EE 430 PROJECT PHASE 1

Sabri Bolkar (1949874) - Selim Borekci (1813831)
Section 1 - Section 3

11 December 2015

Contents

1 Introduction **2**

2 Data Acquisition **2**

3 Spectrogram **2**

4 Graphical User Interface **4**

5 Project Phase-2 **5**

6 Conclusion **5**

7 Appendix **6**

List of Figures

1 Method used for Spectrogram Calculation 3
2 Hamming Window 3
3 Designed Graphical User Interface (Linux Ubuntu Version) 5
4 Buit-in function (on left), designed function (right) 6

Listings

1 Data Acquisition Part 2
2 Spectrogram Part 4
3 All MATLAB Code 6

1 Introduction

In Digital Signal Processing term project, we are assigned to create a comprehensive audio processing system consisting of data acquisition from audio file or directly from microphone, spectrogram analyzer, equalizer, sound affect mixer and remover systems. In the first phase, we implemented data acquisition and spectrogram parts. In addition to main program, we also created a graphical user interface which enables users to change sampling rate, window function types, window length and number of DFT points. All MATLAB codes can be viewed in appendix.

2 Data Acquisition

The acquisition part is readily implemented by MATLAB built-in functions, *audioread* for MP3 reading and *audiorecorder* for microphone reading. Also, *play* and *sound* commands easily enabled us to play the audio data.

```
Fs= 80000; % A/D sampling rate in Hz ----------CONFIGURABLE
T=5; % Record Time in seconds ----------CONFIGURABLE

recObj = audiorecorder(Fs,16,1); % 8 bit - 1 channel sound object
recordblocking(recObj, T);  %record T seconds
data = getaudiodata(recObj); %store sound data to "data"
play(recObj, Fs) %Play the sound

% MP3 data reading

name= 'feel.mp3' ; %% file name to be read
[data2, Fs]= audioread('feel.mp3'); % Get the data
sound('feel.mp3', Fs); %% PLay the sound
```

Listing 1: Data Acquisition Part

3 Spectrogram

For spectrogram implemetation we are restricted not to use built-in MATLAB function *spectrogram*; hence, we designed our own algorithm and program to implement spectrogram processing.

Firstly some research was done on spectrogram and short time Fourier transform. The process is not a cumbersome one but, there were some problems including how to implement the moving window function (Figure 1), which we choose as two kinds (i.e., Hamming and Rectangular). Also, parameters are required to be configurable which created another challenge. The chosen method is to move the window according to entered window size. We would like to design a 50% overlapping system to make sure not to miss any data, therefore half of the window size as hop size is chosen. In this way, we were able to solve the challenge to make sure no data is skipped. In subsequent parts, system multiplies the data points with window function and calculates DFT with MATLAB's built-in *fft* command (using entered DFT points) in a straightforward manner.

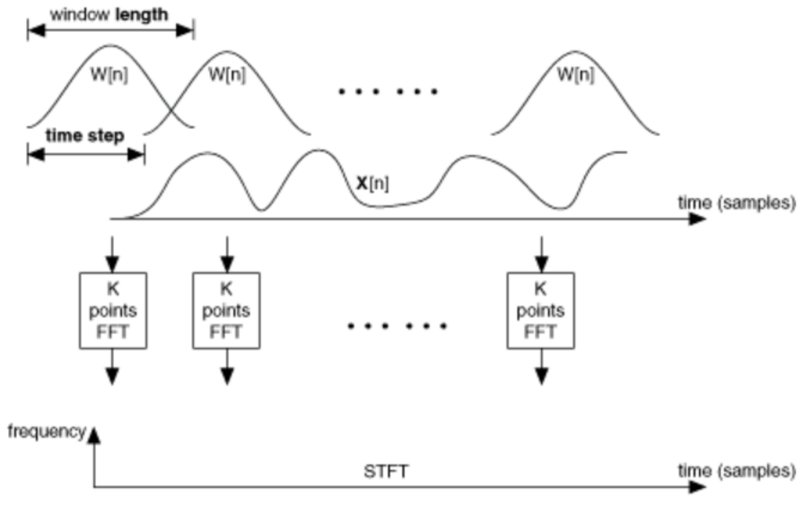


Figure 1: Method used for Spectrogram Calculation

Theoretical formulas can be viewed on right side. Since, it has been dealt with a computer simulation, we have exploited the second formula for discrete time systems. As sliding window functions, Hamming (below figure) and Rect function is used.

$$\mathbf{STFT}\{x(t)\}(\tau,\omega) \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}\,dt$$

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n}$$

$$\text{spectrogram}\{x(t)\}(\tau,\omega) \equiv |X(\tau,\omega)|^2$$

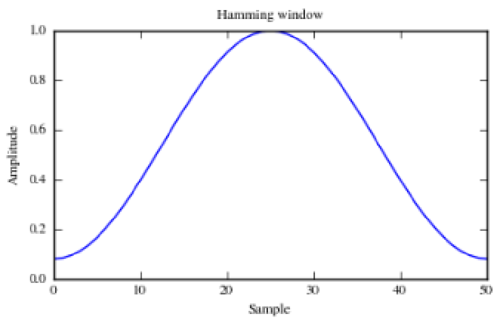


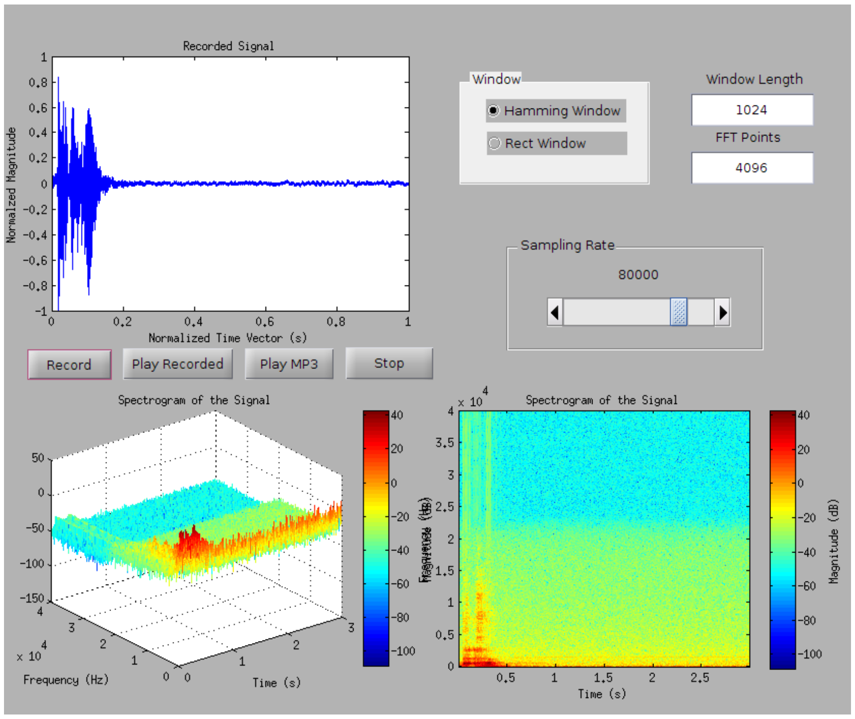
Figure 2: Hamming Window

```
% window1 Hamming
win1 = hamming(wind_length, 'periodic');% ----------CONFIGURABLE
% window2 Rect
win2 = ones(wind_length,1); %----------CONFIGURABLE
% Spect will be the spectrogram
rown = ceil((1+ndft)/2);
coln = 1+fix((data_length-wind_length)/hop);
spect = zeros(rown, coln);
indx = 0;
col = 1;
%%%% DFT calculation
while indx + wind_length <= data_length
    % windowing
    windowed_data = data(indx+1:indx+wind_length).*win1; %%%%% CHANGE WINDOW FUNC

    % FFT
    X = fft(windowed_data, ndft);

     spect(:, col) = X(1:rown);

    indx = indx + hop;
    col = col + 1;
end

t = (wind_length/2:hop:wind_length/2+(coln-1)*hop)/Fs;
f = (0:rown-1)*Fs/ndft;
spect= 20*log10(abs(spect));  %%% TAKING DB of Spectrogram data
```

Listing 2: Spectrogram Part

4 Graphical User Interface

GUI design is one of the most challenging part of the project. Since, it the first time we are creating a graphical user interface for a MATLAB program. It took some time to understand the idea. We mostly used Internet sources to create the graphical user interface. A sample GUI picture of our design can be viewed below for coughing of Sabri.


Window: [✓ Hamming Window] Rect Window

Window Length: 1024

FFT Points: 4096

Sampling Rate: 80000

Figure 3: Designed Graphical User Interface (Linux Ubuntu Version)

5 Project Phase-2

For the remaining part of the project, we started to create algorithms and programs. Real-time spectrum part is already completed, and for equalizer part, we have decided on techniques. We also plan to implement a second graphical user interface for remaining parts of the project.

6 Conclusion

It was a great experience for us to implement DFT for a real-world process. After we compared our program with built-in *spectrogram* command, results amazed us with their almost same outcome for same sampling rate. In below figure, the similarity can be observed. We also observed that, hamming window gives better results for the analysis by smoothing the edges which window is applied to. DFT points and window length have also effects on spectrogram results. But, it can be seen as a trade-off. If a longer window length is chosen, although transform is more accurate the response time is longer which is not desired for some applications where quick analysis is needed.

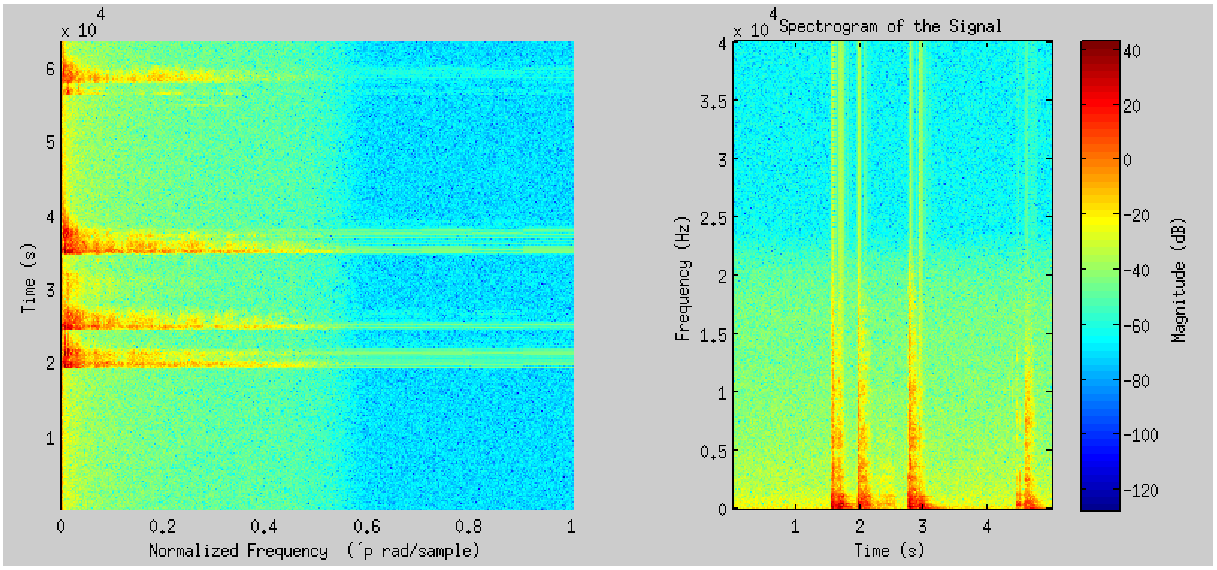


Figure 4: Buit-in function (on left), designed function (right)

7 Appendix

```
%%%%%%%%%%
%
% EE 430 PROJECT Sabri Bolkar — Selim Borekci
%
%%%%%%%%%%

%% STEP 1

%%%%% Microphone reading

Fs= 80000; % A/D sampling rate in Hz ——————————CONFIGURABLE
T=5; % Record Time in seconds ——————————CONFIGURABLE

recObj = audiorecorder(Fs,16,1); % 8 bit — 1 channel sound object
recordblocking(recObj, T);  %record T seconds
data = getaudiodata(recObj); %store sound data to "data"

play(recObj, Fs) %Play the sound

%%
%%%%%%%%%%%%%%%%%%%%
% MP3 data reading

%  name= 'feel.mp3' ; %% file name to be read

%  [data2, Fs]= audioread('feel.mp3'); % Get the data

%  sound('feel.mp3', Fs); %% PLay the sound
```

```
%%
%%%%%%%%%%%%%%%%%%%%
% SPECTROGRAM

% Sample audio data FOR CHECKING
%[data, Fs] = audioread('Applause.mp3');

%%%% CHANGE data with data2 for mp3 processing

data_max = max(abs(data));
data = data/data_max;    %% Normalization

% Default input parameters
data_length = length(data);
wind_length = 1024;     %%% ----------CONFIGURABLE
hop = wind_length/2;     %% hopsize; take windows as 50% overlap  ----------
    CONFIGURABLE
ndft = 4096;          %% fft points ----------CONFIGURABLE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% DFT

% window1 Hamming
win1 = hamming(wind_length, 'periodic');% ----------CONFIGURABLE

% window2 Rect
win2 = ones(wind_length,1); %----------CONFIGURABLE

% Spect will be the spectrogram
rown = ceil((1+ndft)/2);
coln = 1+fix((data_length-wind_length)/hop);
spect = zeros(rown, coln);

indx = 0;
col = 1;

%%%% DFT calculation
while indx + wind_length <= data_length
    % windowing
    windowed_data = data(indx+1:indx+wind_length).*win1; %%%%% CHANGE WINDOW FUNC

    % FFT
    X = fft(windowed_data, ndft);


    spect(:, col) = X(1:rown);

    indx = indx + hop;
    col = col + 1;
end


t = (wind_length/2:hop:wind_length/2+(coln-1)*hop)/Fs;
f = (0:rown-1)*Fs/ndft;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

spect= 20*log10(abs(spect));  %%% TAKING DB of Spectrogram data
```

```
%Play the sound
% sound(data)

%%
% PLOTING %%%%%%%%%%%%%%%%%%%%%

%plot the sound data

subplot(3,1,1)
Tx=linspace(0, T, length(data));
plot(Tx, data);
set(gca,'YDir','normal')
xlabel('Time (s)')
ylabel('Sample Magnitude')
title('Recorded Signal')

%%% plot the spectrogram

%%%% 3D
subplot(3,1,2)
h=surf(t, f, spect); colorbar;
set(h,'LineStyle','none') %% turn linestyle off to see the graph
set(gca,'YDir','normal')
xlabel('Time (s)')
ylabel('Frequency (Hz)')
title('Spectrogram of the Signal')

handle = colorbar;
ylabel(handle, 'Magnitude (dB)')

%%%% 2D
subplot(3,1,3)
imagesc(t, f, spect) %%% Plot it with color info

set(gca,'YDir','normal')
xlabel('Time (s)')
ylabel('Frequency (Hz)')
title('Spectrogram of the Signal')

handle = colorbar;
ylabel(handle, 'Magnitude (dB)')

%%
```
Listing 3: All MATLAB Code