EE679: Computing Assignment 3

Swapnil Bembde 14D070034

October 7, 2017

Question 1

* Value of pitch is used from the last assignment to construct an input, which is 133.33Hz.
* The estimated LP filters for /a/, /i/ and /n/ are with p = 20 because these filters give similar variability in spectral envelope to Fourier transform. But for /s/ p = 40 is used.
* Regarding similarity with the original sound, reconstructed /a/ and /i/ sound almost similar to the original sound. /s/ also sounds similar. But /n/ does not sound similar. I think for /n/ I need to use higher p to model it's zeros of the filter.
*Application - We can generate any IPA and this can be used as a codebook for a speech recognition system or we can build automated voice generator.

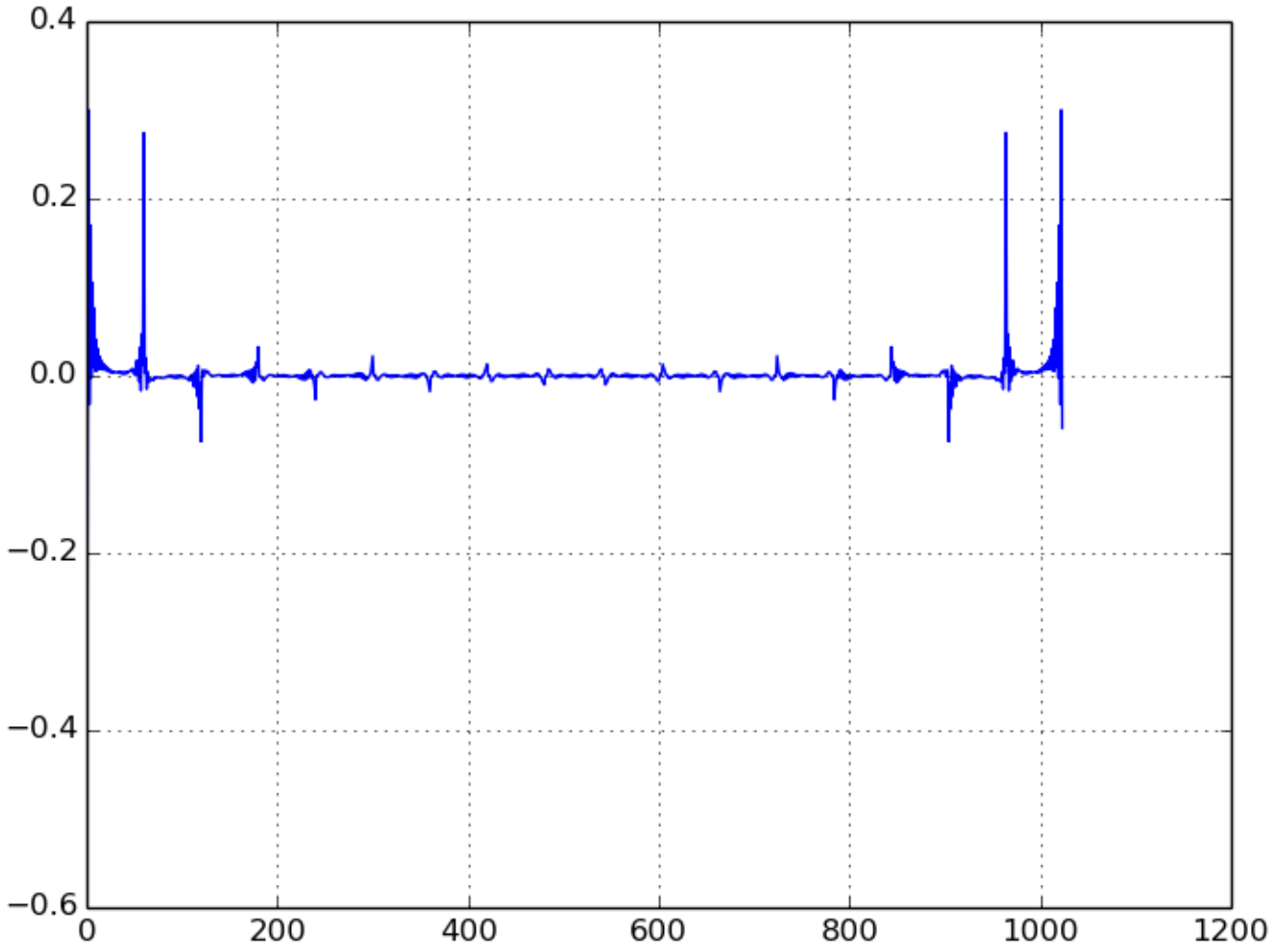
Code snippet:

```
from scipy import signal
from scipy.io.wavfile import read, write
import numpy as np
import matplotlib.pyplot as plt
from math import pi
import sys

# select the segment
# poles and zeros are from last assignment
if sys.argv[1] == 'a':
        # for p = 20
    #a = np.asarray([1.0, 0.236005432799, -0.644092002133, 0.0819119339449, \
    #-0.2124261564, -0.152211140765, -0.0212219709262, -0.13630085139,\
    #-0.0830381940115, -0.0594591290005, 0.0932528522119, -0.138673810679,\
    # -0.0449949546443, 0.160272165261, 0.107773913053, 0.0937237624822,\
    # -0.106615993903, -0.0430479902438, 0.00944250408534, -0.00745641174071,\
    #0.0831723106828])
    # for p = 10
    a = np.asarray([1.0, 0.229278962837, -0.645647130149, 0.0916355495711,\
     -0.207582361967, -0.173742441768, -0.00159815647695, -0.129920337525, \
     -0.0646863732807, -0.00292087909058, 0.0709251746617])

    b = np.asarray([ 3.01612624, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,\
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ])
    SampFre = 8000.0

elif sys.argv[1] == 'n':
    a = np.asarray([1.0, -0.0343093242382, -0.926868858417, 0.0165509083365,\
     -0.00386836371951, 0.218989739824, -0.00834526166107, -0.204479256656, \
        0.0612608771718, -0.00301307871679, 0.00852860873086, -0.000786003244963,\
         -0.0541912219039, 0.228276713792, -0.00981429198171, -0.185870867754,\
         0.00445150087087, -0.0142551645715, -0.130314042554, 0.00322505195405, \
```

```
        0.161683923034])

    b = np.asarray([2.24130211, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,\
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ])
    SampFre = 8000.0

elif sys.argv[1] == 'i':
    a = np.asarray([1.0, 0.244453564833, -1.01664129699, -0.321295111969,\
     0.088749127459, 0.109822570243, 0.0952082699624, 0.207489730971,\
     -0.0215834114004, -0.281867271388, -0.0938234280706, 0.0639360961099,\
      0.0961474114684, 0.207959826937, 0.0511832367678, -0.242036238679,\
      -0.155966156495, 0.139158197106, 0.195869516629, -0.0979195745779,\
       -0.170353127148 ])

    b = np.asarray([ 2.55634885 , 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,\
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0 ])
    SampFre = 8000.0

elif sys.argv[1] == 's':
        #for s p = 40
    a = np.asarray([1.0, -0.00358528757999, 0.385091318531, -0.11943297762,\
     -0.35715722615, -0.166353261538, -0.108766425583, -0.00221571686568,\
     0.0225611469574, -0.244560616563, 0.0622123574262, -0.101027641575,\
      0.0414860380115, -0.0309243387031, -0.00804769088502, -0.0591783440907,\
      -0.00655702747194, -0.0946193074253, 0.0152480326206, -0.0625141681721,\
       0.0254888396516, -0.0952285418858, -0.0607358904417, -0.113405325762,\
       0.0482700542505, -0.0185480613643, 0.0690345230921, -0.0596909779859,\
        0.0340837640503, -0.0993764375533, -0.0184148072237, -0.0289216749101,\
        -0.0919745287535, -0.0727036376883, -0.0187708573377, 0.0352487253815,\
         0.135518645672, 0.0874812212111, 0.108809324213, 0.0222556074524,\
         -0.0287374167142])

    b = np.asarray([4.66706473, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,\
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,\
     0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,\
      0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0])
    SampFre = 16000.0

pitchperiod = 8000.0/60.0
duration = 300.0/1000.0
timepoints = np.linspace(0, duration, duration*SampFre, endpoint=False)

#input generation
if sys.argv[1] == 's':
        # random generator
    inp = np.random.normal(0, 1, duration*SampFre)
else:
        # impulse train
    inp = (1+signal.square(2 * np.pi * pitchperiod * timepoints, duty=0.01))/2

output = signal.lfilter(b, a, inp)
# de emphasis
doutput = signal.lfilter(np.asarray([1.0, 0.0]), np.asarray([1, -0.95]), output)
```

Question 2

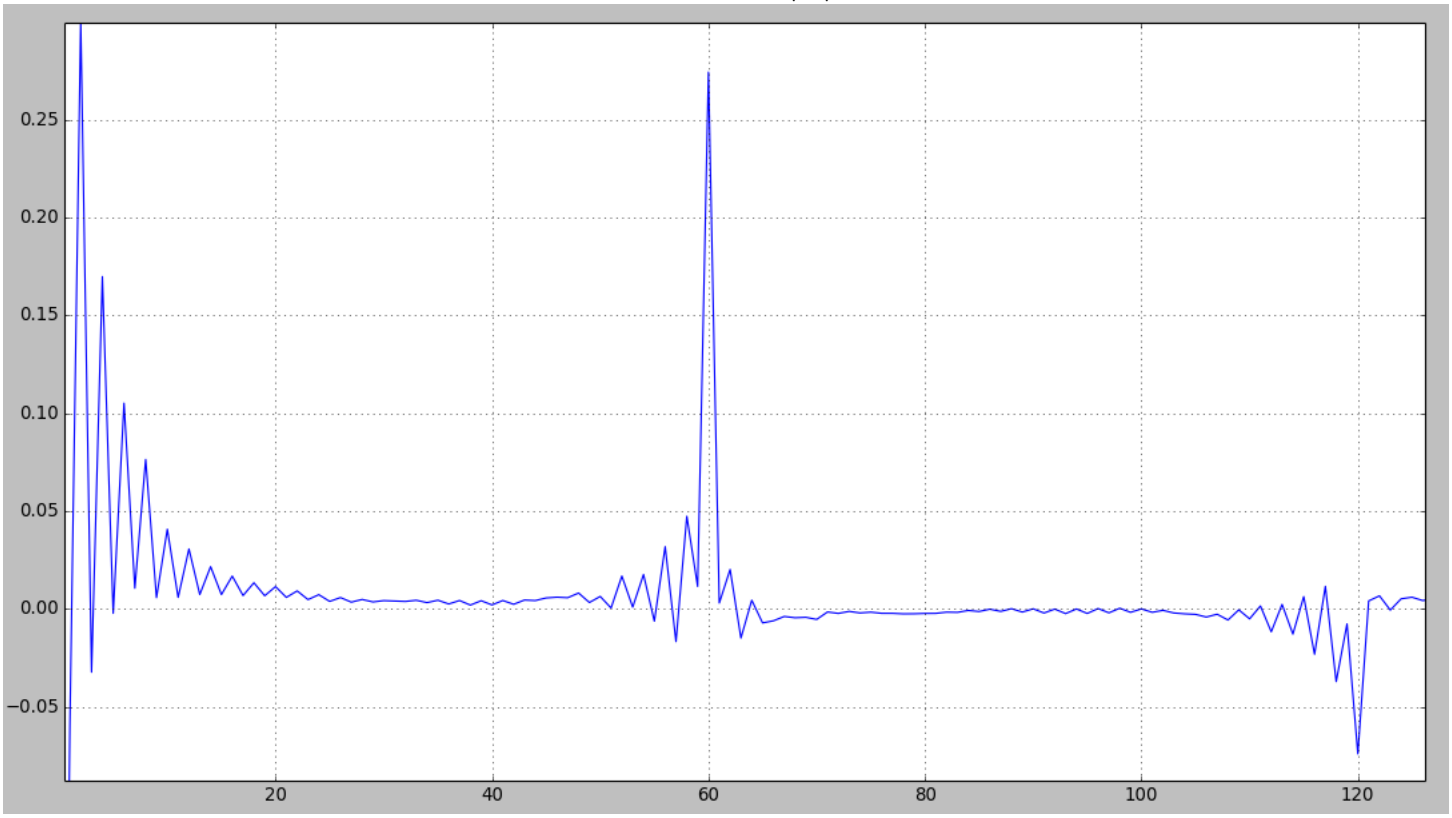
Real cepstrum of synthetic signal /a/



Comments:

* If we compare this real cepstrum with the real cepstrum due to natural phone(which is done in part 3), we can see, periodic peaks die as cepstrum coefficient increases(¡N/2). This periodicity remains because our reconstruction is crude approximation of the natural phone.

Zoomed version of real cepstrum of synthetic signal /a/



Comments:

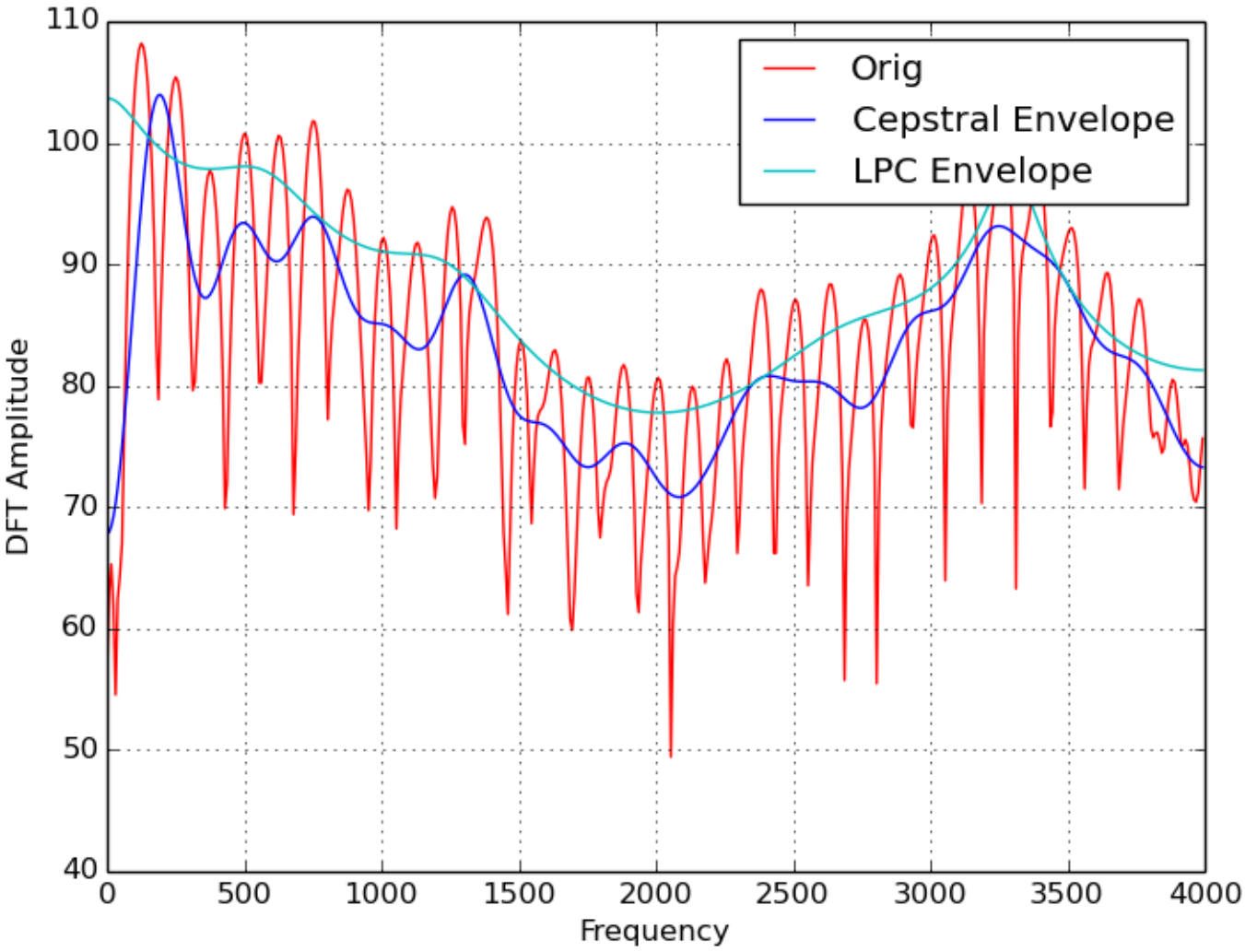
*In this question, we perform the cepstral analysis on the LP synthesized signals from the previous question. As we can see in above figure, the there are peaks at n = 60, 120, 180, ... The values of the peaks are dying out as we increase n till 512. These peaks are due to glottal impulses.

* In the zoomed figure, we can't have lifter's length more than 50 due to the peak at 60. For the relation between cepstral coefficients and spectral envelope next figures are plotted.

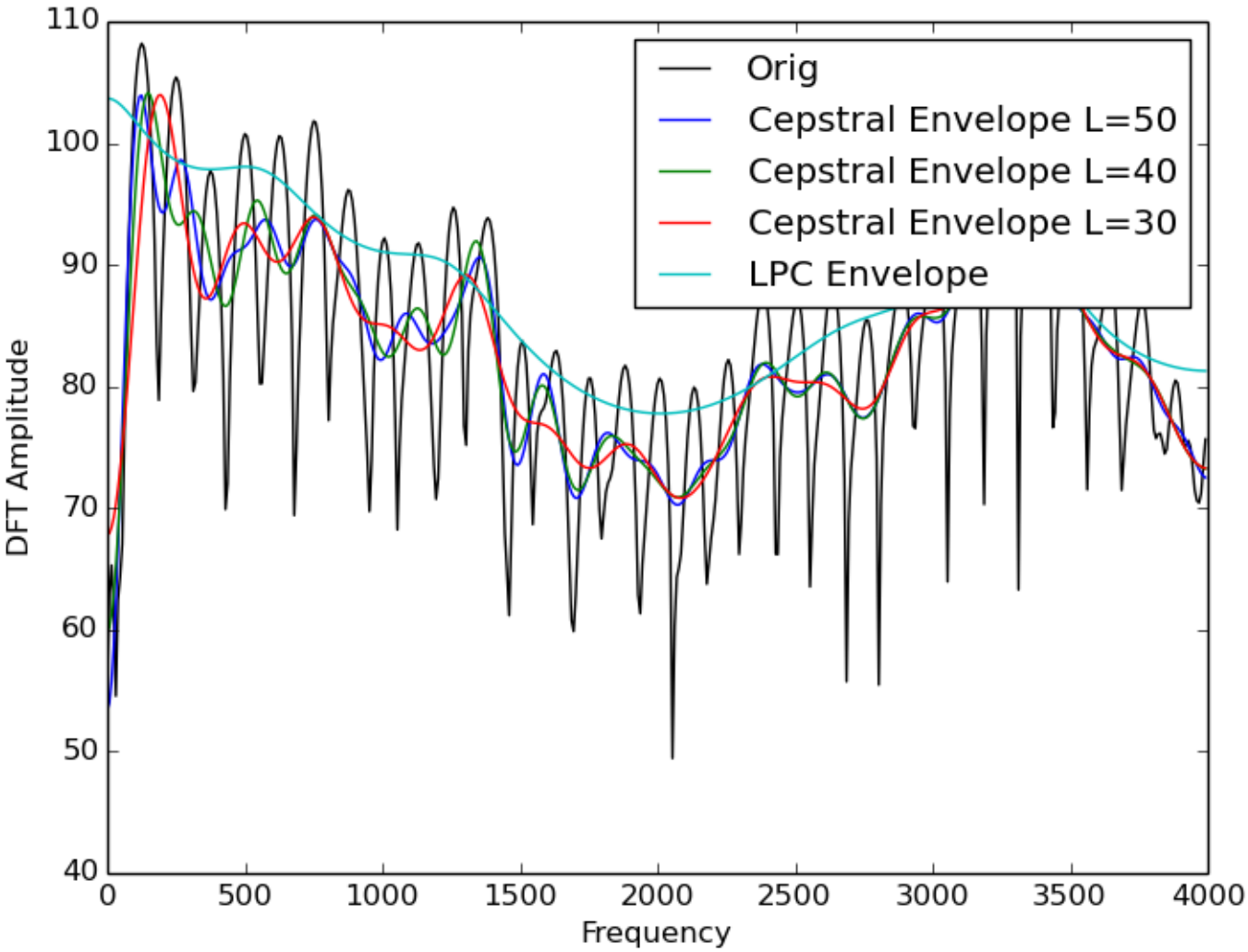
* All observations are based on an assumption that our signal is dying down before an impulse(maximum) shows up in the domain of cepstral coefficients.

Spectral envelope of synthetic signal /a/

* with lifter's length of 25



Spectral envelope of synthetic signal /a/



Comments:

* As we can see in figure, LPC envelope has higher values than cepstral envelopes of different lengths except at a few peaks.
* LPC can't follow the original envelope because we are considering few poles only. In case of lifters of different lengths, we are considering prominent coefficients only. Hence as we are increase length of lifter we increase variability in the envelope of spectrum.
* As we can see blue is changing faster than green and red spectra because blue has liftering length of 50.
* For formant tracking, it depends on the vocal tract's real cepstrum. If it dies down very fast then we don't need more coefficients to represent spectral envelope. More coefficients we result in better estimation of formants because we will get prominent gain corresponding to the formants.

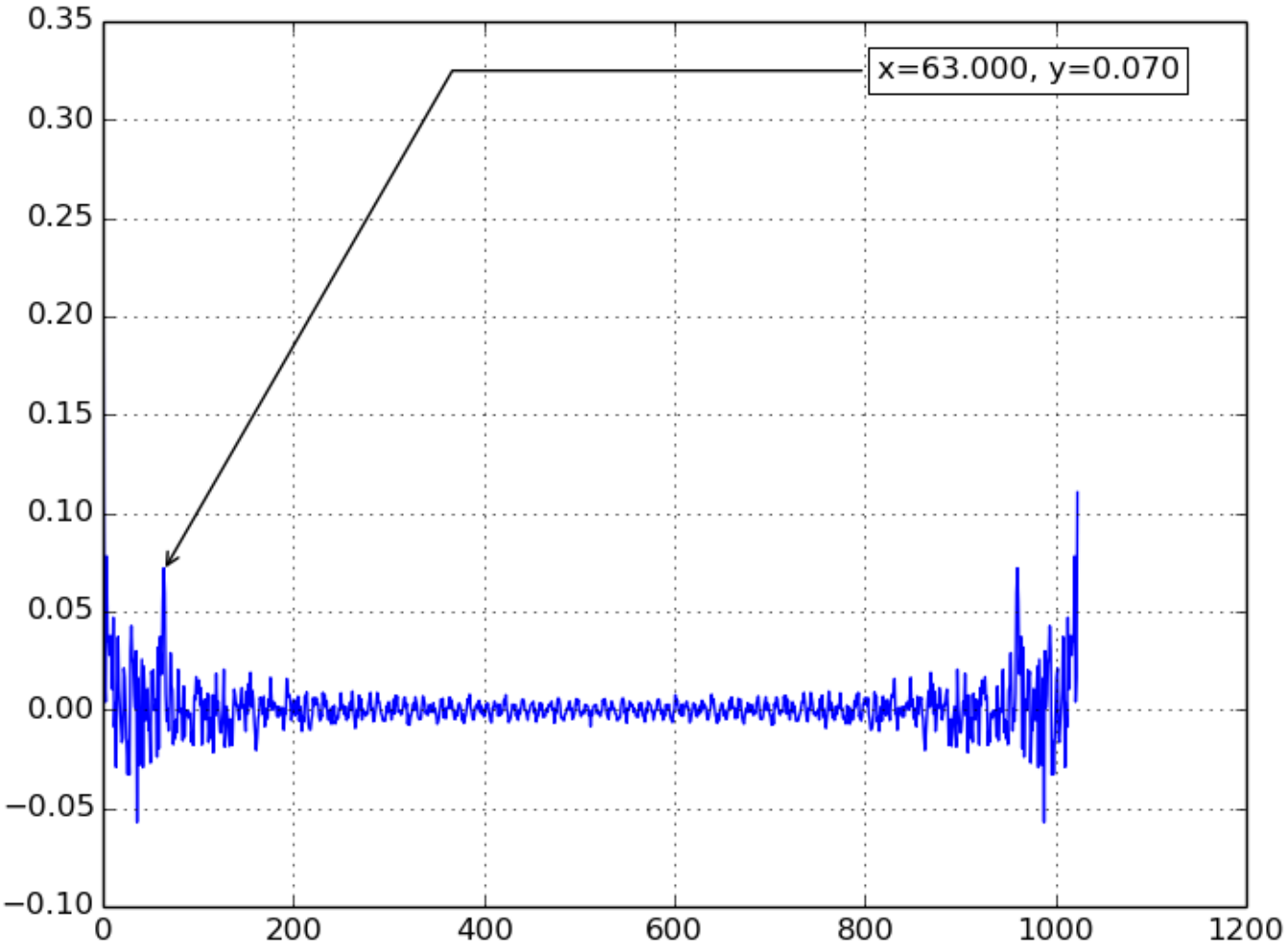
Code Snippet:

```
# Windowing the middle samples
win = data[int((len(data)-SampWin)/2):int((len(data)+SampWin)/2)]*np.hamming(SampWin)
#pre-emphasis
for i in range(1, len(win)):
        win[i] = win[i] - 0.95 * win[i-1]

# Calculating the real cepstrum
dft_c = np.log10(np.abs(np.fft.fft(win, 1024)))
cep = np.real(np.fft.ifft(dft_c))
```

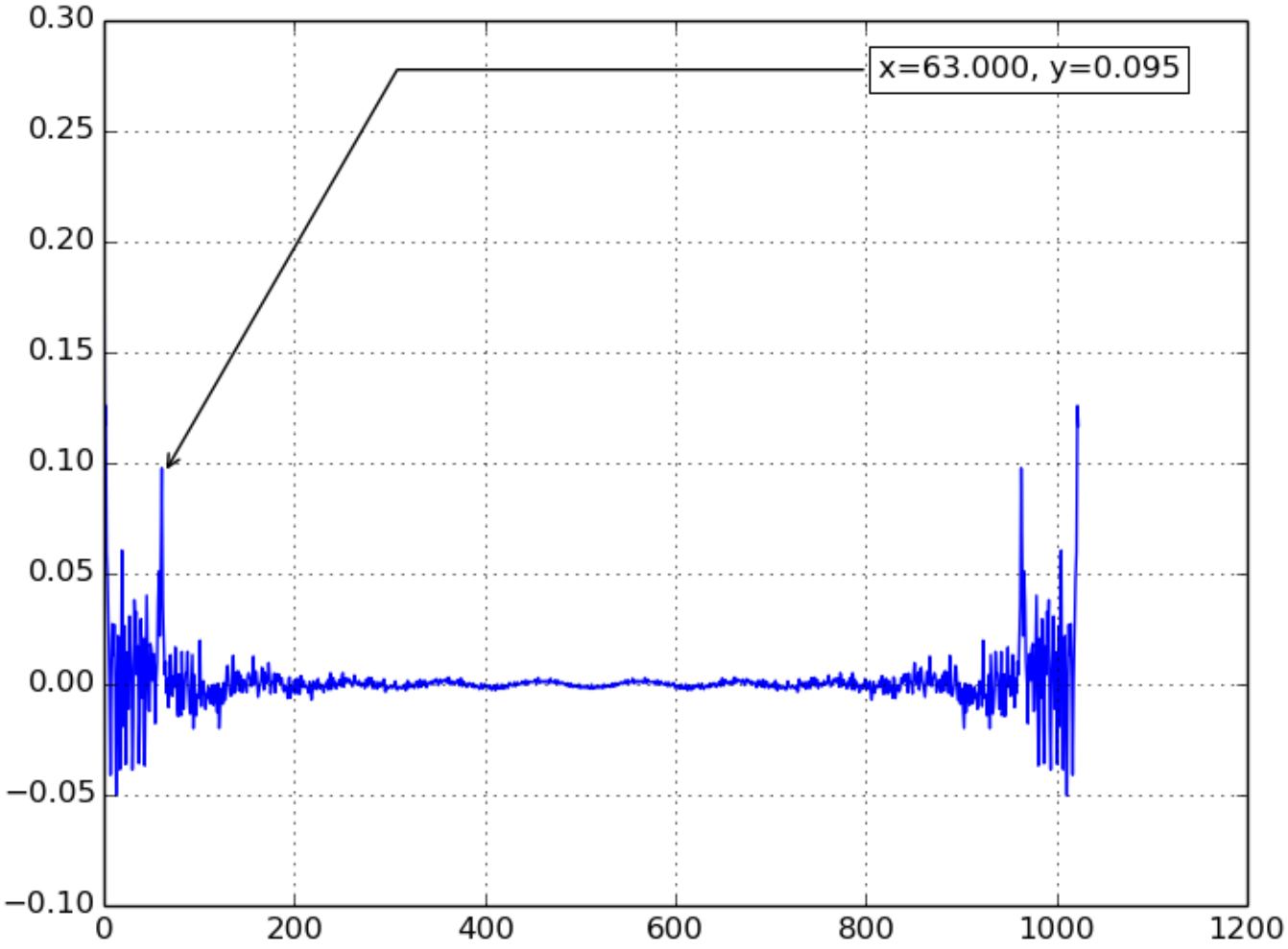
```
# Window the cepstrum

#cep[lift:(cep.shape[-1]-lift)] = 0

#frequency axis
dft_cep_lift = np.abs(np.fft.fft(cep, 1024))
freq_cep_lift = np.fft.fftfreq(dft_cep_lift.shape[-1], 1/float(SampFre))
plt.plot(freq_cep_lift[:len(freq_cep_lift)/2], 20*np.abs(dft_c[:len(dft_c)/2]), 'k')

for i in range (3,0,-1):
        lift = 20 + i*10
        # Window the cepstrum
        cep[lift:(cep.shape[-1]-lift)] = 0
        # Take the DFT

        dft_cep_lift = np.abs(np.fft.fft(cep, 1024))
        freq_cep_lift = np.fft.fftfreq(dft_cep_lift.shape[-1], 1/float(SampFre))
        #plot the DFT
        plt.plot(freq_cep_lift[:len(freq_cep_lift)/2], 20*np.abs(dft_cep_lift[:len(dft_ce
```

Question 3

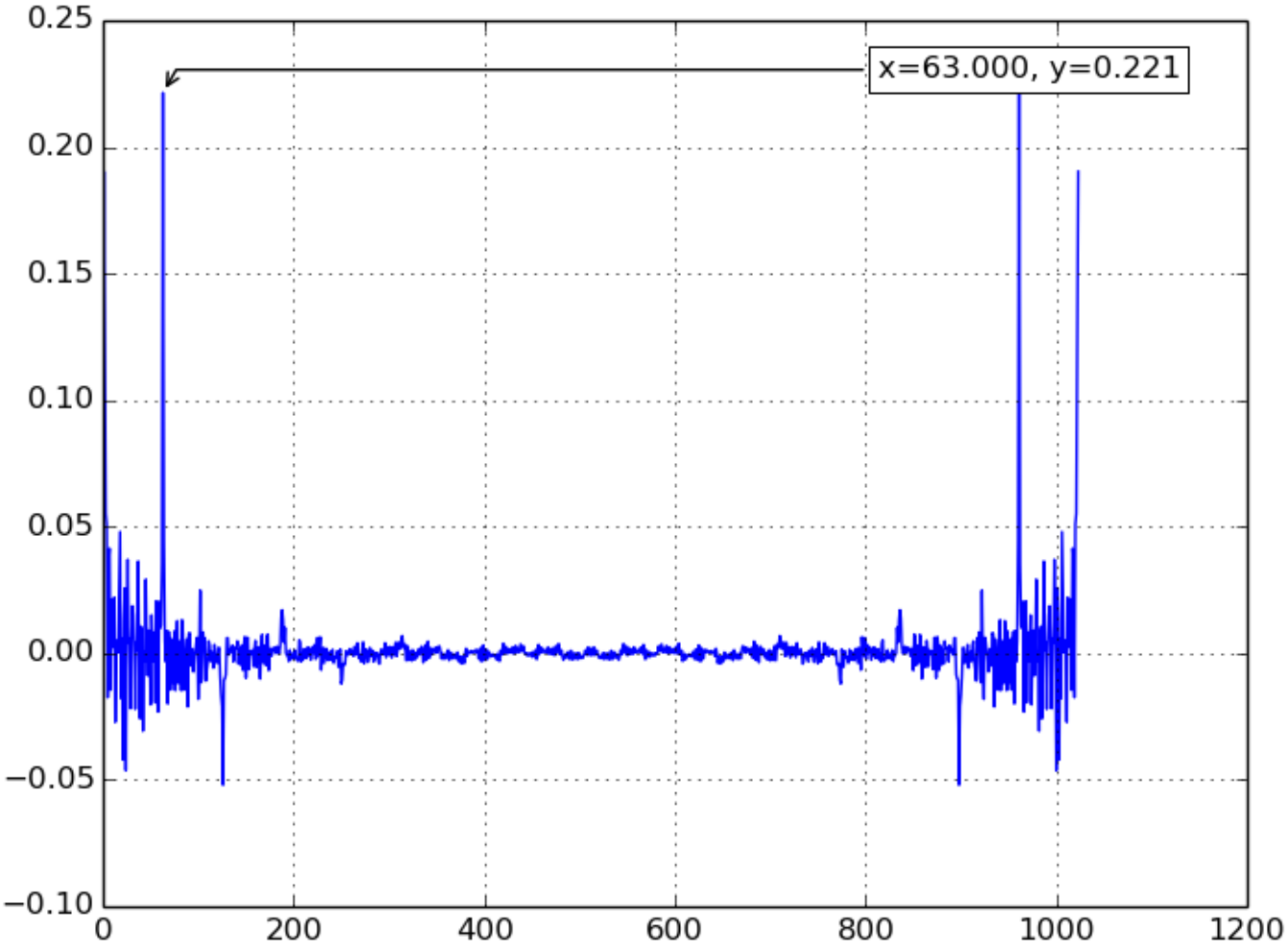
Real Cepstrum for /a/



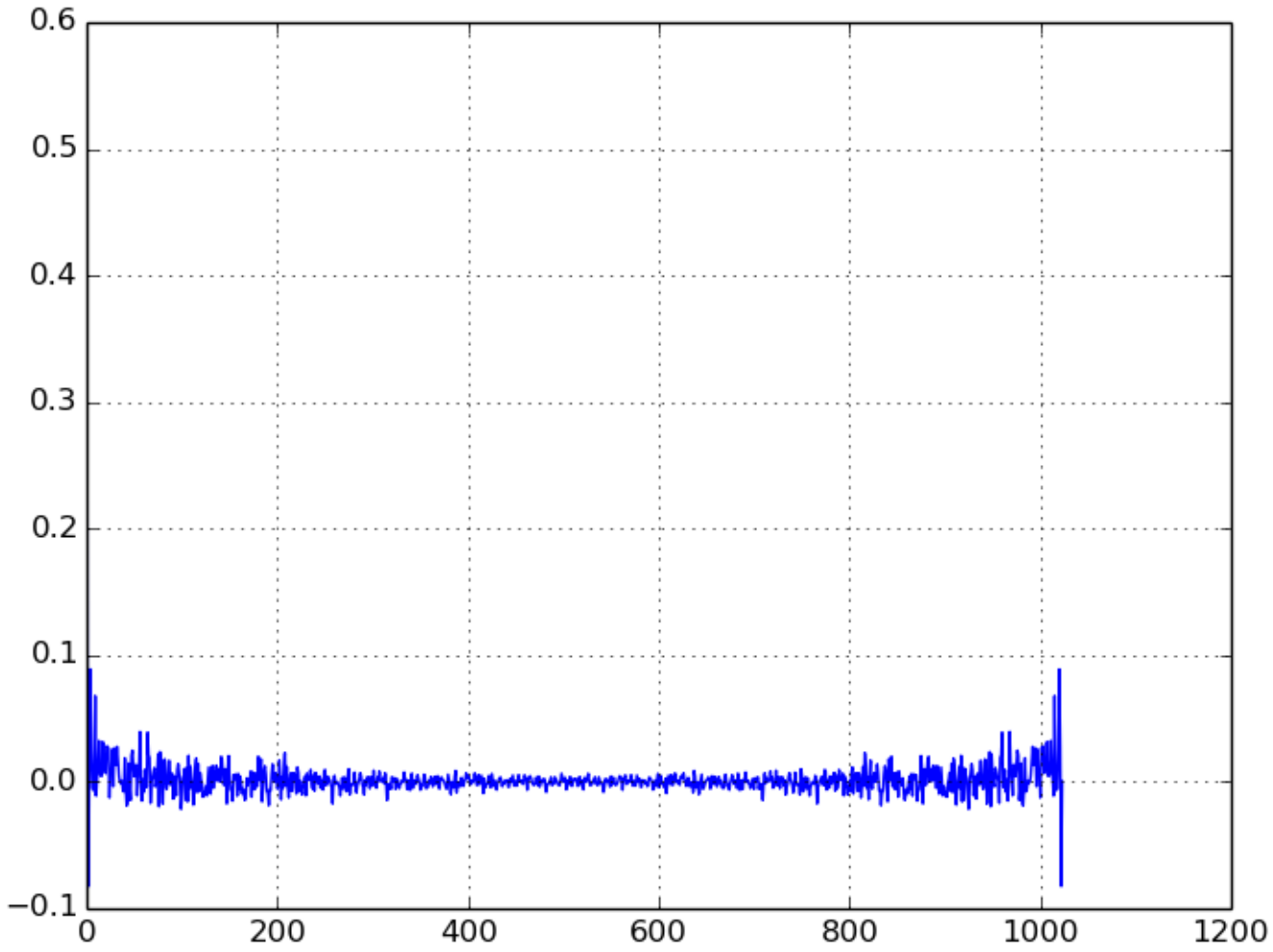
Real Cepstrum for /i/



Real Cepstrum for /n/



Real Cepstrum for /s/

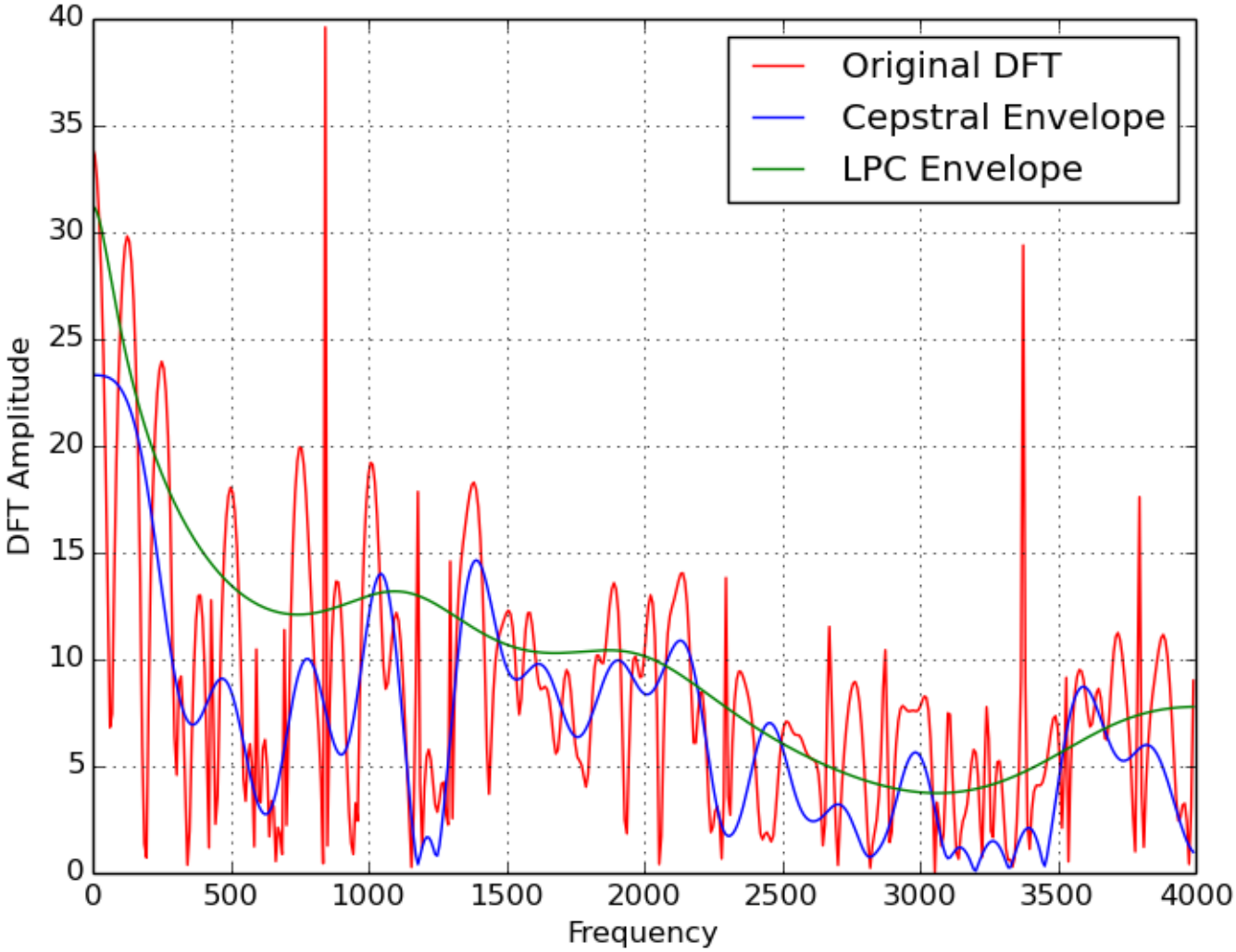


Comments:

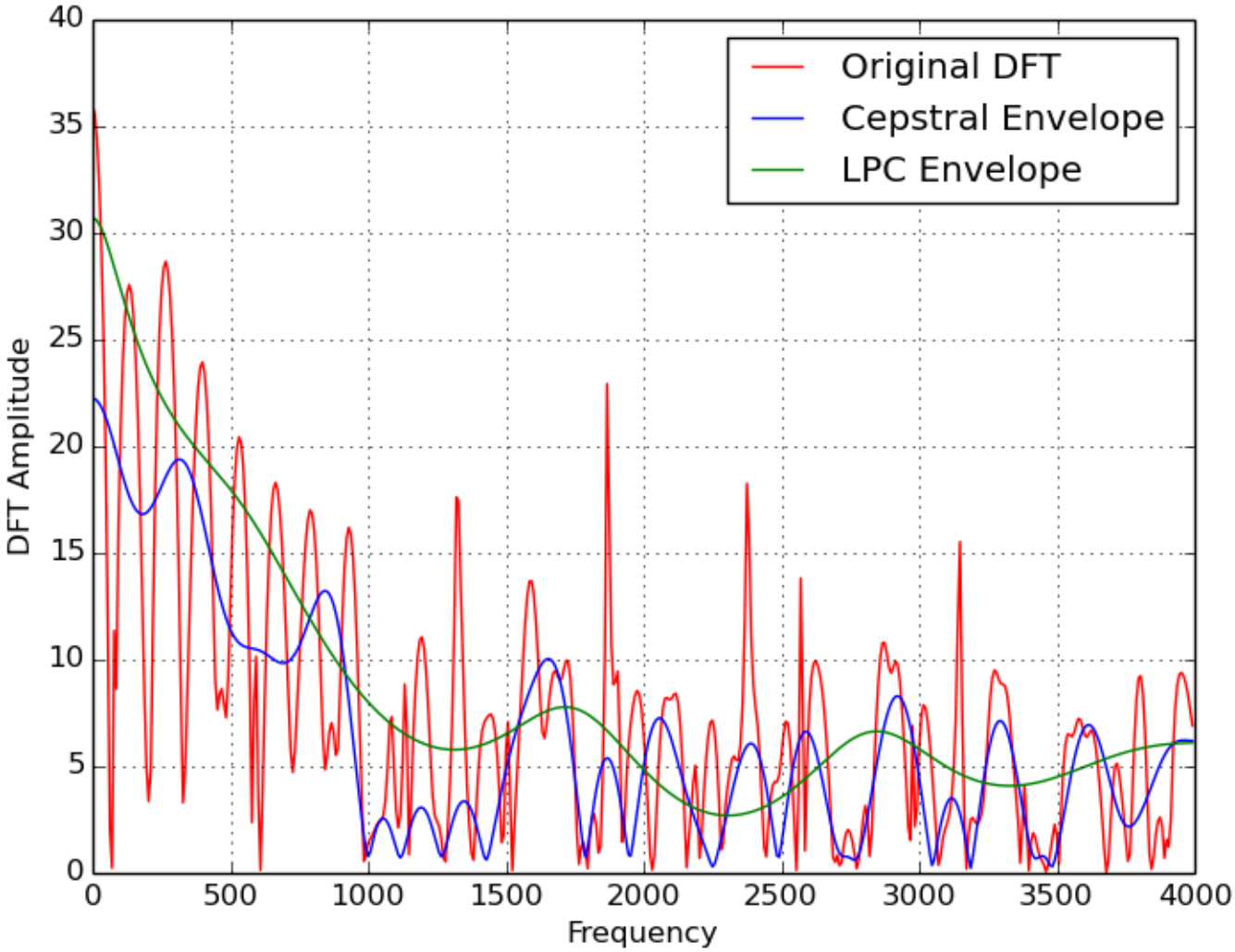
* for /a/, /i/ and /n/ ceptra has a peak around 60 samples. Hence for the pitch of the selected segment, Sampling rate is 8KHz, implies pitch = 8000/60 =133.33Hz.
* for /s/ ceptrum does not have any peak because it is unvoiced.

Spectral envelope for /a/

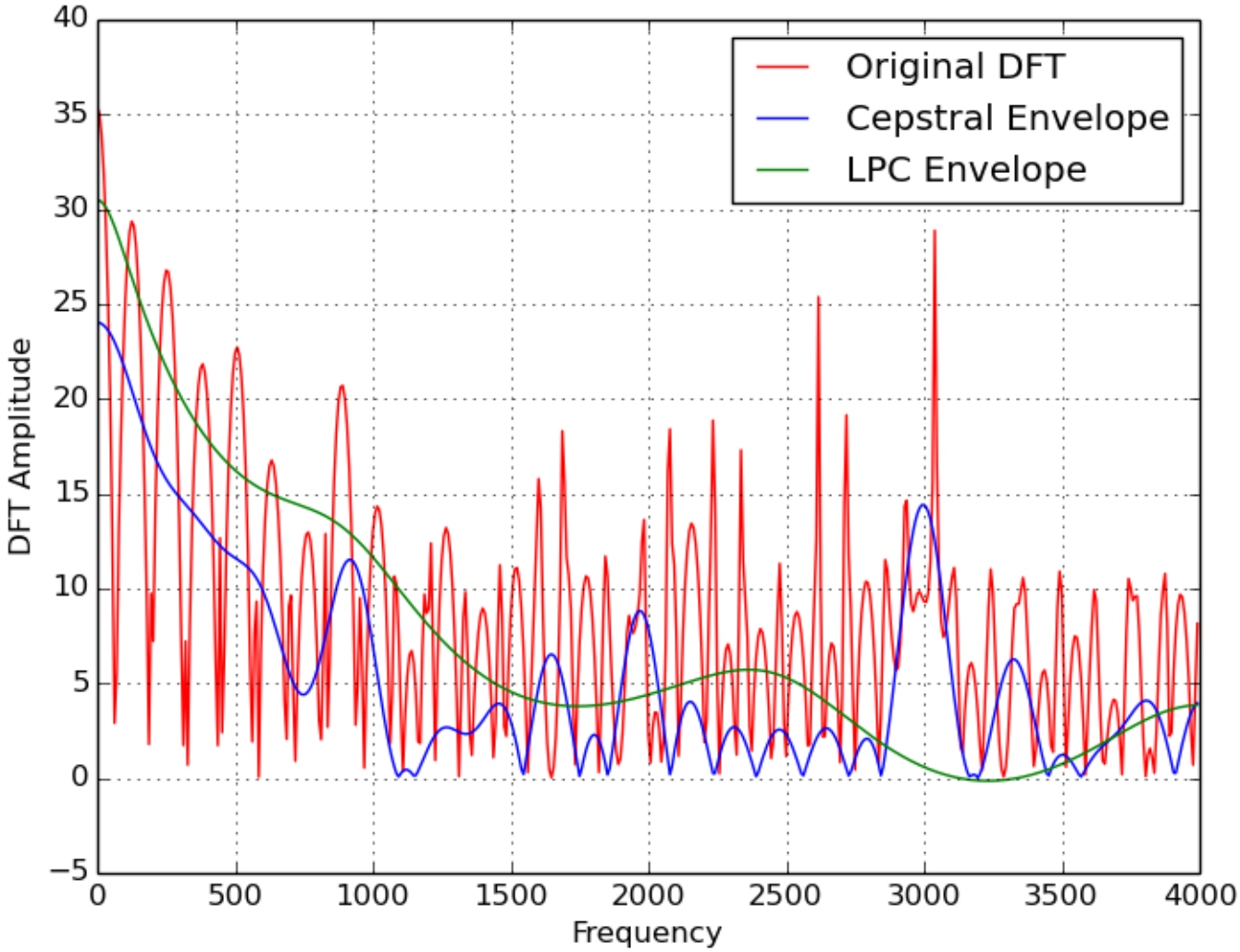
* In the red, Log spectrum of absolute of the DFT of the original signal.
* In the Blue, smoothed spectral envelope of real cepstrum for length of lifter = 30
* In the green, LP magnitude spectrum of p = 10 for /a/,/i/ and /n/ and for /s/ p = 18 is used.



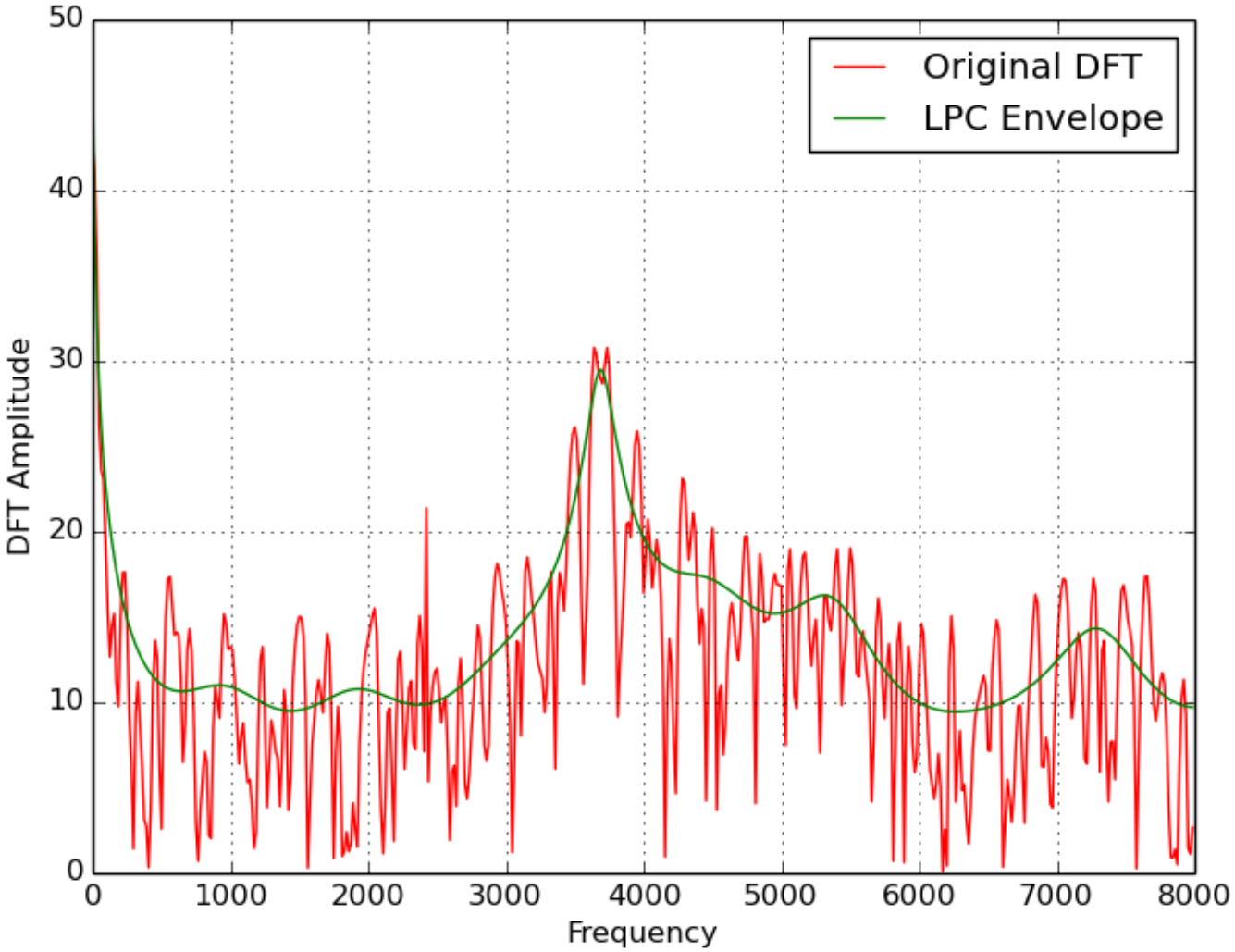
Spectral envelope for /i/



Spectral envelope for /n/



Vocal tract magnitude response of /s/



Code Snippet:

```
# Windowing the middle samples
win = data[int((len(data)-SampWin)/2):int((len(data)+SampWin)/2)]*np.hamming(SampWin)

# Calculating the real cepstrum
dft_c = np.log10(np.abs(np.fft.fft(win, 1024)))
cep = np.real(np.fft.ifft(dft_c))

# Liftering the cepstrum
lift = 30
# include both the sides of the cepstrum
cep[lift:(cep.shape[-1]-lift)] = 0

# Take the smoothed DFT
dft_cep_lift = np.abs(np.fft.fft(cep, 1024))
freq_cep_lift = np.fft.fftfreq(dft_cep_lift.shape[-1], 1/float(SampFre))

#compare with LP magnitude spectrum from last assigment using Levinson's algorithm.
```