xx,L,fs);
output = char(output);
soundsc(xx,fs);
```