LAB 3:

Dinesh Vaithyalingam Gangatharan
Harsha Nandeeshappa Krishnarajanagar

Preparation

5.1 –

a – [0 4 0 0]

b – [0 2 0 2]

Preparation

5.1 DFT computing

$x(n) = e^{j0.5\pi n} \quad n = 0, \dots, 3$

$$X(k) = \sum_{n=0}^{3} x(n) e^{-j\frac{2\pi}{N}kn}$$

$$X(0) = \sum_{0}^{3} x(n) = 0$$

$$X(1) = \sum_{0}^{3} x(n) e^{-j\frac{2\pi}{N}n} = 4$$

$$X(2) = \sum_{0}^{3} x(n) e^{-j\frac{2\pi 2n}{N}} = 0$$

$$X(3) = \sum_{0}^{3} x(n) e^{-j\frac{2\pi 3n}{N}} = 0$$

$$\cos(\pi n) = \frac{e^{j\pi n} + e^{-j\pi n}}{2}$$

$$DFT(\cos(0.5\pi n)) = \frac{X(k) + X(N-k)}{2}$$

$$\therefore X(\cos(0.5\pi n)) = (0, 2, 0, 2)$$

5.2 DTFT of a regular window of length N

$$\text{Window} \quad W(e^{j\omega}) = \sum_{n=0}^{N-1} w(n) e^{-j\omega n}$$

$$= \sum_{n=0}^{N-1} e^{-j\omega n}$$

$$= \frac{1 - e^{j\omega N}}{1 - e^{-j\omega}}$$

$$= \frac{e^{-j\omega\frac{N}{2}} \left(e^{j\omega\frac{N}{2}} - e^{-j\omega\frac{N}{2}}\right)}{e^{-j\omega/2}\left(e^{j\omega/2} - e^{-j\omega/2}\right)}$$

$$= e^{-j\omega\frac{N}{2}} \frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

5.3

5.3 $\bar{x}(n) = e^{j w_0 n}$

$x(n) = \bar{x}(n)\, w(n)$ w(n) is window function

$X(e^{jw}) = \frac{1}{2\pi}\left(\hat{X}(e^{jw}) \circledast W(e^{jw})\right)$

$= \underline{W\left(e^{j(w-w_0)}\right)}$

$X(e^{jw})$ is shifted by w_0 in frequency domain

5.4 –

Sidelobe level - Sidelobe level is the peak of energy in frequency response plot of the window signal between the first and second zero crossing. There can be many sidelobes present and their level are between consequent zero crossing.

Mainlobe width - The mainlobe width in frequency response of a window is the region
between the first zero crossing on either side of the zero frequency(origin)

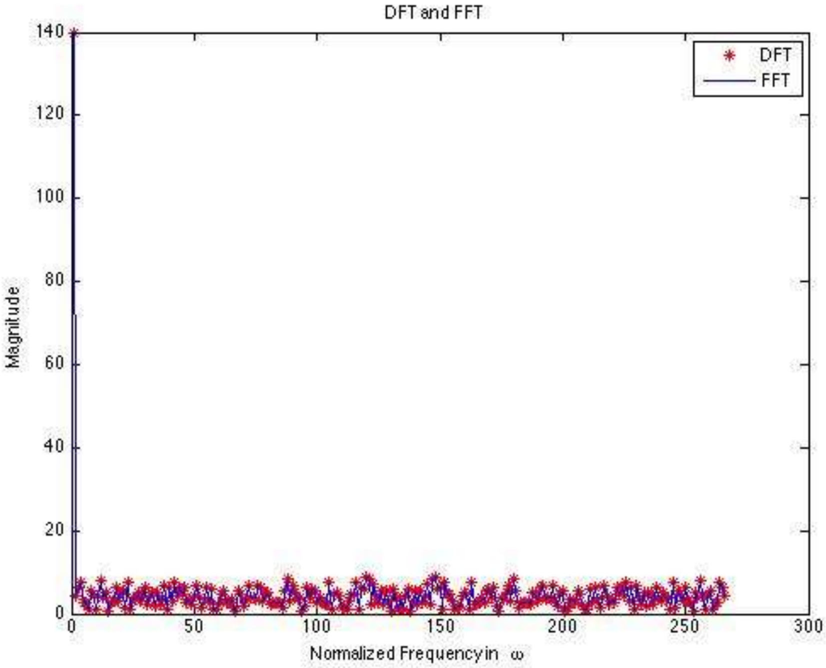
5.5 –
Infinite length of signals – DTFT are infinitely long, therefore computationally intensive. This can be addressed by truncated signals by exploiting the periodic nature.
Continuous nature of DTFT - Instead of storing $X(e^{jw})$ for every possible w, we can store N evenly spaced values of w.

6.1 Computational Complexity of the DFT/FFT

Code

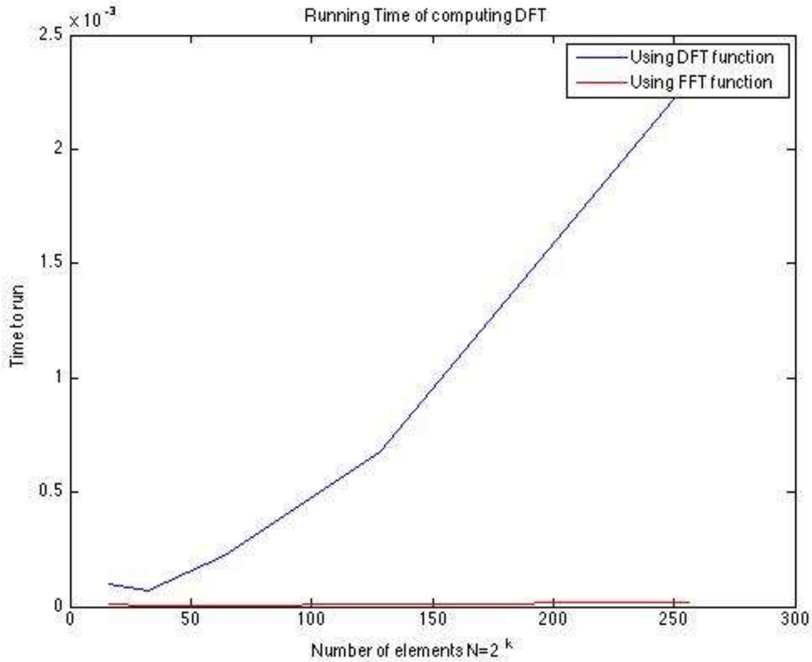
```
%% Computational Complexity of DFT/FFT
%% 6.1.1 Direct DFT and result comparision with FFT

x = rand(256,1);
dftXk = dft(x);
fftXk = fft(x);

figure(1)
plot(1:numel(x),abs(dftXk),'r*',1:numel(x),abs(fftXk),'b-');
title('DFT and FFT');
xlabel('Normalized Frequency in \omega'), ylabel('Magnitude');
legend('DFT','FFT');

dft_t=[];
fft_t=[];
for k = 4:8
    N=2^k;
    x=rand(N,1);
    tic
    dft(x);
    dft_t=[dft_t toc];
    tic
    fft(x);
    fft_t=[fft_t toc];
end

k=4:8;
figure(2)
plot(2.^k,dft_t,2.^k,fft_t,'r' );
legend('Using DFT function','Using FFT function');
xlabel('Number of elements N=2^k');
ylabel('Time to run');
title('Running Time of computing DFT');
```

Result



DFT and FFT
DFT
FFT
Magnitude
Normalized Frequency in ω

s

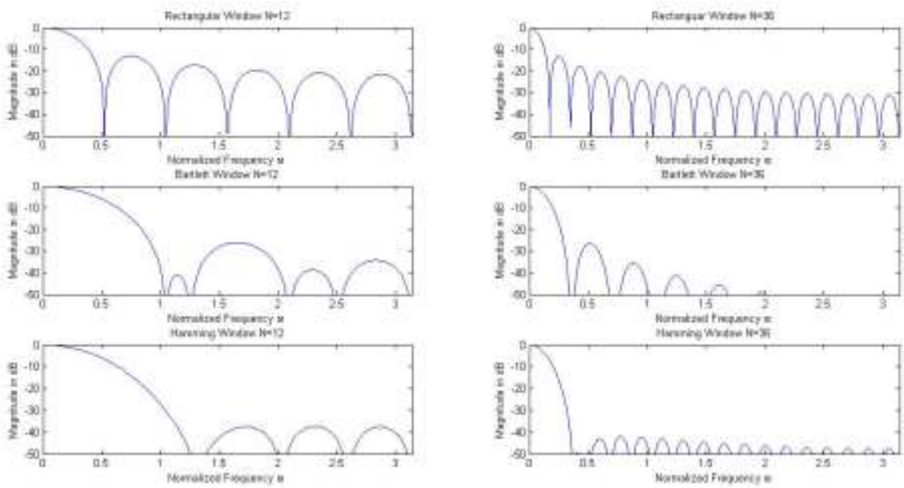


Running Time of computing DFT
Using DFT function
Using FFT function
Time to run
Number of elements N=2 k

6.2 Discret-Time Fourier Transform

```
% 6.2 (1)

function [ X,w ] = dtft( x )
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
M=length(x);
X=fft(x,100*M);
n=0:(100*M-1);
w=2*pi*n/(100*M);
end



% 6.2 (2)
clc;
clear all;

%Varying the window Lengths
windowLen1=12;
windowLen2=36;

% Different windows

recWindow1=rectwin(windowLen1);
recWindow2=rectwin(windowLen2);

bartlettWin1=bartlett(windowLen1);
bartlettWin2=bartlett(windowLen2);

hammingWin1=hamming(windowLen1);
hammingWin2=hamming(windowLen2);

figure(1);

%Rectangular Window N=12
[X1,w1]=dtft(recWindow1);
subplot(3,2,1);
plot(w1,20*log10(abs(X1)/max(abs(X1))));
title('Rectangular Window N=12');
xlim([0 pi]);ylim([-50 0]);
xlabel('Normalized Frequency \omega');
ylabel('Magnitude in dB');

%Rectangular Window N=36
[X2,w2]=dtft(recWindow2);
subplot(3,2,2);
plot(w2,20*log10(abs(X2)/max(abs(X2))));
title('Rectanguar Window N=36');
xlim([0 pi]);ylim([-50 0]);
xlabel('Normalized Frequency \omega');
ylabel('Magnitude in dB');

%bartlett Window N=12
[X3,w3]=dtft(bartlettWin1);
subplot(3,2,3);
plot(w3,20*log10(abs(X3)/max(abs(X3))));
title('Bartlett Window N=12');
```

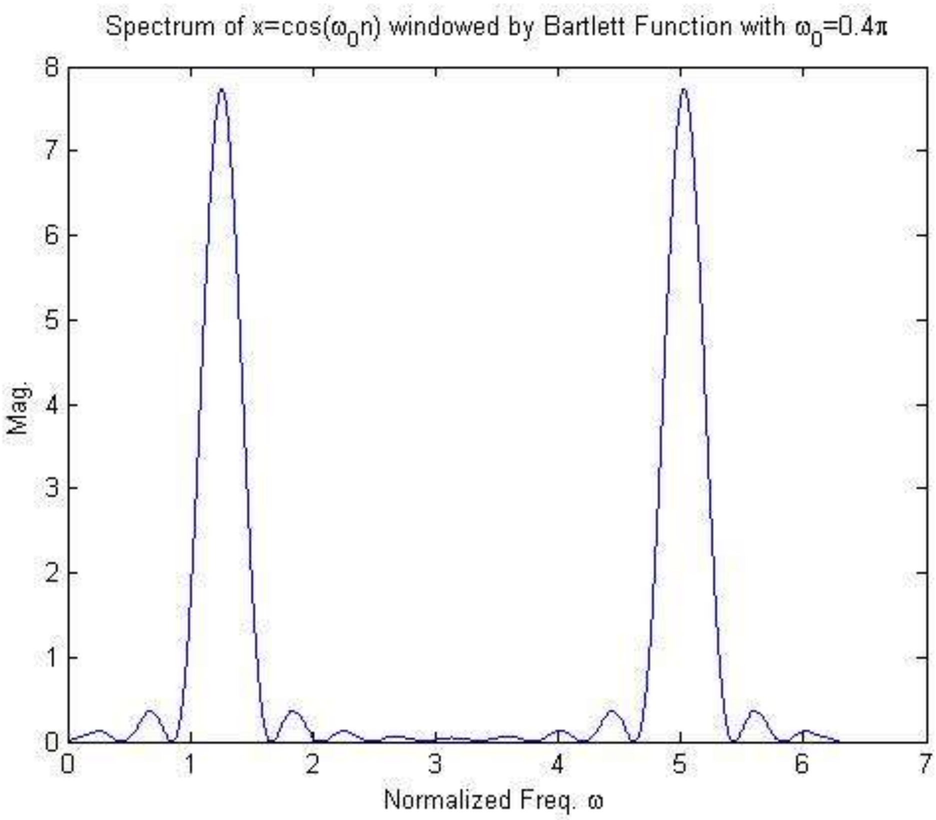
```
xlim([0 pi]);ylim([-50 0]);
xlabel('Normalized Frequency \omega');
ylabel('Magnitude in dB');

%bartlett Window N=36
[X4,w4]=dtft(bartlettWin2);
subplot(3,2,4);
plot(w4,20*log10(abs(X4)/max(abs(X4))));
title('Bartlett Window N=36');
xlim([0 pi]);ylim([-50 0]);
xlabel('Normalized Frequency \omega');
ylabel('Magnitude in dB');

%Hamming Window N=12
[X5,w5]=dtft(hammingWin1);
subplot(3,2,5);
plot(w5,20*log10(abs(X5)/max(abs(X5))));
title('Hamming Window N=12');
xlim([0 pi]);ylim([-50 0]);
xlabel('Normalized Frequency \omega');
ylabel('Magnitude in dB');

%Hamming Window N=36
[X6,w6]=dtft(hammingWin2);
subplot(3,2,6);
plot(w6,20*log10(abs(X6)/max(abs(X6))));
title('Hamming Window N=36');
xlim([0 pi]);ylim([-50 0]);
xlabel('Normalized Frequency \omega');
ylabel('Magnitude in dB');
```



Increasing the samples of the windowing function improves the Mainlobe width of the frequency response. The Sidelobe width is almost the same for both the cases.
Hamming has the highest sidelobe width suppression
Rectangular window has the smallest mainlobe width.

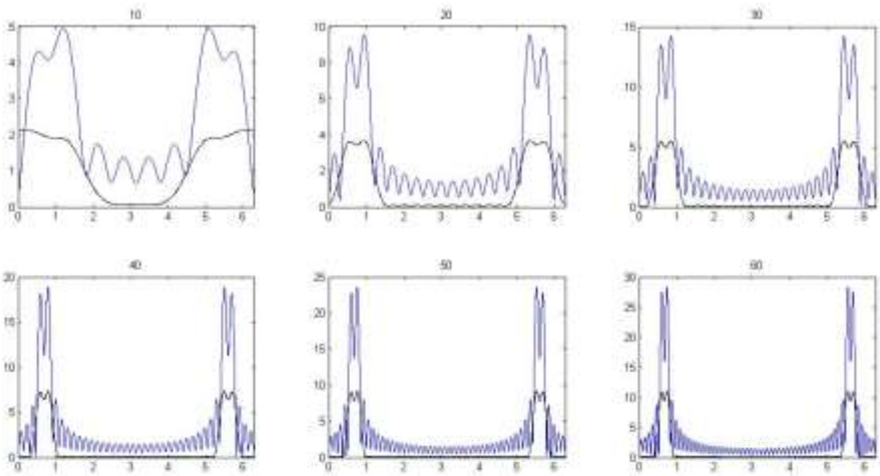
```
%generating DTFT of common windows

clear all;
clc;
```

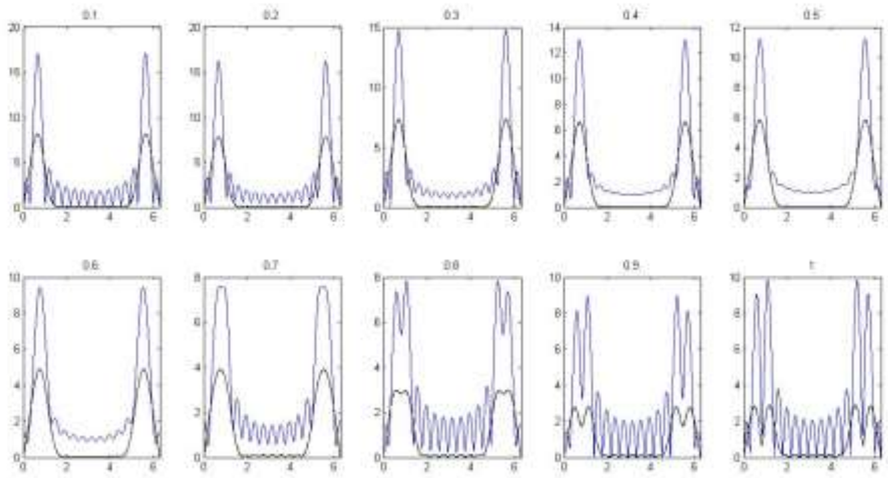
```
M=32;
n=-(M/2):(M/2-1);
w=0.4*pi;
x=sin(w*n).';
tmp=bartlett(M);
size(tmp);
y=x.*tmp;
[X,w0]=dtft(y);
figure(2);
plot(w0,abs(X));
title('Spectrum of x=cos(\omega_0n) windowed by Bartlett Function
with \omega_0=0.4\pi');
xlabel('Normalized Freq. \omega');
ylabel('Mag.');
```




```
clear all;
clc;

N=16;
w1=0.2*pi;

%% a=0.5;
figure(3);
for a=0.1:0.1:1
w=w1+2*pi*a/N;
n=0:N-1;
x=(cos(w1*n)+cos(w*n)).';
win1=rectwin(N);
win2=hamming(N);
x1=x.*win1;
```

```
x2=x.*win2;
[X1,W1]=dtft(x1);
[X2,W2]=dtft(x2);
subplot(2,5,a*10);

plot(W1,abs(X1),'b',W2,abs(X2),'--rs','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',1);
xlim([0 2*pi]);
title(a);
end

%% N=16;
w1=0.2*pi;
a=0.8;
figure(3);
figure(4);
for N=10:10:60
w=w1+2*pi*a/N;
n=0:N-1;
x=(cos(w1*n)+cos(w*n)).';
win1=rectwin(N);
win2=hamming(N);
x1=x.*win1;
x2=x.*win2;
[X1,W1]=dtft(x1);
[X2,W2]=dtft(x2);
subplot(2,3,N/10);
plot(W1,abs(X1),'b',W2,abs(X2),'--rs','LineWidth',1,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',1);
xlim([0 2*pi]);
title(N);
end
```





6.3 DTMF Signal Generation –

Code

```
%% DTMF Signal Decoding
%% 6.3 Generation
x = dtmfdial('7',8000);

Xk = fft(x);
N = length(x);
wk = 2*pi*(0:1/N:1-1/N);
fk = (wk * 8000) / (2 * pi);
figure(1)
plot(fk,abs(Xk));

title('Spectrum of Digit 7');
xlabel('Frequency in Hz');
ylabel('Magnitude');


%% 6.3.3 Decode
mynum =  load('mynumber.mat');
mynum = mynum.xx;

figure(2);
plot(mynum);
title('Original DTMF Signal');
xlabel('Time');
ylabel('Amplitude');
l = floor(length(mynum)/320);
comp_digit = [];
for num = 0:l-1
    digit = mynum(num*320+1:(num*320+256));
    X = fft(digit);
    comp_digit = [comp_digit dtmfcoef(X)];
end

disp(comp_digit);
```

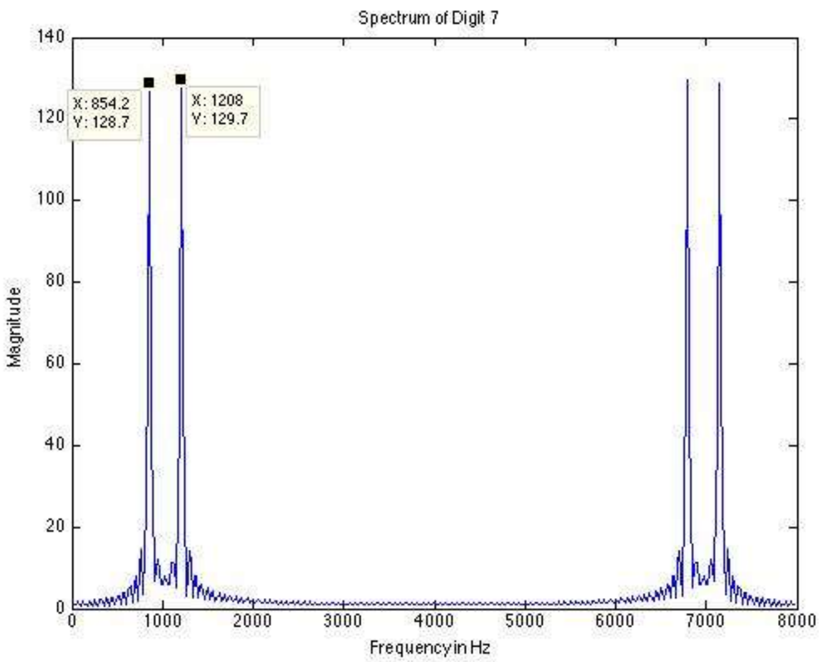
---

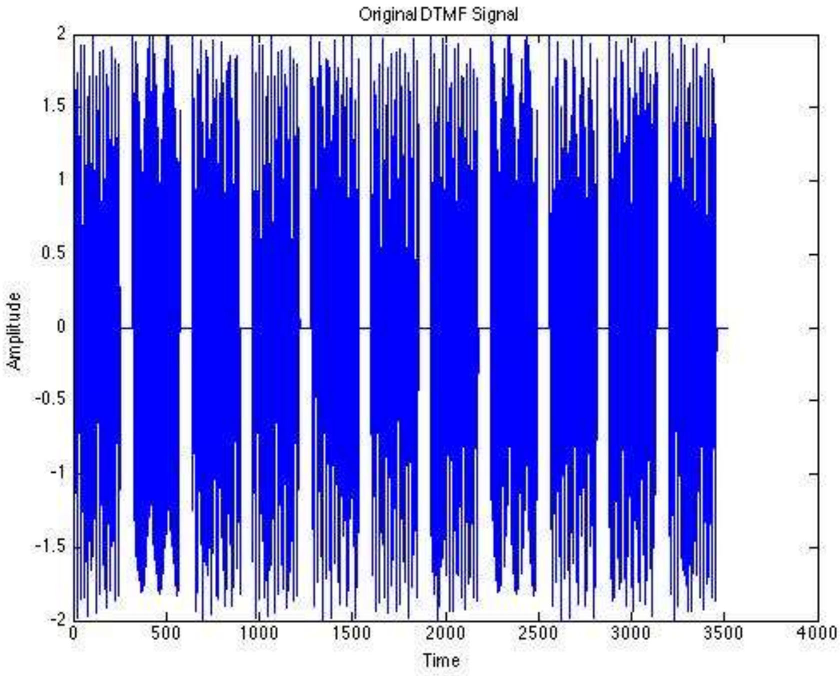
dtmfcoef Function

```
function keydecoded = dtmfcoef( Xk )
%DTMFCOEF Summary of this function goes here
```

```
%   Detailed explanation goes here
dtmf.keys = ...
    ['1','2','3','A';
     '4','5','6','B';
     '7','8','9','C';
     '*','0','#','D'];
ff_cols = [1209,1336,1477,1633];
ff_rows = [ 697; 770; 852; 941];

fs = 8000;
N = 256;

maxXk = max(abs(Xk));

ff_rows_new = ceil(N*ff_rows / fs);
ff_cols_new = ceil(N*ff_cols /fs);
row_val =abs(Xk(ff_rows_new));
col_val=abs(Xk(ff_cols_new));

row=find(row_val > maxXk/4 );
col=find(col_val > maxXk/4 );
keydecoded = dtmf.keys(row,col);
end
```

Results –
Decoded number - 03501783198





6.4 Inverse Filtering for speech deverbrations

```
%Exercise 6.4

clear all;
clc;

%% Generate room Impulse response
fs=44100;
mic=[3 3 3];
rm=[10 10 10];
src=[5 4 3];
n=4;
r=1;
hangerResolution=rir(fs, mic, n, r, rm, src);
figure(1);
stem(hangerResolution);
title('Room Impulse response');
xlabel('Time');
ylabel('Mag');
room = wavread('..\Lab_3\files_lab3\speech.wav');%audioread depreteates
wavread
                                                 %in next release
soundsc(room);

%% Convolute with impulse response
hanger = conv(hangerResolution,room);
soundsc(hanger);
M=length(hangerResolution)+length(hanger)-1;

%% Restore distorted signal
restoredSignal=ifft(fft(hanger,M)./fft(hangerResolution,M));
soundsc(restoredSignal);

%% Signal plots
figure(2);
subplot(3,1,1);
plot((0:1/fs:(length(room)-1)/fs),room);
title('Original signal');
subplot(3,1,2);
```

```
plot((0:1/fs:(length(hanger)-1)/fs),hanger);
title('Distorted signal');
subplot(3,1,3);
plot((0:1/fs:(length(restoredSignal)-1)/fs),restoredSignal);
title('Restored signal');
```