Assigment 1

Professor Preeti Rao
EE 679 - Speech Processing

August 18, 2018

1 Question

Given the following specification for a single-formant resonator, obtain the transfer function of the filter H(z)from the relation between resonance frequency and bandwidth, and the pole radius and angle. Plot its magnitude response (dB magnitude versus frequency) and impulse response.

- F1 (formant) = 900 Hz
- B1(bandwidth) = 200 Hz
- Fs (sampling freq) = 16 kHz

We know the frequency response in analog domain is as follows:

$$H(j\Omega) = \frac{G}{(j\Omega + \sigma_p)^2 + \omega_p^2}$$

To convert in digital domain, we need to use the impulse invariance transform to obtain the mapping of s-plane to z-plane.

The poles in the z domain can be represented as:

$$z_p = r * e^{-j\theta}$$

where,

$$r = e^{-B_i \pi T}$$

$$\theta = 2\pi F_i T$$

$$T = \frac{1}{sampling\_frequency}$$

Hence, we obtain the transfer function of the two pole digital resonator as:

$$H(z) = \frac{k}{(1 - re^{j\theta}z{-}1)(1 - re^{-j^{-1}})}$$

which is equivalent to,

$$H(z) = \frac{k}{1 - 2rcos(\theta)z^{-1} + r^2z^{-2}}$$

The frequency response which I obtained is plotted below:

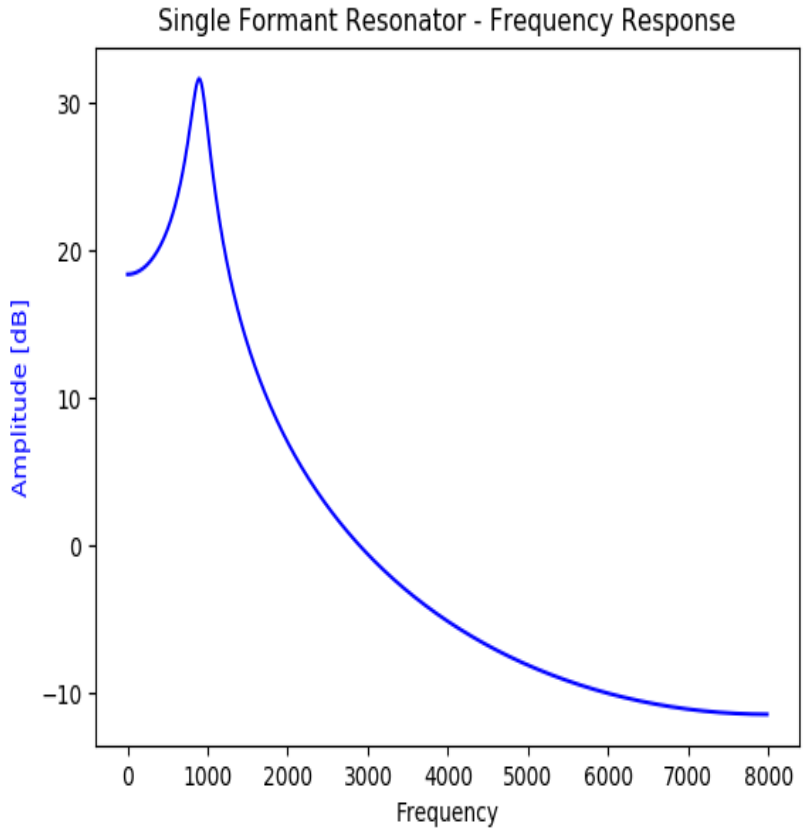

Figure 1: Frequency Response of Single Formant Resonator

The impulse response can be obtained by passing an impulse as an input to this system and calculating the output. The impulse response plot is shown below.

2 Question

Excite the above resonator (filter) with a periodic source excitation of$F0 = 140Hz$. You can approximate the source signal by narrow-triangular pulse train. Compute the output of the source-filter system over the duration of 0.5 second using the difference equation implementation of the LTI system. Plot the time domain waveform over a few pitch periods so that you can observe waveform characteristics. Play out the 0.5 sec duration sound and comment on the sound quality.

The input was approximated as periodic square waveform with small duty cycle.

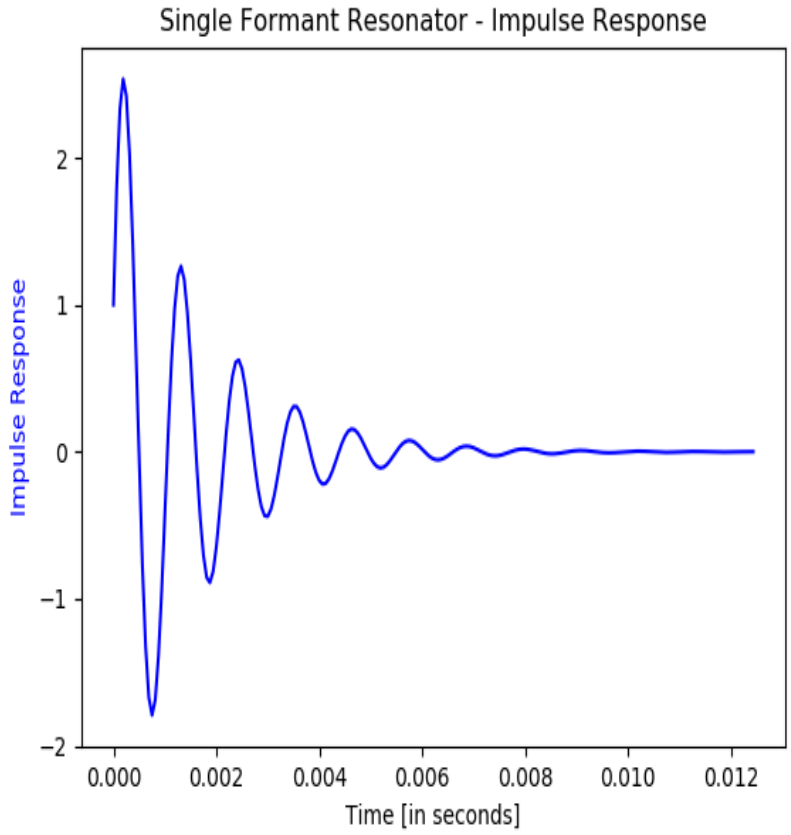

Figure 2: Impulse Response of Single Formant Resonator

An after thought: The input can be approximated as a sawtooth with vertical downslope. The motivation to use such a waveform instead of the hinted waveform was the rapid downslope relative to the upslope which is observed in the glottal excitations. The results after using this waveform were qualitatively better but the interpretability of the waveforms particularly was tedious. It would be interesting to delve into this further which I intend to.

The output was calculated using the difference equation implementation of LTI systems. Every LTI system can be accurately modelled as an recursive filter.
The corresponding difference equation which we obtain is:

$$y[n] = x[n] + 2rcos(\theta)y[n-1] - r^2y[n-2]$$

We will solve this filter recursively where we set the initial conditions, $y[n-1] = 0$ and $y[n-2] = 0$.
The exact implementation in `Python` is mentioned at the end of this paper.

Here is the output of the single formant resonator we designed in previous question when excited by the signal of frequency $140Hz$.

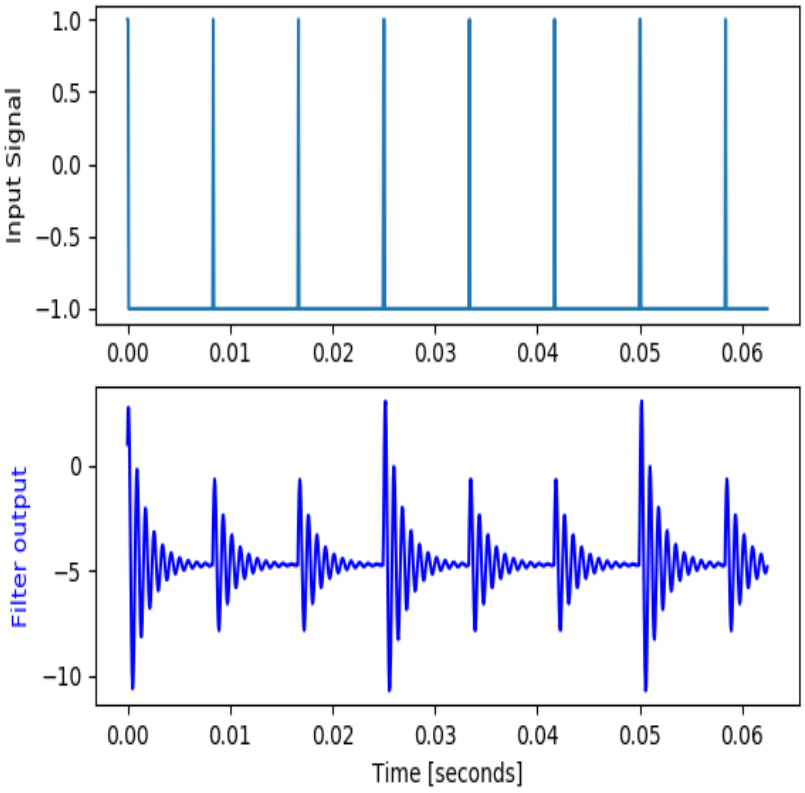

Figure 3: Impulse Response of Single Formant Resonator

The output was of low frequency and rather noisy [difficult to interpret] What I mean by low frequency is that the contribution of the higher frequencies in the signal was relatively low which was apparent in the audio

3 Question 3

Vary the parameters as indicated below and comment on the differences in waveform and sound quality for the different parameter combinations.

- F0=120Hz,F1=300Hz,B1=100Hz
- F0=120Hz,F1=1200Hz,B1=200Hz
- F0=180Hz,F1=300Hz,B1=100Hz

Keeping the frequency of the input signal as 120 Hz, the following plots have been plotted for the given cases in the respective order.

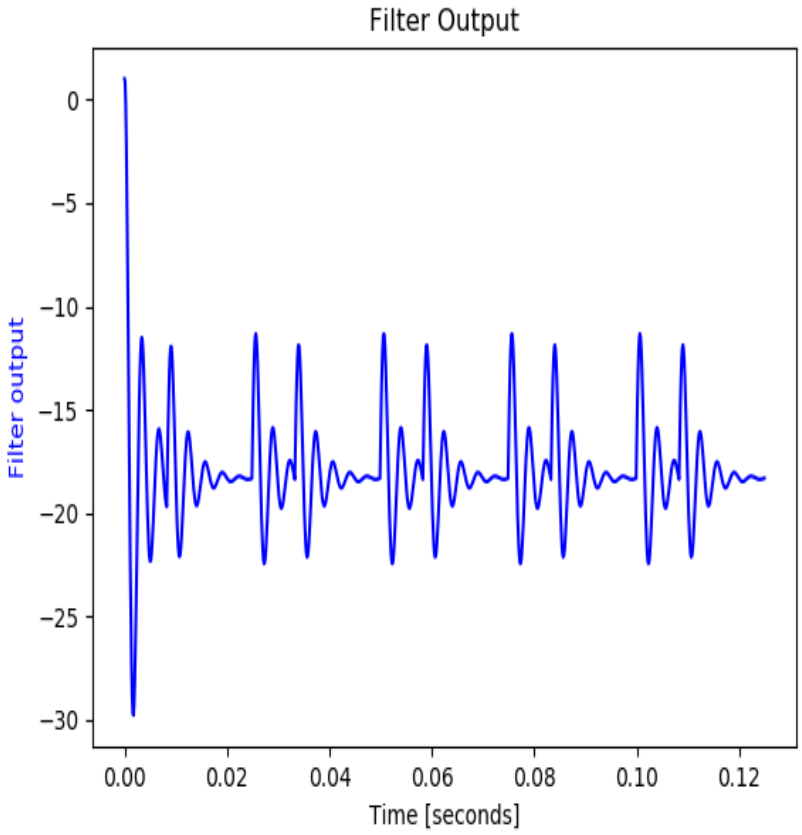

Figure 4: F0=120Hz,F1=300Hz,B1=100Hz

Comparing 1st and 3rd waveform which differ in the input signal frequency, we can observe that the pitch is higher of the 3rd one than that of the first waveform. This fact shows itself when you play out the waveforms. This isn't surprising because the source excitation frequency will definitely control the pitch being perceived. The formant frequency and the bandwidth which are essentially the parameters and this case the only parameters of the system, the nature of sound which we hear doesn't change which is well reflected in the waveforms. If we were to establish an analogy with the human vocal tract, here the vocal tract parameters remain the same but the vibration of the glottis changes in frequency.
Comparing the 1st and 2nd waveforms which differ in the formant frequency and bandwidth, we observe that the waveforms in their cosmetic appearances look very different and which is expected and intuitive if we consider the previously mentioned analogy. When the waveforms are played, the sounds do sound very differently with the pitch of the 2nd waveform dominating over the 1st one. But the pitch is not really a concern in the comparative analysis we are doing here. The second waveform shows a sharp decline to the pulse locally which is not exhibited by the 1st waveform which rather has a more or less smooth waveform. In audio this shows up as abruptness in sounds vs a smooth sound. We can also see that the pitch of the 2nd waveform is clearly controlled by the input signal.

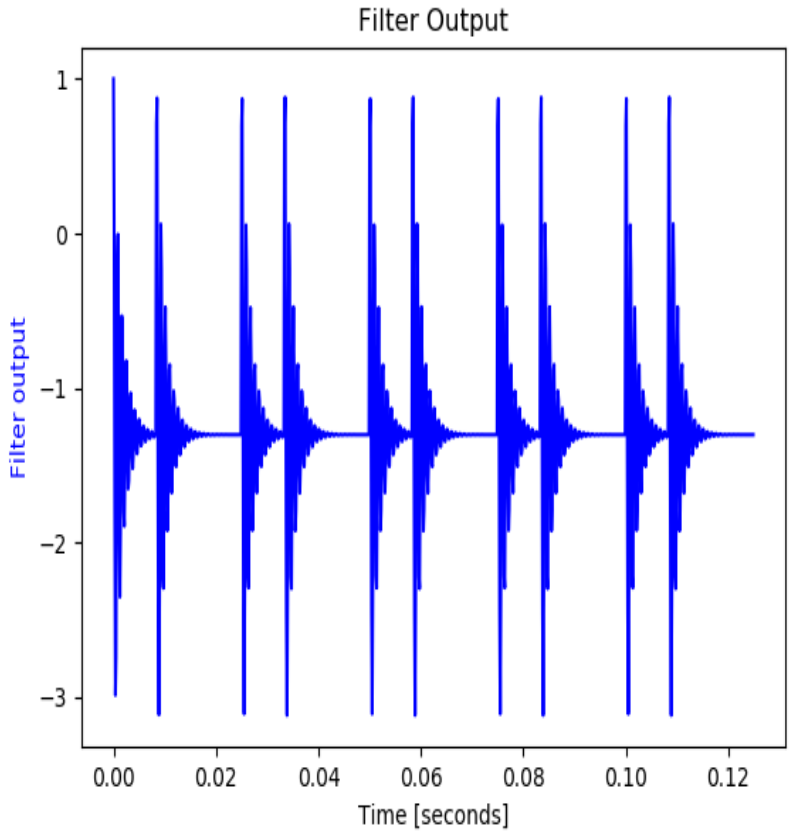

Figure 5: F0=120Hz,F1=1200Hz,B1=200Hz

4 Question 4

In place of the simple single-resonance signal, synthesize the following more realistic vowel sounds at two distinct pitches (F0 = 120 Hz, F0 = 220 Hz). Keep the bandwidths constant at 100 Hz for all formants. Duration of sound: 0.5 sec

Vowel F1, F2, F3
/a/ 730, 1090, 2440
/i/ 270, 2290, 3010
/u/ 300, 870, 2240
The produce vowels though are distinctly interpretable, are noisy nevertheless.
For F0 = 120Hz, following are the waveforms respectively,

For F0 = 220Hz, following are the waveforms respectively,

5 Question

Signal Analysis: Compute the DTFT magnitude (dB) spectrum of any 2 of the vowel sounds you have synthesized in Q3. Use rectangular and Hamming windows of lengths: 5 ms, 10

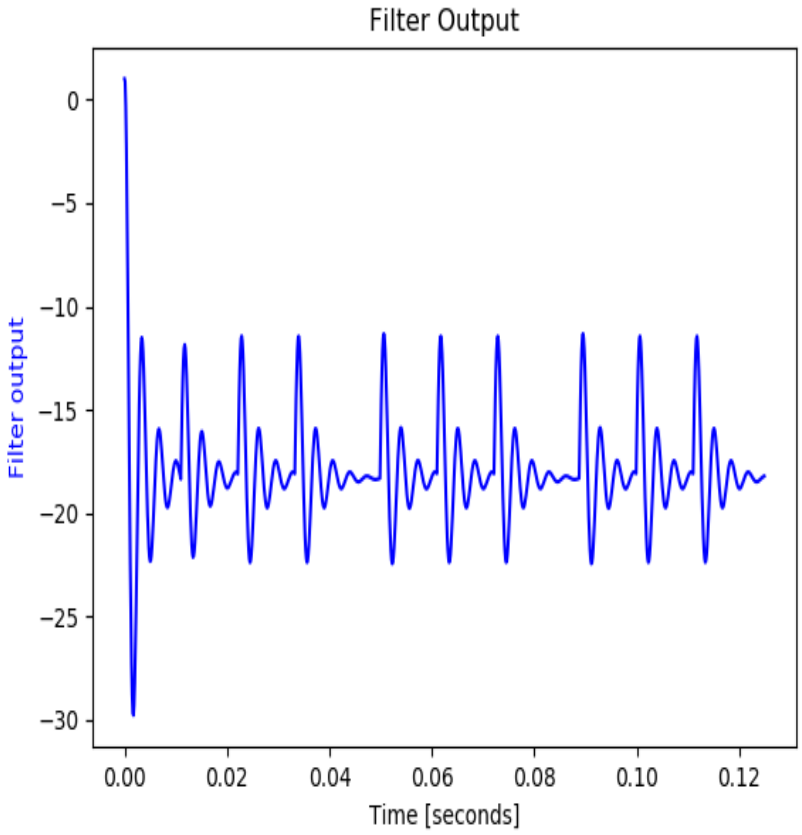

Figure 6: F0=180Hz,F1=300Hz,B1=100Hz

ms, 20 ms, 40 ms, each with a large zero- padded DFT. (i) Comment on the similarities and differences between the different spectra. (ii) Estimate the signal parameters from each of the magnitude spectra and compare with the ground-truth.

The DFT length chosen was 1024 to ensure sufficient padding.
The chosen vowels are /a/ and /i/ at F0 = 220 Hz

The plots are as follows:

We observe when the window length is small, the spectra is very smooth and has wide mainlobe width which is not surprising but as a matter of fact, expected since the width of the main lobe is inversely proportional to the window length. Keeping the window small vanishes out the harmonic(pitch) information. Instead what we can observe are the formants. The pitch information is preserved when we use longer window lengths.
Also, we observe that the spectra of hamming window has wider main lobe than the rectangular window. Also, the rate of decay of energy as we go from the main lobe to the sidelobes is higher in case of hamming window than the rectangular window. Hence we can say that if we want to detect formants, we should use hamming window and if we intend to extract the pitch of the signal, we should use rectangular window.

Signal Feature Extraction

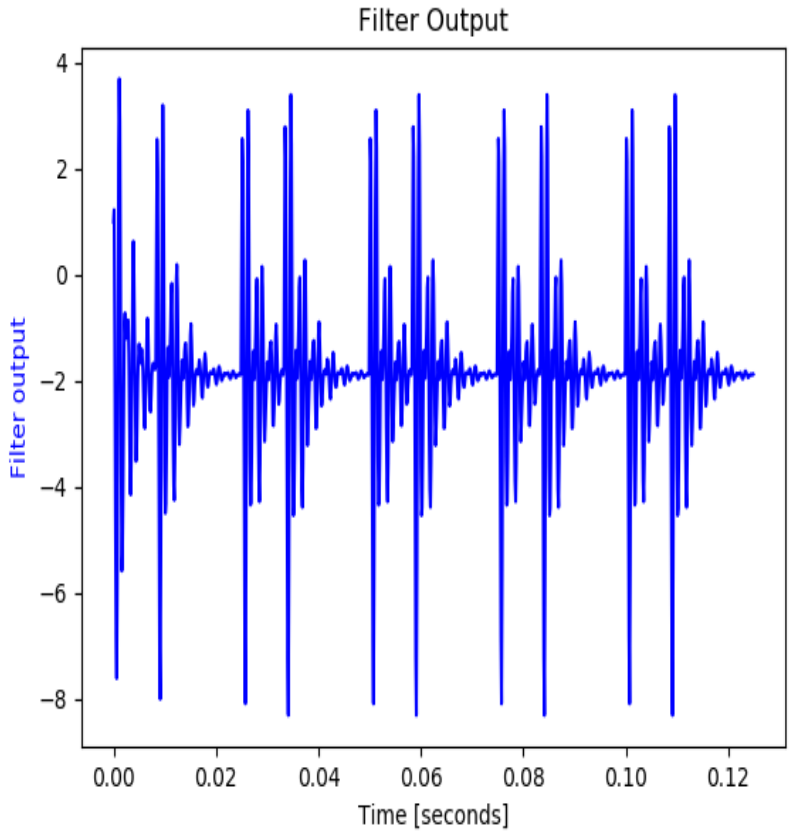

Figure 7: /a/ at pitch=120Hz

For /a/, the formants observed at various window lengths are listed as follows:
Rectangular Window

- At window size = 5ms, formants are: 752.34, 1043.81, 2430.27 (All in Hertz)
- At window size = 10ms formants are: 752.34, 1114.71, 2414.51
- At window size = 20ms formants are: 752.34, 1091.08, 2422.39
- At window size = 40ms formants are: 744.46, 1106.83, 2430.27

Hamming Window

- At window size = 5ms, formants are: 752.34, 1035.93, 2438.15
- At window size = 10ms formants are: 760.22, 1098.95, 2414.51
- At window size = 20ms formants are: 760.22, 1098.95, 2422.39
- At window size = 40ms formants are: 744.46, 1098.95, 2422.39

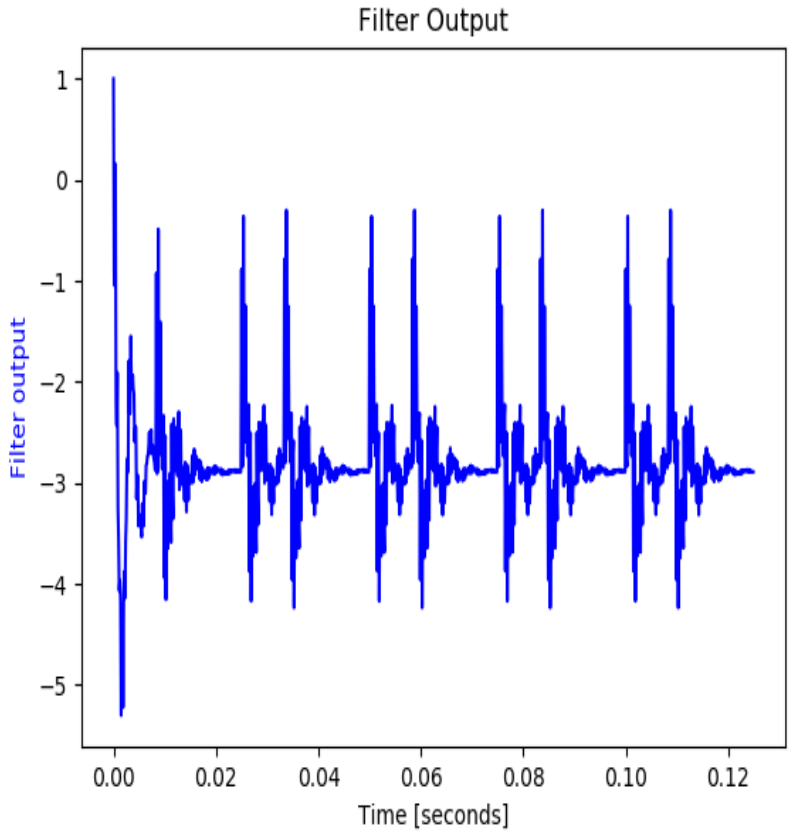

Figure 8: /i/ at pitch=120Hz

Number of harmonics in /a/: Considering the hamming windows of size 20 since the harmonics are best seen in this plot, there are 6 clear harmonics between 0 and 1000 Hz and 5 harmonics between 1000 and 2000 Hz. Hence the pitch is around 200Hz.

For /e/, the formants observed at various window lengths are listed as follows:

Rectangular Window

- At window size = 5ms, formants are: 2312.10, 3005.33
- At window size = 10ms formants are: 295.44, 2296.35, 3005.33
- At window size = 20ms formants are: 311.19, 2296.35, 3060.48
- At window size = 40ms formants are: 303.32, 2288.47, 3021.09

Hamming Window

- At window size = 5ms, formants are: 319.07, 2327.86, 3005.33
- At window size = 10ms formants are: 271.80, 2288.47, 2965.95

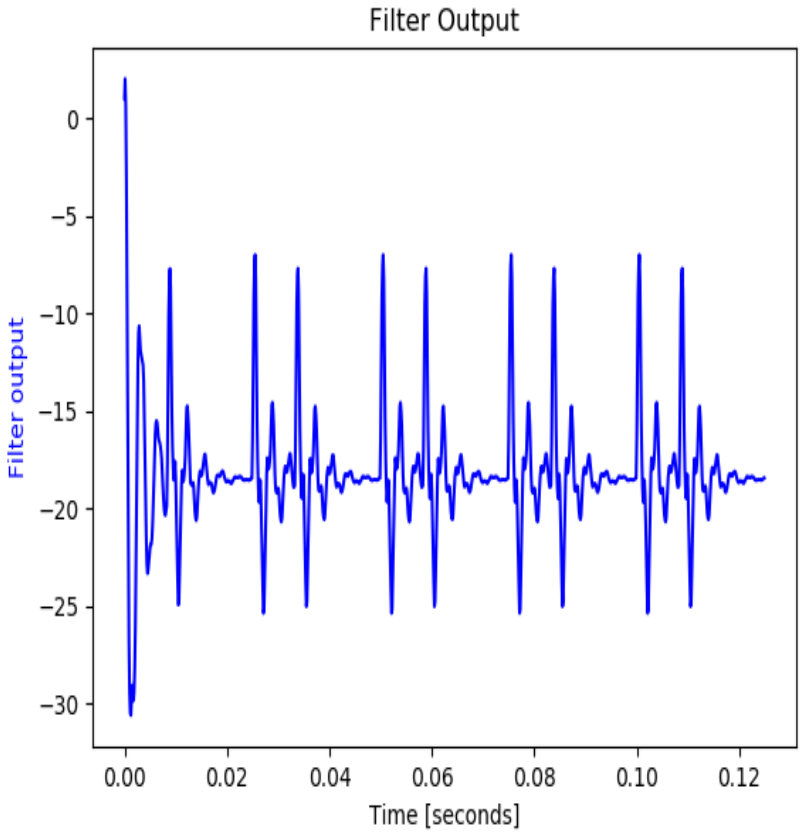

Figure 9: /o/ at pitch=120Hz

Number of harmonics in /b/: Considering the hamming windows of size 20 since the harmonics are best seen in this plot, there are 6 clear harmonics between 0 and 1000 Hz. Hence the pitch is around 170Hz.

6 Code

6.1 parameter import file

```
formant_freq = [730, 1090, 2440]
formant_bw = [100, 100, 100]
samp_freq = 8000.0
sig_freq = 220
time_length = 0.5
win_size = 20
dft_len = 1024
```

Listing 1: Python example

6.2 Question1

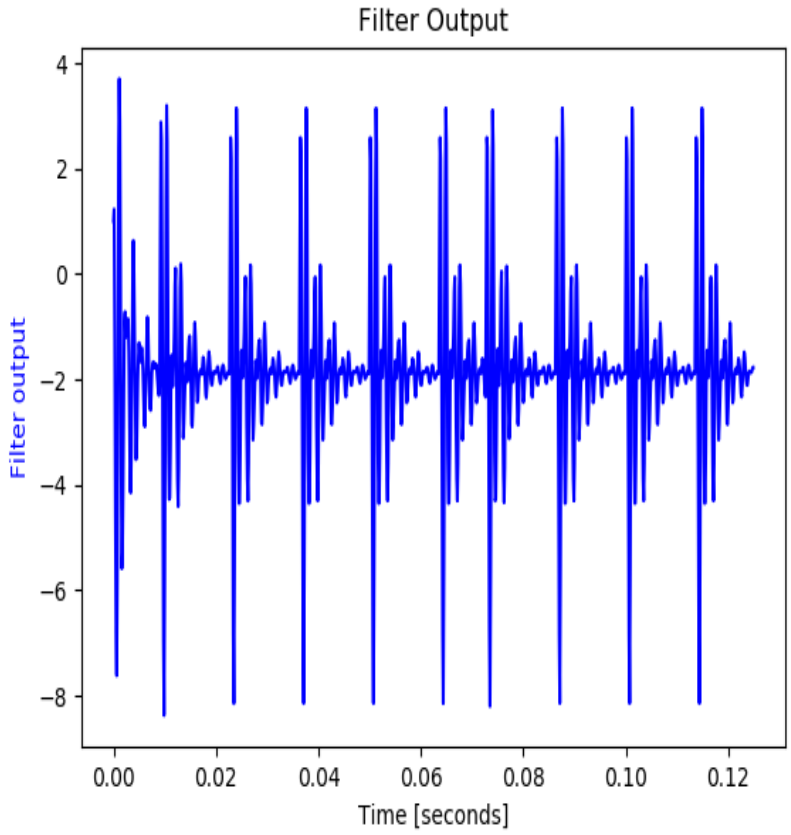

Figure 10: /a/ at pitch=220Hz

```
1
2  from scipy import signal
3  import numpy as np
4  from math import pi
5  import matplotlib.pyplot as plt
6  from scipy.io.wavfile import write
7  import pylab
8
9  F1 = 900
10 B1 = 200
11 Fs = 16000
12 T = 1.0/Fs
13
14
15 # Calculate pole angles and radii
16 R = np.exp(-pi*B1/Fs)
17 theta = 2*pi*F1/Fs
18
19 # Get poles and an equal number of zeros
20 poles = np.array([R * np.exp(1j*theta), R * np.exp(-1j*theta)])
21 zeros = np.zeros(poles.shape, poles.dtype)
```

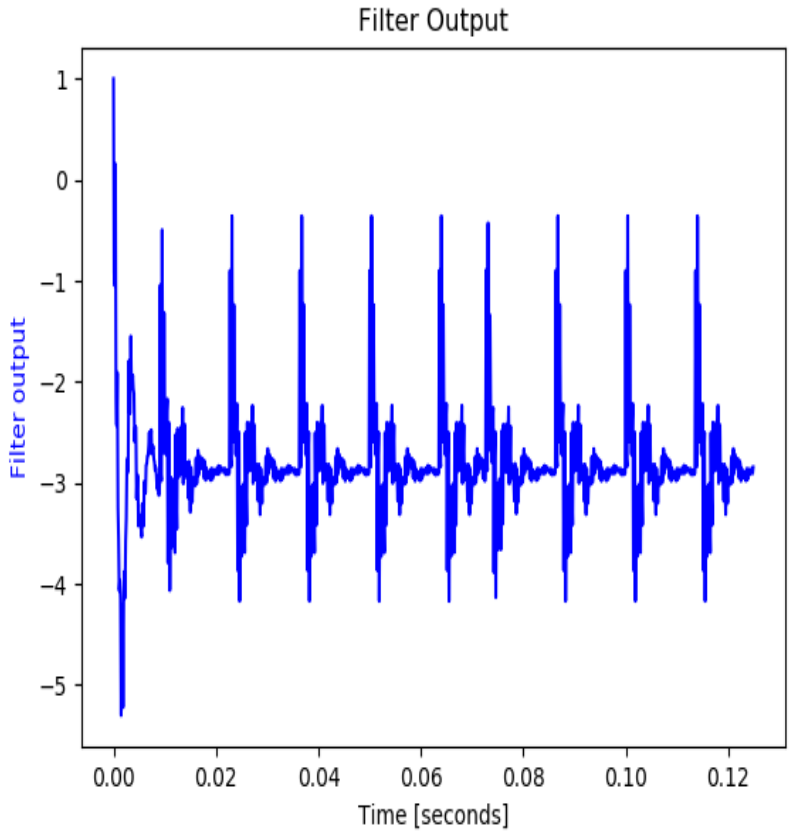

Figure 11: /i/ at pitch=220Hz

```
22
23 b, a = signal.zpk2tf(zeros, poles, 1)
24
25 w, h = signal.freqz(b, a)
26 fig1 = plt.figure()
27 plt.title('Single Formant Resonator - Frequency Response')
28 plt.plot(Fs*w/(2*pi), 20 * np.log10(abs(h)), 'b')
29 plt.ylabel('Amplitude [dB]', color='b')
30 plt.xlabel('Frequency')
31 pylab.savefig('./results/' + 'q1' + '_' + 'freq_response' + '.png')
32
33 fig2 = plt.figure()
34 angles = np.unwrap(np.angle(h))
35 plt.plot(Fs*w/(2*pi), angles, 'g')
36 plt.ylabel('Angle (in radians)', color='g')
37 plt.grid()
38 plt.axis('tight')
39 pylab.savefig('./results/' + 'q1' + '_' + 'phase_response' + '.png')
40
41
42
```

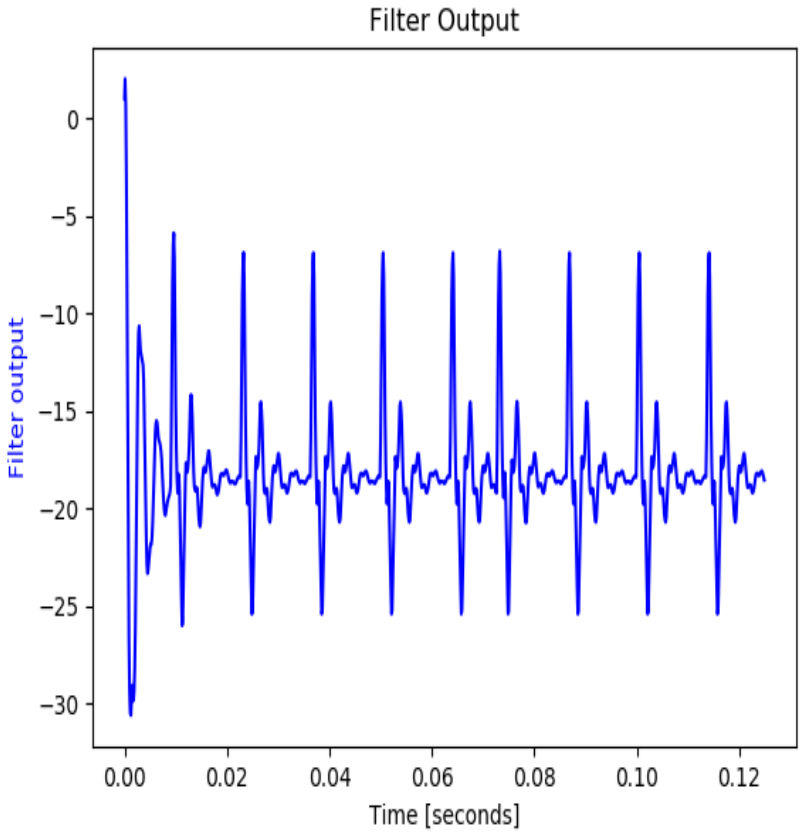

Figure 12: /o/ at pitch=220Hz

```
pulse = np.zeros([200], 'float64')
pulse[0] = 1
y = np.zeros(pulse.shape, pulse.dtype)
time = np.linspace(0, len(pulse)/float(Fs), 200, endpoint=False)

for i in range(len(pulse)):
    y[i] = y[i] + a[0]*pulse[i]
    for j in range(1, len(a)):
        if i-j >= 0:
            y[i] = y[i] - a[j]*y[i-j]
fig3 = plt.figure()
plt.title('Single Formant Resonator - Impulse Response')
ax3 = fig2.add_subplot(111)
plt.plot(time, y, 'b')
plt.ylabel('Impulse Response', color='b')
plt.xlabel('Time [in seconds]')
pylab.savefig('./results/' + 'q1' + '_' + 'impulse_response' + '.png')
```

Listing 2: Python example

6.3 Question2

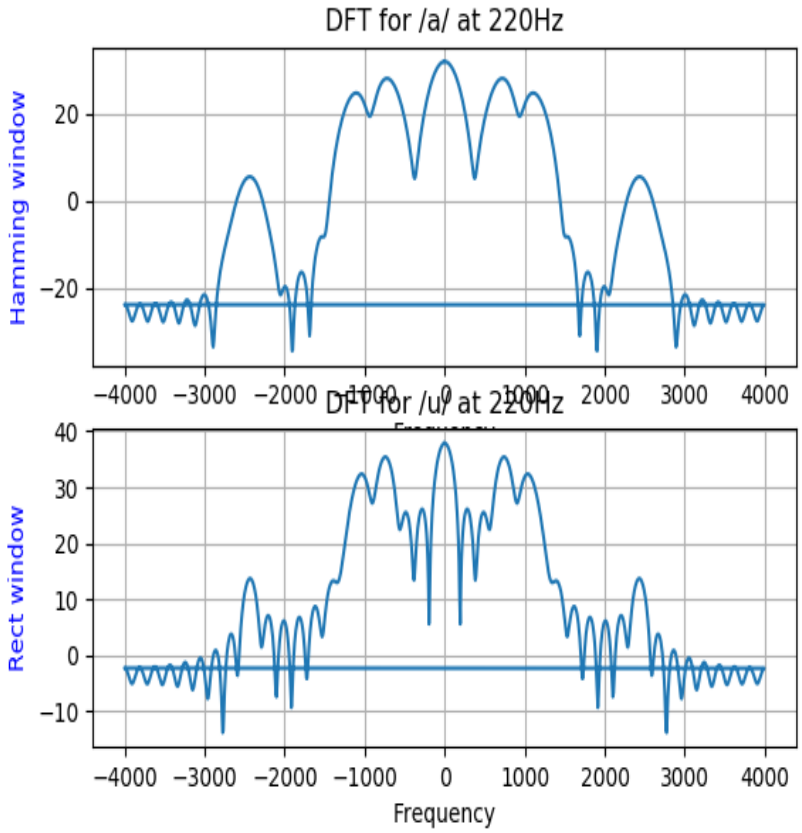

Figure 13: DTFT of /a/ at pitch=220Hz with window length = 5ms

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

# Create a signal of frequency F0Hz.
# Signal time length = t0 seconds
# Signal sampling frequency is Fs

F1 = np.array(hparams.formant_freq)
B1 = np.array(hparams.formant_bw)
Fs = hparams.samp_freq
T = 1.0/Fs
t0 = hparams.time_length
num_samples = Fs*t0
F0 = hparams.sig_freq
```

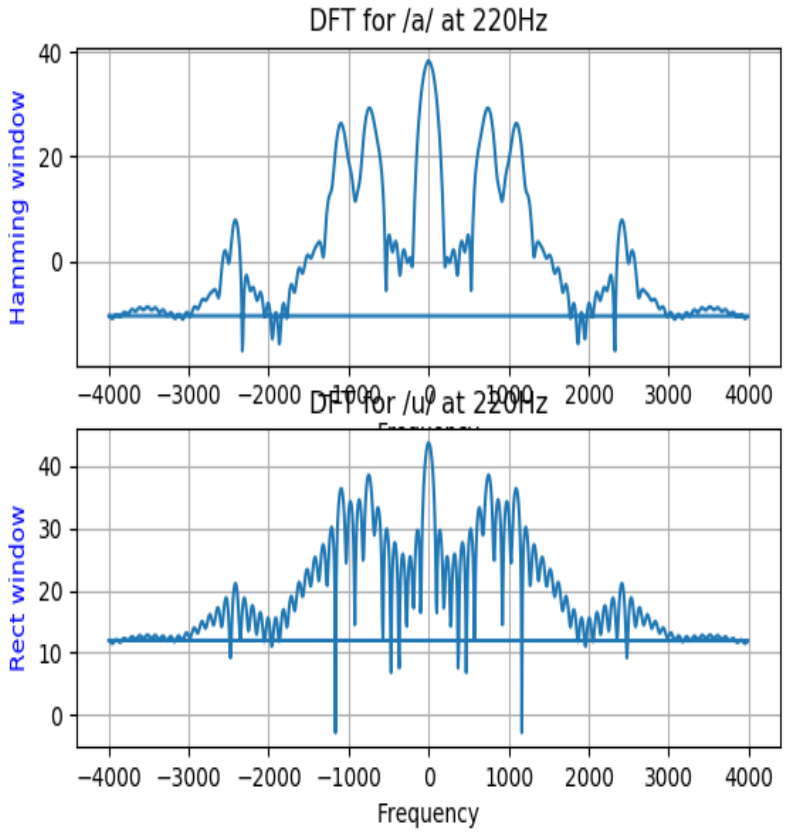

Figure 14: DTFT of /a/ at pitch=220Hz with window length = 10ms

```
t = np.linspace(0, t0, num_samples)
fig = plt.figure()

sig = signal.square(2 * np.pi * F0 * t, duty=0.01)
#sig = signal.sawtooth(2 * np.pi * F0 * t)

ax1 = fig.add_subplot(211)
plt.ylabel('Input Signal')
plt.plot(t[0:1000], sig[0:1000])

# Calculate pole angles and radii
R = np.exp(-pi*B1/Fs)
theta = 2*pi*F1/Fs

# Get poles and an equal number of zeros
poles = np.array([R * np.exp(1j*theta), R * np.exp(-1j*theta)])
zeros = np.zeros(poles.shape, poles.dtype)

b, a = signal.zpk2tf(zeros, poles, 1)

y = np.zeros(sig.shape, sig.dtype)
```

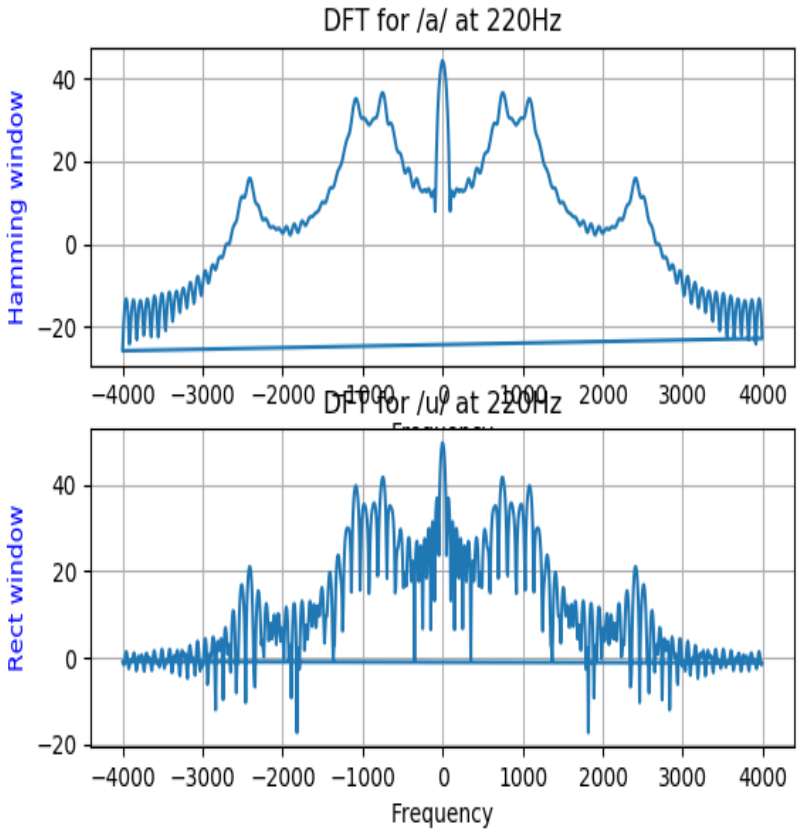

Figure 15: DTFT of /a/ at pitch=220Hz with window length = 20ms

```
43
44 for i in range(len(sig)):
45     y[i] = y[i] + a[0]*sig[i]
46     for j in range(1, len(a)):
47         if i-j >= 0:
48             y[i] = y[i] - a[j]*y[i-j]
49
50 # plt.title('Filter Output')
51 ax2 = fig.add_subplot(212)
52
53 plt.plot(t[0:1000], y[0:1000], 'b')
54 plt.ylabel('Filter output', color='b')
55 plt.xlabel('Time [seconds]')
56 pylab.savefig('./results/' + 'q2' + '_' + 'trial0' + '.png')
57 write('trial0.wav', Fs, y)
58 plt.show()
```

Listing 3: Python example

6.4 Question3

```
1 import numpy as np
```

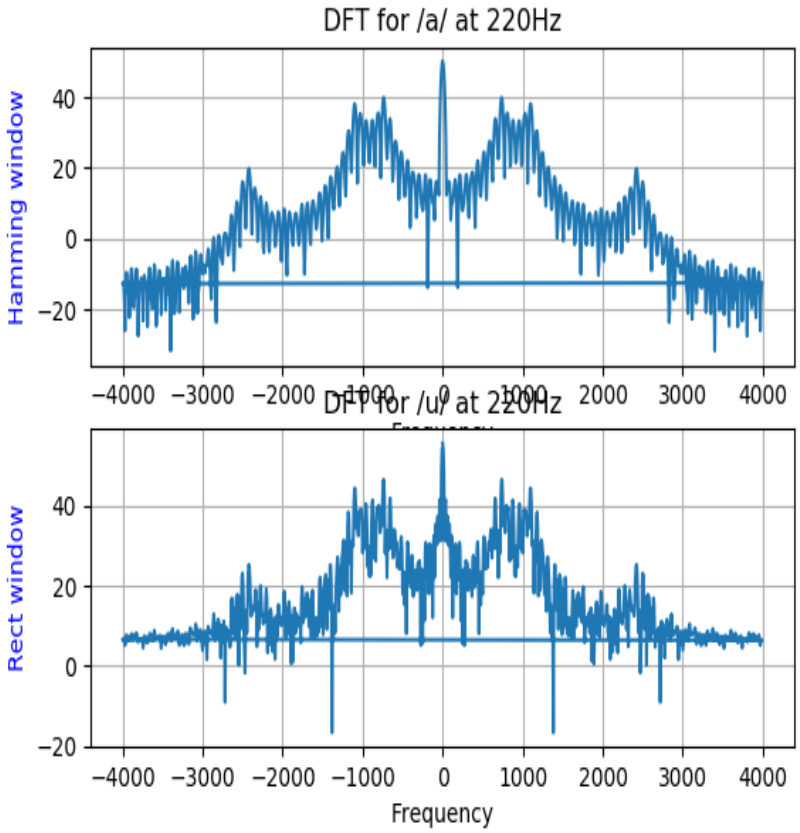

Figure 16: DTFT of /a/ at pitch=220Hz with window length = 40ms

```
from matplotlib import pyplot as plt
import pylab
from scipy import signal
from scipy.io.wavfile import write
import hparams
import sys
from math import pi
from scipy.io.wavfile import write


F1 = np.array(hparams.formant_freq)
B1 = np.array(hparams.formant_bw)
Fs = hparams.samp_freq
T = 1.0/Fs
t0 = hparams.time_length
num_samples = Fs*t0
F0 = hparams.sig_freq
t = np.linspace(0, t0, num_samples)
sig = signal.square(2 * np.pi * F0 * t, duty=0.01)


```

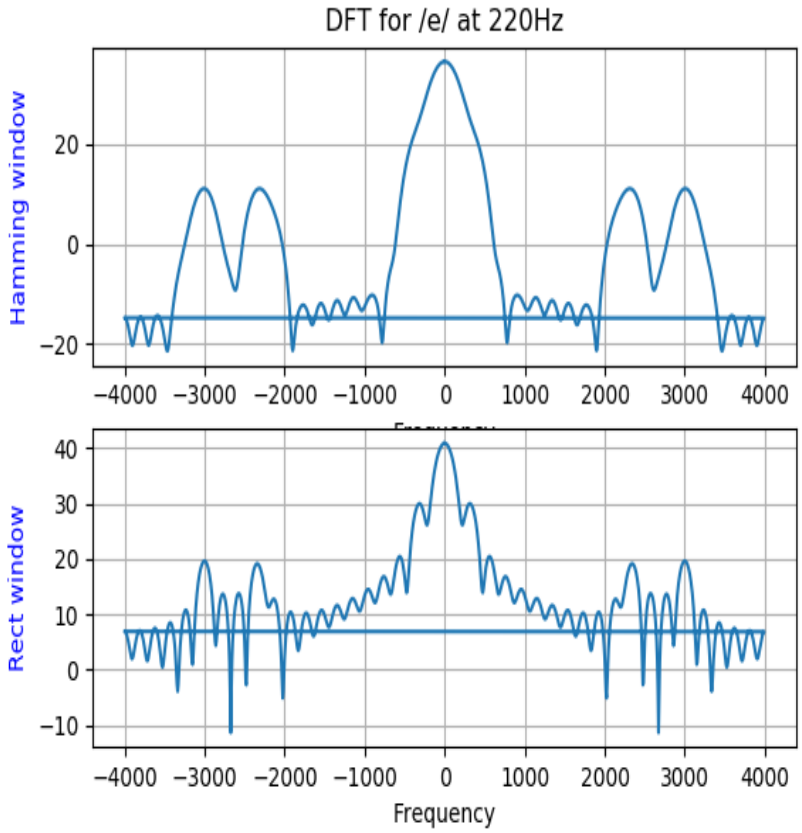

Figure 17: DTFT of /i/ at pitch=220Hz with window length = 5ms

```
23
24 # Calculate pole angles and radii
25 R = np.exp(-pi*B1/Fs)
26 theta = 2*pi*F1/Fs
27
28
29 poles = np.array([R * np.exp(1j*theta), R * np.exp(-1j*theta)])
30 zeros = np.zeros(poles.shape, poles.dtype)
31
32 b, a = signal.zpk2tf(zeros, poles, 1)
33 # import pdb; pdb.set_trace()
34 y = np.zeros(sig.shape, sig.dtype)
35 time = np.linspace(0, num_samples/float(Fs), num_samples, endpoint=False)
36
37 for i in range(len(sig)):
38     y[i] = y[i] + a[0]*sig[i]
39     for j in range(1, len(a)):
40         if i-j >= 0:
41             y[i] = y[i] - a[j]*y[i-j]
42 fig3 = plt.figure()
43 plt.title('Filter Output')
```

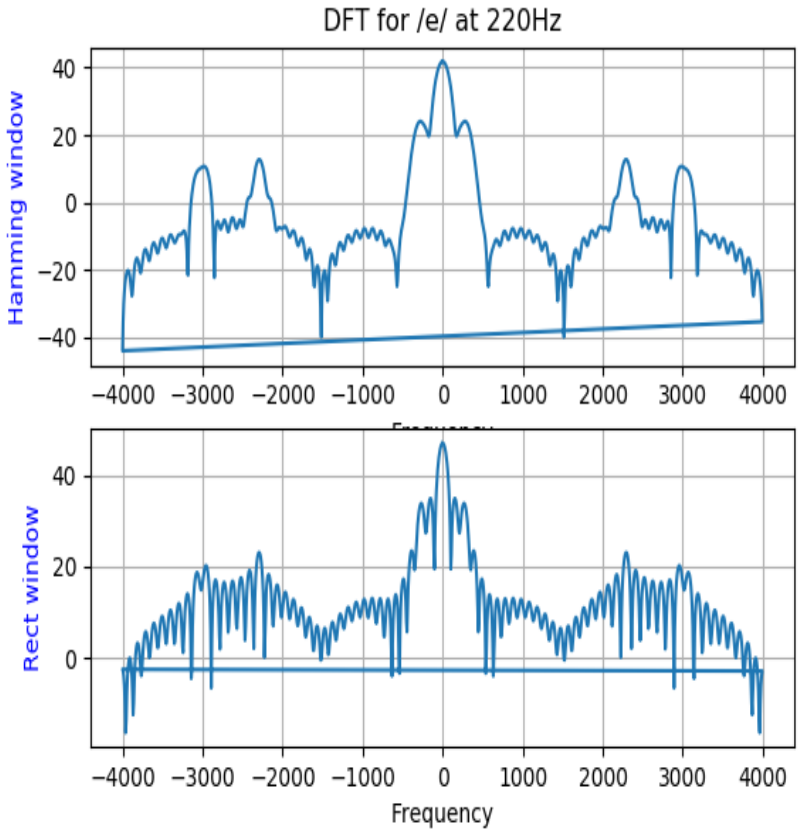

Figure 18: DTFT of /i/ at pitch=220Hz with window length = 10ms

```
# ax3 = fig2.add_subplot(111)
plt.plot(time[0:1000], y[0:1000], 'b')
plt.ylabel('Filter output', color='b')
plt.xlabel('Time [seconds]')
pylab.savefig('./results/' + 'q3' + '_' + 'trial1' + '.png')
write('./results/' + 'q3' + '_' + ' trial1.wav', Fs, y)
plt.show()
```

Listing 4: Python example

6.5 Question4

```

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

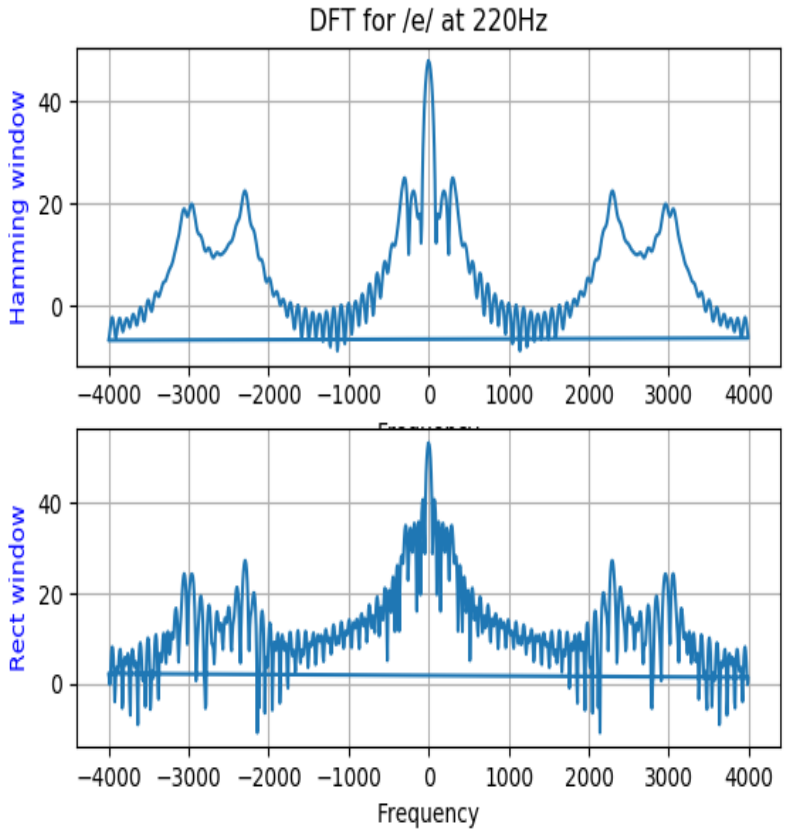

Figure 19: DTFT of /i/ at pitch=220Hz with window length = 20ms

```
11
12
13
14 F1 = np.array(hparams.formant_freq)
15 B1 = np.array(hparams.formant_bw)
16 Fs = hparams.samp_freq
17 T = 1.0/Fs
18 t0 = hparams.time_length
19 num_samples = Fs*t0
20 F0 = hparams.sig_freq
21 t = np.linspace(0, t0, num_samples)
22
23 # sig = signal.sawtooth(2 * np.pi * F0 * t, width=1)
24 sig = signal.square(2 * np.pi * F0 * t, duty=0.01)
25
26 # Calculate pole angles and radii
27 R = np.exp(-pi*B1/Fs)
28 theta = 2*pi*F1/Fs
29
30
31 poles = np.concatenate([R * np.exp(1j*theta), R * np.exp(-1j*theta)])
```

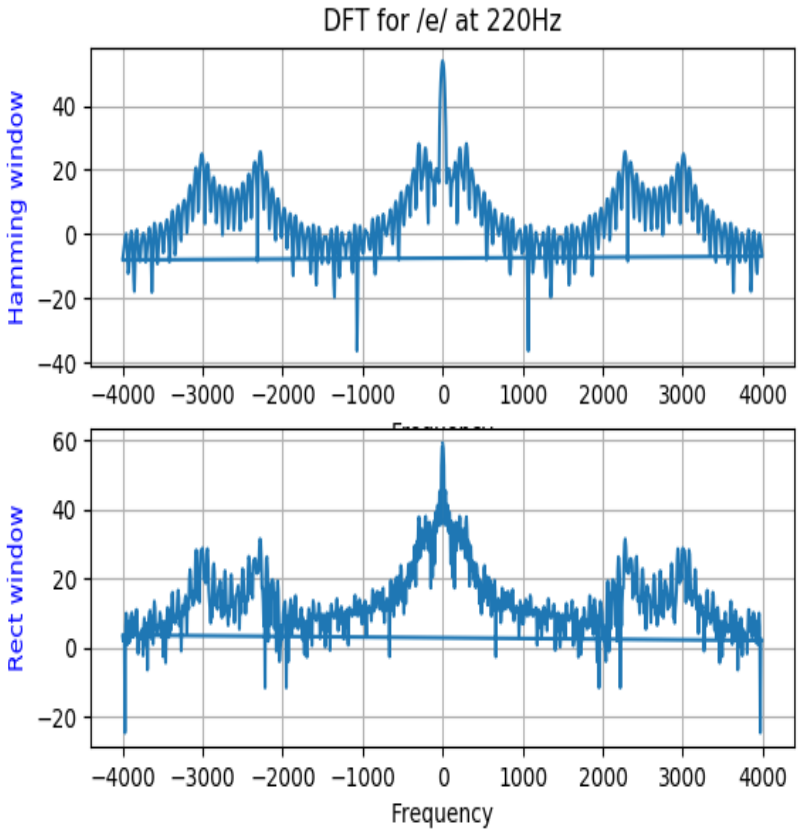

Figure 20: DTFT of /e/ at pitch=220Hz with window length = 40ms

```
zeros = np.zeros(poles.shape, poles.dtype)
b, a = signal.zpk2tf(zeros, poles, 1)

y = np.zeros(sig.shape, sig.dtype)
# time = np.linspace(0, num_samples/Fs, num_samples, endpoint=False)

for i in range(len(sig)):
    y[i] = y[i] + a[0]*sig[i]
    for j in range(1, len(a)):
        if i-j >= 0:
            y[i] = y[i] - a[j]*y[i-j]

plt.title('Filter Output')
plt.plot(t[0:1000], y[0:1000], 'b')
plt.ylabel('Filter output', color='b')
plt.xlabel('Time [seconds]')
pylab.savefig('./results/' + 'q4' + '_' + '120_e' + '.png')
write('./results/' + 'q4' + '_' + ' 120_e.wav', Fs, y)
plt.show()
```

Listing 5: Python example

6.6 Question5

```
1
2  import numpy as np
3  from matplotlib import pyplot as plt
4  import pylab
5  from scipy import signal
6  from scipy.io.wavfile import write
7  import hparams
8  import sys
9  from math import pi
10 from scipy.io.wavfile import write, read
11
12 Fs = hparams.samp_freq
13 win_size = hparams.win_size/1000.0;
14 window_samples = int(win_size*Fs)
15 t0 = hparams.time_length
16 num_samples = Fs*t0
17
18 output = read('./results/q4_ 220_e.wav')
19 output = output[1]
20
21
22 window = output[:window_samples]*np.hamming(window_samples)
23 dft = np.fft.fft(window, n=hparams.dft_len)
24 freq = np.fft.fftfreq(dft.shape[-1], 1/Fs)
25
26
27 f,a = plt.subplots()
28 plt.subplot(211)
29 plt.title('DFT for /e/ at 220Hz')
30 plt.plot(freq, 20*np.log10(np.abs(dft)))
31 plt.ylabel('Hamming window', color='b')
32 plt.xlabel('Frequency')
33 plt.grid()
34 plt.axis('tight')
35
36 def onclick(event):
37     print [event.xdata,event.ydata]
38 f.canvas.mpl_connect('button_press_event', onclick)
39
40 window = output[:window_samples]
41 dft = np.fft.fft(window, n=hparams.dft_len)
42 freq = np.fft.fftfreq(dft.shape[-1], 1/Fs)
43
44 plt.subplot(212)
45 plt.plot(freq, 20*np.log10(np.abs(dft)))
46 plt.ylabel('Rect window', color='b')
47 plt.xlabel('Frequency')
48 plt.grid()
49 plt.axis('tight')
50
51 def onclick(event):
52     print [event.xdata,event.ydata]
```

```
53 f.canvas.mpl_connect('button_press_event', onclick)
54 plt.show()
```

Listing 6: Python example