Implementaion of DTMF Decoding with Matlab

Zheng GONG
Department of Electrical and Computer Engineering
University of Maryland
College Park, US
Email: joeygong@termpmail.umd.edu

***Abstract*—In this paper two method of decoding DTMF(Dual-tone multi-frequency) signaling are accomplished with Matlab. First of them uses the method of FFT and the other one with a filter approaching. Both methods decode an input wave file including multi-digits of code into a readable string of numbers accurately. The decoder can detect every single signal in the input multi-digits wave file then split them so periodic dial signals are not required.**

***Keywords*—DTMF, decoder, Matlab, multi-digits.**

I. Introduction

DTMF is used for telecommunication signaling over analog telephone lines in the voice-frequency band between telephone handsets and other communications devices and the switching center. To make the whole system of telephone communication works, a DTMF decoder plays an important role.

A. DTMF

In DTMF, a 4×4 matrix is formed by each row representing a low frequency, and each column representing a high frequency, as shown in table **??**. Numbers from 0 to 9, letters from A to D and symbols $\star$ and $\#$ are represented by a combination of a low frequency and a high frequency. Also, special tone frequencies are used to represent busy signal, dail tone and so on. frequencies used here are all lower than 600 and differ by countries and will not be disscussed here.

TABLE I. DTMF Keypad Frequencies Match

Frequency/Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	$\star$	0	#	0

B. FFT

Fast Fourier Transform(FFT) is an algorithm to make computation of Discrete Fourier Transform(DFT) faster. By DFT, signals are converted from time domain to frequency domain, which makes it easier to do analysis. In this project, for an single tone of dial, with effect of noise, there should be more than two frequencies having non-zero amplitude after DFT, but two of them should be relatively larger and they are actually the target frequencies.

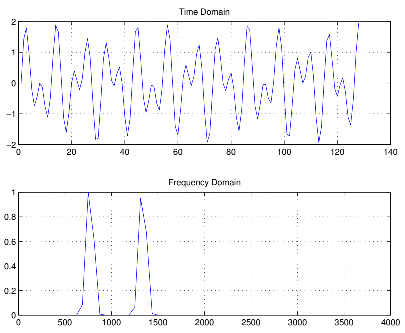


Fig. 1. Example FFT of dialing '5'

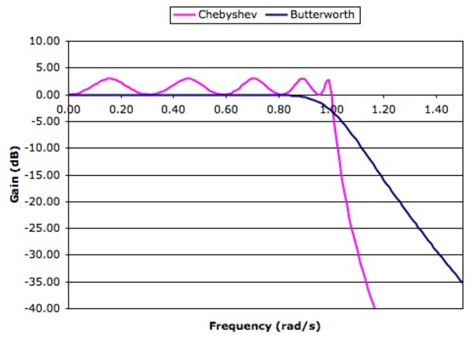


Fig. 2. Compare of Filters

The figure **??** shows an instance of the FFT of signal '5', two of the frequency peak can be easily found. Once we can find the two peaks of the signal, the signal can be identified.

C. Type I Chebyshev Filter

Chebyshev filters(Type I) are filters having a steeper roll-off and more passband ripple than the Butterworth filter(As shown in figure **??**). As in this application, there are frequencies need to be filtered but closed to the needed frequency(The highest busy tone is 620Hz and the lowest dial tone is 697Hz, only 80Hz difference). So to minimize the error, we decided to choose Chebyshev filter rather than Butterworth filter in sacrificing of ripple in the passband.

In addition, we only use the filter to identify frequency but not going to playback, the distortion in passband does not matter.

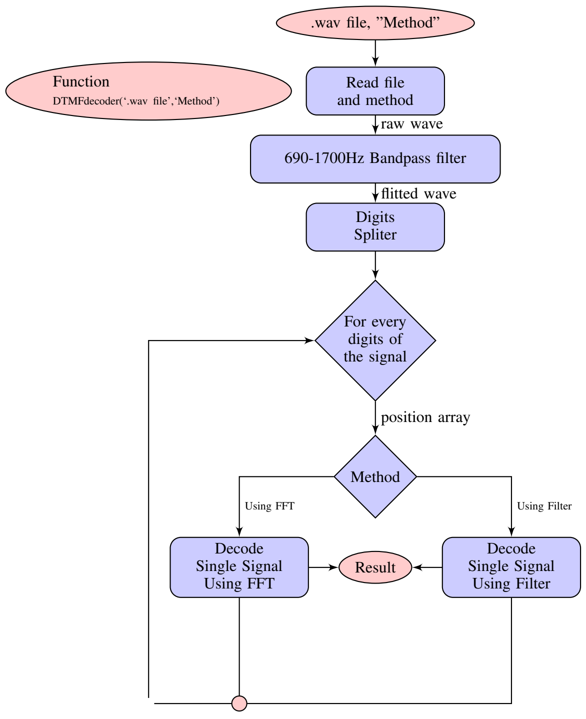

.wav file, "Method"
Function
DTMFdecoder('.wav file','Method')
Read file and method
raw wave
690-1700Hz Bandpass filter
flitted wave
Digits Spliter
For every digits of the signal
position array
Method
Using FFT
Using Filter
Decode Single Signal Using FFT
Result
Decode Single Signal Using Filter

Fig. 3. Flow Chart for the decoder

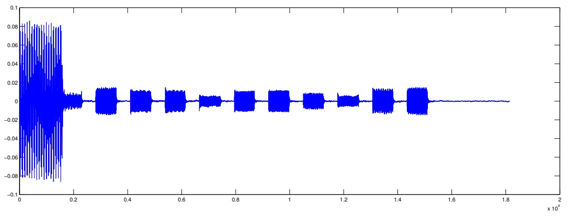


Fig. 4. Waveform of Raw Input Wave

II. Proposed Approach

The decoder is accomplished following the flow chart in figure **??**.

A. Read file

To read the wave file using Matlab, we use the commend:

```
[RawWave,Fs]=wavread(WaveFile);%where ...
    WaveFile equals to the string of file name
```

Then we get the wave as a array in RawWave. In our occasion, we tested a real recored wave file 'realrec1.wav' and get its wave form in figure **??**

B. Pre-filter

From figure **??** we find that there exits lots of wave with large amplitude that we don't what and can cause trouble to

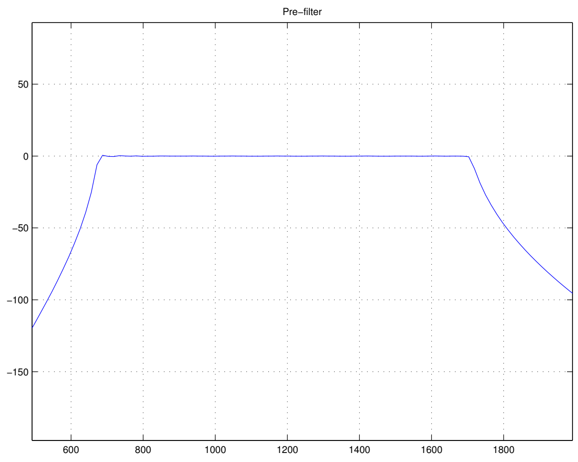


Fig. 5. Raw Wave After the Pre-filter

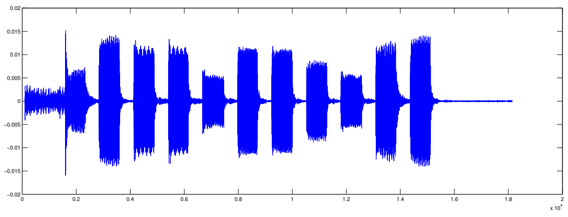


Fig. 6. Pre-filter Frequency Response

wave analyze. So we need a band pass filter to get ride of them.

Using the Matlab code:

```
f1=680;f3=1700;
fsl=640;fsh=1740;
rp=0.1;rs=20;
wp1=2*pi*f1/Fs;
wp3=2*pi*f3/Fs;
wsl=2*pi*fsl/Fs;
wsh=2*pi*fsh/Fs;
wp=[wp1 wp3];
ws=[wsl wsh];
[n,wn]=cheb1ord(ws/pi,wp/pi,rp,rs);
[bz1,az1]=cheby1(n,rp,wp/pi);
FiltedWave=filter(bz1,az1,RawWave);
```

we get the Chebyshev filter in figure **??**. Then the filtered wave will be cut off on the low frequency and the unnecessary high frequency, as shown in figure **??**.

C. Split multi-digits

After we get the wave that only contains the frequencies components we want, we need to cut it into small, monotonous(or should be 'bitonous' here) pieces so that we can analyze them. To achieve this, we need a function 'WaveSpliter', who takes in the raw wave and gives out an

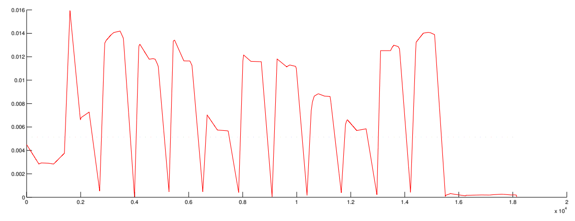


Fig. 7. Envelope of Filtered Raw Wave

$n \times 2$ array that tell us where the dial tone starts and where it ends.

$$PositionArray = \begin{pmatrix} 1st\ dial & 1st\ dial \\ begin\ point & end\ point \\ \vdots & \vdots \\ \vdots & \vdots \\ nth\ dial & nth\ dial \\ begin\ point & end\ point \end{pmatrix}$$

To get this, we first find the envelope of the signals, as shown in figure **??** Then we calculate the mean of the amplitude. Then scale it with proper value, here we divide it by 1.5, then take it like a threshold value. Find every point on the envelope that cross the threshold value, recored it as the beginning or end point of the dialing tone.

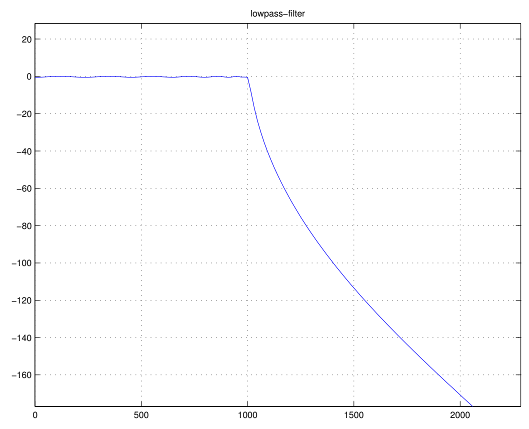
D. Decode Single Digits Using FFT

Since we know where the monotonous signal begin and end, we can analyze them through different ways. The first way of approaching is to use FFT.

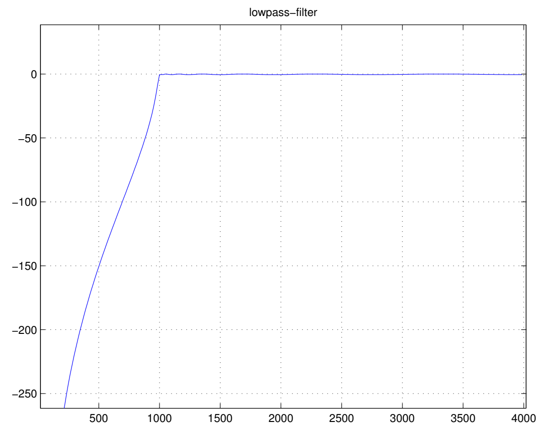
The FFT of the signal will have two peaks, to identify them, we first need to filter the signal through two different filter, lowpass filter at 1000Hz and conjugating high pass filter, as shown in figure **??**. After this, we do FFT to the two signal, and now each signal in frequency domain should has only one peak. Now we can find the maximum of the function to find out the peak value and identify the signal.

E. Decode Single Digits Using Filter Bank

For the filter bank approaching, we test the wave through two bandpass filter separately. There are 8 kind of filters, each of them can filter only one frequency of the DTMF and there are 4×4 combinations. After the signal has passed through the two filter, we add the maximum of the two result signal. At last we will get 16 groups of maximum value, and the maximum of these maximum is the dial tone we want to decide. Figure **??** shows the occasion using filter band method to identify signal '5'



(a) Lowpass Filter



(b) Highpass Filter

Fig 8. Filter Fefore FFT

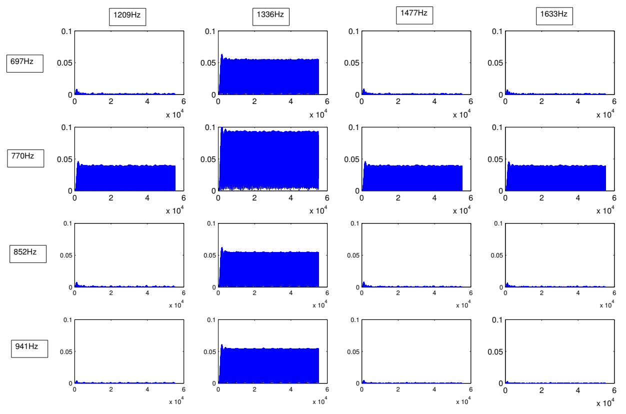


Fig. 9. Filter Band Identify Signal 5

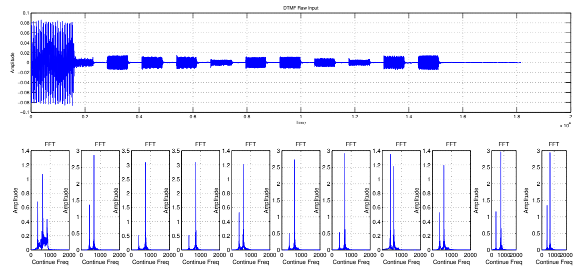


Fig. 10. Filter Band Identify Signal 5

III. Conclusion

Finally we accomplished the function ‘DTMFdecoder’. It has two parameters, the inout wave file name and the methode string. The function can be used as:

```
>>DTMFdecoder('realrec1.wav','FFT'); %for ...
    using FFT method to decode file realrec1.wav
>>DTMFdecoder('realrec1.wav','Filter'); %for ...
    using Filter method to decode file ...
    realrec1.wav
```

and the out put will be like:

```
>> DTMFdecoder('realrec1.wav','FFT');%for ...
    using FFT method to decode file realrec1.wav
The code is 18664557438
>>DTMFdecoder('realrec1.wav','Filter'); %for ...
    using Filter method to decode file ...
    realrec1.wav
The code is 18664557438
```

Further, if the option ’graph’ sets to ’gryphz_on’, the input wave will be played back and the waveform and the FFT of every digits will be shown, see figure **??**.

```
>> DTMFdecoder('realrec1.wav','FFT','graph_on');
%for using FFT method to decode file ...
    realrec1.wav with graph on
The code is 18664557438
```

Appendix A
Code for the Whole Porject

A. *DTMFdecoder.m*

```
function result = DTMFdecoder(WaveFile,Method,Graph)
%read file
if nargin==1,Method='FFT';Graph='graph_off';end
if nargin==2,Graph='graph_off';end
[RawWave,Fs,Bits]=wavread(WaveFile);
assert(strcmp(Method,'FFT')||strcmp(Method,'Filter'),...
    'ERROR: invalid method! The method shoud be ''FFT'' or ''Filter''.');
assert(strcmp(Graph,'graph_on')||strcmp(Graph,'graph_off'),...
    'ERROR: invalid Graph opintion! it shoud be ''graph_on'' or ...
        ''graph_off''.');

if strcmp(Graph,'graph_on'),sound(RawWave,Fs,Bits);end

% Prefileter
f1=690;f3=1700;
fsl=630;fsh=1800;
rp=1;rs=15;
wp1=2*pi*f1/Fs;
wp3=2*pi*f3/Fs;
wsl=2*pi*fsl/Fs;
wsh=2*pi*fsh/Fs;
wp=[wp1 wp3];
ws=[wsl wsh];
[n,wn]=cheb1ord(ws/pi,wp/pi,rp,rs);
[bz1,az1]=cheby1(n,rp,wp/pi);
FiltedWave=filter(bz1,az1,RawWave);
FiltedWave(1:100)=0;

%position signal
HalfSampleLength=1024;
EffPosition = WaveSpliter(FiltedWave);
a=size(EffPosition);

%draw raw wave
if strcmp(Graph,'graph_on')

set(gcf,'Position',[0 20 1200 500]);
set(gca,'Position',[0 0 1 1]);
figure_FontSize=8;
set(get(gca,'XLabel'),'FontSize',figure_FontSize,'Vertical','top');
set(get(gca,'YLabel'),'FontSize',figure_FontSize,'Vertical','middle');
set(findobj('FontSize',10),'FontSize',figure_FontSize);
set(findobj(get(gca,'Children'),'LineWidth',0.5),'LineWidth',2);

subplot(2,a(1),1:a(1))
plot(RawWave);
title('DTMF Raw Input');xlabel('Time');
ylabel('Amplitude');grid;
end

for i=1:a(1)
    %decode single
    MidSig = ceil((EffPosition(i,1)+EffPosition(i,2))/2);
    SigBegin = MidSig-ceil((EffPosition(i,2)-EffPosition(i,1))*3/10);
    SigEnd = MidSig+ceil((EffPosition(i,2)-EffPosition(i,1))*3/10);

    if strcmp(Method,'FFT')
        result(i) = DTMFdecoder_single_FFT(...
            FiltedWave(SigBegin:SigEnd),Fs);
    elseif strcmp(Method,'Filter')
        result(i) = DTMFdecoder_single_Filter(...
            FiltedWave(SigBegin:SigEnd),Fs);
    end
    %do fft and draw
    if strcmp(Graph,'graph_on')
    FFTWave=abs(fft(FiltedWave...
       (SigBegin:SigEnd),2*HalfSampleLength));
    subplot(2,a(1),a(1)+i)
    plot(Fs/length(FFTWave)/2.*(1:length(FFTWave)/2)...
        ,FFTWave(1:length(FFTWave)/2));
    title('FFT');xlabel('Continue Freq');
    ylabel('Amplitude');grid;
    end
end

disp( ['The code is ' result])
end
```

B. *WaveSpliter.m*

```
function Result = WaveSpliter(RawWave)


Env = envelope([1:length(RawWave)],abs(RawWave),400,'top');
Avr = mean(abs(RawWave))/1.5;
ResultRowIndex=1;ResultColIndex2=1;
IsDailing = 0;
for WaveIndex=1:length(RawWave)
    if Env(WaveIndex)>Avr
        if IsDailing==0
            Result(ResultRowIndex,1)=WaveIndex;
            IsDailing = 1;
        end

    else
        if IsDailing==1
            Result(ResultRowIndex,2)=WaveIndex;
            IsDailing=0;
            ResultRowIndex=ResultRowIndex+1;
        end

    end
end
% wave end
if IsDailing==1
   Result(ResultRowIndex,2)=length(RawWave);
end
end
```

C. *envelope.m*

```matlab
function [env] = envelope(x_data, y_data, view, side)
% Function call: env_secant(x_data, y_data, view, side)
% Calculates the top envelope of data <y_data> over <x_data>.
% Method used: 'secant-method'
% env_secant() observates the max. slope of about <view> points,
% and joints them to the resulting envelope.
% An interpolation over original x-values is done finally.
% <side> ('top' or 'bottom') defines which side to evolve.
% Author: Andreas Martin, Volkswagen AG, Germany


side = strcmpi( {'top','bottom'}, side ) * [ 1 ; -1 ];

assert( view > 1, ...
       'Parameter <view> too small!' );
assert( ndims (x_data) == 2, ...
       'Parameter <x_data> has to be vector type!' );
assert( size (x_data, 1) == 1 || size (x_data, 2) == 1, ...
       'Parameter <x_data> has to be vector type (Nx1)!' );
assert( ndims (y_data) == 2, ...
       'Parameter <y_data> has to be vector type (Nx1)!' );
assert( size (y_data, 1) == 1 || size (y_data, 2) == 1, ...
       'Parameter <y_data> has to be vector type (Nx1)!' );
assert( length (x_data) == length (y_data), ...
       'Parameters <x_data> and <y_data> must have same length!' );
assert( side ≠ 0, ...
       'Parameter <side> must be ''top'' or ''bottom''' );

y_data = y_data(:);
data_len = length( y_data );
x_new = [];
y_new = [];

i = 1;
while i < data_len;
    ii = i+1:min( i + view, data_len );
    [ m, idx ] = max( ( y_data(ii) - y_data(i) ) ./ (ii-i)' .* side );

    % Equidistant x_data assumed! Use next row instead, if not:
    %[ m, idx ] = max( ( y_data(ii) - y_data(i) ) ./ ( x_data(ii) - ...
           x_data(i) ) * side );

    % New max. slope: store new "observation point"
    i = i + idx;
    x_new = [ x_new x_data(i) ];
    y_new = [ y_new y_data(i) ];
end;

env = interp1( x_new, y_new, x_data, 'linear', 'extrap' );
```

D. DTMFdecoder_single_FFT.m

```matlab
function ResultCode = DTMFdecoder_single_FFT(PureWave,Fs)
Rp=0.5;
As=25;
Fpass = 1000;
Fstop = 1100;
wp = 2*pi*Fpass/Fs;
ws = 2*pi*Fstop/Fs;
[N,Wn]=cheb1ord(wp/pi,ws/pi,Rp,As);

[bl,al]=cheby1(N,Rp,Wn);
RawWaveFilt_L=filter(bl,al,PureWave);
FFTWave_L=fft(RawWaveFilt_L,length(PureWave));
FFTWave_L_Mag=abs(FFTWave_L(1:ceil(length(PureWave)/2)));

[bh,ah]=cheby1(N,Rp,Wn,'high');
RawWaveFilt_H=filter(bh,ah,PureWave);
FFTWave_H=fft(RawWaveFilt_H,length(PureWave));
FFTWave_H_Mag=abs(FFTWave_H(1:ceil(length(PureWave)/2)));

m=max(abs(FFTWave_L_Mag));n=max(abs(FFTWave_H_Mag));
o=find(m==FFTWave_L_Mag);p=find(n==FFTWave_H_Mag);
j=((o-1)*Fs)/length(PureWave);
k=((p-1)*Fs)/length(PureWave);

ResultCode = '';
if j≤732.59 & k≤1270.91;
    ResultCode = '1';
   elseif j≤732.59 & k≤1404.73;
    ResultCode = '2';
   elseif j≤732.59 & k≤1553.04;
    ResultCode = '3';
   elseif j≤732.59 & k>1553.05;
    ResultCode = 'A';
   elseif j≤809.96 & k≤1270.91;
    ResultCode = '4';
   elseif j≤809.96 & k≤1404.73;
    ResultCode = '5';
   elseif j≤809.96 & k≤1553.04;
    ResultCode = '6';
   elseif j≤809.96 & k>1553.05;
    ResultCode = 'B';
   elseif j≤895.39 & k≤1270.91;
    ResultCode = '7';
   elseif j≤895.39 & k≤1404.73;
    ResultCode = '8';
   elseif j≤895.39 & k≤1553.04;
    ResultCode = '9';
   elseif j≤895.39 & k>1553.05;
    ResultCode = 'C';
   elseif j>895.40 & k≤1270.91;
    ResultCode = '*';
   elseif j>895.40 & k≤1404.73;
    ResultCode = '0';
   elseif j>895.40 & k≤1553.04;
    ResultCode = '#';
   elseif j>895.40 & k>1553.05;
    ResultCode = 'D';
end
end
```

E. DTMFdecoder_single_Filter.m

```matlab
function ResultCode = DTMFdecoder_single_Filter(PureWave,Fs)
DTMFCell=...
    {...
    'Tone1/Tone2'   '1209'  '1336'  '1477'  '1633';...
            '697'   '1'     '2'     '3'     'A';...
            '770'   '4'     '5'     '6'     'B';...
            '852'   '7'     '8'     '9'     'C';...
            '941'   '*'     '0'     '#'     'D'...
    };
Code=DTMFCell( 2:end,2:end)';
Tone1=cellfun(@(x) str2num(x), DTMFCell(1,2:end));
Tone2=cellfun(@(x) str2double(x), DTMFCell(2:end,1)');
[ToneMat1 ToneMat2] =ndgrid(Tone1,Tone2);
k=1;
MaxAmplitude=zeros(numel(Tone1)*numel(Tone2),1);

for TonePair=[ToneMat1(:)';ToneMat2(:)']
    if TonePair(1)≠TonePair(2)
        zewave = PureWave;
        fa1=TonePair(1)-20;fa3=TonePair(1)+20;%
        fsal=TonePair(1)-40;fsah=TonePair(1)+40;%
        rpa=0.1;rsa=20;%
        wpa1=2*pi*fa1/Fs;
        wpa3=2*pi*fa3/Fs;
        wsal=2*pi*fsal/Fs;
        wsah=2*pi*fsah/Fs;
        wpa=[wpa1 wpa3];
        wsa=[wsal wsah];
        [na,wna]=cheb1ord(wsa/pi,wpa/pi,rpa,rsa);
        [bza1,aza1]=cheby1(na,rpa,wpa/pi);
        FiltedWave_1=filter(bza1,aza1,zewave);

        fb1=TonePair(2)-20;fb3=TonePair(2)+20;%
        fsbl=TonePair(2)-40;fsbh=TonePair(2)+40;%
        rpb=0.1;rsb=20;%
        wpb1=2*pi*fb1/Fs;
        wpb3=2*pi*fb3/Fs;
        wsbl=2*pi*fsbl/Fs;
        wsbh=2*pi*fsbh/Fs;
        wpb=[wpb1 wpb3];
        wsb=[wsbl wsbh];
        [nb,wnb]=cheb1ord(wsb/pi,wpb/pi,rpb,rsb);
        [bzb1,azb1]=cheby1(nb,rpb,wpb/pi);
        FiltedWave_2=filter(bzb1,azb1,zewave);

       MaxAmplitude(k)=max(abs(FiltedWave_1))+max(abs(FiltedWave_2));
    else
       MaxAmplitude(k)=0;
    end
 k=k+1;
end
[val Indx]=max(MaxAmplitude);

ResultCode=Code{Indx};
end
```

APPENDIX B

Appendix two text goes here.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.