SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMAKURU-572103

(An Autonomous Institute under Visvesvaraya Technological University, Belagavi)

Speech Processing Lab Report

submitted in partial fulfillment of the requirement for the completion of VI semester of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

submitted by

ANKIT KUMAR 1SI19EC011

under the guidance of

Dr Veena Karjagi

Associate Professor

Department of ECE

SIT, Tumakuru-3

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

2021-22

1. DFT of Speech signal

The information of frequency and amplitude of the speech components is extracted using DFT of the speech signal

CODE

```
import numpy as np
import matplotlib.pyplot as plt
from array import *
x = array('i', [1,2,3,4])
N=len(x)
n = np.arange(N)
k = n.reshape((N, 1))
e = np.exp(-2j * np.pi * k * n / N)
X = np.dot(e, x)
w=2*np.pi*n/N
print(n)
print(k)
print(e)
print(X)
plt.plot(w, np.abs(X))
plt.grid()
plt.show()
```

OUTPUT:

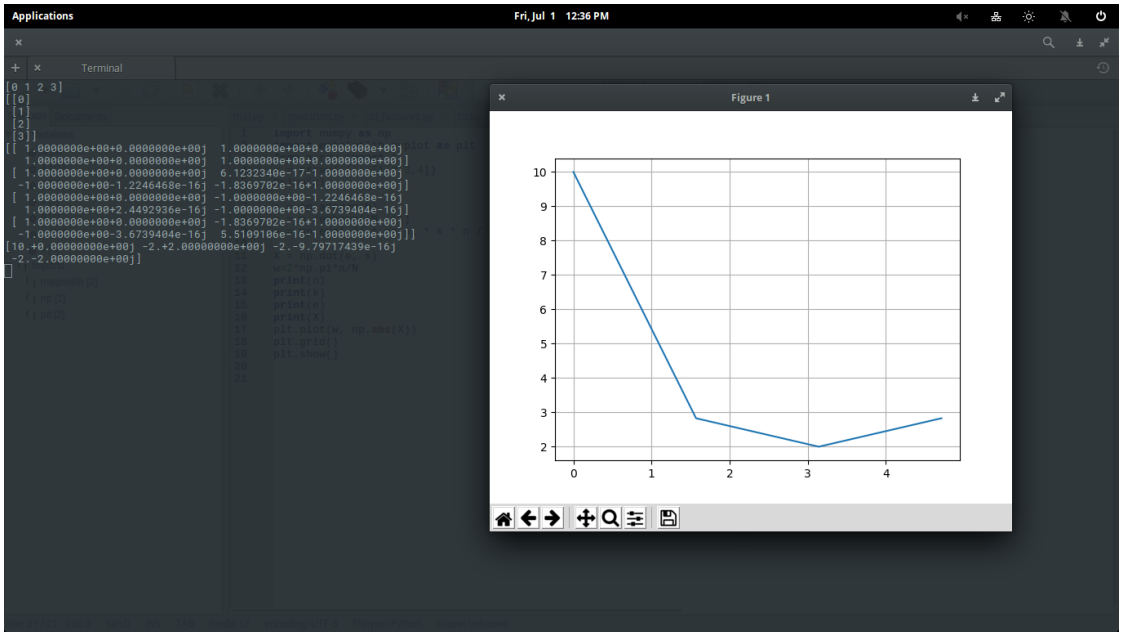

Figure 1 : DFT

2. NARROW BAND WIDE BAND SPECTRUM

Narrow band spectrum is created using a large window and wide band spectrum is created using a relatively smaller window

Spectrum is used to detect pitch and excitation of the speech signal

CODE

```
from scipy.io import wavfile
import scipy.io
import numpy as np
import math
from scipy.signal import hann
from scipy.fftpack import fft
import matplotlib.pyplot as plt


samplerate, data = wavfile.read('\Users\DELL/fa.wav')
print(samplerate)

window = hann(40)

data_win=np.multiply(data[:40],window)

mags = abs(fft(data_win))
mags_dB=20*np.log10(mags)

N=len(data_win)
k=np.arange(N)
f=k*samplerate/N
plt.plot(f,mags_dB)
plt.ylabel("Magnitude (dB)")
plt.xlabel("Frequency in Hz")
plt.title("Spectrum")
plt.show()
```

OUTPUT

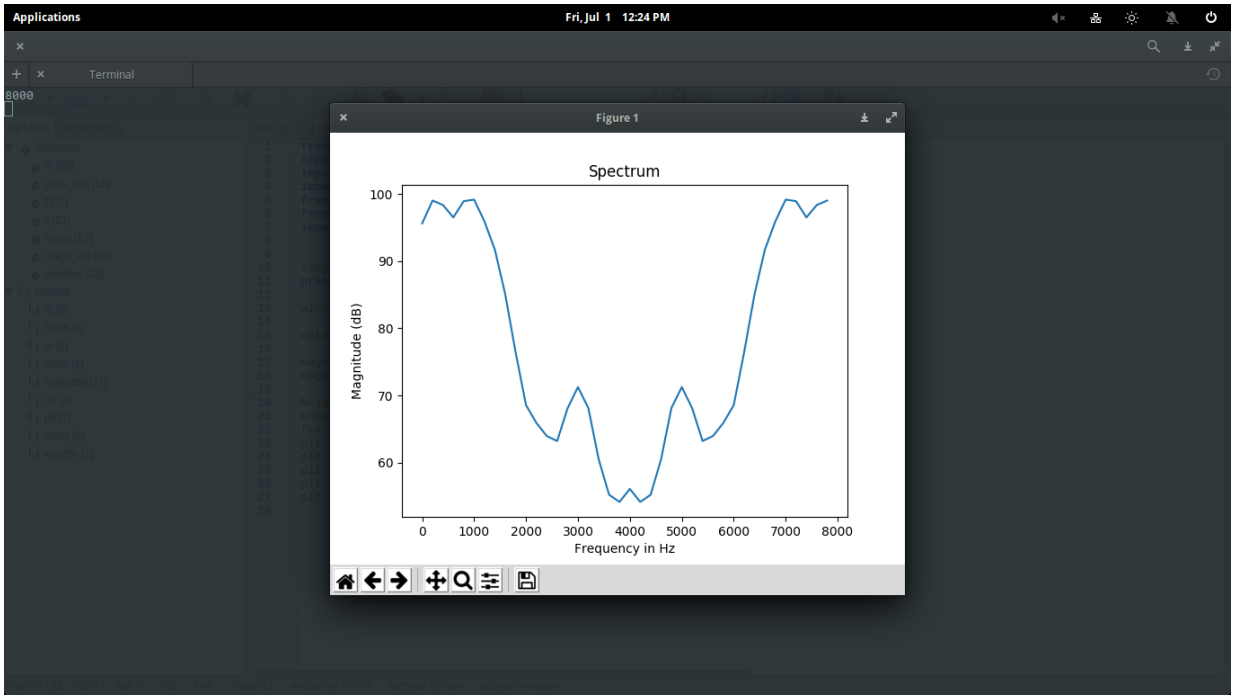

Figure 2 : Wide band spectrum

3. TIME DOMAIN FEATURES (ZERO CROSSING RATE AND SHORT TIME ENERGY)

o differentiate speech from silence and voiced and unvoiced speech, we use Short Time Energy. To differentiate voiced and unvoiced speech, classification of percussive sounds, zero crossing rate is used.

CODE

```
window=hann(240)
nvrlp=120

nframes=len(data)/nvrlp
print(nframes)

nframes=int(nframes)
nframes=nframes-1
#print(nframes)

cols=240
rows=nframes

frames=np.arange(nframes)
samples=np.arange(cols)
'''print(frames)
print(samples)'''

Energy=[[0]*1]*rows
ZCR=[[0]*1]*rows
time=[[0]*1]*rows
print(Energy)

for i in frames:
        frame_start=i*nvrlp
        data_select=data[frame_start:frame_start+240]
        data_win=np.multiply(data_select,window)
        Energy[i]=sum(pow(data_win,2))
        for j in samples:
                ZCR[i]=ZCR[i]+abs(np.sign(data_win[j])-np.sign(data_win[j-1]))
        time[i]=i*(120/8000)

signal_samples=np.arange(len(data))/samplerate
```

```
plt.subplot(311)
plt.plot(signal_samples,data)
plt.subplot(312)
plt.plot(time,Energy)
plt.subplot(313)
plt.plot(time,ZCR)
plt.show()
```

OUTPUT

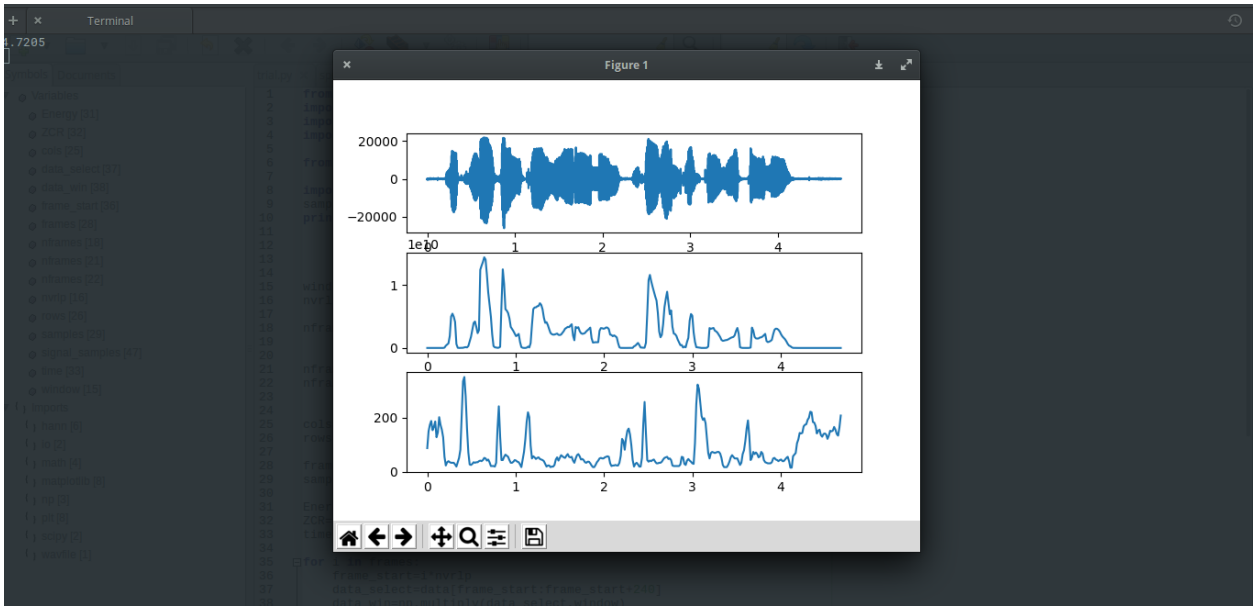

Figure 3 : a)Energy b)ZCR

4. CEPSTRAL ANALYSIS

Cepstrum is the inverse Fourier Transform of logarithm of power spectrum of a signal.
A speech sequence has to be deconvolved into the excitation and vocal tract components in the time domain. For this, multiplication of the two components in the frequency domain has to be converted to a linear combination of the two components. For this purpose cepstral analysis is used for transforming the multiplied source and system components in the frequency domain to linear combination of the two components in the cepstral domain.
Cepstral graph is used to separate vocal tract information from the excitations. The initial few samples provide vocal tract information while excitation is obtained as periodic peaks.

CODE

1. Real Cepstrum

```
from scipy.io import wavfile
import scipy.io
import numpy as np
import math
from scipy.signal import hann
from scipy.fftpack import fft
from scipy.fftpack import ifft
import matplotlib.pyplot as plt
samplerate, data = wavfile.read('/home/sit/Downloads/speech/fa.wav ')
print(samplerate)
mags = np.abs(fft(data[:480],1024))
log_mags=np.log(mags)
ceps=ifft(log_mags)
t=(np.arange(480))/samplerate
#print(np.abs(ceps[:480])
plt.plot(t,np.abs(ceps[:480]))
plt.show()
```

2. Complex cepstrum

```
from scipy.io import wavfile
import scipy.io
import numpy as np
import math
from scipy.signal import hann
from scipy.fftpack import fft
from scipy.fftpack import ifft
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

samplerate,data=wavfile.read('/home/sit/Downloads/speech/fa.wav')
print(samplerate)
```

```
signal=data[1:480]
mags=np.abs(fft(data[:480]))
log_mags=np.log(mags)
ceps=ifft(log_mags)

t=(np.arange(480))
print(np.abs(ceps))
plt.plot(t,np.abs(ceps))
plt.show()
exi=ceps[:10]
vocl=ceps[11:]
exi_fft_exp=np.exp(abs(fft(exi,512)))
vocl_fft_exp=exp(abs(ft(vocl,512)))
k=np.arange(512);
f=samplerate*k/512;
plt.plot(f,exi_fft_exp)
plt.show(
plt.plot(f,vocl_fft_exp)
plt.show()
```

OUTPUT

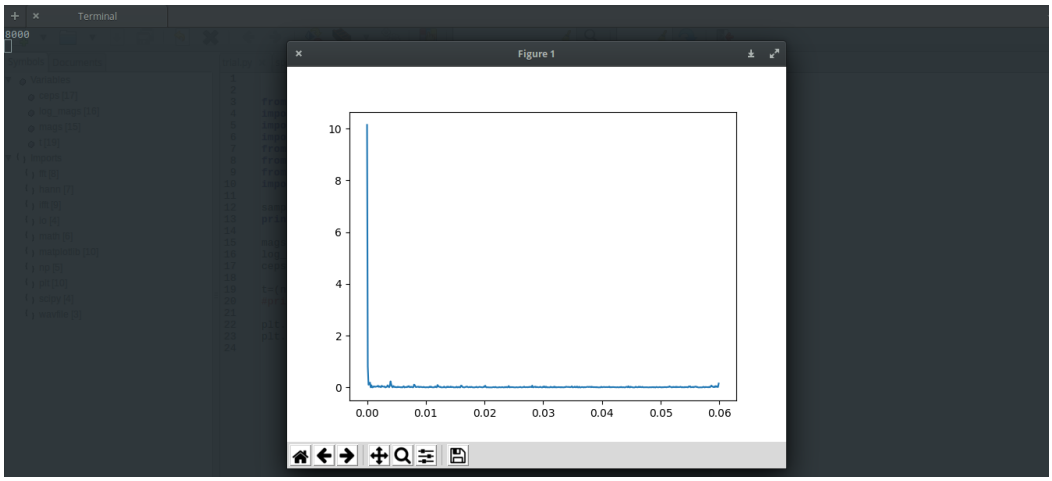

Figure 4 : Real cepstrum

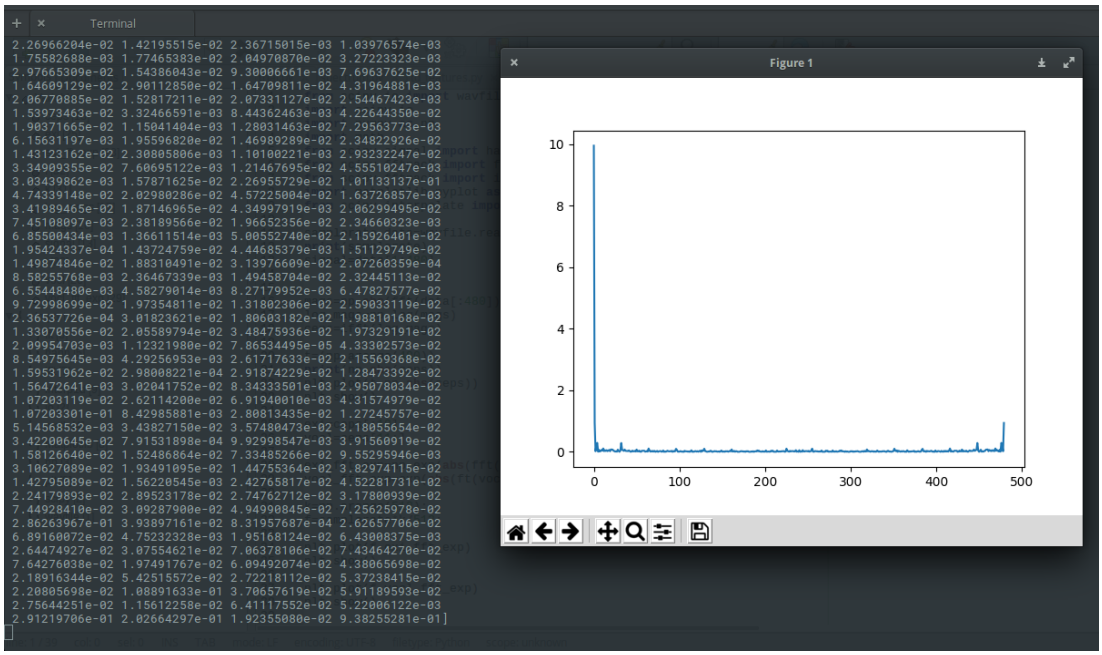

Figure 5 : Complex cepstrum

5. SPECTROGRAM

Spectrogram is a time dependent spectrum of a speech signal. The horizontal axis of a spectrogram represents time and the vertical axis represents frequency. The intensity represents the magnitude.
Information about properties of a speech such as pitch, formants, voiced or unvoiced etc.. is obtained through Spectrogram.

CODE

```
from scipy.io import wavfile
import scipy.io
import numpy as np
import math
from scipy.fftpack import fft
from scipy.signal import hann
import matplotlib.pyplot as plt
samplerate, data = wavfile.read('/home/ecr-02/Documents/machali.wav')
print(len(data)/samplerate)
window=hann(240)
nvrlp=120
nframes=len(data)/nvrlp
nframes=int(nframes)
nframes=nframes-1
rows=nframes
frames=np.arange(nframes)
dft_frame=[[0]*512]*rows
time=[[0]*1]*rows
for i in frames:
        frame_start=i*nvrlp
        data_select=data[frame_start:frame_start+240]
        data_win=np.multiply(data_select,window)
        dft_frame[i]=20*np.log10(np.abs(fft(data_win,512)))
        time[i]=i*(120/8000)
print(np.shape(dft_frame))
dft_frame_t=np.transpose(dft_frame)
plt.imshow(dft_frame_t[257:512])
plt.show()
```

OUTPUT

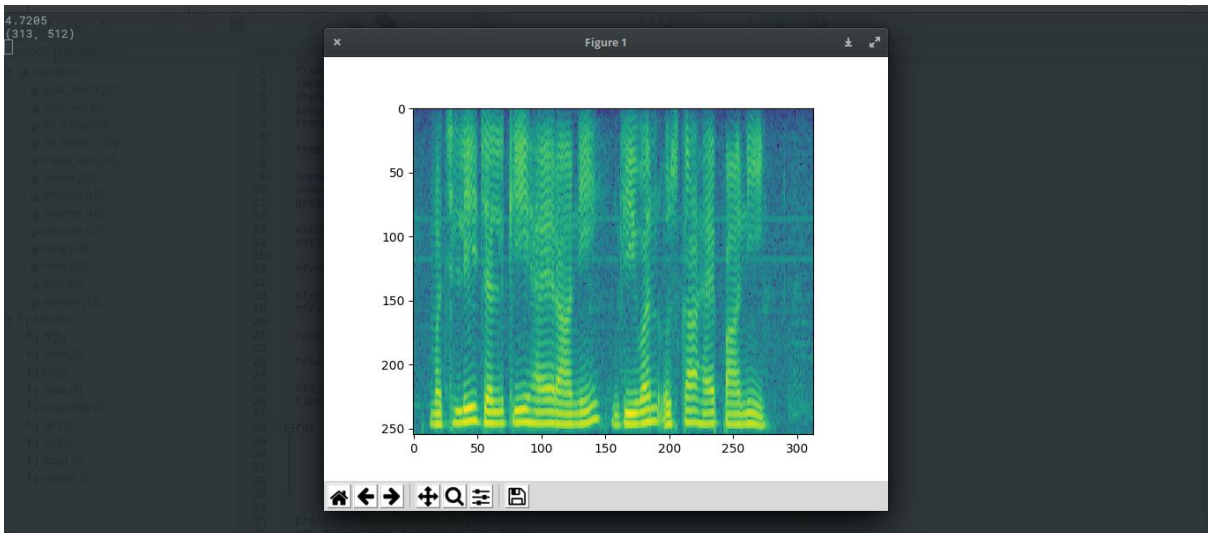

Figure 6: Spectrogram of "Machali" utterance

6. LINEAR PREDICTIVE ANALYSIS

Speech sample can be approximated as a linear combination of past speech samples. By minimizing the sum of squared differences between the actual speech samples and linearly predicted ones, a unique set of predictor coefficients can be determined. This is linear predictive analysis.
It is used to predict pitch, formants, pitch period etc..

CODE

```
from scipy.io import wavfile
import scipy.io
import numpy as np
import math
from scipy.fftpack import fft
from scipy.signal import hann
import matplotlib.pyplot as plt
from scipy.linalg import solve_toeplitz, toeplitz
samplerate, data = wavfile.read('/home/sit/Downloads/speech/fa.wav')
frame=data[:480]/32768
lp_order=10;
len_sig=len(frame)
auto_corr=np.correlate(frame,frame,"full")
st=len_sig-1
en=2*len_sig-1
norm_auto=auto_corr[st:en]/auto_corr[st]
b=np.array((norm_auto[1:lp_order+1]))
c=norm_auto[:lp_order];
r=c;
lpcoeffs=solve_toeplitz((c,r),b)
print(lpcoeffs)
```

OUTPUT

```
Terminal
 2.14681112 -1.89570141  0.53654544 -0.09261142  0.76286507 -0.86370883
 0.11689183  0.35352519 -0.19824816 -0.01905677]

-----------------
program exited with code: 0)
ress return to continue
```

Figure 7 : LP coefficients of "fa"