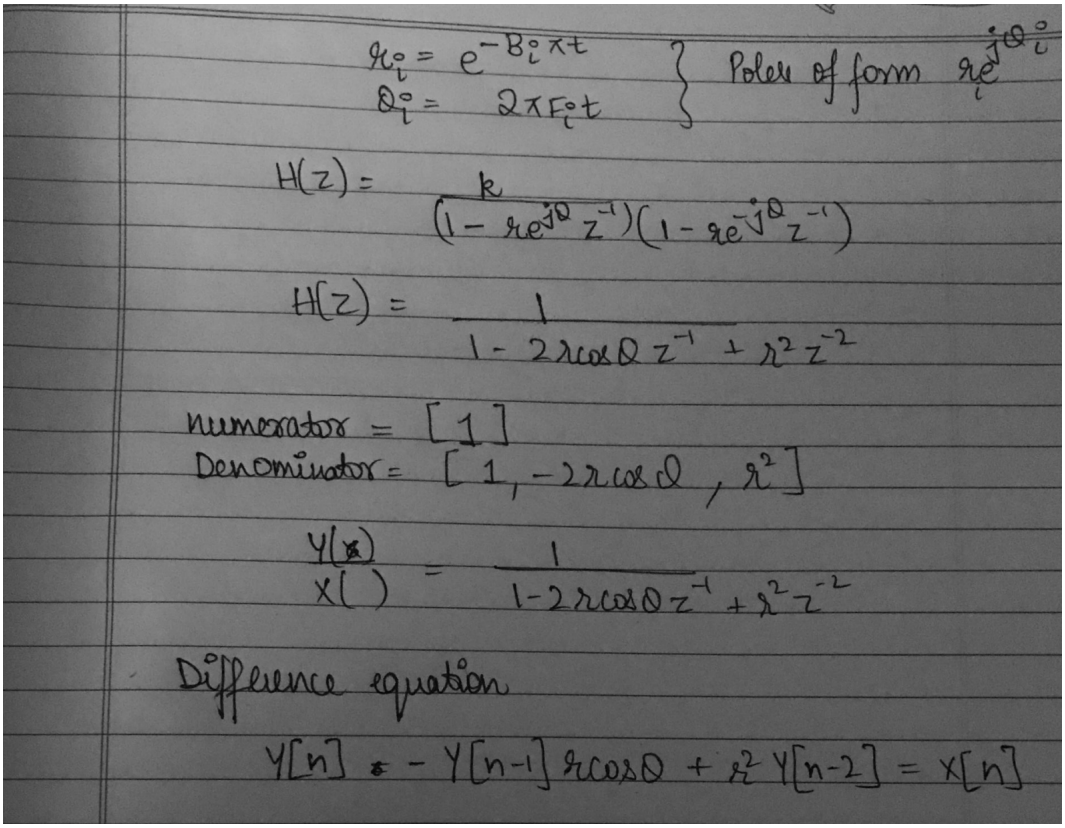
Assignment 1 Speech Processing

Aditya Khanna 17D070036
Faculty mentor : Prof. Preeti Rao

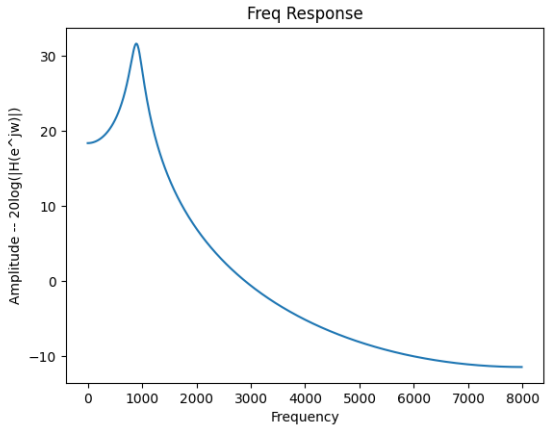
20th September 2020

1 Question 1

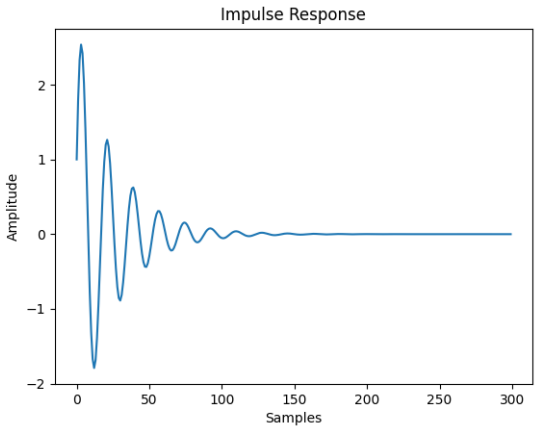
Given the following specification for a single-formant resonator, obtain the transfer function of the filter H(z) from the relation between resonance frequency / bandwidth, and the pole angle /radius. Plot filter magnitude response (dB magnitude versus frequency) and impulse response.
F1 (formant) = 900 Hz
B1(bandwidth) = 200 Hz
Fs (sampling freq) = 16 kHz



The above formula is for the Single formant resonator. Plugging in the values in the equation we observe the following frequency response



To obtain the impulse response, we give an impulse to the function at x=0 and obtain the response using the difference equation. Y[n] depends only on Y[n-1], Y[n-2] and X[n]

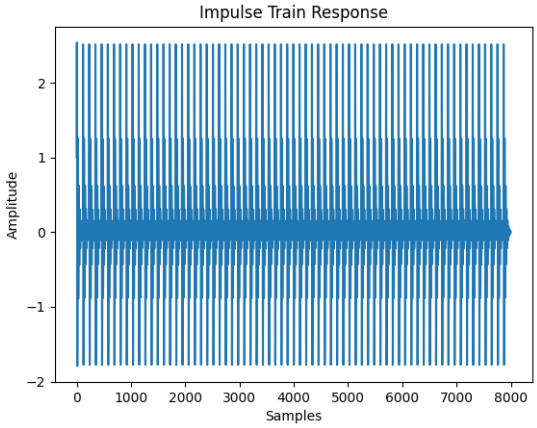


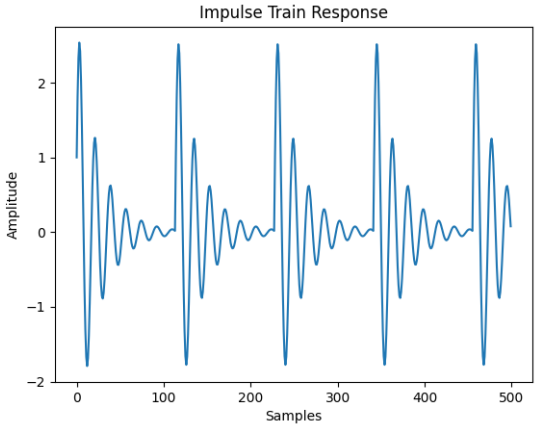

```
import numpy as np
import math
import matplotlib.pyplot as plt
from scipy import signal

# Input Parameters
F_1=900
B_1=200
F_s=16000

r_i = np.exp(-B_1*3.142/F_s)
theta_i = 2*3.142*F_1/F_s
denominator=[1,-2*r_i*math.cos(theta_i),r_i*r_i]
numerator=[1]
w, h = signal.freqz(numerator,denominator)

# plt.title('Freq Response')
# plt.plot(F_s*w/(2*3.142), 20 * np.log10(abs(h)))
# plt.ylabel('Amplitude -- 20log(|H(e^jw)|)')
# plt.xlabel('Frequency')
# plt.savefig('Q1_1.png')
y=[]
time=[]
for i in range(0,300):
  y.append(0)
  time.append(i)
# print(time)
y[0]=1 # impulse
y[1]= -y[0]*denominator[1]
for i in range(2,299):
  y[i] = -y[i-1]*denominator[1]-y[i-2]*denominator[2]; #x[i]=0
plt.title('Impulse Response')
plt.plot(time, y)
plt.ylabel('Amplitude')
plt.xlabel('Samples')
plt.savefig('Q1_2.png')

```

2 Question 2

Excite the above resonator ("filter") with a periodic source excitation of F0 = 140 Hz. 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.

Now instead of using an impulse response, a train of impulses are used as an input. The train of inputs have a frequency of 140 Hz. With a sampling rate of 16K Hz, we have 8000 samples in 0.5 seconds. Out of those 70 samples must be impulses. Thus after every 8000/70 samples there is an impulse.
Using Python we code this in the input variable and pass the same through the Difference Equation to obtain the output response.





Since the frequency is very low its very low pitched and doesnt have such good quality

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

# Input Parameters
F_1=900
B_1=200
playingtime=0.5
F_0=140
F_s=16000

r_i = np.exp(-B_1*3.142/F_s)
theta_i = 2*3.142*F_1/F_s
denominator=[1,-2*r_i*math.cos(theta_i),r_i*r_i]
numerator=[1]
w, h = signal.freqz(numerator,denominator)
x=[]
y=[]
time=[]

total_samples=int(F_s*playingtime)
```

```
    for i in range(0,total_samples):
      x.append(0)
      y.append(0)
      time.append(i)


    i=0
    count=0
    while i<total_samples and count < int(F_0*playingtime):
      x[i]= 1;
      i=i+int(F_s/F_0);
      count+=1

    y[0]=1
    y[1]= x[1]-y[0]*denominator[1]
    for i in range(2,total_samples):
      y[i] = x[i]-y[i-1]*denominator[1]-y[i-2]*denominator[2];
    plt.title('Impulse Train Response')
    # plt.plot(time[0:500], y[0:500])
    plt.plot(time, y)
    plt.ylabel('Amplitude')
    plt.xlabel('Samples')
    # plt.savefig('Q2_1.png')
    plt.savefig('Q2_2.png')

    y = np.array(y)
    write('Q2_1.wav', F_s, y)
```

3 Question 3

Vary the parameters as indicated below and comment on the differences in waveform and sound quality for the different parameter combinations.
(a) F0 = 120 Hz, F1 = 300 Hz, B1 = 100 Hz
(b) F0 = 120 Hz, F1=1200 Hz, B1 = 200 Hz
(c) F0 = 180 Hz, F1 = 300 Hz, B1 = 100 Hz

Using the above code and varying the values of the parameters F0, F1 and B1 we observe the following three graphs and the image sounds

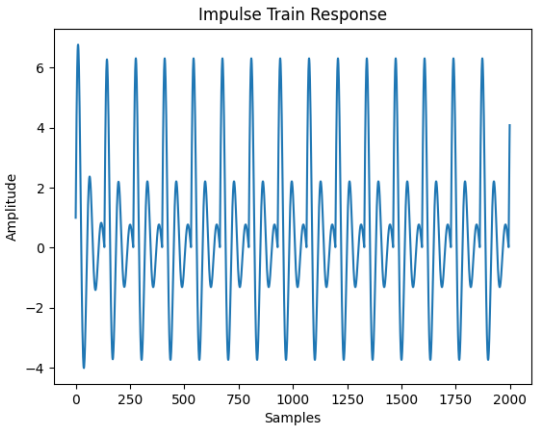


Figure 1: F0 = 120 Hz, F1 = 300 Hz, B1 = 100 Hz

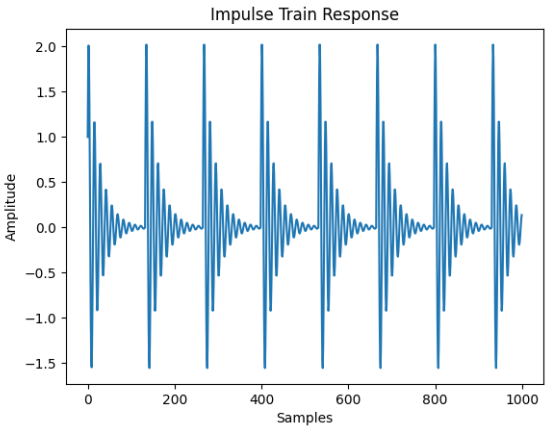


Figure 2: F0 = 120 Hz, F1=1200 Hz, B1 = 200 Hz

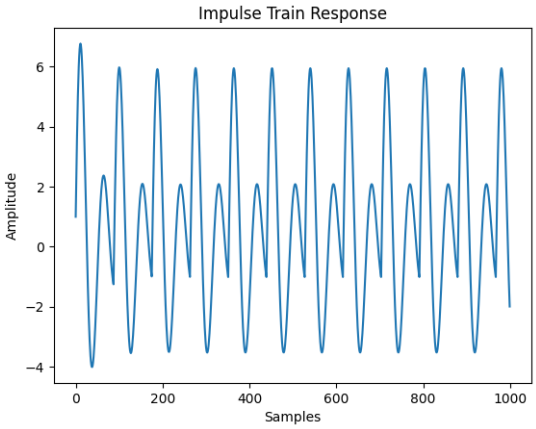


Figure 3: F0 = 180 Hz, F1 = 300 Hz, B1 = 100 Hz

The first and the third sounds have roughly the same waveforms. The 3rd one seems to have a much higher pitch the 1st one and can be easily identified when we play the sounds as well
The first and second have a different bandwidth and thus it takes longer for the 2nd waveform to die than the first one. This is also clearly visible in the number of waves in the 1000 samples. Moreover, 2nd waveform on hearing seems to have a longer decay effect too. Clearly there is a direction correlation of bandwidth and decay time.
The first and second also differ in terms of the formant frequency as well. Which makes the angle of the pole different.The second waveform doesnt experience a sudden decrease. It always decreases gradually.

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

# Input Parameters
F_1=300
B_1=100
playingtime=0.5
F_0=180
F_s=16000

r_i = np.exp(-B_1*3.142/F_s)
theta_i = 2*3.142*F_1/F_s
denominator=[1,-2*r_i*math.cos(theta_i),r_i*r_i]
numerator=[1]
w, h = signal.freqz(numerator,denominator)
x=[]
```

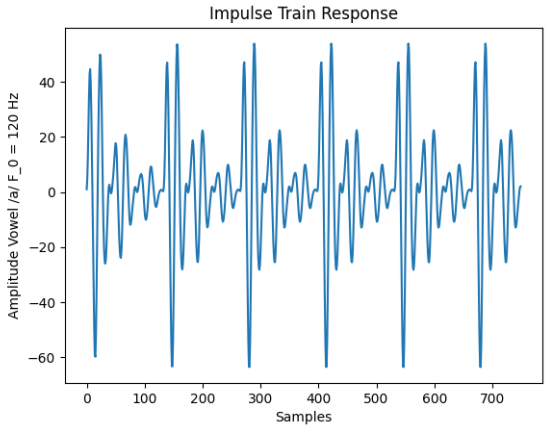
```
    y=[]
    time=[]

    total_samples=int(F_s*playingtime)
    for i in range(0,total_samples):
      x.append(0)
      y.append(0)
      time.append(i)
    i=0
    count=0
    while i<total_samples and count < int(F_0*playingtime):
      x[i]= 1;
      i=i+int(F_s/F_0);
      count+=1

    y[0]=1 # impulse
    y[1]= x[1]-y[0]*denominator[1]
    for i in range(2,total_samples):
      y[i] = x[i]-y[i-1]*denominator[1]-y[i-2]*denominator[2];
    plt.title('Impulse Train Response')
    plt.plot(time[0:1000], y[0:1000])
    plt.ylabel('Amplitude')
    plt.xlabel('Samples')
    # plt.savefig('Q2_1.png')
    plt.savefig('Q3_3.png')

    y = np.array(y)
    write('Q3_3.wav', F_s, y)

```

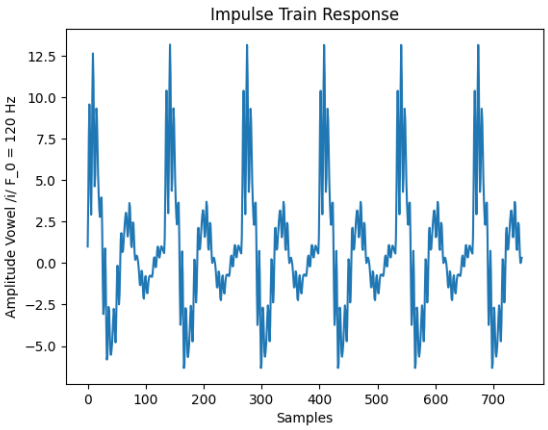
4 Question 4

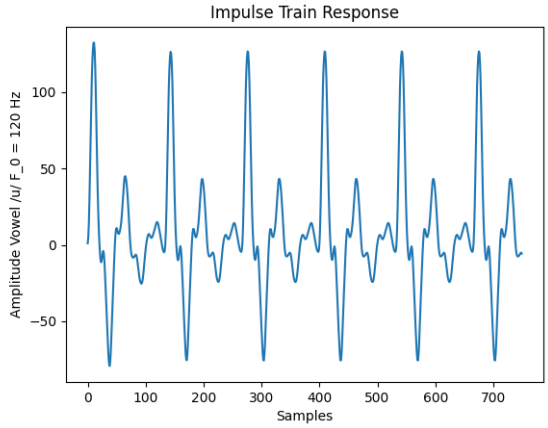
In place of the simple single-resonance signal, synthesize the following more realistic vowel sounds at two distinct pitches (F0 = 120 Hz, F0 = 220 Hz). Keep the bandwidths constant at 100 Hz for all formants. Duration of sound: 0.5 sec
Vowel F1, F2, F3
/a/ 730, 1090, 2440
/i/ 270, 2290, 3010
/u/ 300, 870, 2240

Since we use 3 formants here, We can multiply the transfer functions so as to obtain the final response of the same. To do this, let the first impulse train be bounded by F0 and then passed through the difference equation using F1 to obtain a response Y. This Y is then passed as an input to the difference equation using F2 a response Y1 is obtained. Then the Y1 is passed as an input to the difference equation using F3 and a response Y2 is obtained.This is the final response to the train of impulses.

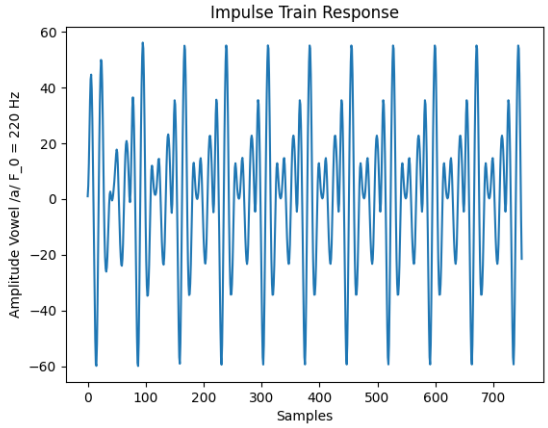
4.1 120 Hz

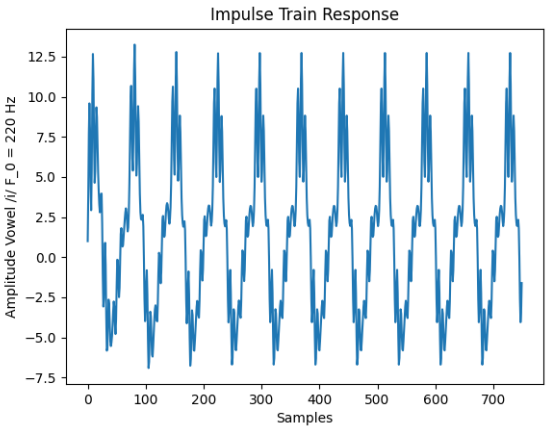


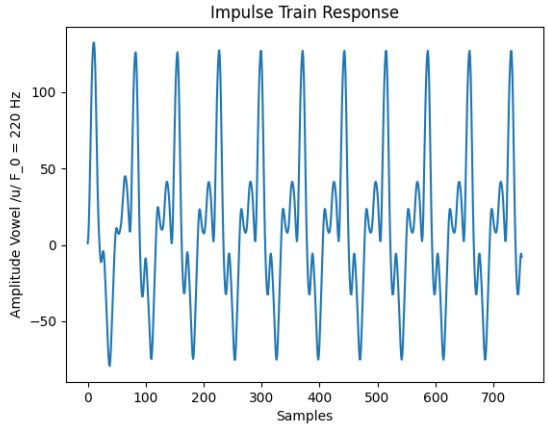

Impulse Train Response
Amplitude Vowel /i/ F_0 = 120 Hz
Samples


Impulse Train Response
Amplitude Vowel /u/ F_0 = 120 Hz
Samples

4.2 220 Hz


Impulse Train Response
Amplitude Vowel /a/ F_0 = 220 Hz
Samples






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


B_1=100
playingtime=0.5
F_0=220
F_s=16000

x=[]
y=[]
y_1=[]
y_2=[]
time=[]
total_samples=int(F_s*playingtime)
for i in range(0,total_samples):
  x.append(0)
  y.append(0)
  y_1.append(0)
  y_2.append(0)
  time.append(i)
i=0
count=0
while i<total_samples and count < int(F_0*playingtime):
  x[i]= 1;
  i=i+int(F_s/F_0);
  count+=1

# Input Parameters
F_1=300
```

```
F_2=870
F_3=2240

def response(x,y,F):
  r_i = np.exp(-B_1*3.142/F_s)
  theta_i = 2*3.142*F/F_s
  denominator=[1,-2*r_i*math.cos(theta_i),r_i*r_i]
  y[0]=x[0] # impulse
  y[1]= x[1]-y[0]*denominator[1]
  for i in range(2,total_samples):
    y[i] = x[i]-y[i-1]*denominator[1]-y[i-2]*denominator[2];
  return y

y=response(x,y,F_1)
y_1=response(y,y_1,F_2)
y_2=response(y_1,y_2,F_3)

plt.title('Impulse Train Response')
plt.plot(time[0:750], y_2[0:750])
plt.ylabel('Amplitude Vowel /u/ F_0 = 220 Hz')
plt.xlabel('Samples')
plt.savefig('Q4_u_220.png')

y_2 = np.array(y_2)
write('Q4_u_220.wav', F_s, y_2)


```

The Audio files of 'A' and 'U' do sound like them, but there is a little roughness in 'I'. But the voices are a little robotic