EE 779: Computing Assignment 1

Kalpesh Patil (130040019)

# Qusetion 1: Impulse response of single formant resonator

Poles of the transfer function can be obtained as follows

This would give us locations of poles in complex plane. The transfer function can be written as

## Code

clear all

*//Function to calculate filter response using difference equation*

function [**y**]=time\_response(**x**, **num**, **den**, **n\_samples**)

**y** = zeros(**n\_samples**,1)

*//numerator is constant (all pole filter)*

**y**(1) = **num**(1)\***x**(1)

*//response by taking coefficients for denominator*

for ii =2:**n\_samples**

temp = k\***x**(ii)

for jj = 1:min(ii-1,length(**den**)-1)

temp = temp - **den**(jj+1)\***y**(ii - jj)

end

**y**(ii) = temp

end

endfunction

*//specifying parameters*

F1 = 1000

B1 = 200

Fs = 16000

*//numerator of filter assumed to be 1*

k = 1

*//number of samples*

n\_samples = 200

*//number of points for fft*

n\_fft = n\_samples\*10

*//computing poles*

r = exp(-B1\*%pi/Fs);

theta = 2\*%pi\*F1/Fs

*//specifying coefficients of discrete time filter*

num = k

den = [1 -2\*r\*cos(theta) r^2]

*//impulse input*

delta\_n = zeros(n\_samples,1)

delta\_n(1) = 1

*//impulse response*

h = time\_response(delta\_n,num,den,n\_samples)

h\_padded = zeros(n\_fft,1)

h\_padded(1:length(h)) = h

n\_time\_samples = 100

time\_array = [0:n\_time\_samples-1]\*Fs/1000

fig = scf()

plot(time\_array, h(1:n\_time\_samples))

xtitle('impulse response','time (ms)','h(n)')

xs2jpg(gcf(), '../plots/Q1/impulse response.jpg');

*//Frequency response of the filter*

fig = scf()

H = fftshift(fft(h\_padded))

H\_mag = abs(H(n\_fft/2 +1:n\_fft))

freq\_array = linspace(0,Fs/2,n\_fft/2)

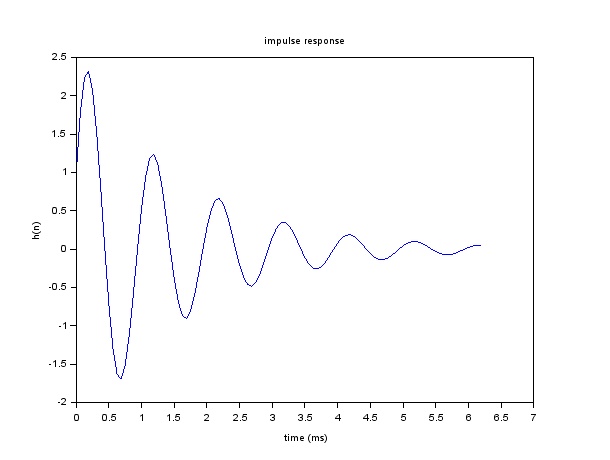
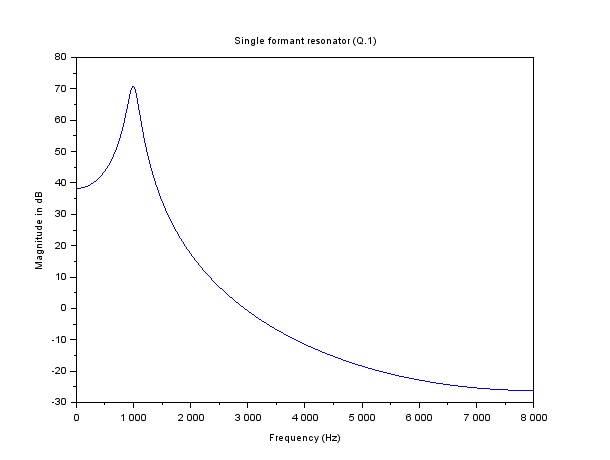
plot(freq\_array,20\*log(H\_mag))

xtitle('Single formant resonator (Q.1)','Frequency (Hz)','Magnitude in dB')

xs2jpg(gcf(), '../plots/Q1/single\_formant\_resonator\_magnitude\_plot.jpg');

## Results & Discussion

Plots for impulse response in time domain and frequency response of the filter are attached below. We can observe that peak in frequency response is around 1000Hz which was formant frequency. Also in the plot of impulse response in time domain, peaks are occurring after the interval of 1ms, which signifies 1000Hz as formant frequency.



# Question 2: Exciting the filter in question 1 with periodic source excitation and compute output

As mentioned in the problem statement, source signal was approximated with narrow triangular pulse train. Following difference equation was implemented, which is obtained using filter described in question 1.

System was assumed to be causal and hence .

## Code

clear all

*//Function to calculate filter response using difference equation*

function [**y**]=time\_response(**x**, **num**, **den**, **n\_samples**)

**y** = zeros(**n\_samples**,1)

*//numerator is constant (all pole filter)*

**y**(1) = **num**(1)\***x**(1)

*//response by taking coefficients for denominator*

for ii =2:**n\_samples**

temp = k\***x**(ii)

for jj = 1:min(ii-1,length(**den**)-1)

temp = temp - **den**(jj+1)\***y**(ii - jj)

end

**y**(ii) = temp

end

endfunction

*//specifying parameters*

F1 = 1000

B1 = 200

Fs = 16000

F0 = 150

t\_duration = 0.5

*//numerator of filter assumed to be 1*

k = 1

*//specifying input signal*

n\_samples = t\_duration\*Fs + 1

n\_fft = n\_samples\*10

x = zeros(n\_samples,1)

*//impulse is approximated by narrow triangular pulse as described in*

*//problem statement*

for i = 2:t\_duration\*F0

x(floor((i-1)\*Fs/F0))= 1

x(floor((i-1)\*Fs/F0 + 1))= 0.75

x(floor((i-1)\*Fs/F0 + 2))= 0.5

x(floor((i-1)\*Fs/F0 + 3))= 0.25

x(floor((i-1)\*Fs/F0 - 1))= 0.75

x(floor((i-1)\*Fs/F0 - 2))= 0.5

x(floor((i-1)\*Fs/F0 - 3))= 0.25

end

*//computing poles*

r = exp(-B1\*%pi/Fs);

theta = 2\*%pi\*F1/Fs

*//specifying coefficients of discrete time filter*

num = k

den = [1 -2\*r\*cos(theta) r^2]

*//output from filter*

y = time\_response(x,num,den,n\_samples)

*//plotting time domain for response*

t\_obs = 25 *//we will observe signal for 25ms*

n\_time\_samples = floor(t\_obs\*Fs/1000)

time\_array = linspace(0,t\_obs,n\_time\_samples)

temp = find(y~=0)

init = temp(1) *//capture n\_time\_samples here onwards*

fig = scf()

plot(time\_array, y(init:init+n\_time\_samples-1))

xtitle('Time domain output','time (ms)','Fiter output')

xs2jpg(gcf(), '../plots/Q2/time\_domain\_output.jpg');

*//converting to y to sound by limiting amplitude in [-1,1]*

*//as required by the wavewrite*

y\_snd = y'/max(y')

playsnd(y\_snd,Fs);

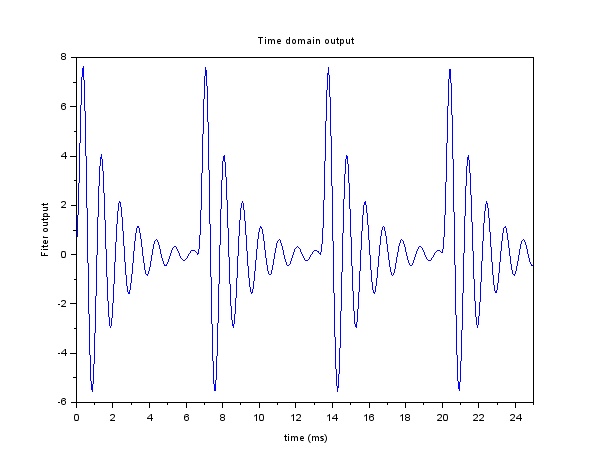
wavfile = '../wav\_files/Q2/output\_signal\_F0\_150\_F1\_1000.wav';

wavwrite(y\_snd, Fs, wavfile);

## Results & Discussion

Time domain waveform over few periods is plotted below. It is observed that pattern of waveform is repeating after 6.6 ms, which corresponds to 151.5 Hz. The pitch of the input given was 150 Hz. Hence we were able to observe this. Also the difference between two consecutive decaying peaks was observed to be 1ms which corresponds to 1000 Hz formant frequency used in the input.

Quality of the sound clip played for 0.5 sec was not good. It was a bit rough and artificial (not signifying any human voice component). Roughness can be attributed to abrupt changes in waveform.



# Question 3: Vary the parameters and observe differences in waveform and sound quality

Following code implements task of varying parameters and plotting and storing waveforms.

## Code

clear all

*//Function to calculate filter response using difference equation*

function [**y**]=time\_response(**x**, **num**, **den**, **n\_samples**)

**y** = zeros(**n\_samples**,1)

*//numerator is constant (all pole filter)*

**y**(1) = **num**(1)\***x**(1)

*//response by taking coefficients for denominator*

for ii =2:**n\_samples**

temp = k\***x**(ii)

for jj = 1:min(ii-1,length(**den**)-1)

temp = temp - **den**(jj+1)\***y**(ii - jj)

end

**y**(ii) = temp

end

endfunction

F0\_list = [120 120 180]

F1\_list = [300 1200 300]

B1\_list = [100 200 100]

for iter = 1:length(F0\_list)

*//specifying parameters*

F1 = F1\_list(iter)

B1 = B1\_list(iter)

Fs = 16000

F0 = F0\_list(iter)

t\_duration = 0.5

*//numerator of filter assumed to be 1*

k = 1

*//specifying input signal*

n\_samples = t\_duration\*Fs + 1

n\_fft = n\_samples\*10

x = zeros(n\_samples,1)

*//impulse is approximated by narrow triangular pulse*

for i = 2:t\_duration\*F0

x(floor((i-1)\*Fs/F0))= 1

x(floor((i-1)\*Fs/F0 + 1))= 0.75

x(floor((i-1)\*Fs/F0 + 2))= 0.5

x(floor((i-1)\*Fs/F0 + 3))= 0.25

x(floor((i-1)\*Fs/F0 - 1))= 0.75

x(floor((i-1)\*Fs/F0 - 2))= 0.5

x(floor((i-1)\*Fs/F0 - 3))= 0.25

end

*//computing poles*

r = exp(-B1\*%pi/Fs);

theta = 2\*%pi\*F1/Fs

*//specifying coefficients of discrete time filter*

num = k

den = [1 -2\*r\*cos(theta) r^2]

*//output from filter*

y = time\_response(x,num,den,n\_samples)

*//plotting time domain for response*

t\_obs = 25 *//we will observe signal for 25ms*

n\_time\_samples = floor(t\_obs\*Fs/1000)

time\_array = linspace(0,t\_obs,n\_time\_samples)

temp = find(y~=0)

init = temp(1) *//capture n\_time\_samples here onwards*

fig = scf()

plot(time\_array, y(init:init+n\_time\_samples-1))

plot\_title = strcat(['Output signal for F0 = ',string(F0),'Hz,F1 = ',string(F1),'Hz,B1 = ',string(B1),'Hz'])

xtitle(plot\_title,'time (ms)','y(n)')

xs2jpg(gcf(), strcat(['../plots/Q3/',plot\_title,'.jpg']));

*//converting to y to sound by limiting amplitude in [-1,1]*

*//as required by the wavewrite*

y\_snd = y'/max(y')

playsnd(y\_snd,Fs);

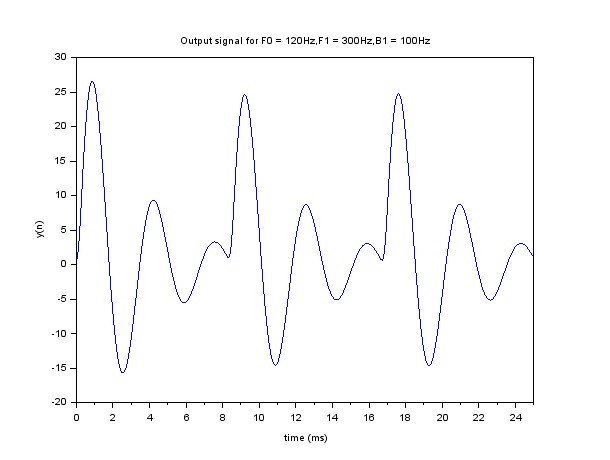
wavfile = strcat(['../wav\_files/Q3/',plot\_title,'.wav']);

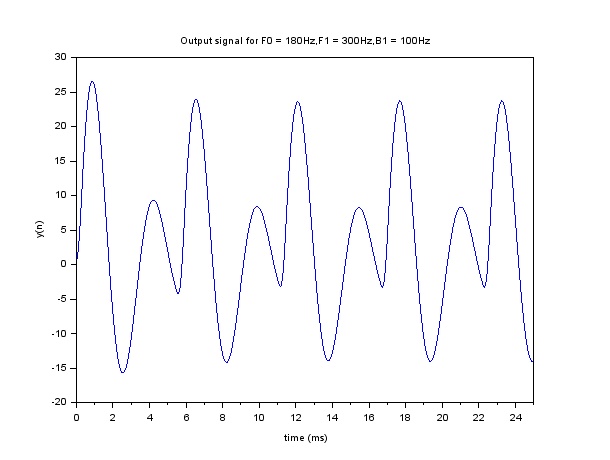
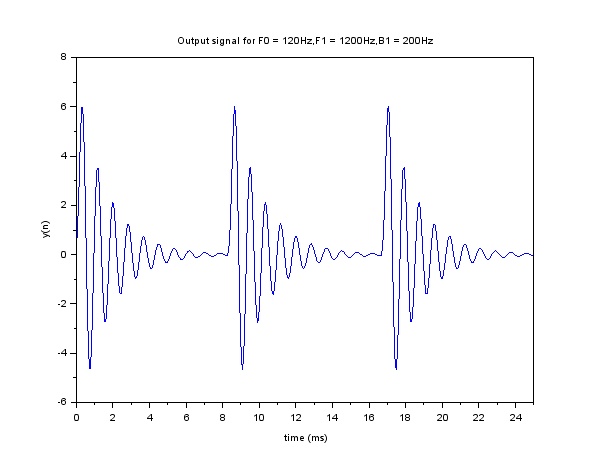
wavwrite(y\_snd, Fs, wavfile);

end

## Results

Plots for the given combinations are as follows. All the plots are for a sample of output captured for 25ms of duration.





## Discussion

1. F0 is same for waveform 1 and waveform 2. But due to changing other parameters we can observe that there is hardly any interference adjacent impulse responses in waveform 2, while lot of inference is present in waveform 1. Waveform 2 decays much faster compared to waveform 1.
2. Pitch of the output depends upon F0. For a fixed time frame (25 ms), first two waveforms had less number of impulse responses (major peaks), while 3rd waveforms had 5 responses.
3. Sound of waveform 2 seems rough compared to other two sounds, because it has huge abrupt changes compared to others. Sound of waveform 3 appears to be the most pleasant amongst all.

# Question 4: Generate given vowels

Cascade of multiple single formant resonators were used to generate a given vowel. Transformation function of cascade system is product of individual transformation functions.

## Code

clear all

*//Function to calculate filter response using difference equation*

function [**y**]=time\_response(**x**, **num**, **den**, **n\_samples**)

**y** = zeros(**n\_samples**,1)

*//numerator is constant (all pole filter)*

**y**(1) = **num**(1)\***x**(1)

*//response by taking coefficients for denominator*

for ii =2:**n\_samples**

temp = k\***x**(ii)

for jj = 1:min(ii-1,length(**den**)-1)

temp = temp - **den**(jj+1)\***y**(ii - jj)

end

**y**(ii) = temp

end

endfunction

*//function to find discrete coefficients of cascade of multiple filters*

function [**num**, **den**]=find\_cascade\_filter(**F\_list**, **B\_list**, **Fs**)

**num** = 1

**den** = [1]

for iter = 1:length(**F\_list**)

F = **F\_list**(iter)

B = **B\_list**(iter)

*//finding poles for current iteration*

r = exp(-B\*%pi/**Fs**)

theta = 2\*%pi\*F/**Fs**

*//current filter*

num\_curr = 1

den\_curr = [1 -2\*r\*cos(theta) r^2]

*//multiplication of polynomials can be computed using their*

*//convolution. Numerator is constant(1). Hence only*

*//denominator needs to be multiplied*

**den** = conv(**den**,den\_curr)

end

endfunction

*//plots frequency response given impulse response*

function plot\_frequency\_response(**h**, **n\_fft**, **vowel**)

h\_padded = zeros(**n\_fft**,1)

h\_padded(1:length(**h**)) = **h**

*//Frequency response of the filter*

fig = scf()

H = fftshift(fft(h\_padded))

H\_mag = abs(H(**n\_fft**/2 +1:**n\_fft**))

freq\_array = linspace(0,Fs/2,**n\_fft**/2)

plot(freq\_array,20\*log(H\_mag))

plot\_title = strcat(['Frequency response of vowel ',**vowel**])

xtitle(plot\_title,'Frequency (Hz)','Magnitude in dB')

xs2jpg(gcf(), strcat(['../plots/Q4/',plot\_title,'.jpg']));

endfunction

*//Plots output signal, plays and store the sound*

function plot\_y(**y**, **F0**, **Fs**, **vowel**)

*//plotting time domain for response*

t\_obs = 25 *//we will observe signal for 25ms*

n\_time\_samples = floor(t\_obs\***Fs**/1000)

time\_array = linspace(0,t\_obs,n\_time\_samples)

temp = find(**y**~=0)

init = temp(1) *//capture n\_time\_samples here onwards*

fig = scf()

plot(time\_array, **y**(init:init+n\_time\_samples-1))

plot\_title = strcat(['Output signal of vowel ',**vowel**,' for F0 = ',string(**F0**)])

xtitle(plot\_title,'time (ms)','y(n)')

xs2jpg(gcf(), strcat(['../plots/Q4/',plot\_title,'.jpg']));

*//converting to y to sound by limiting amplitude in [-1,1]*

*//as required by the wavewrite*

y\_snd = **y**'/max(**y**')

playsnd(y\_snd,**Fs**);

wavfile = strcat(['../wav\_files/Q4/',plot\_title,'.wav'])

wavwrite(y\_snd, **Fs**, wavfile);

endfunction

F0\_list = [120,220]

vowel\_frequency\_matrix = [730, 1090, 2440; *//F1,F2,F3 for vowell a*

270, 2290, 3010; *//F1,F2,F3 for vowell i*

300, 870, 2240] *//F1,F2,F3 for vowell u*

vowel\_list = ['a','i','u']

Fs = 16000

for i\_f0 = 1:length(F0\_list)

F0 = F0\_list(i\_f0)

t\_duration = 0.5

*//numerator of filter assumed to be 1*

k = 1

*//specifying input signal*

n\_samples = t\_duration\*Fs + 1

n\_fft = n\_samples\*10;

x = zeros(n\_samples,1);

*//impulse is approximated by narrow triangular pulse*

for i = 2:t\_duration\*F0

x(floor((i-1)\*Fs/F0))= 1

x(floor((i-1)\*Fs/F0 + 1))= 0.75

x(floor((i-1)\*Fs/F0 + 2))= 0.5

x(floor((i-1)\*Fs/F0 + 3))= 0.25

x(floor((i-1)\*Fs/F0 - 1))= 0.75

x(floor((i-1)\*Fs/F0 - 2))= 0.5

x(floor((i-1)\*Fs/F0 - 3))= 0.25

end

*//Create delta input for finding impulse response*

delta\_n = zeros(n\_samples,1)

delta\_n(1) = 1

for j = 1:length(vowel\_frequency\_matrix(1,:))

F\_list = vowel\_frequency\_matrix(j,:)

*//Bandwidth is constant for all formants*

B\_list = [100,100,100]

*//find output of cascade filter*

[num,den] = find\_cascade\_filter(F\_list,B\_list,Fs)

h = time\_response(delta\_n,num,den)

y = time\_response(x,num,den)

*//plot responses and save sound files*

plot\_frequency\_response(h,n\_fft,vowel\_list(j))

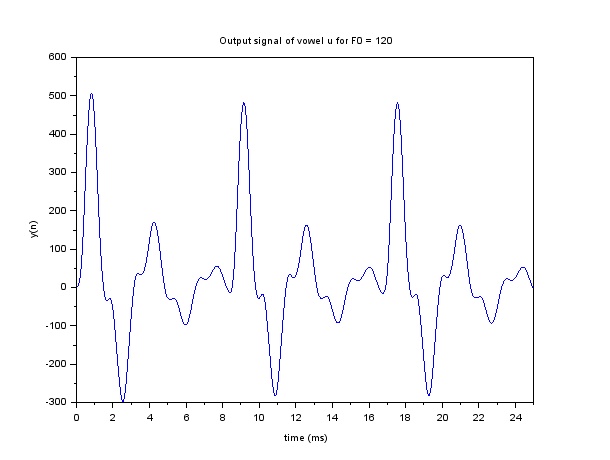
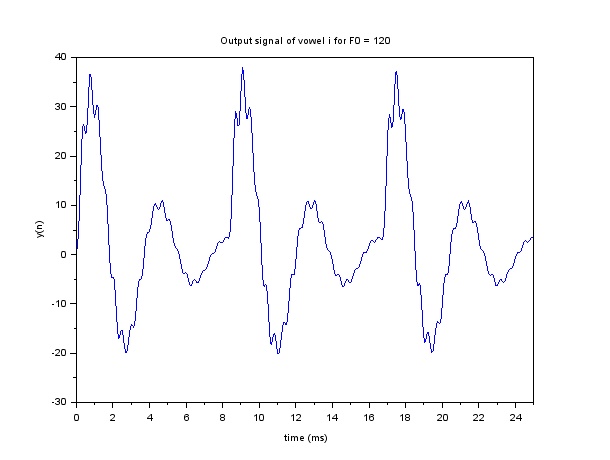
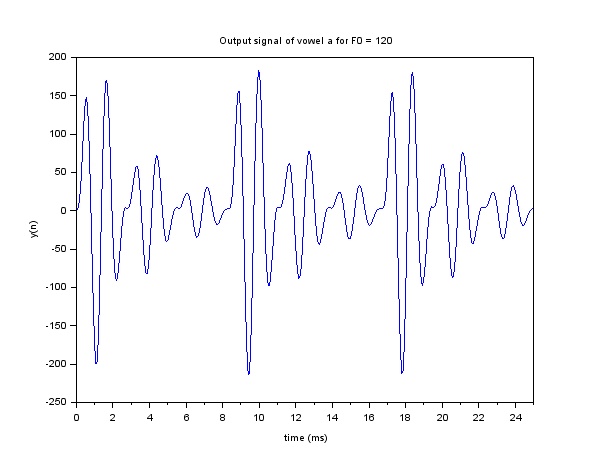
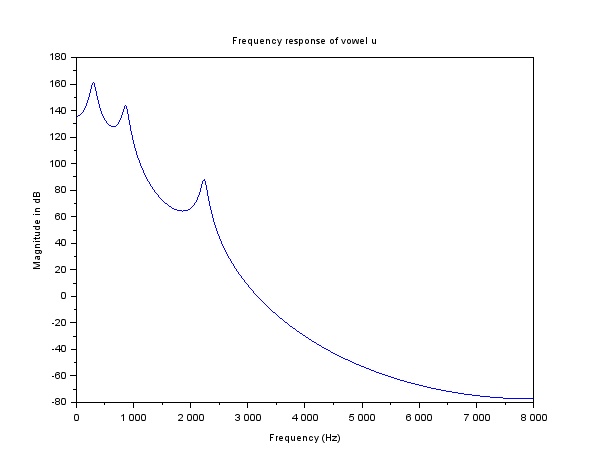
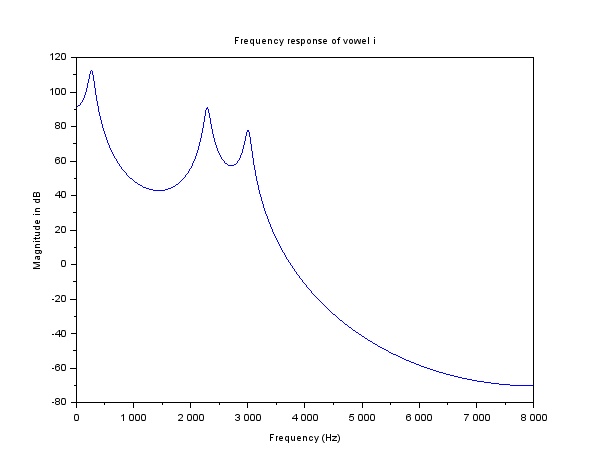
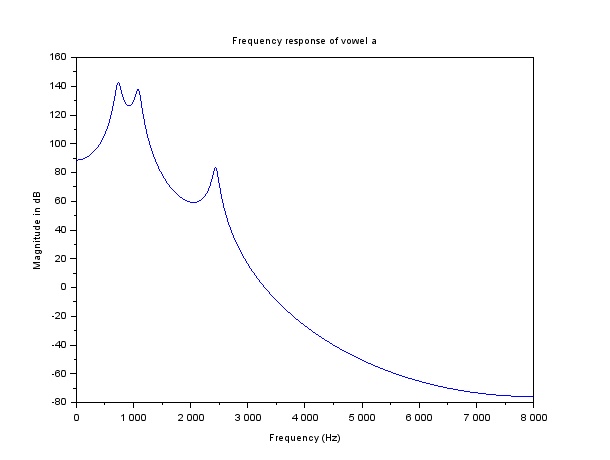
plot\_y(y,F0,Fs,vowel\_list(j))

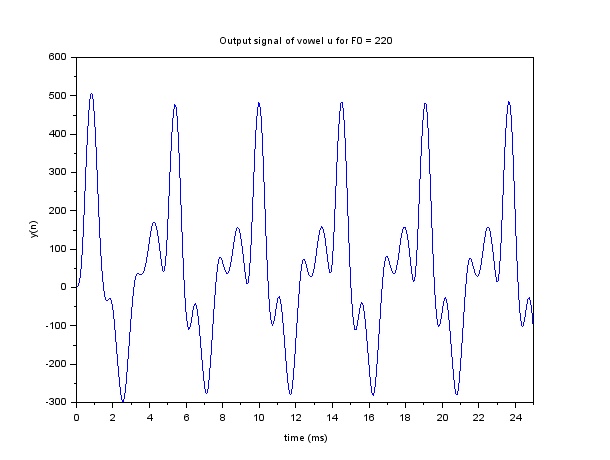
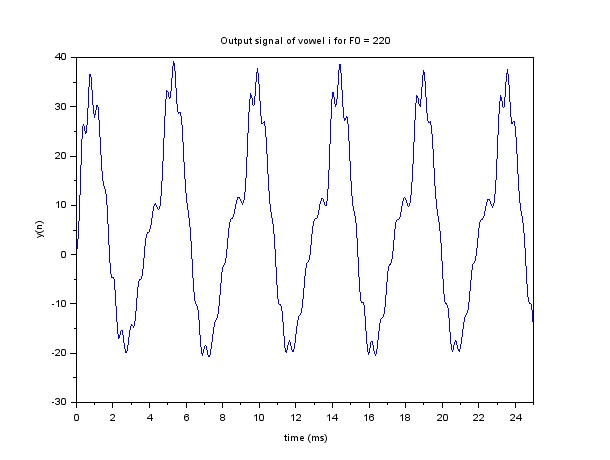
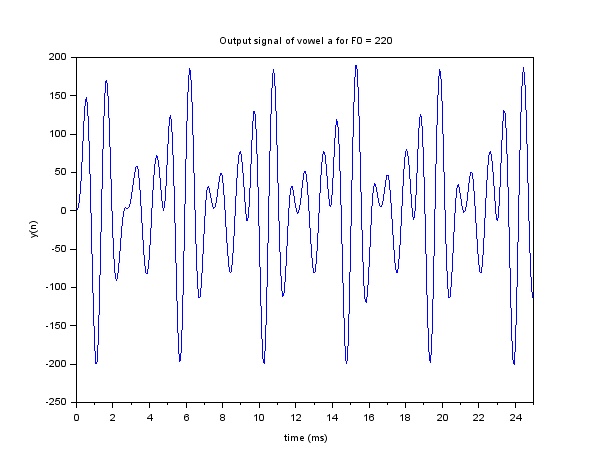
end

end

## Results

The plots for time domain output at different pitches and for different vowels are shown below. Also the frequency response of the cascade filter used is plotted.





## Discussion

The sound for all three vowels is coming properly. The sounds can be better observed if listen to them one after the other. As F0 changes, number of repetitive patterns in the given time frame (25 ms) will change. This is observed. Also the peak frequencies observed in frequency response are matching with the formant frequencies of the cascaded filters.

# Question 5: Signal analysis using different windows

The wavfiles of different vowels generated were loaded. Windowed DFT was calculated for different windows and time durations

## Code

clear all

*//function to read previously generated wavefiles*

function **y**=read\_vowel(**vowel**, **F0**)

wavefile = strcat(['Output signal of vowel ',**vowel**,' for F0 = ',string(**F0**),'.wav'])

**y** = loadwave(strcat(['../wav\_files/Q4/',wavefile]));

endfunction

function [**X\_mag**]=find\_windowed\_FFT(**x**, **window\_type**, **N**, **n\_fft**)

*//we can put this window anywhere on the signal. We will put it*

*//at the centre*

x\_init = (length(**x**)+1)/2 - (**N**-1)/2

select **window\_type**

case 'hamming' then

win\_hamming = window('hm', **N**);

windowed\_x = **x**(x\_init: x\_init + **N** - 1).\*win\_hamming;

case 'rect' then

windowed\_x = **x**(x\_init: x\_init + **N** - 1)

end

*//zero padding*

pad = zeros(1,ceil((**n\_fft** - **N**)/2));

padded\_x = [pad windowed\_x pad]

*//finding fft*

freq\_x = fftshift(fft(padded\_x))

**X\_mag** = abs(freq\_x(**n\_fft**/2 +1:**n\_fft**))

endfunction

window\_times = [5, 10, 20, 40] *//time in ms*

Fs = 16000

*//finding window lengths*

window\_lengths = window\_times\*Fs/1000 + 1

n\_fft = max(window\_lengths)\*10

vowel\_list = ['a','i','u','a','i','u']

F0\_list = [120 120 120 220 220 220]

for p = 1:length(F0\_list)

vowel = vowel\_list(p)

F0 = F0\_list(p)

for window\_type = ['hamming','rect']

for i = 1:length(window\_times)

N = window\_lengths(i)

x = read\_vowel(vowel,F0)

[X\_mag] = find\_windowed\_FFT (x,window\_type,N,n\_fft)

freq\_array = linspace(0,Fs/2,n\_fft/2)

fig = scf()

plot(freq\_array,20\*log(X\_mag))

plot\_title = strcat(['FFT for vowel ',vowel,' using ',window\_type,' window of ',string(window\_times(i)),' ms',' for F0 = ',string(F0)])

xtitle(plot\_title,'Frequency (Hz)','Magnitude in dB')

xs2jpg(gcf(), strcat(['../plots/Q5/',plot\_title,'.jpg']));

end

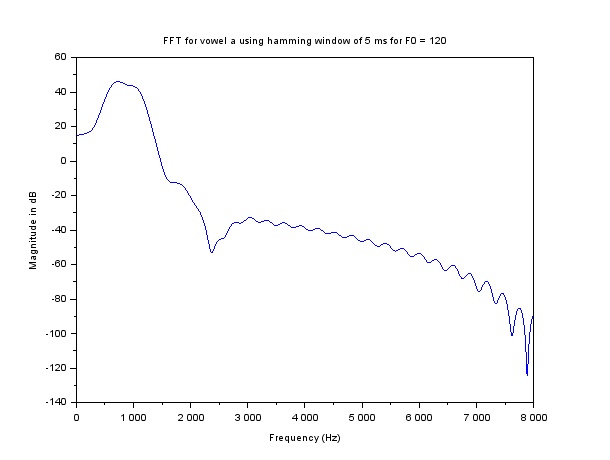
end

end

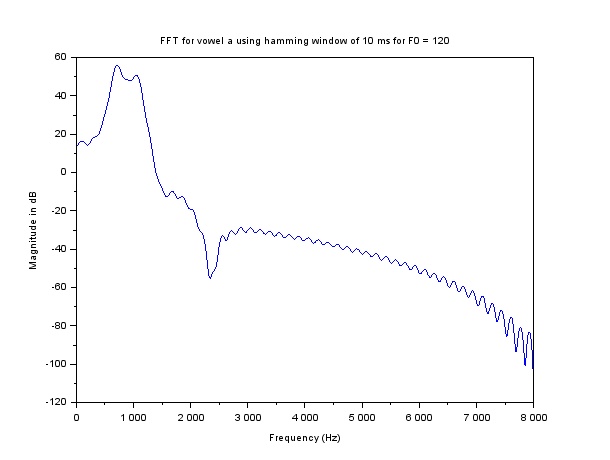
## Results

**Vowel /a/ using hamming window at F0 = 120 Hz**

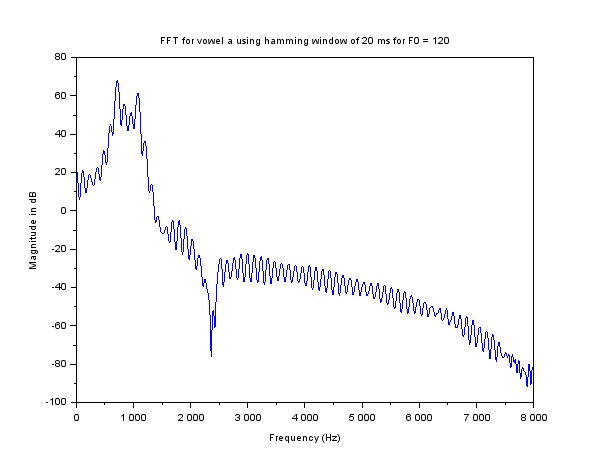
t = 5ms



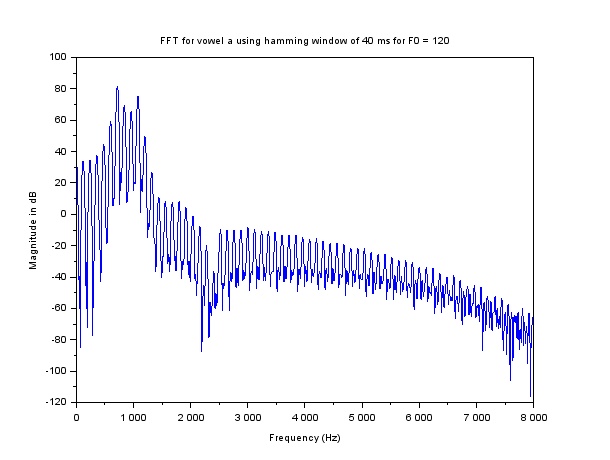
t = 10 ms



t = 20 ms

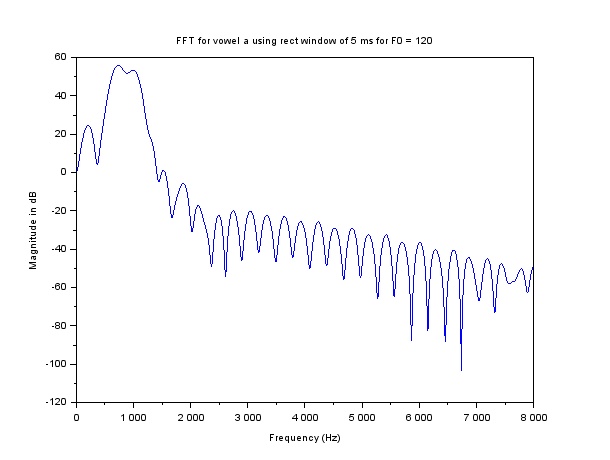


t = 40 ms

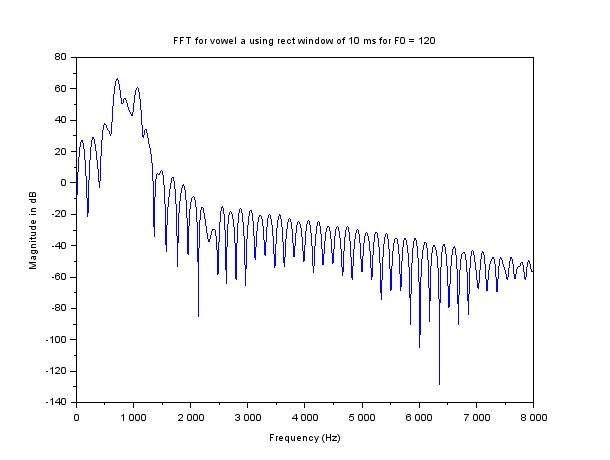


**Vowel /a/ using rectangular window at F0 = 120 Hz**

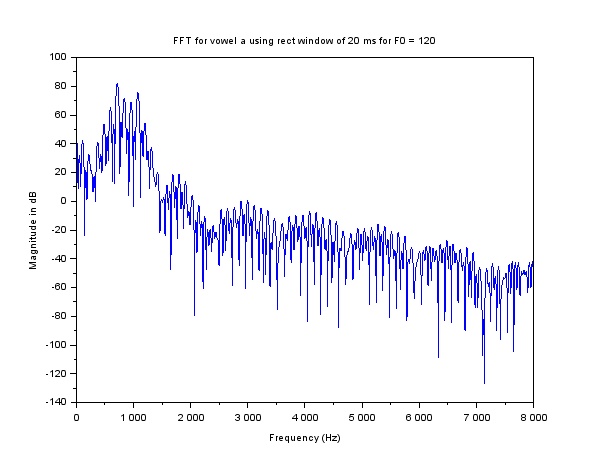
t = 5 ms

****

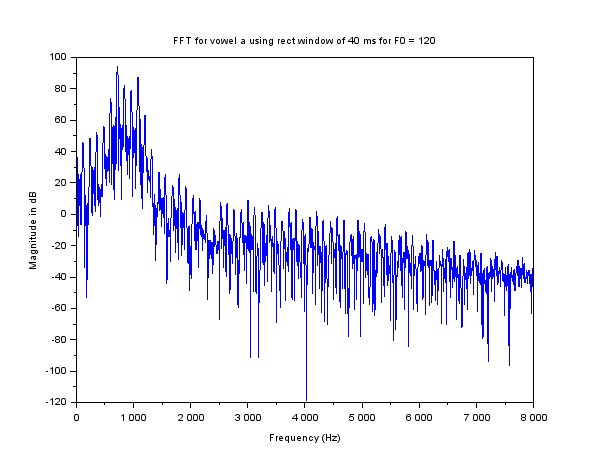
t = 10 ms



t = 20 ms

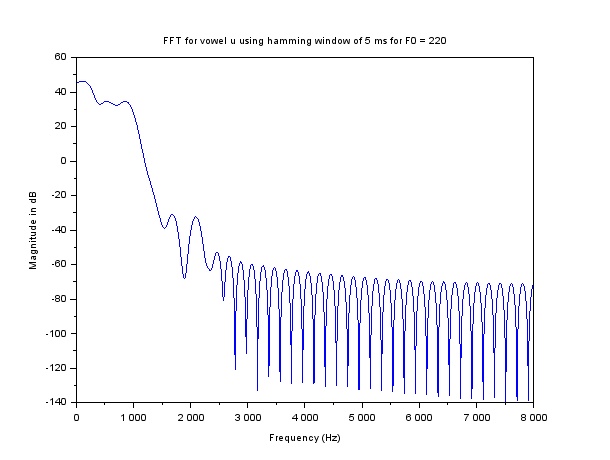


t = 40 ms

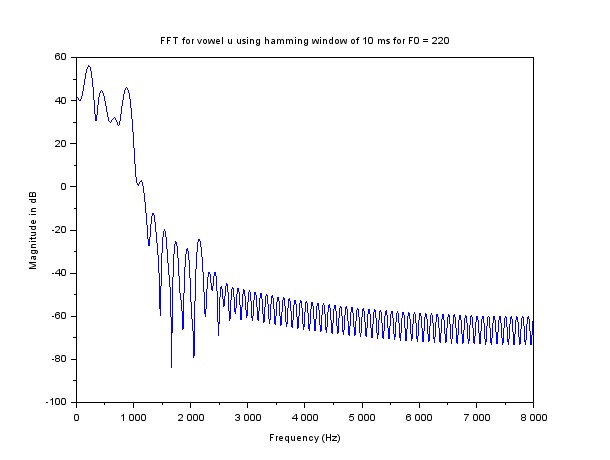


**Vowel /u/ using hamming window at F0 = 220 Hz**

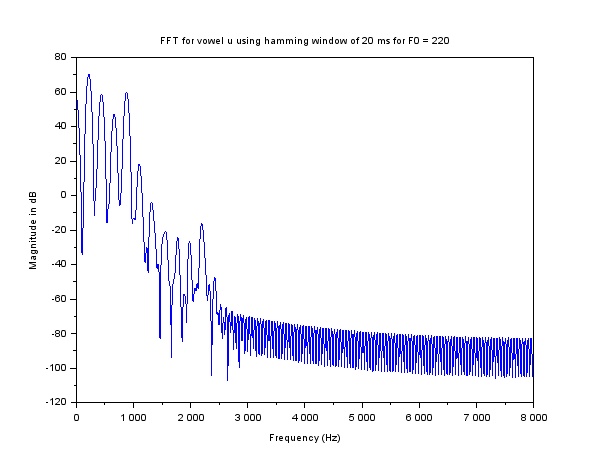
t = 5 ms



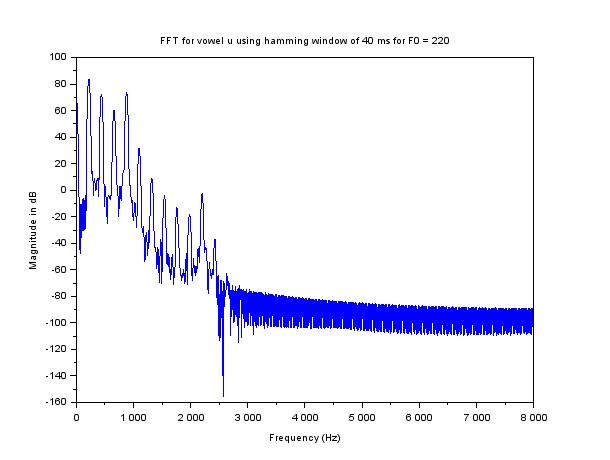
t = 10 ms



t = 20 ms

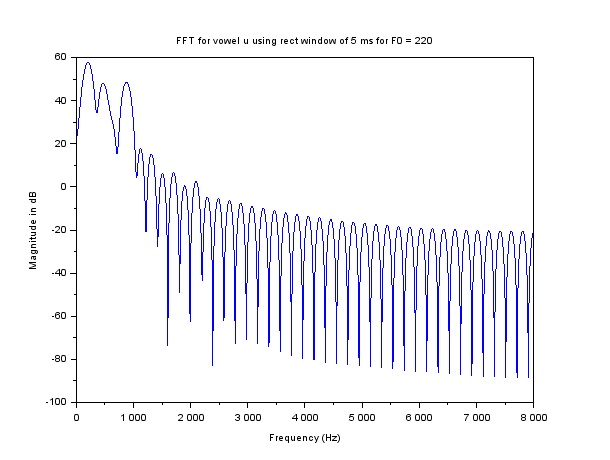


t = 40 ms

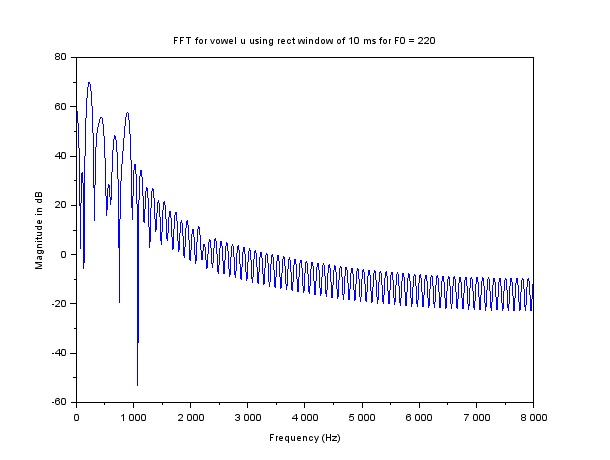


**Vowel /u/ using rectangula window at F0 = 220 Hz**

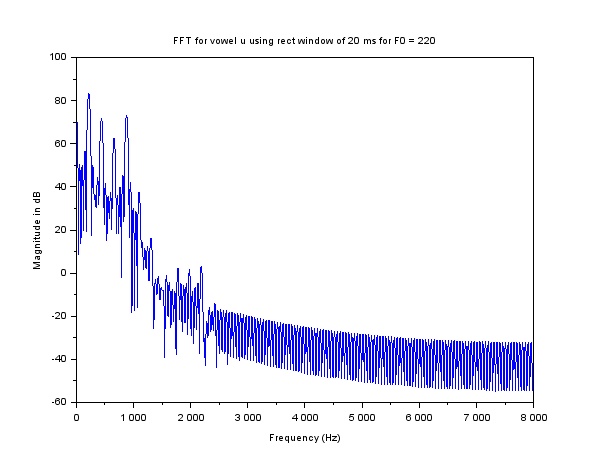
t = 5 ms



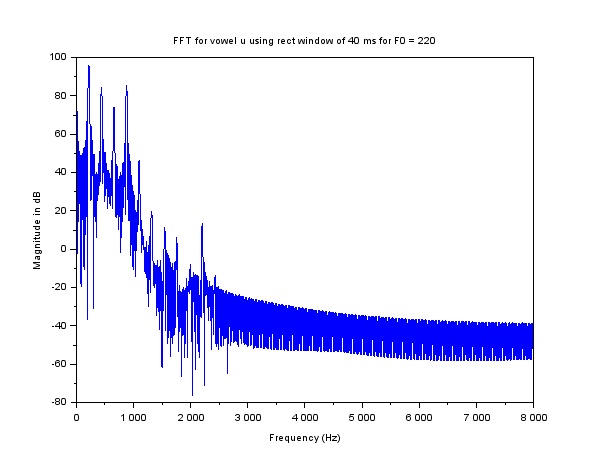
t = 10 ms



t = 20 ms



t = 40 ms



## Discussion

1. **Similarities and differences between different spectra**There are mainly two things changing for a given vowel, window type and length of window.

a. Effect of window size on spectrum

It is observed that spectra with low window size (windows of shorter duration) are smoother compared to that of longer durations. This phenomenon was observed for both Hamming as well as rectangular window. This is expected because for a given window type shorter length in time domain implies more spread in frequency domain. Original spectrum when convolved with this window having more spread in frequency domain, small variations will be removed and we will obtain a smoother window. This destroys pitch information (since multiple peaks will be merged together). But we can’t use longer windows in real life scenarios, because real signals are not stationary over longer duration.

b. Effect of window type on spectrum

Main lobe width of Hamming window is more than that of rectangular window. Therefore Hamming window spreads the spikes in original signal more than rectangular window. We can observe that peaks in case of Hamming window are broader compared to rectangular window. But sidelobes are more significant in case of rectangular window than Hamming window. Therefore detecting formants is more difficult in case rectangular window due to high energy sidelobs. Therefore hamming window should be preferred for detecting formants. But to find the exact frequency of the detected formant, we should use rectangular window, since peaks in that case will be sharper.

1. **Signal Parameters**
2. Formant frequencies

Formant frequencies for all combinations both the vowels were found out manually by zooming into the spectrum

For vowel /a/ at F0 = 120 Hz

|  |  |  |  |
| --- | --- | --- | --- |
| Window | Formant 1 | Formant 2 | Formant 3 |
| Real value (ground truth from Q4) | 730 | 1090 | 2440 |
| Hamming 5 ms | 718 | 988 | 2792 |
| Hamming 10 ms | 707 | 1058 | 2550 |
| Hamming 20 ms | 726 | 1077 | 2527 |
| Hamming 40 ms | 716 | 1076 | 2517 |
| Rectangular 5 ms | 724 | 992 | 2721 |
| Rectangular 10 ms | 721 | 1068 | 2312 |
| Rectangular 20 ms | 718 | 1073 | Not clear |
| Rectangular 40 ms | 718 | 1079 | Not clear |

For vowel /u/ at F0 = 220

|  |  |  |  |
| --- | --- | --- | --- |
| Window | Formant 1 | Formant 2 | Formant 3 |
| Real value (ground truth from Q4) | 300 | 870 | 2240 |
| Hamming 5 ms | 150 | 860 | 2098 |
| Hamming 10 ms | 225 | 875 | 2142 |
| Hamming 20 ms | 222 | 877 | 2192 |
| Hamming 40 ms | 224 | 878 | 2209 |
| Rectangular 5 ms | 200 | 868 | 2090 |
| Rectangular 10 ms | 225 | 894 | 2135 |
| Rectangular 20 ms | 213 | 874 | 2184 |
| Rectangular 40 ms | 217 | 878 | 2202 |

1. Bandwidth

Bandwidth is calculated by considering envelope of the spectrum and finding a frequency difference between the points 3dB below the formant frequencies. For generation we simplistically assumed B = 100 Hz for all formants for all vowels. But when we try to measure it using the method mentioned above, there are lots of parameters involved. Hence bandwidth estimate isn’t much reliable as compared formant.

1. Pitch

Pitch is estimated by frequency difference between two points divided by number of peaks in between. Here we assume that every spike in signal is captured as a peak in estimated spectrum. But that is not the case always. Hence estimated pitch was varying between 100 Hz to 200 Hz for 120 Hz vowels.