# EE-679 Assignment 1A

Soham Naha
Roll : 193079003

September 19, 2020

## Q1.

**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**

**Solution**
We know that the Vocal Tract transfer function $H(z)$, for given formant frequency $F_i$ and bandwidth $B_i$ is given by the formula:

$$H(z) = \begin{cases} \frac{1}{(1-re^{-j\theta}z^{-1})(1-re^{j\theta}z^{-1})} \\ \frac{1}{1-2rcos(\theta)z^{-1}+r^2z^{-2}} \end{cases} \tag{1}$$

$$\text{where, r} = e^{-\sigma_p T_s} \text{ and } \theta = \Omega_p * T_s$$

$$\text{and } \sigma_p = \pi B_i \text{ and } \Omega_p = F_i$$

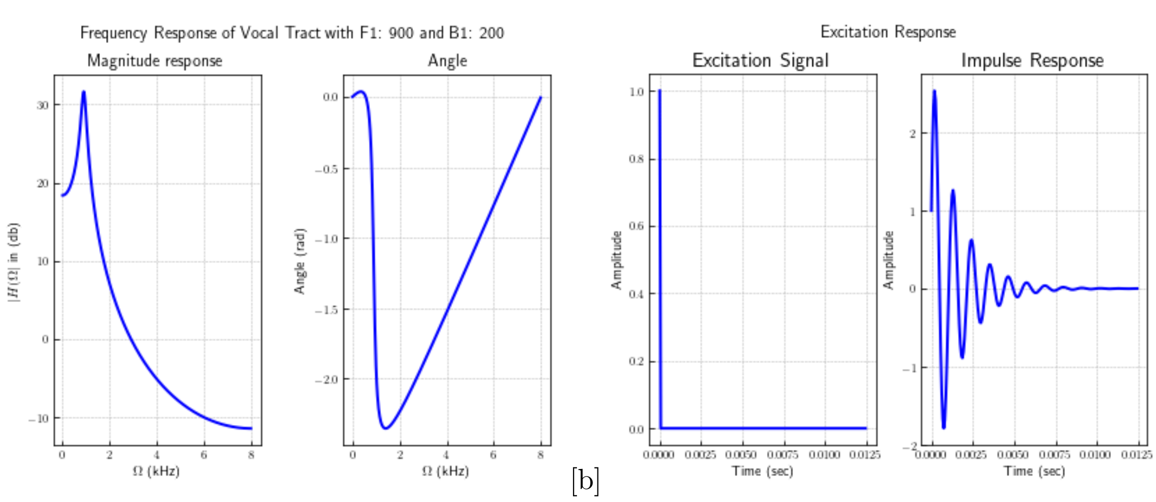
The Impulse Response of the filter is calculated using the difference equation formula:

$$\sum_{k=0}^{M} b[k]y[n-k] = \sum_{k=0}^{N} a[k]x[n-k] \tag{2}$$

where, y[n] is the Impulse Response,
b[k],a[k] are the numerator and denominator coefficients of $H(z)$
and x[n] is the input Impulse signal

```python
# initial package imports
import numpy as np
from scipy.signal import zpk2tf,freqz,sawtooth,square,impulse
from math import pi
from numpy import exp,zeros_like,cos,sin,log10,angle
from numpy import convolve as conv

# given data
f1 = 900 #formant frequency
b1 = 200 #bandwidth
fs = 16000 #sampling frequency
ts = 1.0/fs # sampling time

# calculation of poles and zeros from F1 and B1 and Coeffs
r = np.exp(-pi*b1*ts)
theta = 2*pi*f1*ts
poles = [r*exp(1j*theta) , r*exp(-1j*theta)]
zeros = zeros_like(poles)
b,a = zpk2tf(zeros,poles,k=1)

#### Frequency Response calculation ###
```

```python
w,h = freqz(b,a)
plt.figure()
plt.subplot(1,2,1)
plt.plot(fs * w/(2*pi),20*log10(abs(h)),'b')
s="Frequency Response of Vocal Tract with F1: {} and B1: {}"
plt.suptitle(s.format(f1,b1),fontsize=12)
plt.title(r"Magnitude response",fontsize=12)
plt.ylabel(r"$|H(\Omega|$ in (db)",fontsize=10)
plt.xlabel(r"$\Omega$")
plt.subplot(1,2,2)
angles = np.angle(h)
plt.plot(fs * w/(2*pi),angles,'b')
plt.title(r"Angle",fontsize=12)
plt.ylabel(r"Angle (rad)",fontsize=10)
plt.xlabel(r"$\Omega$",fontsize=10)
plt.subplots_adjust(left=0.125,
                    wspace=0.4)
plt.savefig("Question1.png",bbox_inches="tight",pad=-1,format="png")

#### Impulse Response calculation ###
# forming the impulse input
pulse = np.zeros((200,1))
pulse[0] = 1

# initializing the impulse response
y = zeros_like(pulse)
time = np.linspace(0,len(pulse)*1.0/fs , 200, endpoint=False)

for n in range(len(pulse)):
    y[n] += b[0] * pulse[n]
    for k in range(1,len(a)):
        if (n-k)>=0:
            y[n] -= a[k] * y[n-k]

plt.figure()
plt.suptitle(r"Excitation Response",fontsize=12)
plt.subplot(1,2,1)
plt.plot(time,pulse,'b')
plt.title("Excitation Signal")
plt.ylabel(r"Amplitude",fontsize=10)
plt.xlabel(r"Time (sec)",fontsize=10)
plt.subplot(1,2,2)
plt.plot(time,y,'b')
plt.title("Impulse Response")
plt.ylabel(r"Amplitude",fontsize=10)
plt.xlabel(r"Time (sec)",fontsize=10)
plt.savefig("Question1 Impulse Response.png",bbox_inches="tight",pad=-1,format="png")
```




[a] [b]

Figure 1: (a) Frequency Response (b) Impulse Response

## Q2

**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**

### Solution

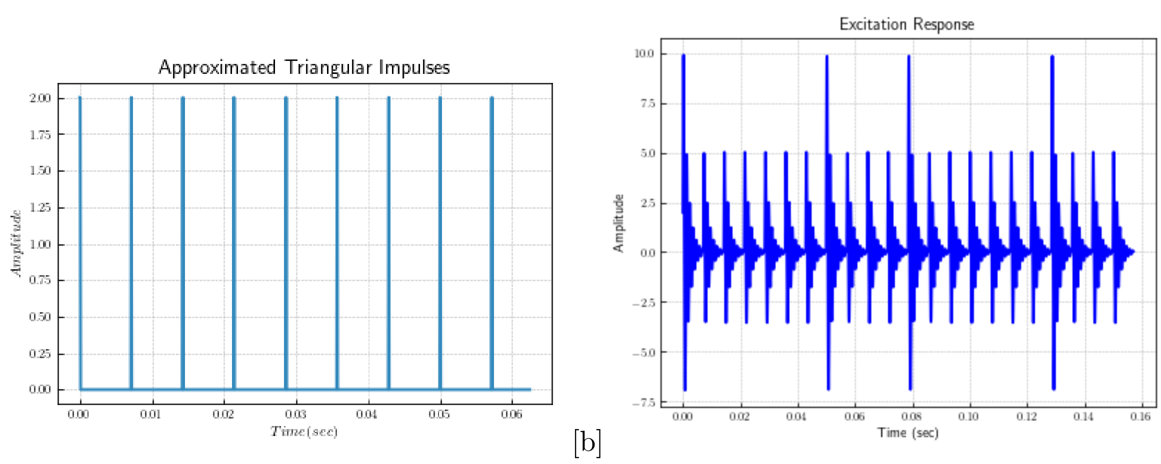
The input waveform was approximated using a square-wave having duty-cycle=0.01.
The excitation signal response is calculated using the same difference equation as used in Q1.:

$$\sum_{k=0}^{M} b[k]y[n-k] = \sum_{k=0}^{N} a[k]x[n-k] \tag{3}$$

where, y[n] is the Impulse Response,
b[k],a[k] are the numerator and denominator coefficients of $H(z)$
and x[n] is the input Impulse signal

```
# initial package imports
import numpy as np
from scipy.signal import zpk2tf,freqz,sawtooth,square,impulse
from math import pi
from numpy import exp,zeros_like,cos,sin,log10,angle
from numpy import convolve as conv
from scipy.io.wavfile import write

# input data
f1 = 900 #formant frequency
b1 = 200 #bandwidth
f_sampling = 16000
f_signal = 140
time = 0.5
t_sampling = 1/f_sampling
num_samples = int(f_sampling*time)

r = np.exp(-pi*b1*ts)
theta = 2*pi*f1*ts
poles = [r*exp(1j*theta) , r*exp(-1j*theta)]
zeros = 0
b,a = zpk2tf(zeros,poles,k=1)

# Excitation signal formation
t = np.linspace(0,time,num_samples)
# sawtooth approximation using square
sig = square(2 * pi * f_signal* t, duty=0.01)+1

plt.figure()
plt.plot(t[:1000],sig[:1000])
plt.xlabel("$Time (sec)$",fontsize=10)
plt.ylabel("$Amplitude$",fontsize=10)
plt.savefig("Question2 Triangular Impulses.png",bbox_inches="tight",pad=-1,format="png")

#Calculation Excitation response
y = zeros_like(sig)
# difference equation
for n in range(len(sig)):
    for k in range(len(b)):
            if (n-k)>=0:
                y[n] += b[k] * sig[n-k]
    for k in range(1,len(b)):
        if (n-k)>=0:
            y[n] += b[k] * sig[n-k]
```

```
        for k in range(1,len(a)):
            if (n-k)>=0:
                y[n] -= a[k] * y[n-k]

    #plotting the excitation response
    plt.figure()
    plt.title("Excitation Response",fontsize=12)
    plt.plot(t[:2514],y[:2514],'b')
    plt.ylabel("Amplitude",fontsize=10)
    plt.xlabel("Time (sec)",fontsize=10)
    plt.savefig("Question2 Response.png",bbox_inches="tight",pad=-1)

    # saving the wav file
    write("Q2output"+"_".join([str(f_signal),str(f1),str(b1)])+".wav",f_sampling,y)
```




[a] [b]

Figure 2: (a) Excitation Signal of F0=140Hz (b) Excitation Response

## Q3

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

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

**Solution**

```
# initial module imports
import numpy as np
from scipy.signal import zpk2tf,freqz,sawtooth,square,impulse
from math import pi
from numpy import exp,zeros_like,cos,sin,log10,angle
from numpy import convolve as conv

def generate_signal_response(time,sig,b,a):
    """
    Given the signal, its duration, the filter numerator and denominator
    coefficients, this function calculates the excitation signal response
    of the filter, saves the plots in the plots directory and returns the
    response.
    inputs: time (time duration of the signal)
            sig (the excitation signal to the filter)
            b,a : filter numerator and denominator coefficients
    outputs: filter response y
    """
    y = zeros_like(sig)
```

```python
    # difference equation
    for n in range(len(sig)):
        for k in range(len(b)):
            if (n-k)>=0:
                y[n] += b[k] * sig[n-k]
        for k in range(1,len(a)):
            if (n-k)>=0:
                y[n] -= a[k] * y[n-k]
    return y

def plot_and_save_waveform(t,y,f_signal,f1,b1,f_sampling):
    """
    Plots and saves the output of the filter excited with the signal upto a few pitch periods.
    inputs: t(time-vector of the excitation signal)
            y( output response of the filter)
            f_signal ( excitation signal frequency )
            f1 (formant frequency of the filter)
            b1 (bandwidth of the filter)
            f_sampling (sampling frequency)
    outputs: None
    """
    plt.figure()
    plt.title(r"Excitation Response",fontsize=12)
    plt.plot(t[:2514],y[:2514],'b')
    plt.ylabel(r"Amplitude",fontsize=10)
    plt.xlabel(r"Time (sec)",fontsize=10)
    plt.savefig("Q3_Signal_Response"+str(f1)+"_"+str(b1)+".png",bbox_inches="tight",pad=-1,format="png")
    write("output"+"_".join([str(f_signal),str(f1),str(b1)])+".wav",f_sampling,y)

def plot_magnitude_response(b,a,f1,b1):
    """
    Plots the magnitude and phase response of the filter using the numerator and denominator
    coefficients of the filter.
    inputs: b,a (filter numerator and denominator coefficients)
            f1,b1 (formant frequency and bandwidth, used to save the figure only)
    outputs: None (saves the magnitude and frequency response)
    """
    # frequency response calculation
    w,h = freqz(b,a)
    plt.figure()
    s = "Frequency response of vocal tract with F1: {}Hz and B1: {}Hz"
    plt.suptitle(s.format(f1,b1),fontsize=12)
    plt.subplot(1,2,1)
    plt.plot(fs * w/(2*pi),20*log10(abs(h)),'b')
    plt.title(r"Magnitude response",fontsize=12)
    plt.ylabel(r"$|H(\Omega|$",fontsize=10)
    plt.xlabel(r"$\Omega$")
    plt.subplot(1,2,2)
    angles = np.angle(h)
    plt.plot(fs * w/(2*pi),angles,'b')
    plt.title(r"Angle",fontsize=12)
    plt.ylabel(r"Angle (rad)",fontsize=10)
    plt.xlabel(r"$\Omega$",fontsize=10)
    plt.subplots_adjust(left=0.125,
                    wspace=0.4, )
    plt.savefig("Q3_Freq_resp_"+str(f1)+"_"+str(b1)+".png",bbox_inches="tight",pad=-1,format="png")

def generate_waveform(f1,b1,f_signal,fs=16000):
    """
    Compiles all the support functions to produce the output
    inputs: f1 (first formant frequency of the filter)
            b1 (bandwidth around the first formant frequency)
            f_signal (excitation signal frequency)
            fs (sampling frequency)
    output: None
```

```
    """
    time = 0.5 # total time duration
    ts = 1/fs # sampling time
    num_samples = int(f_sampling*time) # total number of signal samples
    r = np.exp(-pi*b1*ts) #radius in z-plane
    theta = 2*pi*f1*ts #angle in z-plane

    poles = [r*exp(1j*theta) , r*exp(-1j*theta)] #poles : 2 for every formant
    zeros = zeros_like(poles) # zeros

    b,a = zpk2tf(zeros,poles,k=1)
    plot_magnitude_response(b,a,f1,b1)
    t = np.linspace(0,time,num_samples)

    # sawtooth approximation using square
    sig = square(2 * pi * f_signal* t, duty=0.01)+1

    #
    response = generate_signal_response(t,sig,b,a)
    plot_and_save_waveform(t,response,f_signal,f1,b1,fs)

formant_frequencies = [300, 1200, 300]
bandwidths= [100, 200, 100]
signal_frequencies = [120,120,180]

for i,j,k in list(zip(formant_frequencies,bandwidths,signal_frequencies)):
    generate_waveform(i,j,k)
```

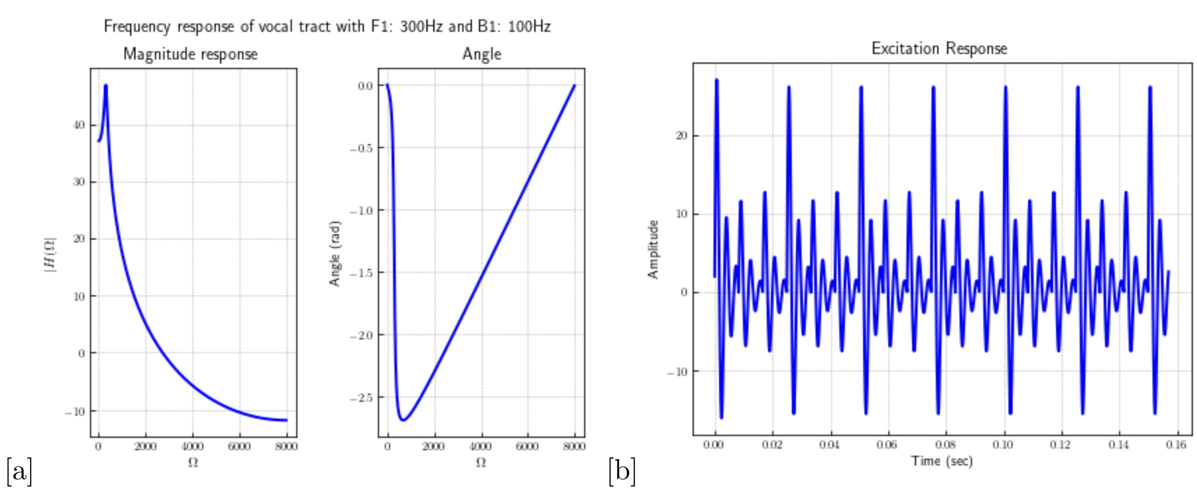



Figure 3: (a) Frequency response at F0= 120 Hz, F1= 300 Hz and B1= 100Hz (b) Time response at F0= 120 Hz, F1= 300 Hz and B1= 100Hz

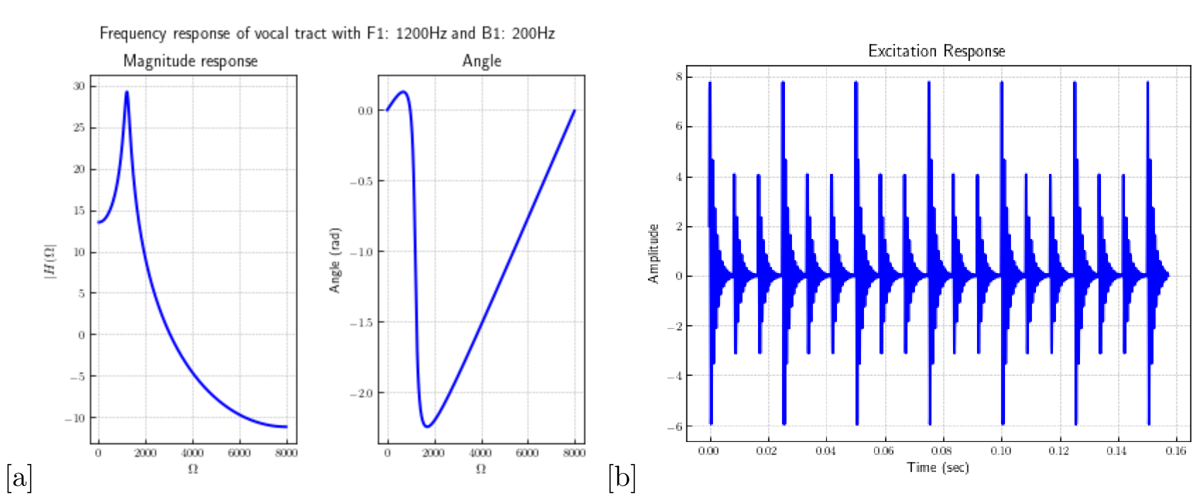
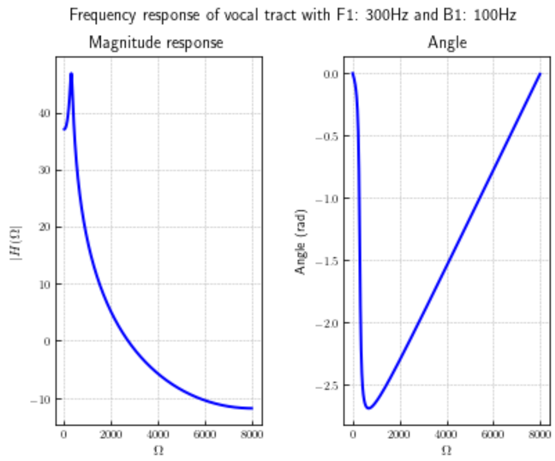



Figure 4: (a) Frequency response at F0= 120 Hz, F1= 1200 Hz and B1= 200Hz (b) Time response at F0= 120 Hz, F1= 1200 Hz and B1= 200Hz

[a] 

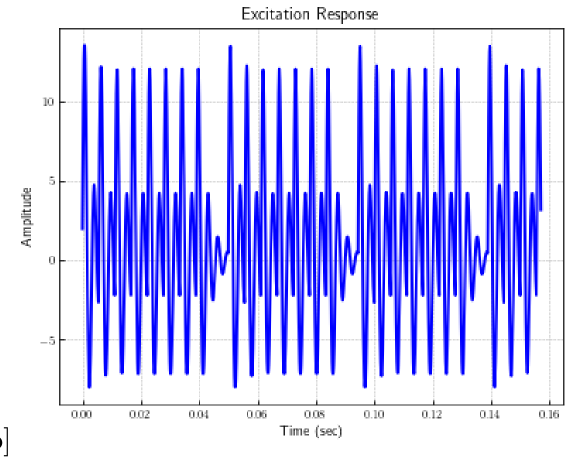

[b] 


Figure 5: (a) Frequency response at F0= 180 Hz, F1= 300 Hz and B1= 100Hz (b) Time response at F0= 180 Hz, F1= 300 Hz and B1= 100Hz

**Observations**

- From the case where the excitation signal frequency ($F_0$) changes, from 120Hz to 180Hz, although the formant frequencies are still the same (and hence the filter , i.e. the vocal tract characteristics), the pitch is higher in the case of $F_0$ = 180 Hz,as the source excitation controls the perceived pitch, i.e. the glottal vibration.
- In case where the formant frequency changes, the vocal tract filter characteristics change. So, although the excitation signal frequency (glottal vibration) is the same, the output waveform seems to be a bit different, with the pitch of the $2^{nd}$ waveform (with $F_1$ = 1200Hz and $B_1$ = 200Hz) dominating over the $1^{st}$ one (with $F_1$ = 300Hz and $B_1$ = 100Hz).The $1^{st}$ waveform shows a slightly smooth time response, whereas, the $2^{nd}$ waveform has sharp declines to the pulse locally. In the wavfile, this shows up as abruptness in sounds vs a smooth sound.
- The wavfile with $F_1$ = 300Hz and $B_1$= 100Hz, with $F_0$= 120hz sounds a lot like the vowel /u/.

# Q4

**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 band-widths 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**

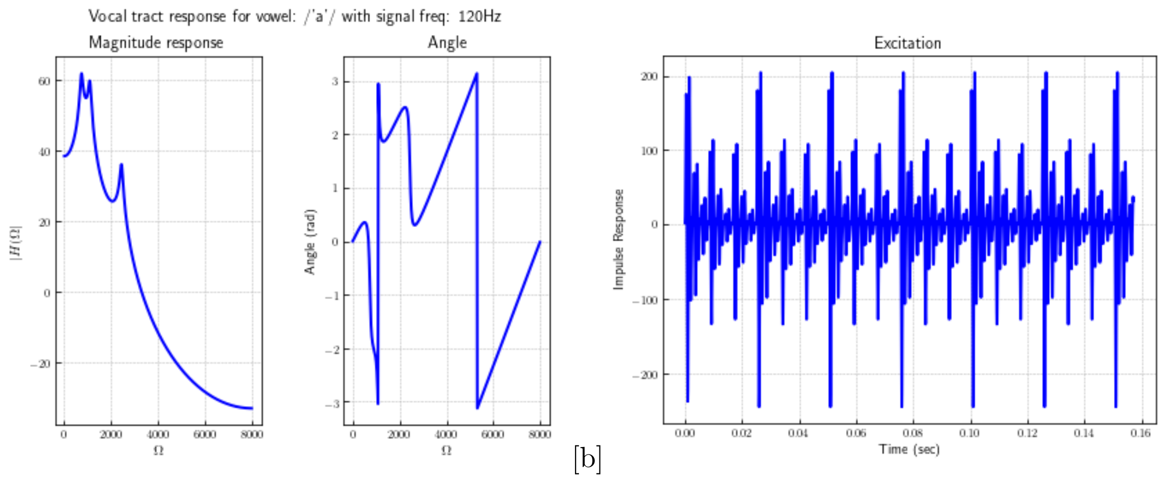
**(Optional: Use glottal pulse shaping and lip radiation filtering. Add a small amount of aspiration noise and pitch jitter.)**

**Solution**

```python
# initial package imports
import numpy as np
from scipy.signal import zpk2tf,freqz,sawtooth,square,impulse
from math import pi
from numpy import exp,zeros_like,cos,sin,log10,angle
from numpy import convolve as conv

def generate_signal_response(t,sig,b,a):
    y = zeros_like(sig)
    # difference equation
    for n in range(len(sig)):
        for k in range(len(b)):
```

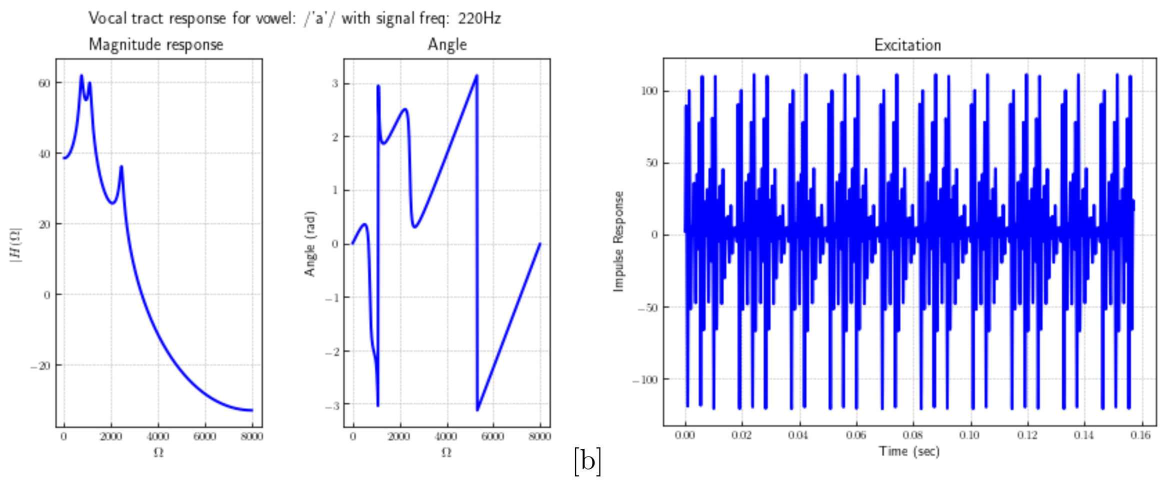
```
            if (n-k)>=0:
                y[n] += b[k] * sig[n-k]
        for k in range(1,len(a)):
            if (n-k)>=0:
                y[n] -= a[k] * y[n-k]
    return y

def plot_magnitude_response(b,a,vowel,f0):
    """
    Plots the magnitude and phase response of the filter using the numerator and denominator
    coefficients of the filter.
    inputs: b,a (filter numerator and denominator coefficients)
            vowel (the vowel parameters being used)
            f0 (excitation signal frequency)
    outputs: None (saves the magnitude and frequency response)
    """
    w,h = freqz(b,a)
    plt.figure()
    s = "Vocal tract response for vowel: /'{}'/ with signal freq: {}Hz"
    plt.suptitle(s.format(vowel,f0) ,fontsize=12,weight=2)
    plt.subplot(1,2,1)
    plt.plot(fs * w/(2*pi),20*log10(abs(h)),'b')
    plt.title("Magnitude response",fontsize=12)
    plt.ylabel(r"$|H(\Omega|$",fontsize=10)
    plt.xlabel(r"$\Omega$")
    plt.subplot(1,2,2)
    angles = np.angle(h)
    plt.plot(fs * w/(2*pi),angles,'b')
    plt.title(r"Angle",fontsize=12)
    plt.ylabel(r"Angle (rad)",fontsize=10)
    plt.xlabel(r"$\Omega$",fontsize=10)
    plt.subplots_adjust(left=0.125,
                wspace=0.4)
    plt.savefig("plots/Q4_Freq_resp_"+vowel+"_"+str(f0)+".png",bbox_inches="tight",pad=-1,format="png")

def plot_and_save_waveform(t,y,f_signal,f_sampling,vowel):
   """
    Plots and saves the output of the filter excited with the signal upto a few pitch periods.
    inputs: t(time-vector of the excitation signal)
            y( output response of the filter)
            f_signal ( excitation signal frequency )
            f_sampling (sampling frequency)
            vowel (the vowel being coded)
    outputs: None
    """
    plt.figure()
    plt.title("Excitation",fontsize=12)
    plt.plot(t[:2514],y[:2514],'b')
    plt.ylabel("Impulse Response",fontsize=10)
    plt.xlabel("Time (sec)",fontsize=10)
    plt.savefig("Q4_Signal_Response"+str(f_signal)+"_"+vowel+".png",bbox_inches="tight",pad=-1,format="png")
    write("output"+"_"+str(f_signal)+"_"+vowel+".wav",f_sampling,y)

def vocal_tract(formant_frequencies):
    """
    Given the formant frequencies calculates the numerator and denominator coefficients
    by convolving between the different formant frequencies
    inputs: formant_frequencies (list of the formant frequencies)
    outputs: numerator and denominator coefficients
    """
    global bw
    r = []
    theta = []
    for i in formant_frequencies:
        r.append(np.exp(-pi*bw*ts)) #radius in z-plane
```

```
        theta.append(2*pi*i*ts) #angle in z-plane

    denom_coeffs = []
    num_coeffs = []
    convolved_a = 1
    for radius,angle in zip(r,theta):
        poles = [radius*exp(1j*angle),radius*exp(-1j*angle)]
        zeros = zeros_like(poles)
        b,a = zpk2tf(zeros,poles,k=1)
        num_coeffs.append(b)
        denom_coeffs.append(a)
        convolved_a = conv(convolved_a,a)

    denom_coeffs = zeros_like(convolved_a)
    denom_coeffs[0] = 1

    return denom_coeffs,convolved_a

def generate_vowels(formant_frequencies,bandwidth,signal_frequency,vowel,time,f_sampling):
    ts = 1/f_sampling # sampling time
    num_samples = int(f_sampling*time) # total number of signal samples

    b,a = vocal_tract(formant_frequencies)
    plot_magnitude_response(b,a,vowel,signal_frequency)

    t = np.linspace(0,time,num_samples)

    # sawtooth approximation using square
    sig = square(2 * pi * signal_frequency* t, duty=0.01)+1

    response = generate_signal_response(t,sig,b,a)
    plot_and_save_waveform(t,response,signal_frequency,f_sampling,vowel)

f0 = [120,220]
f1 = [730,270,300]
f2 = [1090,2290,870]
f3 = [2440,3010,2240]
bw = 100
vow = ["a","i","u"]
duration = 0.5
fs = 16000 #sampling frequency
vowels = {}
for i in range(len(vow)):
    vowels[vow[i]] = {"formants":[f1[i],f2[i],f3[i]]}

for sig_freq in f0:
    for vowel in vowels:
        generate_vowels(vowels[vowel]["formants"],bw,sig_freq,vowel,duration,fs)
```



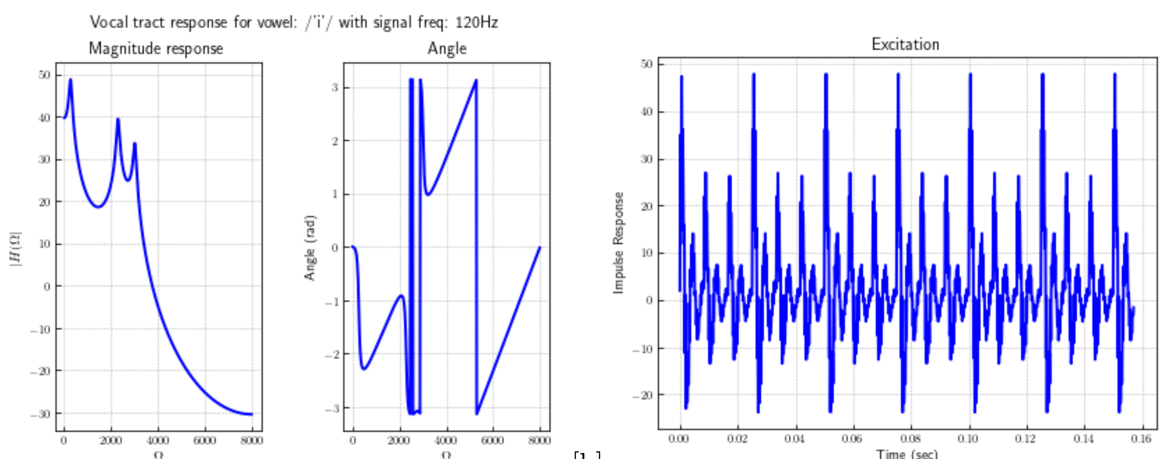

[a] [b]

Figure 6: (a) Frequency response at F0: 120 Hz and vowel: /a/ (b) Time response at F0: 120 Hz and vowel: /a/



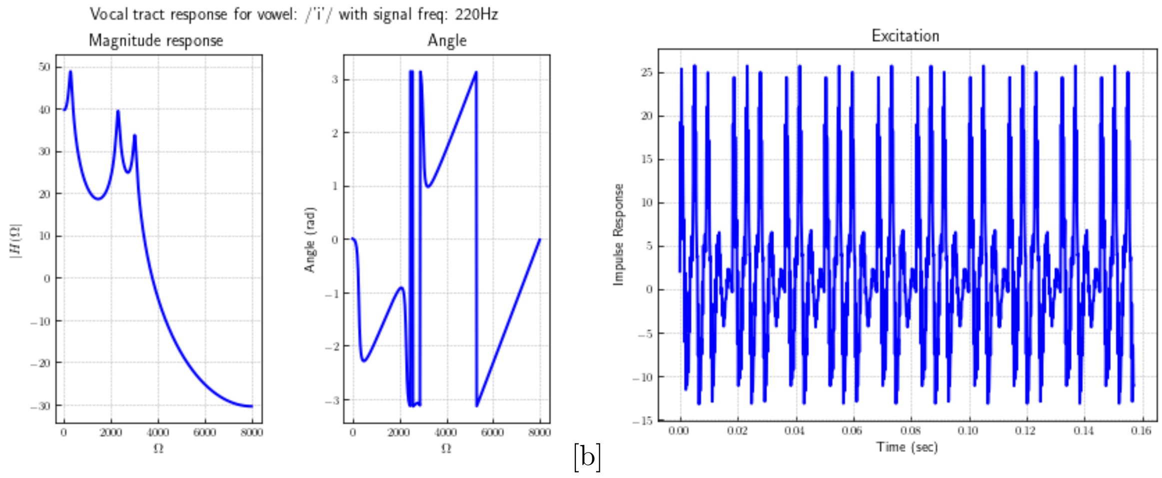

[a] [b]

Figure 7: (a) Frequency response at F0: 220 Hz and vowel: /a/ (b) Time response at F0: 220 Hz and vowel: /a/



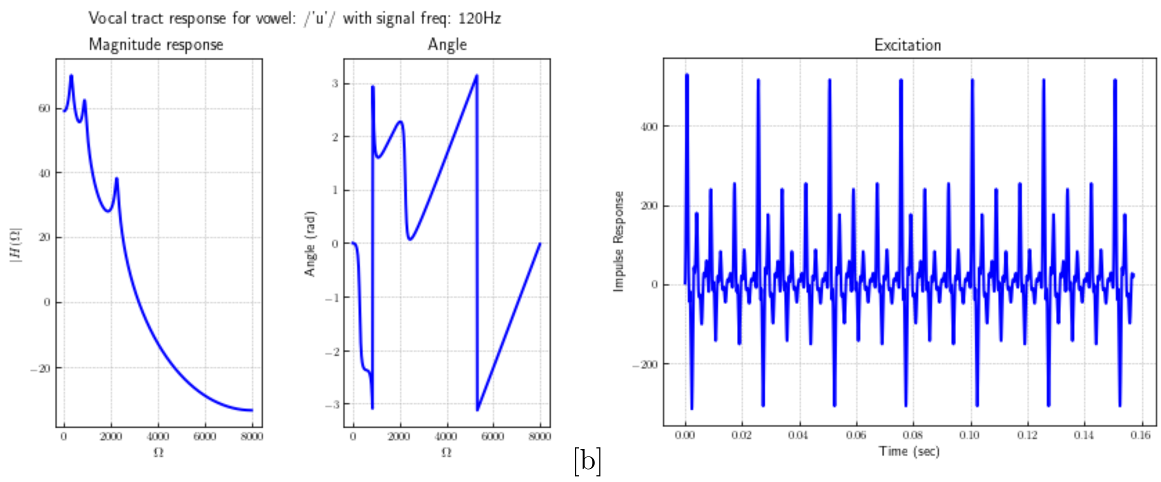

[a] [b]

Figure 8: (a) Frequency response at F0: 120 Hz and vowel: /i/ (b) Time response at F0: 120 Hz and vowel: /i/



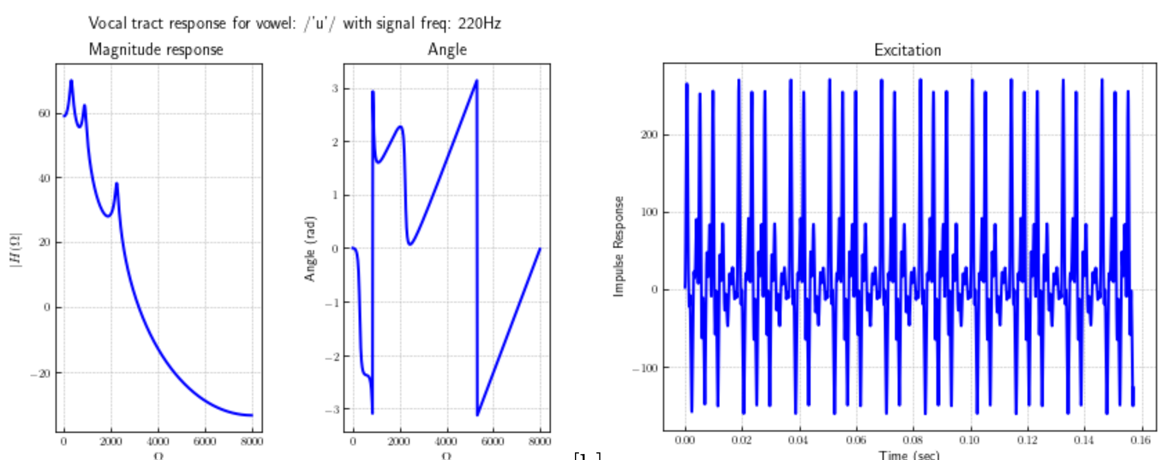

[a] [b]

Figure 9: (a) Frequency response at F0: 220 Hz and vowel: /i/ (b) Time response at F0: 220 Hz and vowel: /i/




[a] [b]

Figure 10: (a) Frequency response at F0: 120 Hz and vowel: /u/ (b) Time response at F0: 120 Hz and vowel: /u/




[a] [b]

Figure 11: (a) Frequency response at F0: 220 Hz and vowel: /u/ (b) Time response at F0: 220 Hz and vowel: /u/

**Observations** The vowels are distinguishable, but the