

---

# Problem 1

## Table of Contents

FFT Compression Demo .....	1
Reconstruction and obtaining error signal .....	1
Take FFT of the error and shift the frequency components to low frequency .....	1
Multiply the shifted frequency with the exponential function .....	2
Saving audio files .....	2
Listening to audio files .....	2
Plots in time domain .....	2
Plots in frequency domain .....	3
FFT Compress - Function definition .....	4
FFT Extract - Function definition .....	4

## FFT Compression Demo

```
clearvars
close all
[x, fs] = audioread('x.wav');

n=length(x); %number of samples
window_length=1; % in seconds
percentage=0.1; % percentage of most significant fft coefficients
               left after thresholding

% Splitting the input signal to 1 sec clips and taking their FFT

sld_window=1:fs*window_length; % samples for 1 sec window
fft_coef_total=[];
while sld_window(end) <= n
    fft_coef=fft_compress(x(sld_window),percentage);
    fft_coef_total=[fft_coef_total fft_coef(:)]; % each column
               contains 10 % of FFT of 1 sec
    sld_window=sld_window+fs*window_length;
end
```

## Reconstruction and obtaining error signal

```
x1=fft_extract(fft_coef_total, fs, window_length);
e=x-x1;
```

## Take FFT of the error and shift the frequency components to low frequency

```
Nfft = length(e); % same as length(x) or length(x_reconstruct)
```

```
k0 = percentage * Nfft;
e_fft = fft(e);
shift_e_fft = [e_fft(k0: Nfft-1-k0); zeros(k0*2,1)];
x2=ifft(shift_e_fft);
```

## Multiply the shifted frequency with the exponential function

```
for n=1:length(x2)
    x3(n) = x2(n) .* exp(1j*2*pi/Nfft*k0*(n-1));
end
```

## Saving audio files

```
audiowrite('x1.wav',real(x1),fs)
audiowrite('e.wav',real(e),fs)
audiowrite('x2.wav',real(x2),fs)
audiowrite('x3.wav',real(x3),fs)
```

## Listening to audio files

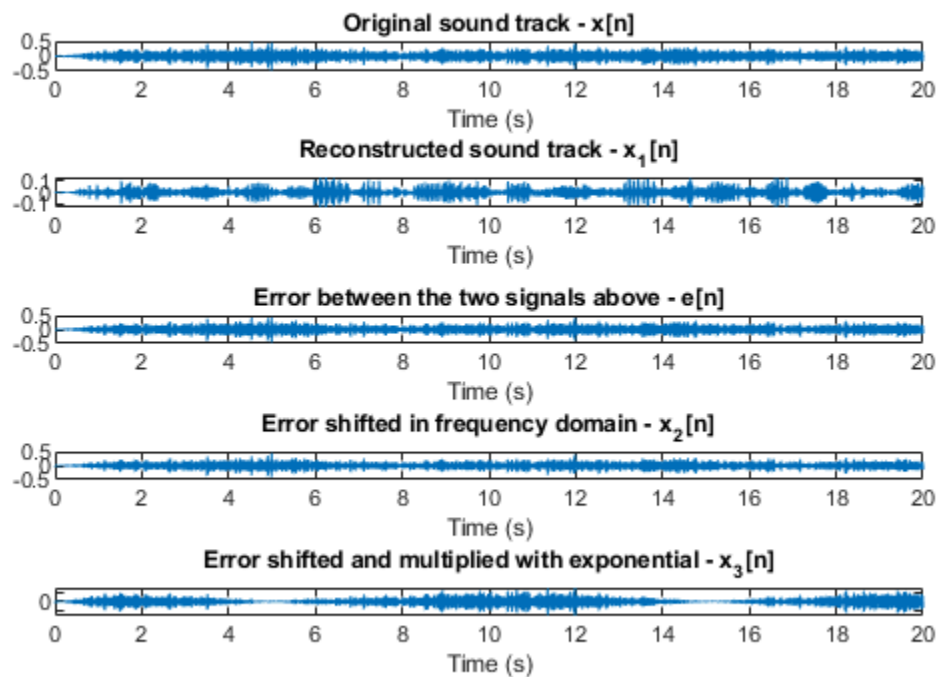
```
sound(real(x1),fs)
pause(25)
sound(real(e),fs)
pause(25)
sound(real(x2),fs)
pause(25)
sound(real(x3),fs)
pause(25)
```

## Plots in time domain

```
t=(0:length(x)-1)./fs;
subplot(511)
plot(t,x)
xlabel('Time (s)')
title('Original sound track - x[n]')
subplot(512)
plot(t,real(x1))
xlabel('Time (s)')
title('Reconstructed sound track - x_1[n]')
subplot(513)
plot(t,real(e))
xlabel('Time (s)')
title('Error between the two signals above - e[n]')
subplot(514)
plot(t,real(x2))
xlabel('Time (s)')
title('Error shifted in frequency domain - x_2[n]')
```

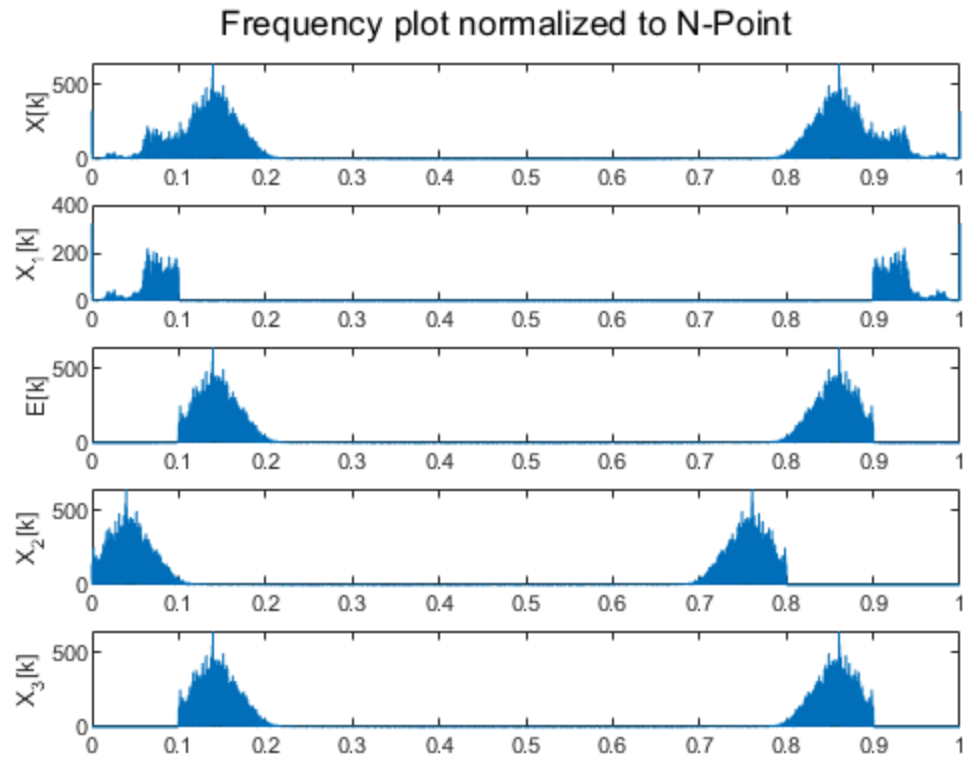
```
subplot(515)
plot(t,real(x3))
xlabel('Time (s)')
title('Error shifted and multiplied with exponential - x_3[n]')
sgtitle('Time domain Plot')
```

### Time domain Plot



## Plots in frequency domain

```
figure
w=(0:Nfft-1)./(Nfft-1);
subplot(511)
plot(w,abs(fft(x)))
ylabel('X[k]')
subplot(512)
plot(w,abs(fft(x1.')))
ylabel('X_1[k]')
subplot(513)
plot(w,abs(fft(e.')))
ylabel('E[k]')
subplot(514)
plot(w,abs(fft(x2.')))
ylabel('X_2[k]')
subplot(515)
plot(w,abs(fft(x3.')))
ylabel('X_3[k]')
sgtitle('Frequency plot normalized to N-Point')
```



## FFT Compress - Function definition

```
function fft_coef = fft_compress(x,percentage)
    N_fft=length(x);
    fft_x=fft(x,N_fft);
    N= round(percentage*N_fft);
    fft_coef=fft_x(1:N);
end
```

## FFT Extract - Function definition

```
function x = fft_extract(fft_coef_total,fs>window_length)
    x=[];
    N=size(fft_coef_total,1);
    for i=1:size(fft_coef_total,2) % i=1:20 (each column)
        fft_tmp=zeros(fs>window_length,1);
        fft_tmp(1:N)=fft_coef_total(:,i);
        fft_tmp(end-N+2:end)=fft_coef_total(end:-1:2,i)'; % To make it
        conjugate symmetric
        tmp=ifft(fft_tmp);
        x=[x;tmp(1:fs>window_length)];
    end
end
```

*Published with MATLAB® R2020a*