DTMF Coder / Decoder Design using FIR Banks
by S.Sircar

Introduction

Analog DTMF telephone signaling is based on encoding standard telephone Keypad digits and symbols in two audible sinusoidal signals of frequencies FL and FH. Thus the scheme gets its name as dual tone multi frequency (DTMF).

Hz	1209	1336	1477	1633
697	**1**	**2**	**3**	**A**
770	**4**	**5**	**6**	**B**
852	**7**	**8**	**9**	**C**
941	*	**0**	#	**D**

Figure 1

Each digit or symbol represented in figure 1 has 2 distinct high and low frequency components. Thus each high-low frequency pair uniquely identifies the corresponding telephone keypad digit or symbol. Each key pressed can be represented as a discrete time signal of form

$$d_t[n] = sin[\omega_L n] + sin[\omega_L n] \ , \ 0 \le n \le N\text{-}1 \quad (1)$$

Where N is defined as number of samples taken. Typically in the sampling frequency used is 8khz. Thus if the two individual frequency components of the signal can be identified then the number dialed can be decoded.

Note :- In this report I have used (*dual tone* and *digit/symbols)* interchangeably but both mean the same. Dual tone means the encoded samples of the corresponding DTMF digits/symbols.

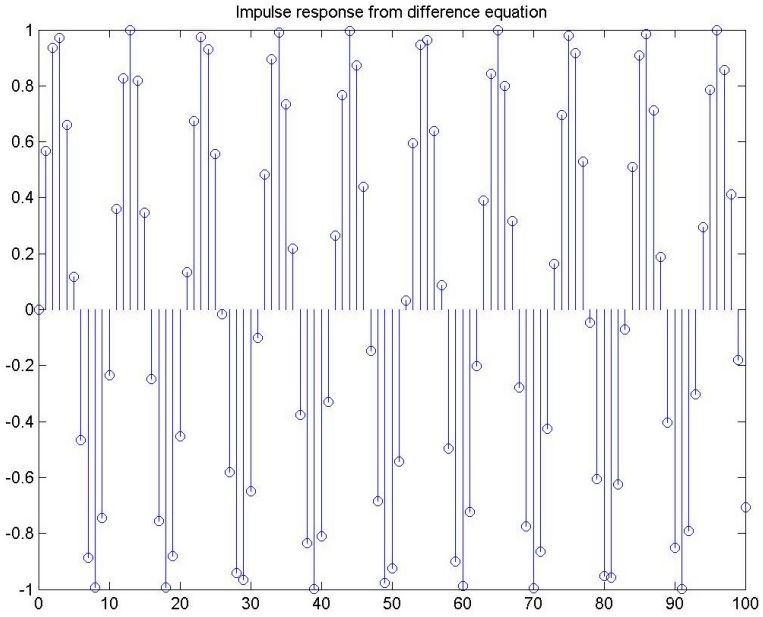
Implementation of DTMF Encoder

The DTMF encoder is implemented in MATLAB function dtmfe.m. The implementation is based on a digital oscillator, that will generate sinusoidal tones at frequencies F_o in response to an input signal x[n] = δ[n].

$$x[n] \longrightarrow H[n] \longrightarrow y[n] \quad \{y[n] = x[n]*H[n]$$

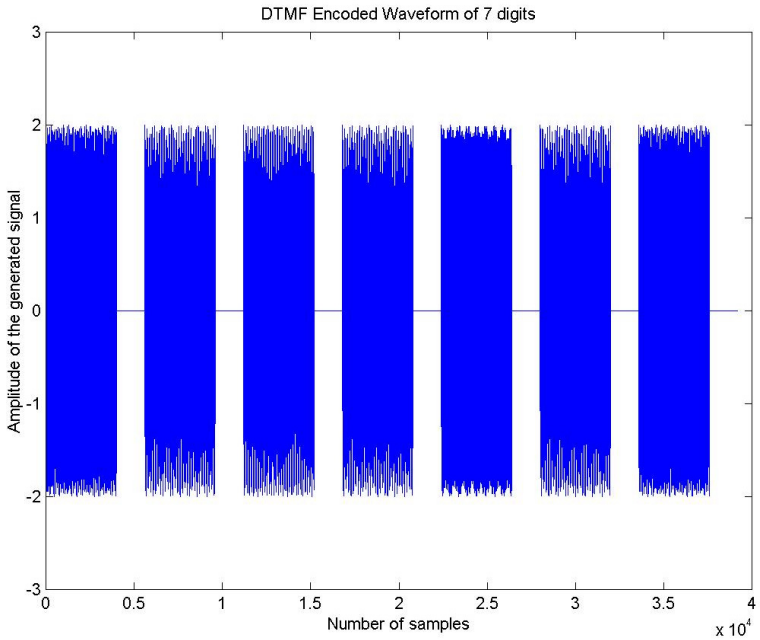
Consider a causal filter with
*y(n) - 2*cos(2*pi * f *Ts)y(n-1) + y(n-2) =0*x(n) sin(f)x(n-1) + 0*x(n-2).*
The impulse response of this system tells us that this indeed is a digital oscillator.

The H[n] is plotted and is sinusoidal and hence any input to this system will oscillate

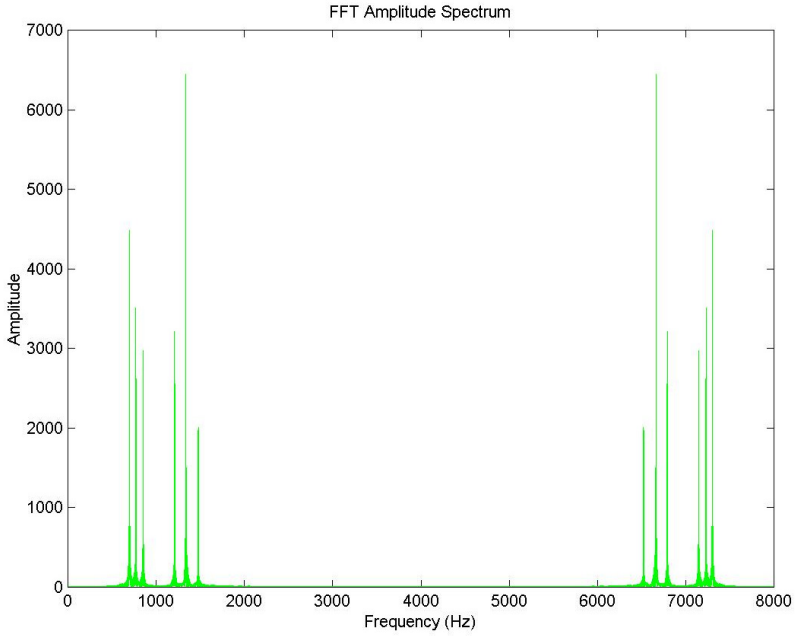


The impulse response of the system is shown in the figure above. This is generated by using the MATLAB dimpulse() function for 100 points.

The encoder is coded so that it inserts silence samples in between each digits/symbols so that they are separated from each other.
Starting position = (total number of digits be encoded-1)(length of each dual tone + number of silence samples to be inserted) + 1.*
This information is used at the decoding step to get each dual tone sample length and how many samples to skip for the next digit/symbol. The resulting waveform after encoding is shown in figure 2 below. The waveform shows an encoded sequence of my phone number 7262527. There are 7 digits in the dialed string and hence the waveform shows 7 distinct sub samples of the resulting DTMF encoded waveform.



The frequency spectrum is shown in the figure below of the dialed dtmf digits. The plot shows both sided frequency spectrum and is simply a mirror of the base band.



Implementation of DTMF decoder

The input to the decoder is a vector containing DTMF tones that are encoded by the encoder. A FIR (Finite Impulse Response) band pass filter is implemented which is centered at the frequencies of interest for decoding each key pressed. The decoding process takes place in iterative form. Starting from row 1 to row 4,in each iteration a FIR band pass filter centered at each F_H is implemented and the signal strength around the band is compared against a threshold. If the mean amplitude of the filter output is more than a set threshold, the frequency component tested is considered to be strong. To prevent cancellation of the sinusoids while calculating the mean, the values are squared.
Similar approach is taken in order to find the F_L present in the signal. Once a row-column pair has been detected the digit encoded is uniquely identified. This approach has been taken in order to suppress the effects of noise in the encoded signal. *This approach has a significant advantage that is the band pass filter coefficients can be manipulated individually to produce a narrower filter and detection process can be enhanced in presence of noise.*

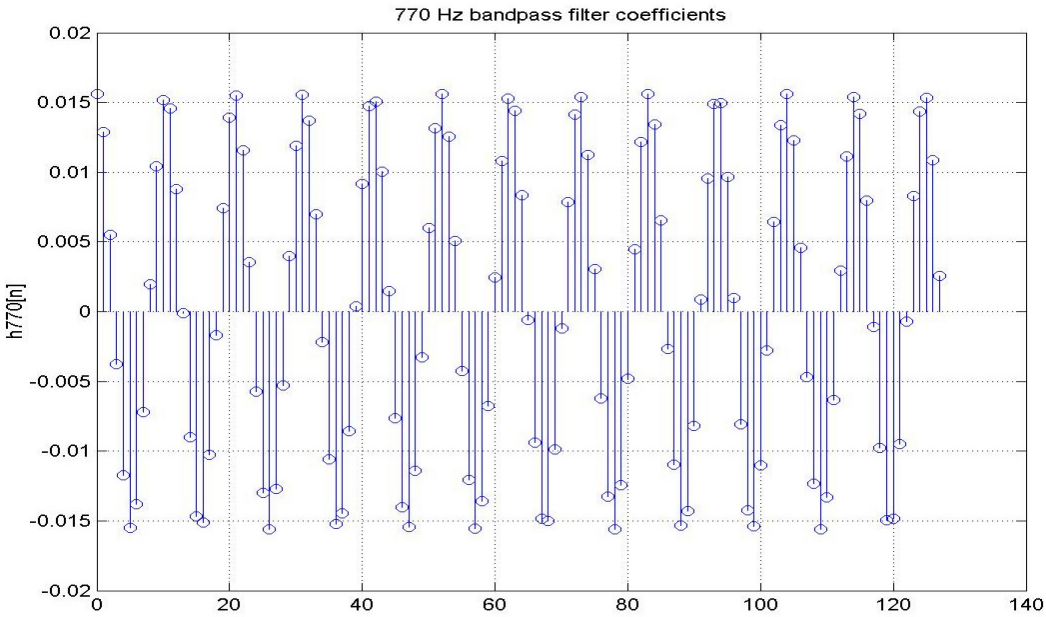
In order to decode a string of dialed symbols/digits the decoding step assumes that the encoder has inserted silence between each dual tone. Each dual tone length is tracked and silence lengths are calculated. The decoding step loops over for the number of digits to be decoded that is easily calculated from the total length of the signal divided by the sum of individual dual tone length and silence length. Thus after decoding the first digit certain number of samples are skipped and the next set of samples are decoded. This means sampling and decoding a small part of the resulting waveform in each iteration, where one iteration relates to one key pressed.

Conclusion

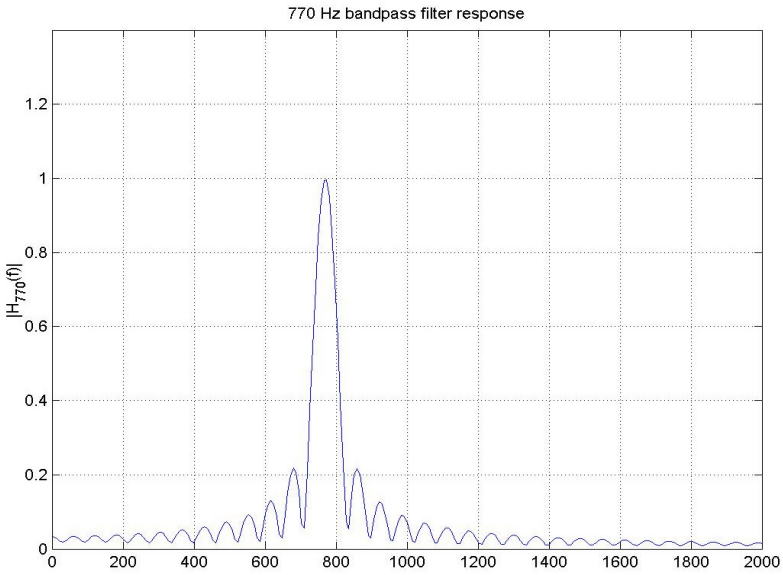
No attempt has been made to test the system performance in a noisy environment. However, since I have used a band pass filter in order to decode every DTMF tone the band pass filter length can be manipulated in order to get good results in noisy environment. A filter of length L (L=128 currently used) can be used to increase the immunity of noise (type of noise). However a higher value of L would detect tones even of lower amplitude better, but also increases the chance of error in detection. This **matched filter** approach definitely reduces the mean square error between two bands of detection. The system has been tested and works according to specification in a noise free environment.

Analysis Results

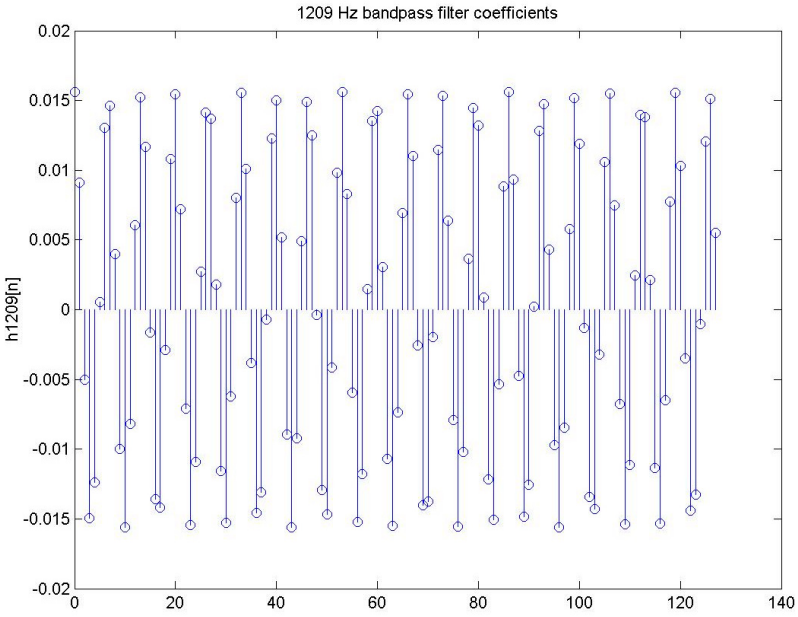
Consider Detecting digit 4 of DTMF which comprise of Fl = 770, Fh = 1209. Filter coefficients and frequency response of 770 hz center frequency band pass FIR filter.



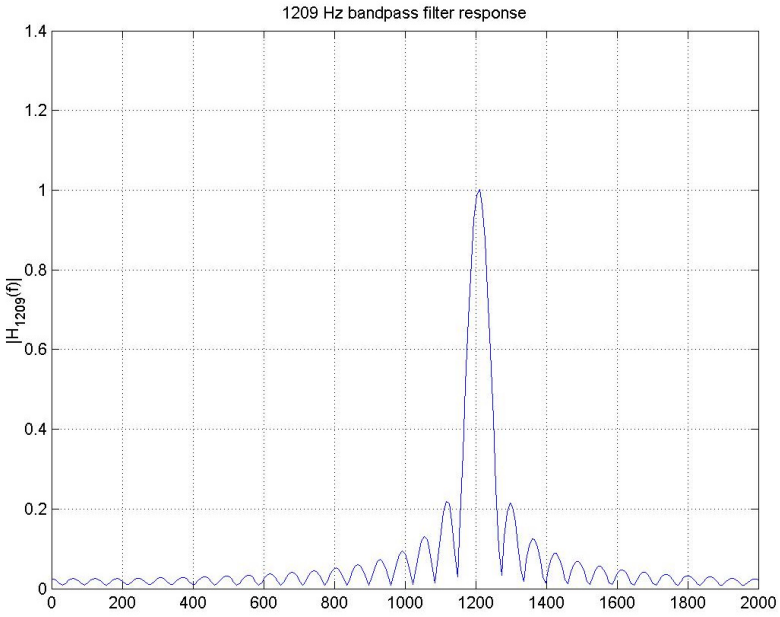
Frequency response of the 770 hz band pass filter is shown in the plot below.



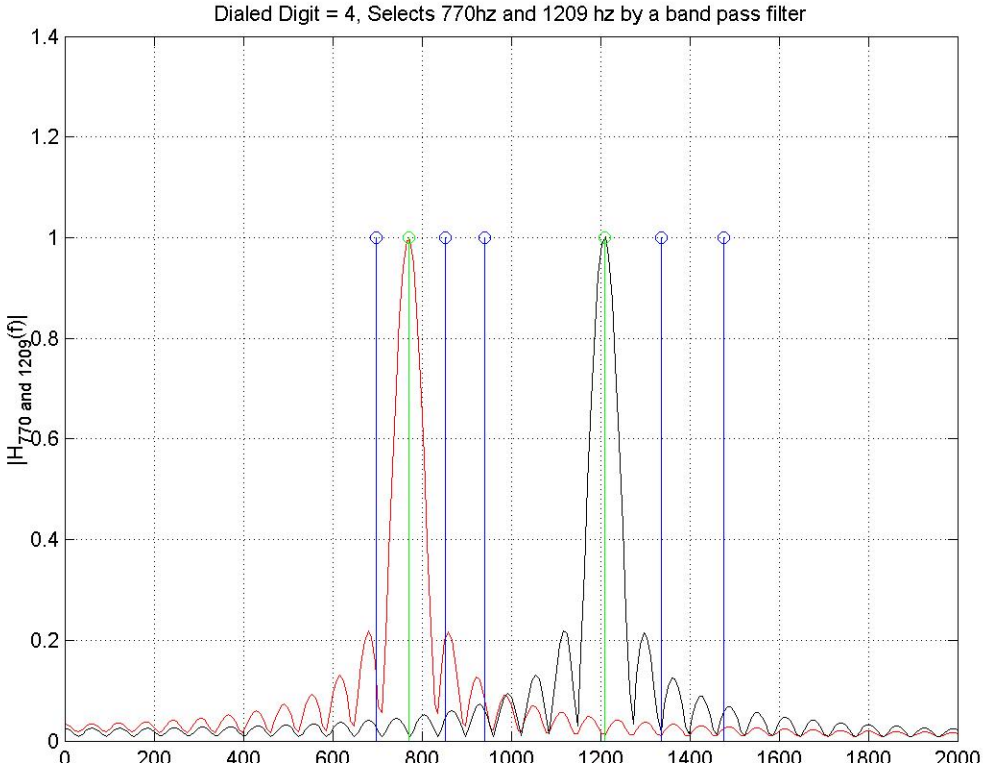
Filter coefficients and frequency response of 1209 hz center frequency band pass FIR filter.



Frequency response of the 770 hz band pass filter is shown in the plot below.



Suppose the frequency we are looking for is 770hz and 1209 hz in order to detect dialed digit 4. The decoding takes place by implementing a band pass filter at every iteration step looking for a particular Fl and Fh. The decoded digit in the following diagram is 4. Take a look at how 770hz and 1209 hz are allowed to pass from the band pass filter.



Source Code for DTMF Encoder/Decoder

```
global dtmf_rfreq
global dtmf_cfreq
global dtmf_key
global dtmf_fs
global dtmf_silencelen
global dtmf_tonelen

% Execute this Script at the begining of encoding and decoding
% This script sets all the global paramneter for encoding and decoding
% Define the frequencies used in synthesizing and decoding the DTMF signal
% Each frequency corresponds to a row or column of keys on the keypad,
%   eg. pressing '4' means that the frequencies for row 2 and column 1 should
%       be used. The resulting DTMF tone is the two waveforms summed together
dtmf_rfreq = [ 697  770  852  941]; % frequency components for each row
dtmf_cfreq = [1209 1336 1477 1633]; % frequency components for each column
dtmf_key =  [ '1'  '2'  '3'  'A';
              '4'  '5'  '6'  'B';
              '7'  '8'  '9'  'C';
              '*'  '0'  '#'  'D'];

    % Sampling Frequency
    dtmf_fs = 8000;
    disp(['DTMF samplerate set to ' num2str(dtmf_fs) ' Hz']);
    %Duration of the Tone to generate
    dtmf_tonelen = round(dtmf_fs * 0.50); % Convert length from seconds to samples
    disp(['DTMF tone length set to ' num2str(dtmf_tonelen) ' samples']);
    %Insert Silence lengths between tones for detection
    dtmf_silencelen = round(dtmf_fs * 0.20);    % Convert length from seconds to samples
    disp(['DTMF break length set to ' num2str(dtmf_silencelen) ' samples']);
```

Code for Digital Oscillator to generate the sinusoids

```
% Generates a sinosoidal output at a specified frequency
% This is generated by implementing a digital oscillator using the following difference equation
% y(n) - 2*cos(2*pi * f *Ts)y(n-1) + y(n-2) =0*x(n) sin(f)x(n-1) + 0*x(n-2)
% The following co-efficient are as follows
% A = [1 -2*cos(2*pi * f *Ts) 1];
% B = [0 sin(2*pi * f *Ts) 0 ];
%
%       Shiladitya Sircar 2002
%       DSP Course Project
%
% Usage -:
% n - must be a vector
% Ts = 1/dtmf_fs;
% n = 0:dtmf_tonelen-1;
% f = some frequency for the tone to be generated
% Y = gensino(f,n,Ts);

function Y = gensino(f,n,Ts)
X = zeros(1,length(n));X(1) = 1; % generate the impulses
A = [1 -2*cos(2*pi * f *Ts) 1];
B = [0 sin(2*pi * f *Ts) 0 ];
Y = filter(B,A,X);
```

Code for DTMF Encoder

```
function encodedTones = dtmfe(num)
% This function translates key pressed in digital telephone keypad to corresponding
% dual tone frequencies
% Example encodedTones = dtmfe('5670531')
% 567031 is string of numbers pressed. The function converts them to their
% corresponding dual tones as per defined in dtmf main (FH,FL)
%
%       Shiladitya Sircar 2002
%       DSP Course Project


%Bring the global variables defined in dtmfmain to the scope of this function

global dtmf_rfreq
global dtmf_cfreq
global dtmf_key
global dtmf_fs
global dtmf_silencelen
global dtmf_tonelen

% Can use the setparameter function in order to just evaluate this function
% to do this just un-comment the setparameter. This script will set up the global
%variable required to execute dtmf encoding.

%setparameter;

num = sprintf('%s',num);
Ts = 1/dtmf_fs;             % Sample length
n = 0:dtmf_tonelen-1;       % Vector to sample from 0 < n <=N-1
%Total Number of Elements required in the ecoded vector space
numcount = length(num);
tlen = numcount*(dtmf_tonelen + dtmf_silencelen);
%Generate the vector first that contains the tones
encodedTones = zeros(1,tlen);

% loop through the individual keys in the input vector

for i = 1:numcount
   % Find out which column and row this key belongs to

   [row,col] = find( dtmf_key == upper(num(i)));

   if ( isempty(row) )
      disp('One of the Key Pressed is not a valid DTMF Key');
   end

   % calculate the start and end position of the current tone
   startpos = (i-1)*(dtmf_tonelen + dtmf_silencelen) + 1;
   endpos = startpos + dtmf_tonelen - 1;
   encodedTones(startpos:endpos) = gensino(dtmf_cfreq(col) , n , Ts ) + gensino(dtmf_rfreq(row) , n , Ts
);
end
```

Code for DTMF Decoder

```
function digitstr = dtmfd(encstr)
% This function translates the encoded tones to corresponding
% keys/key that are pressed.
% Example digitstr = dtmfd(vec)
% Where vec is the vector that contains the encoded tones
%
%       Shiladitya Sircar 2002
%       DSP Course Project
global dtmf_rfreq
global dtmf_cfreq
global dtmf_fs
global dtmf_silencelen
global dtmf_tonelen
global dtmf_key

% Can use the setparameter function in order to just evaluate this function
% to do this just un-comment the setparameter. This script will set up the global
%variable required to execute dtmf encoding.

%setparameter;

% Length of Band Pass Filter
% If the bandpass is too wide, this can be increased to produce a more narrow filter
% for finer precision. However computation time increases a value L = 64 gives
% satisfactory results for no noise environment
L = 64;
filt_n = 0:L-1;

dtmf_digilen = length(encstr)/(dtmf_tonelen+dtmf_silencelen);

for deco_seq = 1:dtmf_digilen
  % Loop through each row and Check if the row-i frequency exist in the signal
  % Define the pointer at start of the sequence
  startpos = (deco_seq - 1)*(dtmf_tonelen + dtmf_silencelen) + 1;
  endpos = startpos + dtmf_tonelen - 1;
  encstr_frag = encstr(startpos:endpos);
  for i = 1:4
    % Calculate filter coefficients for the bandpass filter for row frequencies
    % given the center frequency
    hh = 2/L*cos(2*pi*dtmf_rfreq(i)*filt_n/dtmf_fs);
    ss = mean(conv(encstr_frag,hh).^2) > mean(encstr_frag.^2)/5;% compare mean amplitude and if
    % more than 20% ( /5 ), the frequency
    % component is considered to be strong
    if (ss)
      row = i;
      break % we already have our row - no need to check the rest
    end
  end

  % Loop through each column and check if the column-i frequency exist in the signal
  for i = 1:4
    % Calculate filter coefficients for the bandpass filter for column frequencies
    % given the center frequency
    hh = 2/L*cos(2*pi*dtmf_cfreq(i)*filt_n/dtmf_fs);
    ss = mean(conv(encstr_frag,hh).^2) > mean(encstr_frag.^2)/5;
    if (ss)
      col = i;
```

```
        break  % we already have our column - no need to check the rest
      end
    end
    numstr(1,deco_seq) = dtmf_key(row,col);
end
digitstr = numstr;
```

Code for DTMF main

```
function return_digits = dtmfmain(num2encode)
% This program simulates the telephone DTMF Dialing Scheme
% Please See dtmfe.m for how encoding is done
% Please See dtmfd.m for how decoding is done
% Usage Examples
% return_digits = dtmfmain('123A#*')
%
%       Shiladitya Sircar 2002
%       DSP Course Project

%Define Global variable used by dtmfe and dtmfd functions
global dtmf_rfreq
global dtmf_cfreq
global dtmf_key
global dtmf_fs
global dtmf_silencelen
global dtmf_tonelen

% Define the frequencies used in synthesizing the DTMF signal
% Each frequency corresponds to a row or column of keys on the keypad,
%   eg. pressing '*' means that the frequencies for row 4 and column 1 should
%       be used. The resulting DTMF tone is the two waveforms summed together
dtmf_rfreq = [ 697  770  852  941]; % frequency components for each row
dtmf_cfreq = [1209 1336 1477 1633]; % frequency components for each column
% Define the telephone keypad as a 2d matrix
dtmf_key =  [ '1'  '2'  '3'  'A';
  '4'  '5'  '6'  'B';
  '7'  '8'  '9'  'C';
  '*'  '0'  '#'  'D'];

% Sampling Frequency
dtmf_fs = 8000;
disp(['DTMF samplerate set to ' num2str(dtmf_fs) ' Hz']);
%Duration of the Tone to generate
dtmf_tonelen = round(dtmf_fs * 0.50); % Convert length from seconds to samples
disp(['DTMF tone length set to ' num2str(dtmf_tonelen) ' samples']);
%Insert Silence lengths between tones for detection
dtmf_silencelen = round(dtmf_fs * 0.01);    % Convert length from seconds to samples
disp(['DTMF break length set to ' num2str(dtmf_silencelen) ' samples']);

% Call Encoder Function
encodedTones = dtmfe(num2encode);

%sound(encodedTones,dtmf_fs);

%Plot the encoded Tones
X = fft(encodedTones);
N = length(encodedTones);
f = (0:N-1)/N*dtmf_fs;
figure(1),plot(f, abs(X),'g')
```

```
xlabel('Frequency (Hz)')
ylabel('Amplitude')
title('FFT Amplitude Spectrum')
%Return The Decoded Tones
Decoded_Digits = dtmfd(encodedTones);

if (strncmp(Decoded_Digits,num2encode,length(num2encode)) == 1)
   disp(' ');
   disp('Succesfully decoded the dialed number');
   disp(' ');
   disp('The dialed Telephone Number was'); num2encode
   disp(' ');
   disp('The Decoded Telephone Number is '); Decoded_Digits
else
   disp('Error In Decoding'); Decoded_Digits

end
```