Name: Sreyan Biswas

Dual Tone Multi Frequency (DTMF)

Encoder/Decoder

❖ Aim of the project:

To create study and analyse DTMF (Dual Tone Multi Frequency) coder/decoder using Digital FIR filter on MATLAB

❖ Theory:

Coder:

- In digital signal processing ideal filters with Analog DTMF telephone signalling is based on encoding standard telephone Keypad digits and symbols in two audible sinusoidal signals of frequencies $\mathbf{F_{low}}$ and $\mathbf{F_{high}}$. Thus, the scheme gets its name as dual tone multi frequency (**DTMF**).

- In the below figure, each digit or symbol represented has 2 distinct high and low frequency components. Thus, each high-low frequency pair uniquely identifies the corresponding telephone keypad digit or symbol.

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

- Each key pressed can be represented as a discrete time signal of form:

$$d_i[n] = \sin[\omega_L n] + \sin[\omega_L n] \ , \ 0 \le n \le N\text{-}1$$

where, N is defined as number of samples taken. The sampling frequency used was **5kHz** in my case.

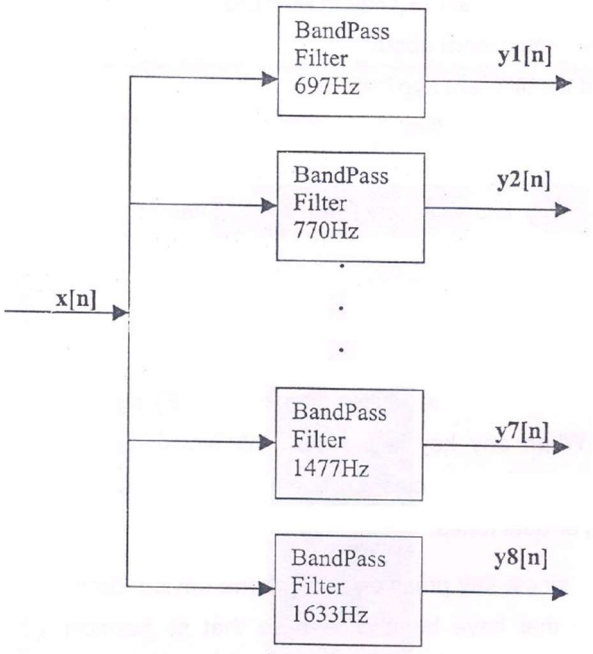
De-Coder:

- The input to the decoder is a vector containing DTMF tones that are encoded by the encoder. A FIR band pass filter is implemented which is centered at the respective 8 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 $\mathbf{F_{high}}$ is implemented and the signal strength around the band is compared amongst all the BPFs.

- The **comparison process** involves calculating the **RMS** of the **filtered output** from each filter and then finding the **top 2 RMS** values which are prominent.

- Generally, the RMS value for prominent bands is more than 25 times than the absent bands. To prevent cancellation of the sinusoids while calculating the mean, the values are squared and rms was taken.
- 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.
- The figure below depicts the decoding algorithm structure.



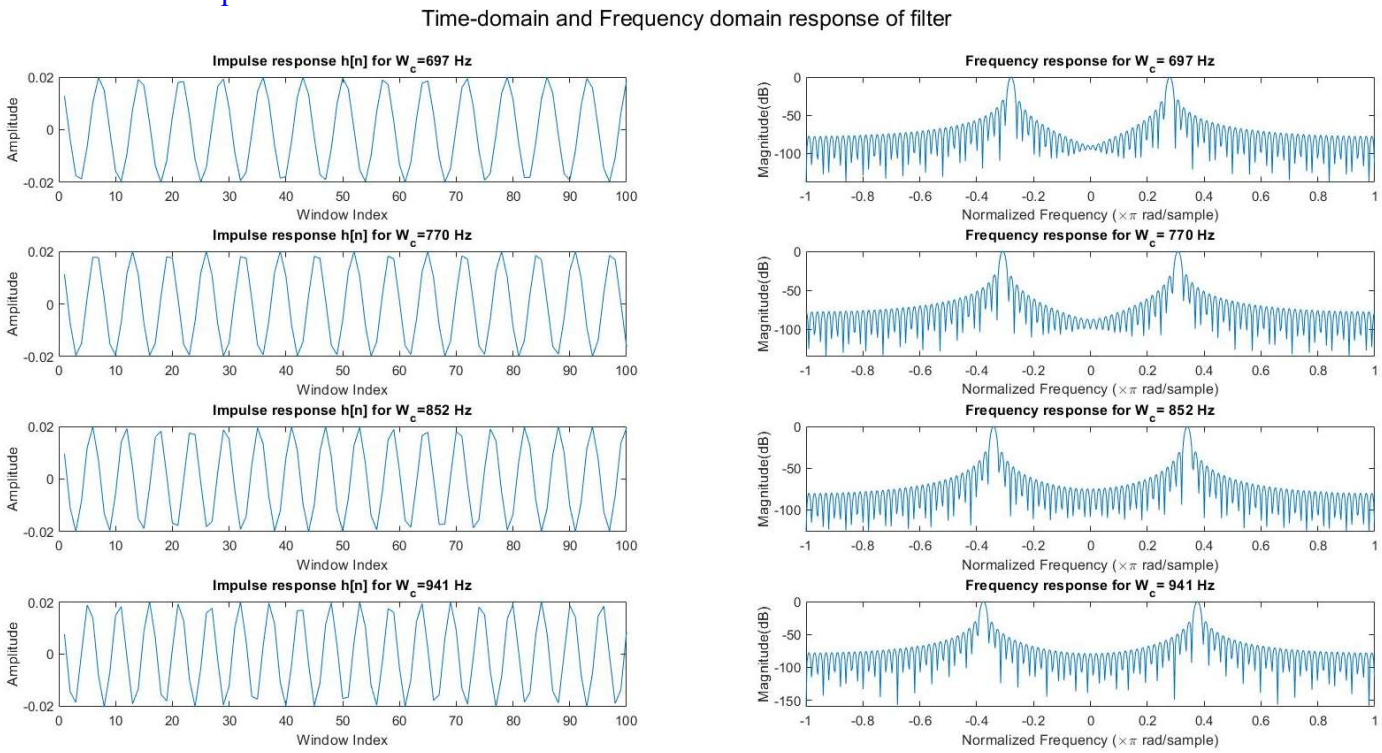
❖ Simulation specifications:

- The **N** was chosen to 64. This is the number of points I took in time domain for each keypress.
- The sampling frequency for the time domain signal was chosen to be **Fs=5000Hz**
- A **vector** was declared for all the possible keypress characters.
- The input sequence was taken as a string array. Then the array was iterated character by character.
- Each of the character has some specific pair of frequencies associated with it. It was found out by using a 2D matrix using the **meshgrid** function
- Then we found the coordinates within the **2D matrix** by using a mathematical **formula** which related the index of the original vector array and the coordinates of the 2D matrix
- A function `plot_signal(f1,f2,x,N,fs,key)` was declared to plot the time domain signal and its frequency spectra for each of the keypress character
- `function [h]= bpf_single(wc,fs,N,L)` was declared to make the individual filters
- `function [out]= bpf_bank (x_all,N,fs,vec)` was declared to make an array of filters and then detect the correct sequence using the RMS value of the output signals of the filter
- `function plot_filter_out(y_arr,N_FFT,fs,wc,index)` was declared to plot the filter output in time domain and in frequency domain
- `function plot_freq_filter(h_arr,wc)` was declared to plot the transfer characteristics of the filters

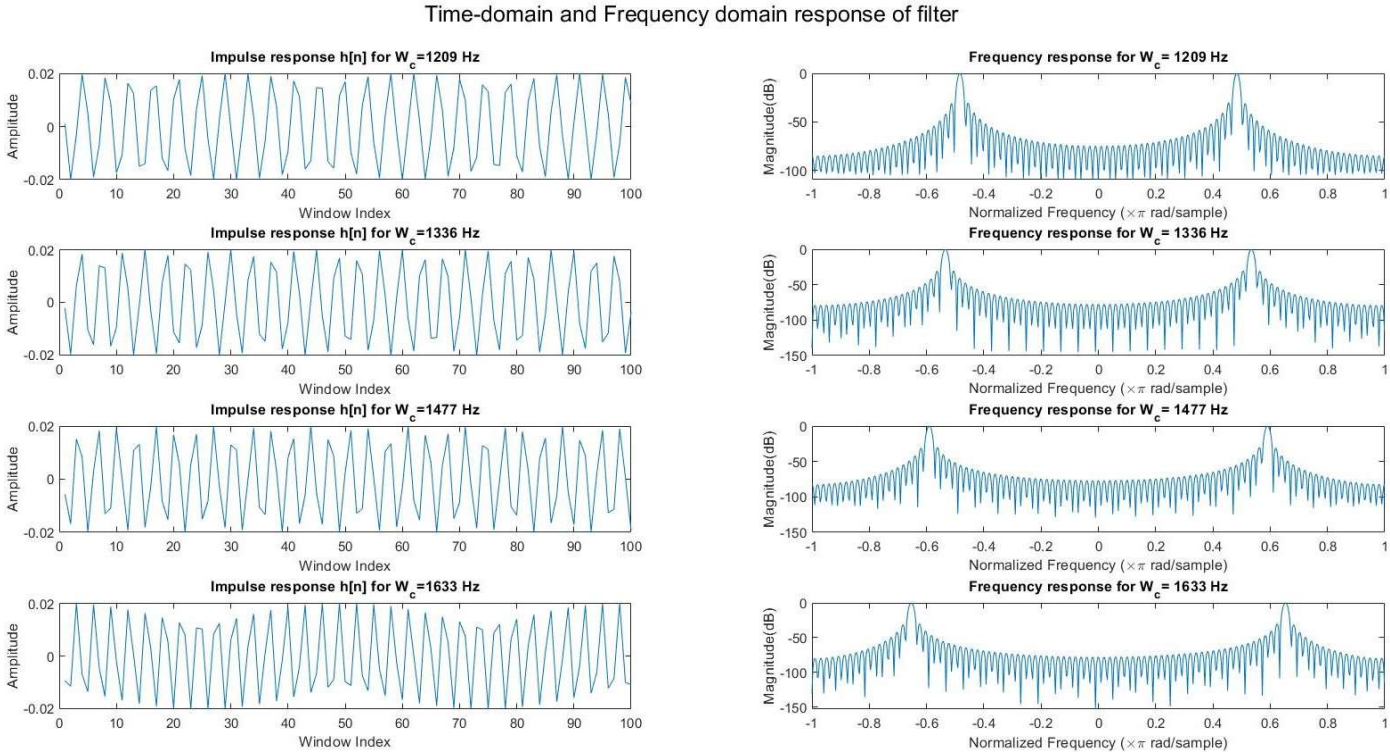
❖ Results:

a) **Time domain and Frequency domain representation of the impulse response of the filters**

For the lower frequencies:



For the Higher frequencies:



We can observe the Cutoff frequency **(Wc)** of the filter increasing as we can see the peak in the frequency response of the each of the succeeding filters are spreading outwards

b) **DTMF encoder-decoder output**

We have given the **input** as the sequence: **892451ABC***

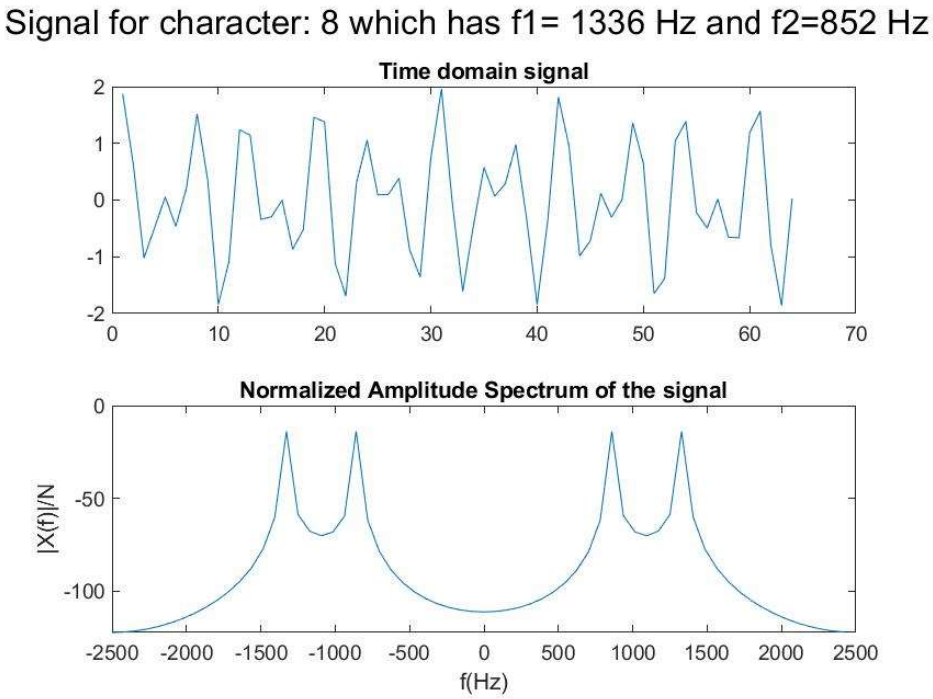
And then the output of the DTMF encoder-decoder is:

```
Command Window
  Write sequence
  892451ABC*
  Detected sequence: 892451ABC*
fx >>
```

The output sequence matches the input. Hence, we conclude that the given input sequence was **correctly decoded** by the DTMF encoder/decoder

c) **Encoder output:**

The plot below is only for the **first character '8'**. The plots for the other characters are also correct but they have not been shown in this report for the sake of simplicity



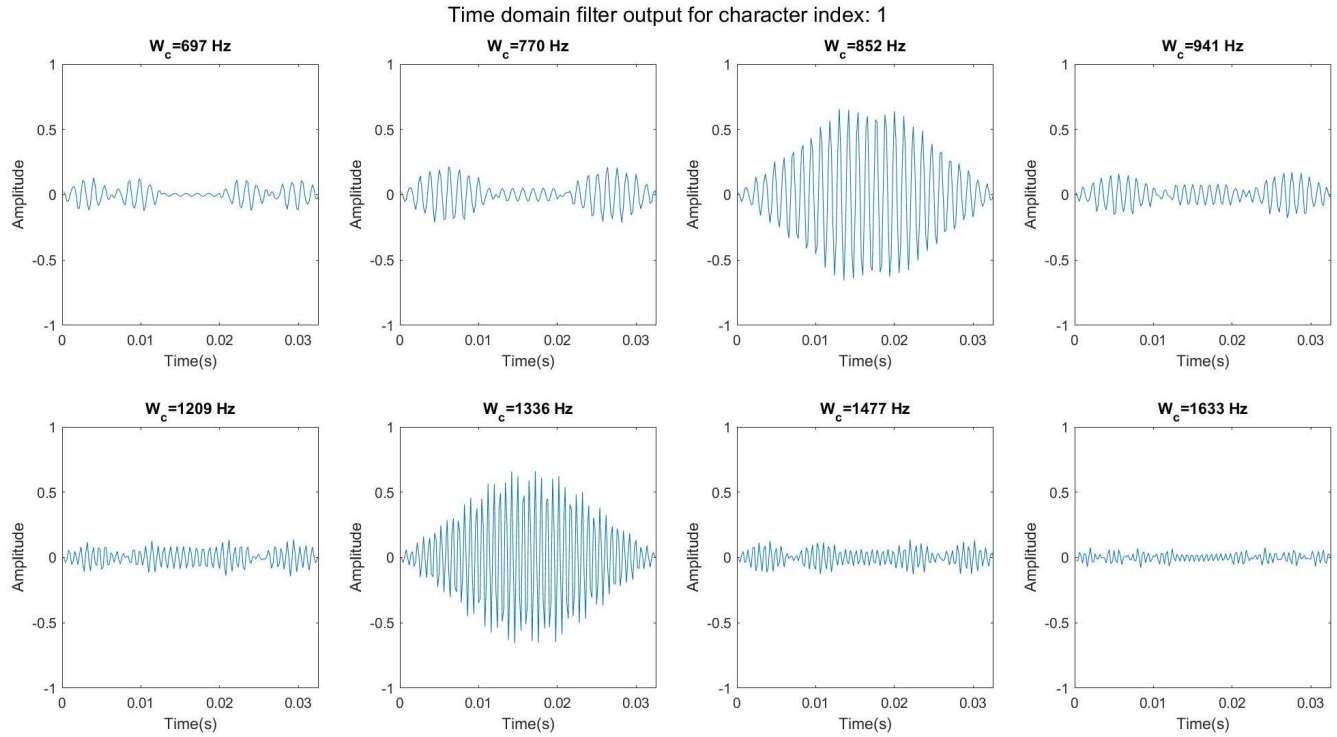
- We can clearly see that the two frequencies **1336Hz** and **852Hz** are present from the frequency spectrum as the two correspond to the frequencies in that range

- We **concatenate** all the signals generated for each character and then send it to the decoder for decoding which characters were originally present in the passed signal **'x(t)'**

d) **Decoder output**

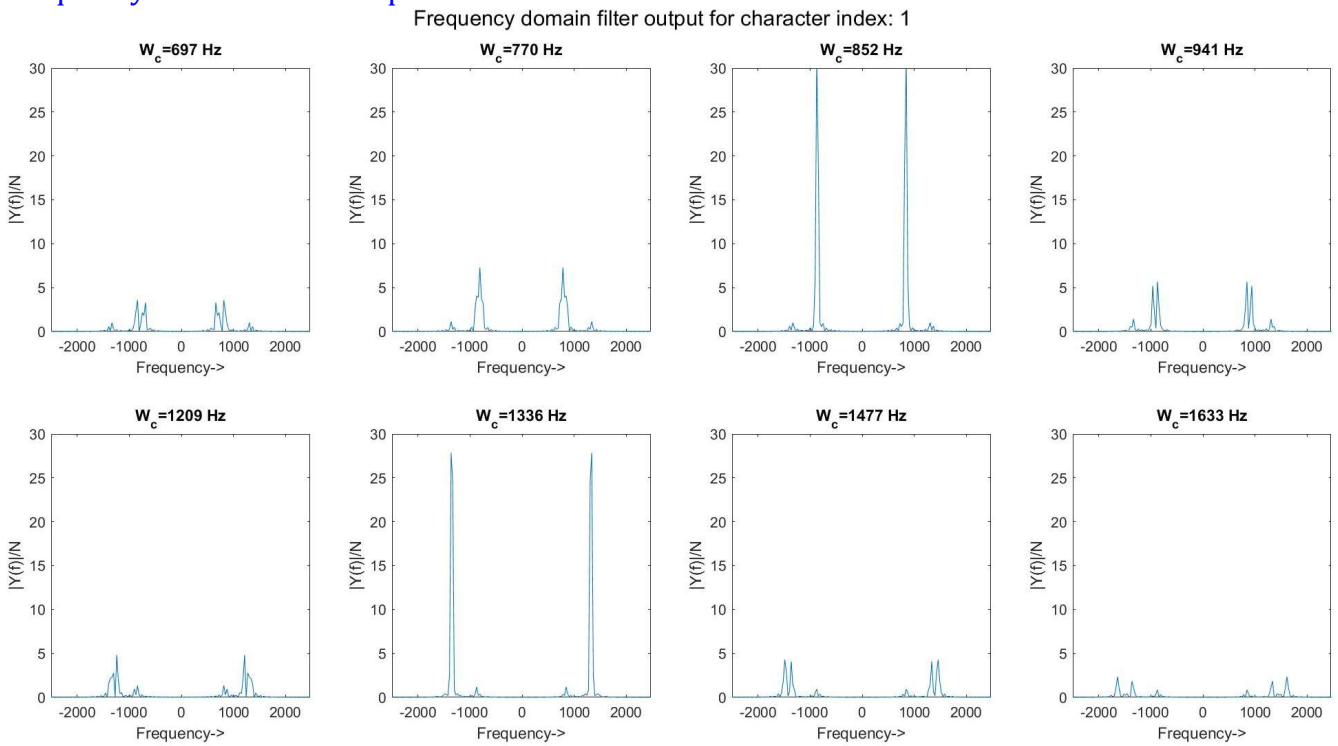
The plot below is only for the **first character '8'**. The plots for the other characters are also correct but they have not been shown in this report for the sake of simplicity.

Time-domain filter output**:**



- This is the output for all the 8 filters with different cut-off frequencies.
- We can clearly see that for the Cut-off frequency **Wc= 852Hz** and **Wc=1336Hz** the time domain amplitude is comparatively higher than the other filter outputs.
- Thus this leads us to a conclusion that these two frequencies were originally present in the input signal 'x'.
- These 2 frequencies are then **back-tracked** by using the original key-mapping for the frequencies of each of the key. Thus, this leads us to the decoding of the character which was present. This is done for the entire signal by breaking the signal into chunks of small signals which match the length of each of the original signal length in time domain

Frequency-domain filter output**:**



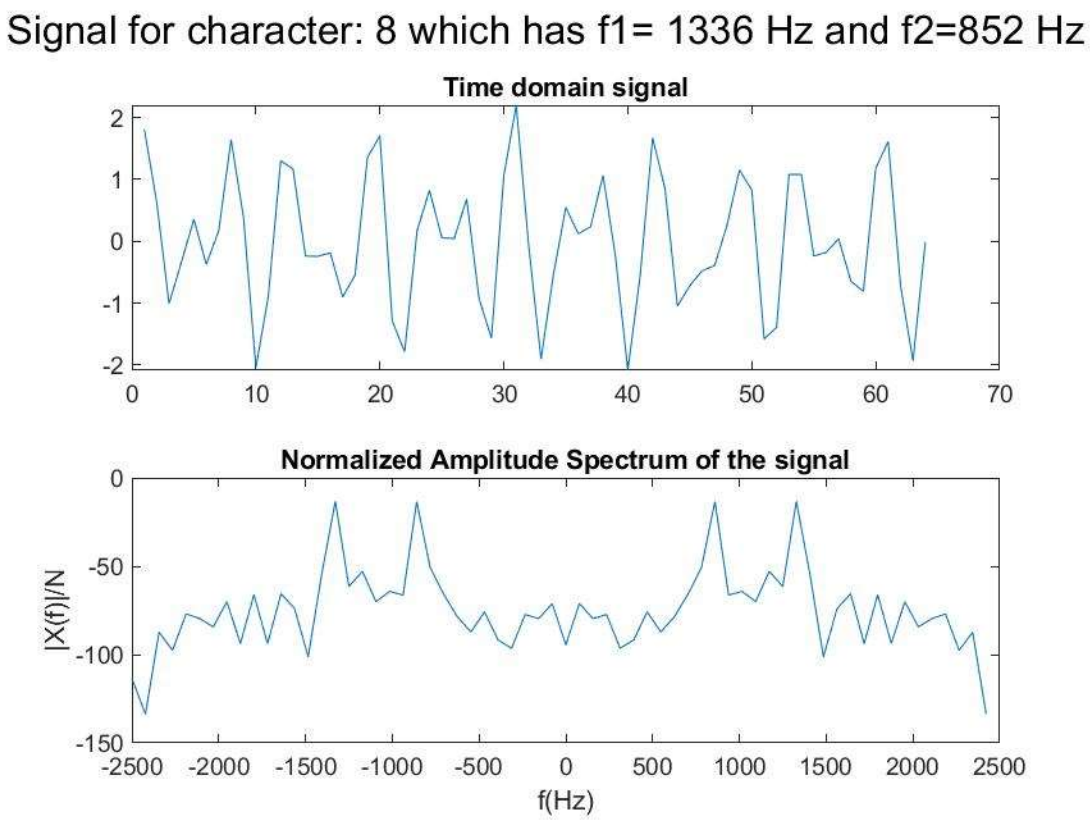
- The above plot is the Normalized Amplitude **spectra** for the output of the 8 filters.

- This plot also verifies that the filter output for the filters with cut-off frequencies **Wc=852Hz** and **Wc=1336Hz** have higher amplitude in its pass-band. Thus these 2 frequencies were present in the original signal.

❖ Optional Part: GUI and Noise

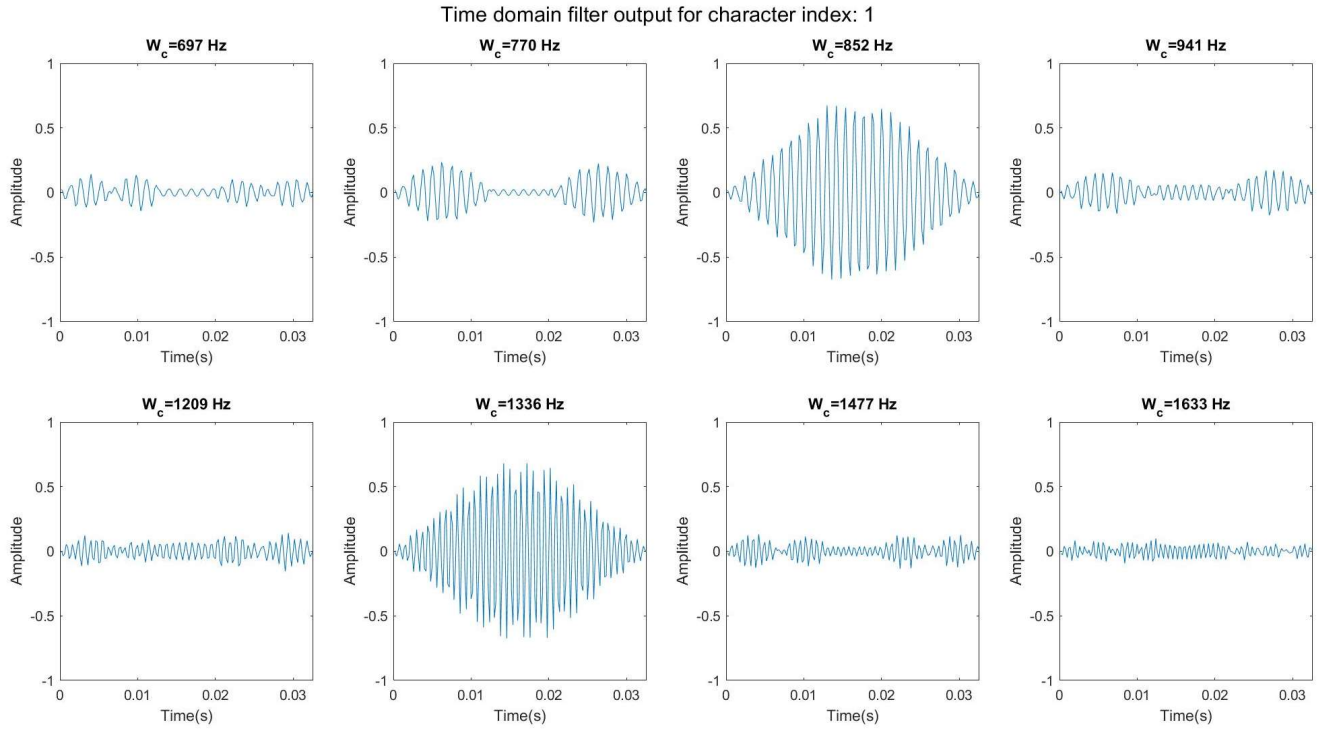
a) Adding Noise:

Given sequence: **892451ABC***



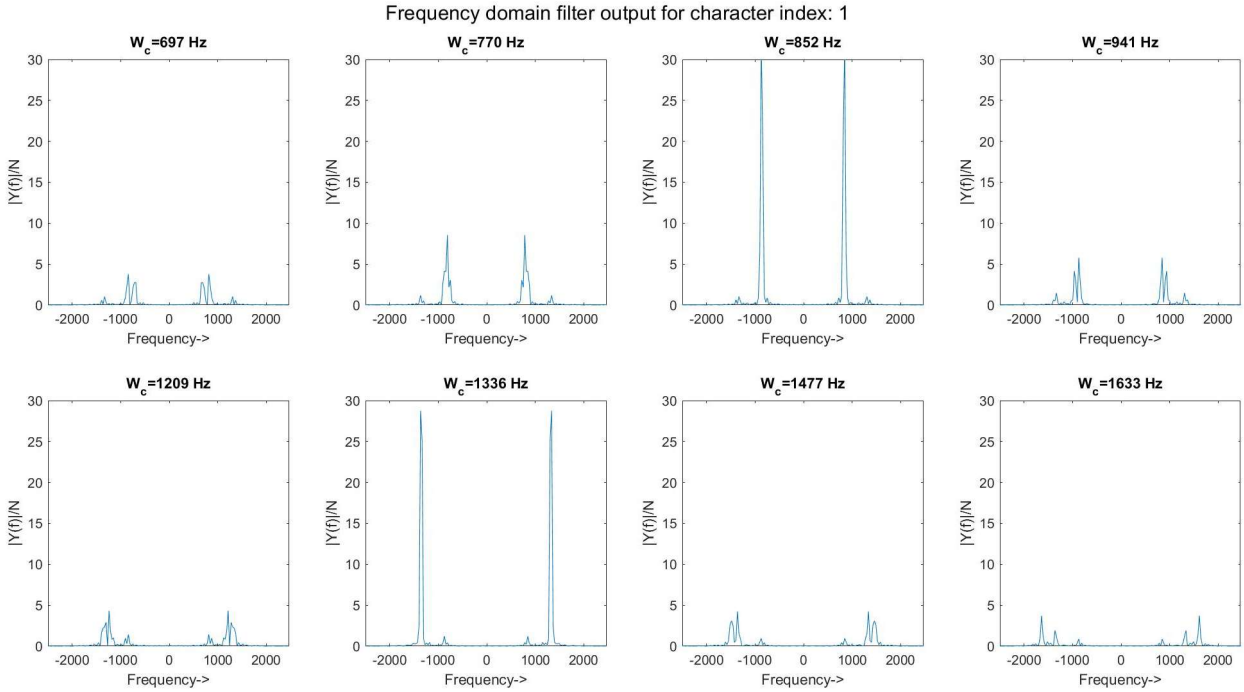
It is evident from the frequency spectra that there is a significant amount of **noise** in the signal generated for the character '8'

Time-domain filter output:



Even with the added noise the time domain signals for the filter outputs are still arguably different. Hence it is possible to decode the sequence correctly even with the added noise

Frequency-domain filter output:



The frequency domain filter outputs are also considerably different. Hence it is possible to decode the sequence correctly even with the added noise

```
Command Window
  Write sequence
  892451ABC*
  Detected sequence: 892451ABC*
fx >>
```

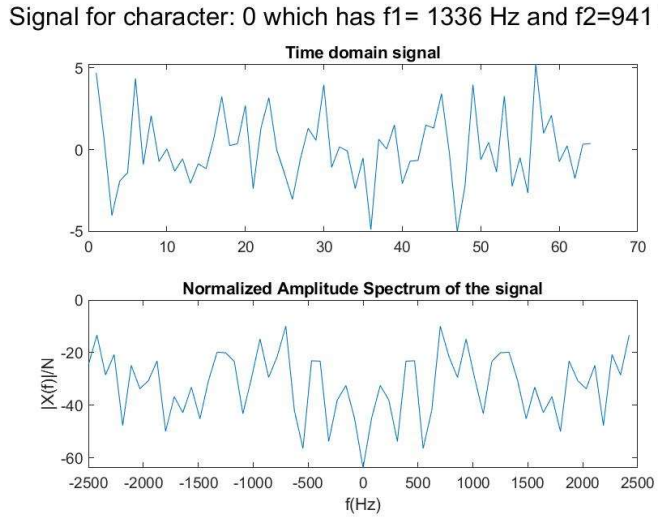
It was observed that till around **1.2** amplitude of the maximum value of the **noise** signal, the DTMF have been able to correctly decode the signal. If we increase it further then the characters were not detected correctly.

b) Adding noise sufficient enough to give incorrect decoding results

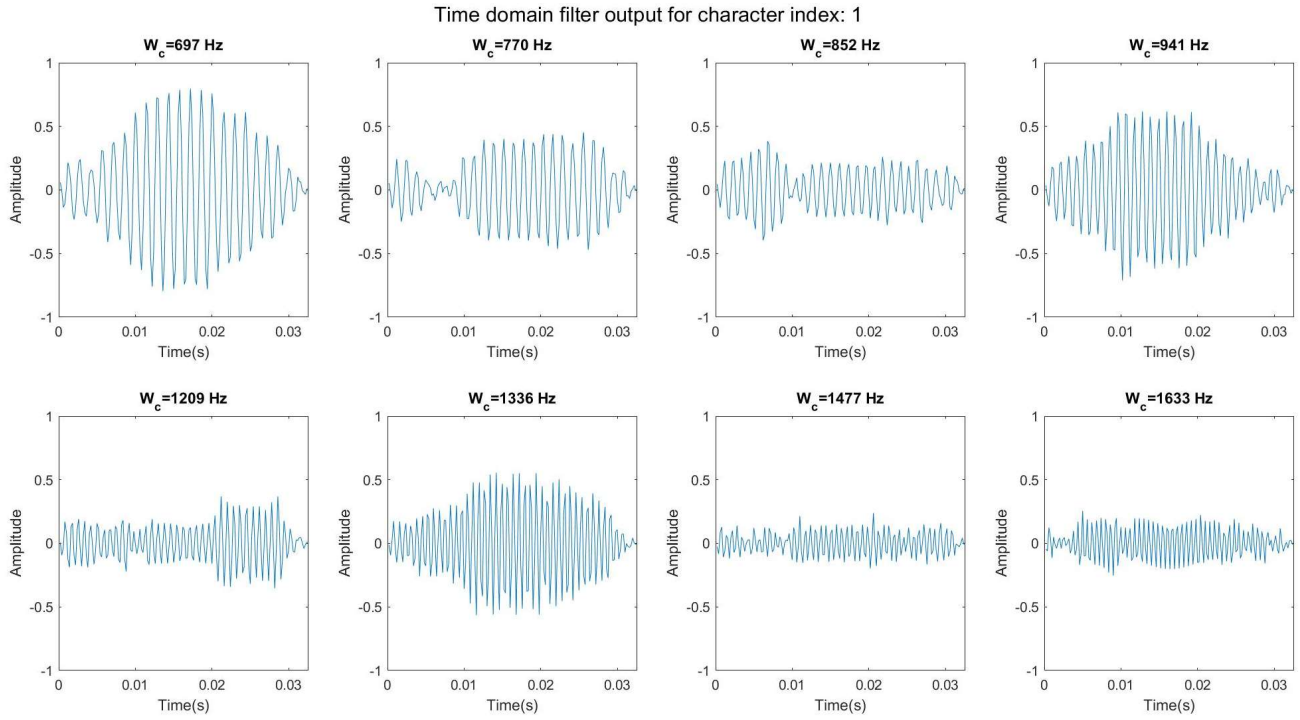
Noise amplitude: 1.8
Given input sequence: 0123456789ABCD*#
Detected sequence: x1x34x6789AB7D*#

The 'x' represents that the decoder was unable to find two maximum frequencies one of them which would represent the row and the other would represent the column.

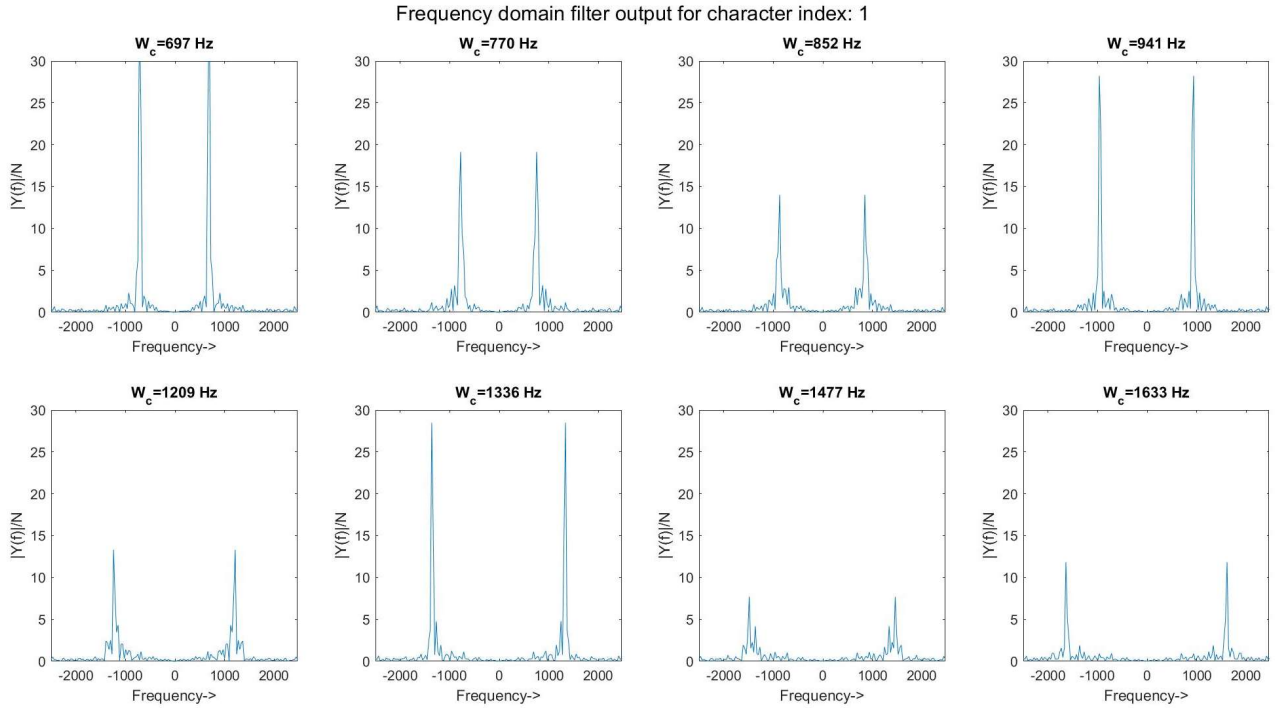
Generated signal plot for the character '0':



Time-domain filter output:



Frequency-domain filter output:



In this signal due to the excessive noise, the character '0' was not able to get detected correctly.

MATLAB Output:

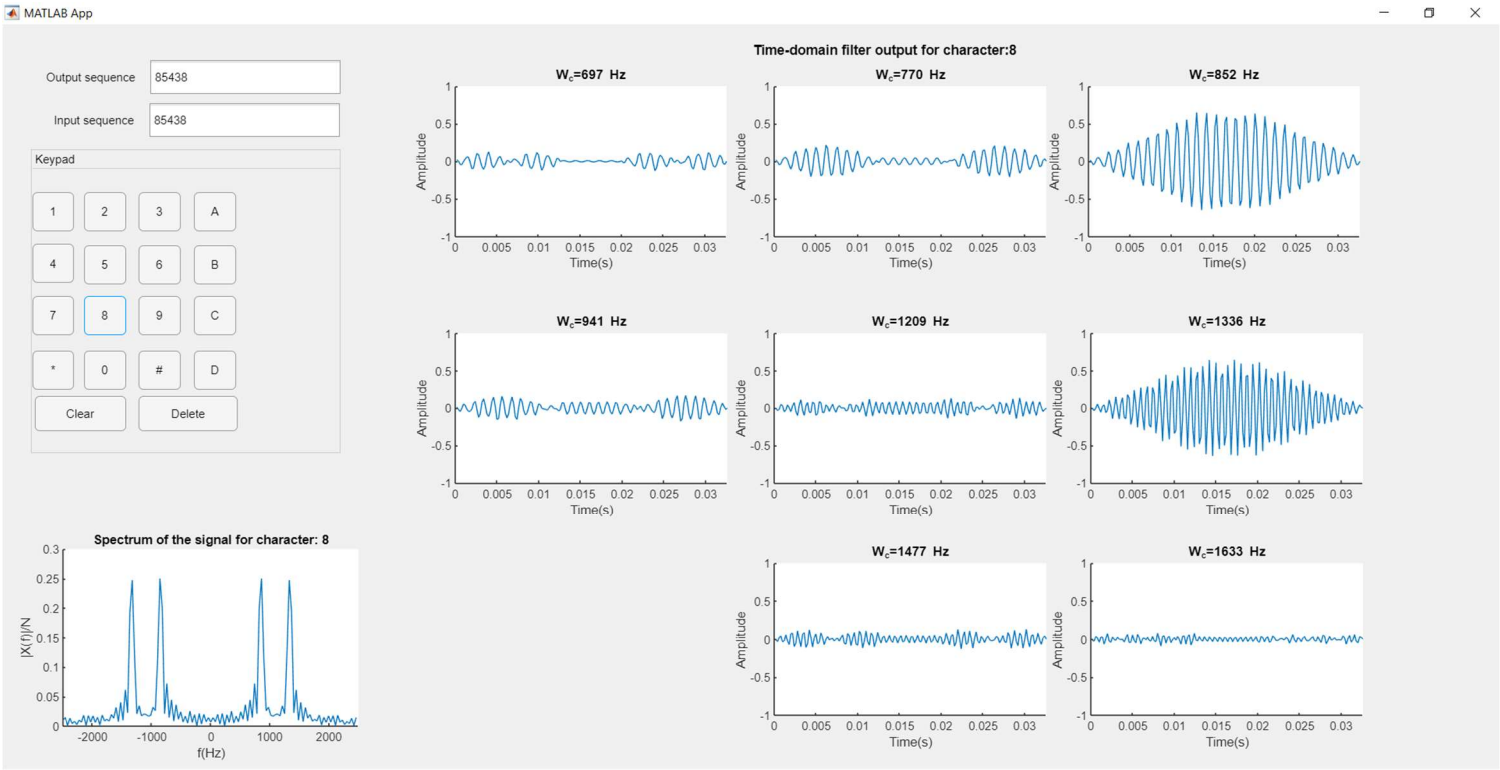
```
Write sequence
0
Detected sequence: x
```

c) **GUI:**

The GUI was made by using the '**App Designer'** functionality of the MATLAB software. This app was made from scratch by coding in the MATLAB App designer which follows **Object oriented programming** design approach.

The older version of MATLAB had '**GUIDE**' for the development User Interface, but it has been discontinued in the latest releases of MATLAB.

Link for GUI app:
https://iitkgpacin-my.sharepoint.com/:u:/g/personal/sb123_iitkgp_ac_in/EZWxHtGDp2NLnDZMmWAz3QEBuwGb4rX9eERihzuvsk7XJw?e=QhWWub



- The GUI has a "**Keypad**" which is for entering the sequence.
- It also has '**Clear**" button which clears the entire input text field
- The '**Delete'** button clears one single character from the sequence
- The Time-domain **filter output** for each of the character is displayed **dynamically** as we keep entering the individual digits
- A **sound** corresponding to the signal containing its two frequencies is produced each time a key is pressed. This was done by using the '**sound'** function of MATLAB
- The **frequency spectra** of the character which was most recently pressed is shown in the bottom left portion of the app window

❖ Discussion:

- The fact that the rms value for the filtered output from the two prominent bandpass filters is higher than those of absent bands by factor of 25 gives us information about the SNR of the bandpass filters.
- All the symbols transmitted in the sequence **892451ABC*** with N samples/ symbol were received back correctly in the aforementioned sequence itself.
- Also all the characters were checked for the correctness in the decoding. It was found that within practical limits of noise, the signals were perfectly capable of being decoded.
- The tones in the DTMF were not harmonics of one another and the difference of two frequencies didn't produce any other frequency.

- **Reason for not using harmonics:**
  The signal in real life passes through various channels and at some part we have an RF-frontend which involves metallic junctions (which might get oxidised) or special non-metallic fibers which can cause intermodulation effect.
  It means that when a signal with several frequency components pass through these non-ideal channels, these junctions due to their inherent non-linear characteristics cause the frequencies to get added up (multiplication of two sinusoids yields two sinusoids with their frequencies as sum and difference of constituents).
  Thus in-case we use frequencies in the touchpad which are sum and/or difference of distinct frequencies already on keypad, then some of the filter banks cannot distinguish between intermodulated (unwanted) signal and desired signal and thus decode a wrong key touch.

Some devices may cause harmonic distortion leading to generation of multiples of fundamental harmonics. Thus for these reasons we avoid using multiples of harmonics and sum/difference of harmonics in our DTMF.

❖ MATLAB Code:

Complete Code of encoder and decoder:

```
clc;
clear all ;

prompt = 'Write sequence \n' ;
num = input(prompt, 's' );
ori=num;%storing the original sequence

vec=[ '1' , '2' , '3' , 'A' , '4' , '5' , '6' , 'B' , '7' , '8' , '9' , 'C' , '*' , '0' , '#' , 'D' ];
N= 64; %number of points to take for each key in time domain

fs= 5000 ;%sampling frequency of the time domain signal

x_all=zeros ( 1 ,N*length(num));
for i =1:length(num)
    ch=num(i);
    index=find(vec==ch);
    x_all(( 1 +( i -1 )*N: i *N)) = signal(vec(index),N,fs,vec);
end
bpf_bank(x_all,N,fs,vec);

function [x]= signal(key,N,fs,vec)
    t= 1:N;

    [X,Y]= meshgrid ([ 1209 1336 1477 1633 ],[ 697 770 852 941 ]);

    for i = 1 : 16
        if (key==vec( i ))
            a= 1 + fix (( i -1 )/ 4 ); % i want to go to the next row only when the i=5, so i-1/4
            b= 1 + rem (( i -1 ), 4 );
        end
    end


    x(t)= sin (2* pi *X(a,b)*t/fs)+ sin (2*pi*Y(a,b)*t/fs);
    plot_signal(X(a,b),Y(a,b),x,N,fs,key);

end

function [h]= bpf_single(wc,fs,N,L)
    n= 1 :L;
    b= 1 ;
    T= 1 /fs;
    h(n)=b* cos( 2*pi*wc*n*T);
    omega = -pi:0.01:pi;
    H = freqz(h,1,omega);
    B = 1/max(abs(H));
    h = B*h;
end

function [out]= bpf_bank (x_all,N,fs,vec)
    wc=[ 697 770 852 941 1209 1336 1477 1633 ]; %8 frequencies
    n_sig=length(x_all)/N;
    RMStable= zeros ( n_sig , 8 );
    L=100; %length of time domain impulse response of the filter

    h_arr= zeros(8,L);
    for i=1:8
        h_arr(i,:)= bpf_single(wc(i),fs,N,L);
    end
    plot_freq_filter(h_arr,wc); %Plot the freqz of the filers created
    %Find the filter output
    y_arr= zeros(8, N+L-1);
    for i=1:n_sig
        x=x_all( 1 +( i -1 )*N: i *N);
        for j = 1 : 8
            h=h_arr(j,:);
            y_arr(j,:) = conv(h,x); %length of this 'y' is N+L-1 (convulution property)
            RMStable( i , j )=rms(y_arr(j,:));
```

```
        end
        %PLOT the filter output
        N_FFT=N+L-1;
        plot_filter_out(y_arr,N_FFT,fs,wc,i);
    end

    %find the first 2 elements from each row and make a comumn vector of size(n_sig,2)
    %the lower index will give the lower frequency
    %and thus that will give the x-coodinate(row) in the touchpad matrix

    [~, fmax_index]=sort(RMStable, 2 , 'descend' );  %we store the index of the RMS numbers in a sorted
manner
    row=min(fmax_index(:, 1 : 2 ),[], 2 );
    col=max(fmax_index(:, 1 : 2 ),[], 2 ) -4 ;
    out= zeros ( size (row, 1 ), 1 );
    %disp ([row col])
    for k= 1 : size (row, 1 ) %we iterate till n_sig
        out(k)=vec( 4 *( row(k)-1 )+col(k)); %this is like the row-wise allocation in c++.Check the
touchpad matrix
    end
    fprintf("Detected sequence: %s\n",out);
end


function plot_filter_out(y_arr,N_FFT,fs,wc,index)
    figure('WindowState','maximized')
    sgtitle("Frequency domain filter output for character index: "+index);
    t1=(1:N_FFT)*(1/fs);
    for i=1:8
        y=y_arr(i,:);
        ysft=abs(fftshift( fft(y,N_FFT) )); %fft of N+L-1 point
        subplot( 2 , 4 , i );
        freq=fs*(-(N_FFT/2):(N_FFT/2)-1)/N_FFT;
        plot(freq,ysft );
        title("W_c="+ wc(i) +" Hz" )
        xlabel( 'Frequency->' )
        ylabel( '|Y(f)|/N' )
        ylim([ 0 30 ]) %to see the relative difference in peaks
    end
    saveas(gcf,"Freq_output_" + index+".jpg");

    figure('WindowState','maximized')
    sgtitle("Time domain filter output for character index: "+index);
    for i=1:8
        y=y_arr(i,:);
        subplot( 2 , 4 , i );
        plot(t1,y)

        title("W_c="+ wc(i) +" Hz" )
        xlabel('Time(s)');
        ylabel('Amplitude');
        ylim([ -1 1 ])
        xlim([0 t1(end)])
    end
    saveas(gcf,"Time_output_" + index+".jpg");
end


function plot_freq_filter(h_arr,wc)
    figure
    sgtitle("Time-domain and Frequency domain response of filter")
    k=1;
    for i=1:8
        if(i==5) %for i=1 to 4 we will use fig1
            figure
            sgtitle("Time-domain and Frequency domain response of filter")
            k=1;
        end
        h=h_arr(i,:);
        omega = -pi:0.01:pi;
        H = freqz(h,1,omega);
        subplot(4,2,k)
        plot(h)

        title( "Impulse response h[n] for W_c="+wc(i)+" Hz" )
        ylabel('Amplitude');
        xlabel('Window Index');
```

```
        k=k+1;

        subplot(4,2,k)
        plot(omega/pi,20*log(abs(H)));
        title("Frequency response for W_c= "+wc(i)+" Hz" )
        xlabel('Normalized Frequency (\times\pi rad/sample)')
        ylabel('Magnitude(dB)')
        k=k+1;
    end
end

function plot_signal(f1,f2,x,N,fs,key)
    figure
    subplot(211)
    plot(x)
    title("Time domain signal")

    subplot(212)
    %N=3*N;
    y1=fftshift(fft(x,N));
    k=fs*(-(N/2):(N/2)-1)/N;
    plot(k,20*log(abs(y1)/N));
    xlabel('f(Hz)')
    ylabel('|X(f)|/N')
    title("Normalized Amplitude Spectrum of the signal")
    sgtitle("Signal for character: "+key+" which has f1= "+f1+" Hz and f2="+f2+" Hz")
    if(key~= '*')
        saveas(gcf,"Signal_plot_" + key+".jpg");
    end

end
```

Code for GUI:

```
classdef EXP_3 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                        matlab.ui.Figure
        TimedomainfilteroutputLabel     matlab.ui.control.Label
        InputsequenceEditField          matlab.ui.control.EditField
        InputsequenceEditFieldLabel     matlab.ui.control.Label
        OutputsequenceEditField         matlab.ui.control.EditField
        OutputsequenceEditFieldLabel    matlab.ui.control.Label
        KeypadPanel                     matlab.ui.container.Panel
        DeleteButton                    matlab.ui.control.Button
        ClearButton                     matlab.ui.control.Button
        DButton                         matlab.ui.control.Button
        Button_hash                     matlab.ui.control.Button
        Button_0                        matlab.ui.control.Button
        Button_star                     matlab.ui.control.Button
        CButton                         matlab.ui.control.Button
        Button_9                        matlab.ui.control.Button
        Button_8                        matlab.ui.control.Button
        Button_7                        matlab.ui.control.Button
        BButton                         matlab.ui.control.Button
        Button_6                        matlab.ui.control.Button
        Button_5                        matlab.ui.control.Button
        Button_4                        matlab.ui.control.Button
        AButton                         matlab.ui.control.Button
        Button_3                        matlab.ui.control.Button
        Button_2                        matlab.ui.control.Button
        Button_1                        matlab.ui.control.Button
        UIAxes_9                        matlab.ui.control.UIAxes
        UIAxes_8                        matlab.ui.control.UIAxes
        UIAxes_7                        matlab.ui.control.UIAxes
        UIAxes_6                        matlab.ui.control.UIAxes
        UIAxes_5                        matlab.ui.control.UIAxes
        UIAxes_4                        matlab.ui.control.UIAxes
        UIAxes_3                        matlab.ui.control.UIAxes
        UIAxes_2                        matlab.ui.control.UIAxes
        UIAxes                          matlab.ui.control.UIAxes
    end


    methods (Access = private)
```

```
        function [out]=decode(app,ch,N)
            vec=[ '1' , '2' , '3' , 'A' , '4' , '5' , '6' , 'B' , '7' , '8' , '9' , 'C' , '*' , '0' , '#' ,
'D' ];
            %N= 64;
            fs= 5000;

            index=(vec==ch);

            x1=signal(vec(index),N,fs,vec); %calls the local function
            %PLOT THE SIGNAL SPECTRUM
            N_F=128;
            y1=fftshift(fft(x1,N_F));
            k=fs*(-(N_F/2):(N_F/2)-1)/N_F;
            plot(app.UIAxes_9,k,abs(y1)/N_F);
            xlabel(app.UIAxes_9,'f(Hz)')
            ylabel(app.UIAxes_9,'|X(f)|/N')
            xlim(app.UIAxes_9,[-fs/2 , fs/2]);
            title(app.UIAxes_9,"Spectrum of the signal for character: "+ch)

            out=bpf_bank(x1,N,fs,vec);

            function [h]= bpf_single(wc,fs,L)
                n= 1 :L;
                b= 1 ;
                T= 1 /fs;
                h(n)=b* cos( 2*pi*wc*n*T);
                omega = -pi:0.01:pi;
                H = freqz(h,1,omega);
                B = 1/max(abs(H));
                h = B*h;
            end
            function [x]= signal(key,N,fs,vec)
                t= 1:N;

                [X,Y]= meshgrid ([ 1209 1336 1477 1633 ],[ 697 770 852 941 ]);

                for i = 1 : 16
                    if (key==vec( i ))
                        a= 1 + fix (( i -1 )/ 4 );
                        b= 1 + rem (( i -1 ), 4 );
                    end
                end

                x(t)= sin (2* pi *X(a,b)*t/fs)+ sin (2*pi*Y(a,b)*t/fs);
                noise=randn(1,length(x));
                noise=((max(x)/10 )*noise)/abs(max(noise));
                x=noise+x;

                %plot_signal(X(a,b),Y(a,b),x,N,fs,key);

            end


            function [out]= bpf_bank (x,N,fs,vec)
                wc=[ 697 770 852 941 1209 1336 1477 1633 ];
                RMStable= zeros ( 1 , 8 );
                L=100; %length of time domain impulse response of the filter

                h_arr= zeros(8,L);
                for i=1:8
                    h_arr(i,:)= bpf_single(wc(i),fs,L);
                end
                y_arr= zeros(8, N+L-1);

                for j = 1 : 8
                    h=h_arr(j,:);
                    y_arr(j,:) = conv(h,x);
                    RMStable(j)=rms(y_arr(j,:));
                end
                N_FFT=N+L-1;

                [~, fmax_index]=sort(RMStable, 2 , 'descend' );
                row=min(fmax_index(:, 1 : 2 ),[], 2 );
                col=max(fmax_index(:, 1 : 2 ),[], 2 ) -4 ;
                out=vec( 4 *( row-1 )+col);
```

```
            end
        end

        function [x,fs]= signal(app,ch,fs)
            vec=[ '1' , '2' , '3' , 'A' , '4' , '5' , '6' , 'B' , '7' , '8' , '9' , 'C' , '*' , '0' , '#' ,
'D' ];
            %fs= 8000;
            t = 0:1/fs:0.2;

            [X,Y]= meshgrid ([ 1209 1336 1477 1633 ],[ 697 770 852 941 ]);

            for i = 1 : 16
                if (ch==vec( i ))
                    a= 1 + fix (( i -1 )/ 4 );
                    b= 1 + rem (( i -1 ), 4 );
                    break
                end
            end

            x= 0.1*(sin (2* pi *X(a,b)*t)+ sin (2*pi*Y(a,b)*t) );

         end

         function [y_arr,wc,fs]= filter_output (app,key,N)
            wc=[ 697 770 852 941 1209 1336 1477 1633 ];
            vec=[ '1' , '2' , '3' , 'A' , '4' , '5' , '6' , 'B' , '7' , '8' , '9' , 'C' , '*' , '0' , '#' ,
'D' ];
            L=100; %length of time domain impulse response of the filter
            %N=64;
            fs=5000;
            h_arr= zeros(8,L);
            for i=1:8
                h_arr(i,:)= bpf_single(wc(i),fs,L);
            end
             function [h]= bpf_single(wc,fs,L)
                n= 1 :L;
                b= 1 ;
                T= 1 /fs;
                h(n)=b* cos( 2*pi*wc*n*T);
                omega = -pi:0.01:pi;
                H = freqz(h,1,omega);
                B = 1/max(abs(H));
                h = B*h;
            end
            x=signal(key,N,fs,vec);
            for j = 1 : 8
                h=h_arr(j,:);
                y_arr(j,:) = conv(h,x);

            end


            function [x]= signal(key,N,fs,vec)
            t= 1:N;
            [X,Y]= meshgrid ([ 1209 1336 1477 1633 ],[ 697 770 852 941 ]);

            for i = 1 : 16
                if (key==vec( i ))
                    a= 1 + fix (( i -1 )/ 4 );
                    b= 1 + rem (( i -1 ), 4 );
                end
            end
            x(t)= sin (2* pi *X(a,b)*t/fs)+ sin (2*pi*Y(a,b)*t/fs);
            noise=randn(1,length(x));
            noise=((max(x)/10 )*noise)/abs(max(noise));
            x=noise+x;
            end
        end


        function func1(app,key)
            fs=8000;
            N=64;

            [x,fs]=signal(app,key,fs);
            sound(x,fs);
            temp=app.InputsequenceEditField.Value;
```

```
            app.InputsequenceEditField.Value =[temp key];

            out=decode(app,key,N);
            temp=app.OutputsequenceEditField.Value;
            app.OutputsequenceEditField.Value =[temp out];
            plot_filter_out(app,key,N);
            %app.MainHeading.Value ="Time-domain filter output for character:"+key;
            app.TimedomainfilteroutputLabel.Text="Time-domain filter output for character:"+key;

        end

        function plot_filter_out(app,key,N)
        [y_arr,wc,fs]= filter_output(app,key,N);

        t1=(1:length(y_arr))*(1/fs);

ax_arr=[app.UIAxes,app.UIAxes_2,app.UIAxes_3,app.UIAxes_4,app.UIAxes_5,app.UIAxes_6,app.UIAxes_7,app.UIAxes
_8];

        %sgtitle(app.UIFigure,"Time domain filter output for key: "+key);
        for i=1:8
            y=y_arr(i,:);
            %app.UIFigure;

            %%subplot( 2 , 4 , i );
            plot(ax_arr(i),t1,y);
            title(ax_arr(i),"W_c="+ wc(i) +" Hz" )
            xlabel(ax_arr(i),'Time(s)');
            ylabel(ax_arr(i),'Amplitude');
            ylim(ax_arr(i),[ -1 1 ])
            xlim(ax_arr(i), [0 t1(end)])
        end
        end


    end


    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: Button_1
        function Button_1Pushed(app, event)
            key='1';
            func1(app,key);
        end

        % Button pushed function: Button_2
        function Button_2Pushed(app, event)
            key='2';
            func1(app,key);
        end

        % Button pushed function: Button_3
        function Button_3Pushed(app, event)
            key='3';
            func1(app,key);
        end

        % Button pushed function: AButton
        function AButtonPushed(app, event)
            key='A';
            func1(app,key);
        end

        % Button pushed function: Button_4
        function Button_4Pushed(app, event)
            key='4';
            func1(app,key);
        end

        % Button pushed function: Button_5
        function Button_5Pushed(app, event)
            key='5';
            func1(app,key);
        end
```

```
        % Button pushed function: Button_6
        function Button_6Pushed(app, event)
            key='6';
            func1(app,key);
        end

        % Button pushed function: BButton
        function BButtonPushed(app, event)
            key='B';
            func1(app,key);
        end

        % Button pushed function: Button_7
        function Button_7Pushed(app, event)
            key='7';
            func1(app,key);
        end

        % Button pushed function: Button_8
        function Button_8Pushed(app, event)
            key='8';
            func1(app,key);
        end

        % Button pushed function: Button_9
        function Button_9Pushed(app, event)
            key='9';
            func1(app,key);
        end

        % Button pushed function: CButton
        function CButtonPushed(app, event)
            key='C';
            func1(app,key);
        end

        % Button pushed function: Button_star
        function Button_starPushed(app, event)
            key='*';
            func1(app,key);
        end

        % Button pushed function: Button_0
        function Button_0Pushed(app, event)
            key='0';
            func1(app,key);
        end

        % Button pushed function: Button_hash
        function Button_hashPushed(app, event)
            key='#';
            func1(app,key);
        end

        % Button pushed function: DButton
        function DButtonPushed(app, event)
            key='D';
            func1(app,key);
        end

        % Button pushed function: ClearButton
        function ClearButtonPushed(app, event)
            app.InputsequenceEditField.Value ="";
            app.OutputsequenceEditField.Value ="";

ax_arr=[app.UIAxes,app.UIAxes_2,app.UIAxes_3,app.UIAxes_4,app.UIAxes_5,app.UIAxes_6,app.UIAxes_7,app.UIAxes
_8,app.UIAxes_9];
            for i=1:length(ax_arr)
                cla(ax_arr(i));
            end
        end

        % Button pushed function: DeleteButton
        function DeleteButtonPushed(app, event)
            temp=app.InputsequenceEditField.Value;
            if(length(temp)>=1)
                temp(end)='';
```

```
            app.InputsequenceEditField.Value = temp;
            app.OutputsequenceEditField.Value = temp;
        end


    end
end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.Position = [100 100 1459 745];
        app.UIFigure.Name = 'MATLAB App';

        % Create UIAxes
        app.UIAxes = uiaxes(app.UIFigure);
        title(app.UIAxes, 'Title')
        xlabel(app.UIAxes, 'X')
        ylabel(app.UIAxes, 'Y')
        zlabel(app.UIAxes, 'Z')
        app.UIAxes.Position = [401 512 308 192];

        % Create UIAxes_2
        app.UIAxes_2 = uiaxes(app.UIFigure);
        title(app.UIAxes_2, 'Title')
        xlabel(app.UIAxes_2, 'X')
        ylabel(app.UIAxes_2, 'Y')
        zlabel(app.UIAxes_2, 'Z')
        app.UIAxes_2.Position = [710 512 308 192];

        % Create UIAxes_3
        app.UIAxes_3 = uiaxes(app.UIFigure);
        title(app.UIAxes_3, 'Title')
        xlabel(app.UIAxes_3, 'X')
        ylabel(app.UIAxes_3, 'Y')
        zlabel(app.UIAxes_3, 'Z')
        app.UIAxes_3.Position = [1013 512 308 192];

        % Create UIAxes_4
        app.UIAxes_4 = uiaxes(app.UIFigure);
        title(app.UIAxes_4, 'Title')
        xlabel(app.UIAxes_4, 'X')
        ylabel(app.UIAxes_4, 'Y')
        zlabel(app.UIAxes_4, 'Z')
        app.UIAxes_4.Position = [401.666666666667 277 308 192];

        % Create UIAxes_5
        app.UIAxes_5 = uiaxes(app.UIFigure);
        title(app.UIAxes_5, 'Title')
        xlabel(app.UIAxes_5, 'X')
        ylabel(app.UIAxes_5, 'Y')
        zlabel(app.UIAxes_5, 'Z')
        app.UIAxes_5.Position = [710 277 308 192];

        % Create UIAxes_6
        app.UIAxes_6 = uiaxes(app.UIFigure);
        title(app.UIAxes_6, 'Title')
        xlabel(app.UIAxes_6, 'X')
        ylabel(app.UIAxes_6, 'Y')
        zlabel(app.UIAxes_6, 'Z')
        app.UIAxes_6.Position = [1016 277 308 192];

        % Create UIAxes_7
        app.UIAxes_7 = uiaxes(app.UIFigure);
        title(app.UIAxes_7, 'Title')
        xlabel(app.UIAxes_7, 'X')
        ylabel(app.UIAxes_7, 'Y')
        zlabel(app.UIAxes_7, 'Z')
        app.UIAxes_7.Position = [710 59 308 192];

        % Create UIAxes_8
        app.UIAxes_8 = uiaxes(app.UIFigure);
```

```
title(app.UIAxes_8, 'Title')
xlabel(app.UIAxes_8, 'X')
ylabel(app.UIAxes_8, 'Y')
zlabel(app.UIAxes_8, 'Z')
app.UIAxes_8.Position = [1016 59 308 192];

% Create UIAxes_9
app.UIAxes_9 = uiaxes(app.UIFigure);
title(app.UIAxes_9, 'Title')
xlabel(app.UIAxes_9, 'X')
ylabel(app.UIAxes_9, 'Y')
zlabel(app.UIAxes_9, 'Z')
app.UIAxes_9.Position = [16 48 334 215];

% Create KeypadPanel
app.KeypadPanel = uipanel(app.UIFigure);
app.KeypadPanel.Title = 'Keypad';
app.KeypadPanel.Position = [30 308 297 311];

% Create Button_1
app.Button_1 = uibutton(app.KeypadPanel, 'push');
app.Button_1.ButtonPushedFcn = createCallbackFcn(app, @Button_1Pushed, true);
app.Button_1.Tag = 'Key1';
app.Button_1.Position = [1 227 43 40];
app.Button_1.Text = '1';

% Create Button_2
app.Button_2 = uibutton(app.KeypadPanel, 'push');
app.Button_2.ButtonPushedFcn = createCallbackFcn(app, @Button_2Pushed, true);
app.Button_2.Tag = 'Key2';
app.Button_2.Position = [54 227 43 40];
app.Button_2.Text = '2';

% Create Button_3
app.Button_3 = uibutton(app.KeypadPanel, 'push');
app.Button_3.ButtonPushedFcn = createCallbackFcn(app, @Button_3Pushed, true);
app.Button_3.Tag = 'Key3';
app.Button_3.Position = [110 227 43 40];
app.Button_3.Text = '3';

% Create AButton
app.AButton = uibutton(app.KeypadPanel, 'push');
app.AButton.ButtonPushedFcn = createCallbackFcn(app, @AButtonPushed, true);
app.AButton.Tag = 'KeyA';
app.AButton.Position = [167 227 43 40];
app.AButton.Text = 'A';

% Create Button_4
app.Button_4 = uibutton(app.KeypadPanel, 'push');
app.Button_4.ButtonPushedFcn = createCallbackFcn(app, @Button_4Pushed, true);
app.Button_4.Tag = 'Key4';
app.Button_4.Position = [1 174 43 40];
app.Button_4.Text = '4';

% Create Button_5
app.Button_5 = uibutton(app.KeypadPanel, 'push');
app.Button_5.ButtonPushedFcn = createCallbackFcn(app, @Button_5Pushed, true);
app.Button_5.Tag = 'Key5';
app.Button_5.Position = [54 173 43 40];
app.Button_5.Text = '5';

% Create Button_6
app.Button_6 = uibutton(app.KeypadPanel, 'push');
app.Button_6.ButtonPushedFcn = createCallbackFcn(app, @Button_6Pushed, true);
app.Button_6.Tag = 'Key6';
app.Button_6.Position = [110 173 43 40];
app.Button_6.Text = '6';

% Create BButton
app.BButton = uibutton(app.KeypadPanel, 'push');
app.BButton.ButtonPushedFcn = createCallbackFcn(app, @BButtonPushed, true);
app.BButton.Tag = 'KeyB';
app.BButton.Position = [167 173 43 40];
app.BButton.Text = 'B';

% Create Button_7
app.Button_7 = uibutton(app.KeypadPanel, 'push');
```

```
app.Button_7.ButtonPushedFcn = createCallbackFcn(app, @Button_7Pushed, true);
app.Button_7.Tag = 'Key7';
app.Button_7.Position = [1 121 43 40];
app.Button_7.Text = '7';

% Create Button_8
app.Button_8 = uibutton(app.KeypadPanel, 'push');
app.Button_8.ButtonPushedFcn = createCallbackFcn(app, @Button_8Pushed, true);
app.Button_8.Tag = 'Key8';
app.Button_8.Position = [54 121 43 40];
app.Button_8.Text = '8';

% Create Button_9
app.Button_9 = uibutton(app.KeypadPanel, 'push');
app.Button_9.ButtonPushedFcn = createCallbackFcn(app, @Button_9Pushed, true);
app.Button_9.Tag = 'Key9';
app.Button_9.Position = [110 121 43 40];
app.Button_9.Text = '9';

% Create CButton
app.CButton = uibutton(app.KeypadPanel, 'push');
app.CButton.ButtonPushedFcn = createCallbackFcn(app, @CButtonPushed, true);
app.CButton.Tag = 'KeyC';
app.CButton.Position = [167 121 43 40];
app.CButton.Text = 'C';

% Create Button_star
app.Button_star = uibutton(app.KeypadPanel, 'push');
app.Button_star.ButtonPushedFcn = createCallbackFcn(app, @Button_starPushed, true);
app.Button_star.Tag = 'Keystar';
app.Button_star.Position = [1 65 43 40];
app.Button_star.Text = '*';

% Create Button_0
app.Button_0 = uibutton(app.KeypadPanel, 'push');
app.Button_0.ButtonPushedFcn = createCallbackFcn(app, @Button_0Pushed, true);
app.Button_0.Tag = 'Key0';
app.Button_0.Position = [54 65 43 40];
app.Button_0.Text = '0';

% Create Button_hash
app.Button_hash = uibutton(app.KeypadPanel, 'push');
app.Button_hash.ButtonPushedFcn = createCallbackFcn(app, @Button_hashPushed, true);
app.Button_hash.Tag = 'Keyhash';
app.Button_hash.Position = [110 65 43 40];
app.Button_hash.Text = '#';

% Create DButton
app.DButton = uibutton(app.KeypadPanel, 'push');
app.DButton.ButtonPushedFcn = createCallbackFcn(app, @DButtonPushed, true);
app.DButton.Tag = 'KeyD';
app.DButton.Position = [167 65 43 40];
app.DButton.Text = 'D';

% Create ClearButton
app.ClearButton = uibutton(app.KeypadPanel, 'push');
app.ClearButton.ButtonPushedFcn = createCallbackFcn(app, @ClearButtonPushed, true);
app.ClearButton.Position = [4 22 93 36];
app.ClearButton.Text = 'Clear';

% Create DeleteButton
app.DeleteButton = uibutton(app.KeypadPanel, 'push');
app.DeleteButton.ButtonPushedFcn = createCallbackFcn(app, @DeleteButtonPushed, true);
app.DeleteButton.Position = [110 22 102 36];
app.DeleteButton.Text = 'Delete';

% Create OutputsequenceEditFieldLabel
app.OutputsequenceEditFieldLabel = uilabel(app.UIFigure);
app.OutputsequenceEditFieldLabel.HorizontalAlignment = 'right';
app.OutputsequenceEditFieldLabel.Position = [40 682 97 22];
app.OutputsequenceEditFieldLabel.Text = 'Output sequence';

% Create OutputsequenceEditField
app.OutputsequenceEditField = uieditfield(app.UIFigure, 'text');
app.OutputsequenceEditField.Position = [152 676 175 34];

% Create InputsequenceEditFieldLabel
```

```
            app.InputsequenceEditFieldLabel = uilabel(app.UIFigure);
            app.InputsequenceEditFieldLabel.HorizontalAlignment = 'right';
            app.InputsequenceEditFieldLabel.Position = [49 638 87 22];
            app.InputsequenceEditFieldLabel.Text = 'Input sequence';

            % Create InputsequenceEditField
            app.InputsequenceEditField = uieditfield(app.UIFigure, 'text');
            app.InputsequenceEditField.Position = [151 632 175 34];

            % Create TimedomainfilteroutputLabel
            app.TimedomainfilteroutputLabel = uilabel(app.UIFigure);
            app.TimedomainfilteroutputLabel.FontSize = 14;
            app.TimedomainfilteroutputLabel.FontWeight = 'bold';
            app.TimedomainfilteroutputLabel.Position = [750 709 348 22];
            app.TimedomainfilteroutputLabel.Text = 'Time-domain filter output';

            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = EXP_3

            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)

            if nargout == 0
                clear app
            end
        end

        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```