EE679 Speech Processing - Assignment 1 b

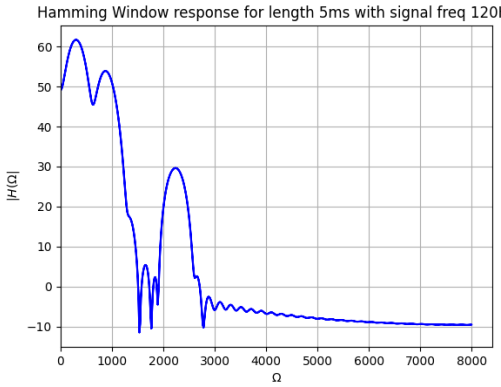
Shreya Laddha - 180070054

23/09/2021

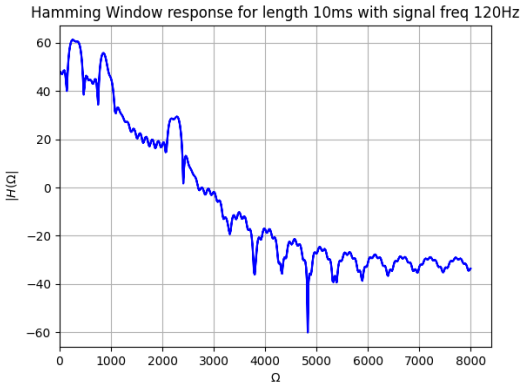
1 Question 1

Use your previous synthesized vowel /u/ at two distinct pitches (F0 = 120 Hz, F0 = 220 Hz). Keep the bandwidths constant at 100 Hz for all formants.
Vowel F1, F2, F3
/u/ 300, 870, 2240
We would like to use the DFT computed with various window lengths and shapes to estimate the vowel's F0 and formant frequencies and study the obtained accuracies with reference to our 'ground truth' values. For the analysis, use a single waveform segment near the centre of your synthesized vowel.
Plot the magnitude (dB) spectrum with rectangular and Hamming windows of lengths: 5 ms, 10 ms, 20 ms, 40 ms, each with a large zero-padded DFT. (i) Comment on the similarities and differences between the different computed spectra. (ii) Estimate the signal parameters from each of the magnitude spectra and report the error with respect to the ground-truth.

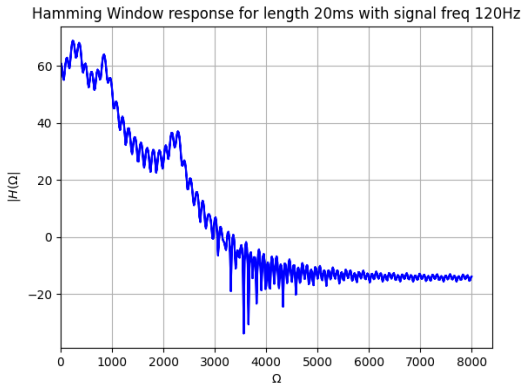
The function to get waveform for a vowel given formant frequencies was used as is from assignment 1 a. This was based on obtaining first the transfer function parameters and then finding output using difference equations.
The DFT is calculated using the appropriate window functions (hamming or rectangular) and output is plotted. For obtaining F0 from graph, peaks within 1kHz are counted and F0= 1000/peakCount.
Formant frequencies are obtained by observing the largest local maxima peak in the graph (magnified for obtaining near accurate value from graph).
We also see overlap case, in this scenario only the important peaks are counted for F0 calculation.
In cases where two peaks are very similar in magnitudes, the one with slightly higher magnitude is taken as formant freq.
Any aliasing and overlap is ignored for calculating number of peaks.
Below are the graphs and observations -



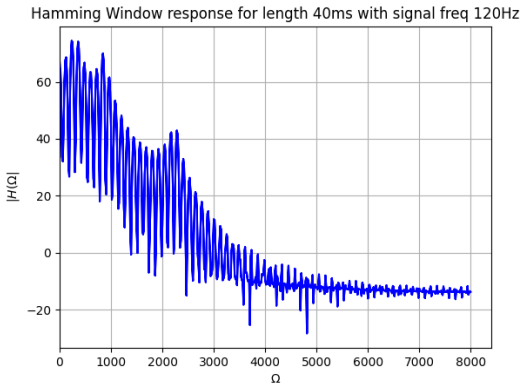
(a) 5 ms



(b) 10 ms

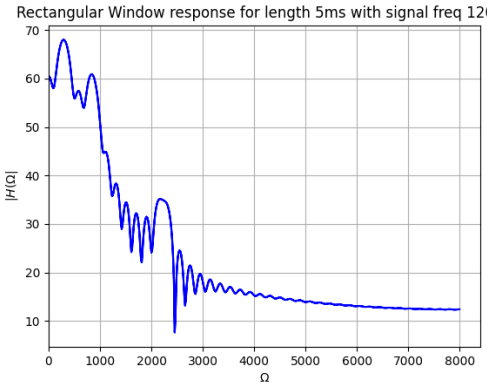


(c) 20 ms

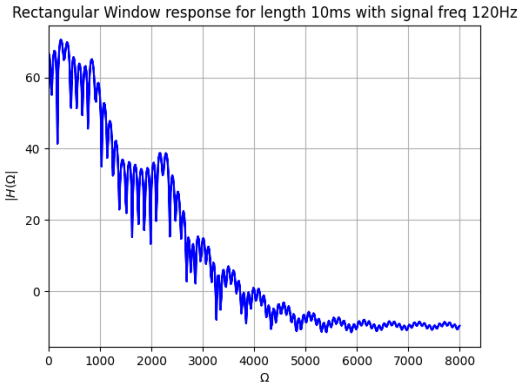


(d) 40 ms

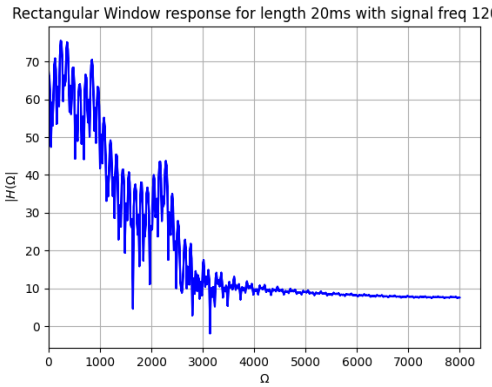
Figure 1: Hamming window outputs for 120 Hz F0 for different window lengths



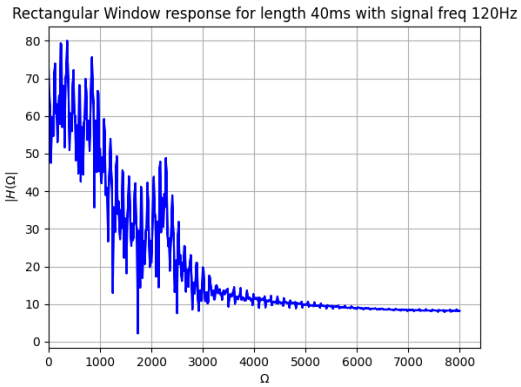
(a) 5 ms



(b) 10 ms

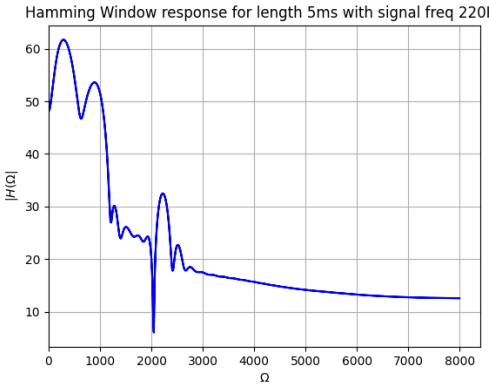


(c) 20 ms

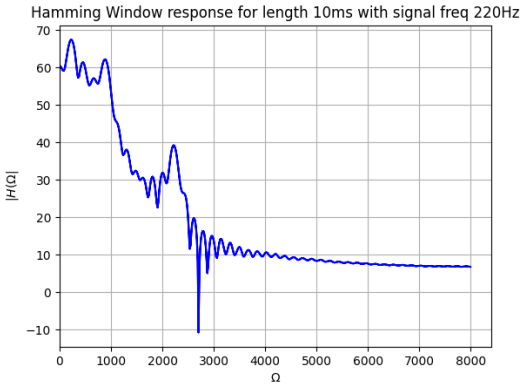


(d) 40 ms

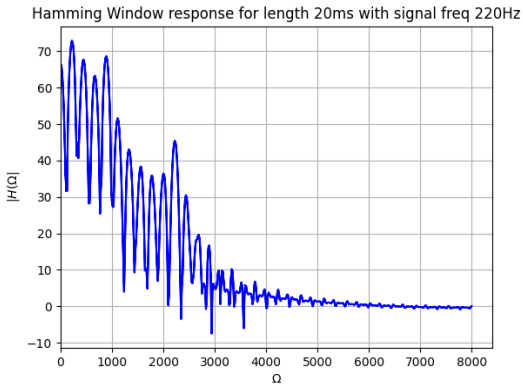
Figure 2: Rectangular window outputs for 120 Hz F0 for different window lengths



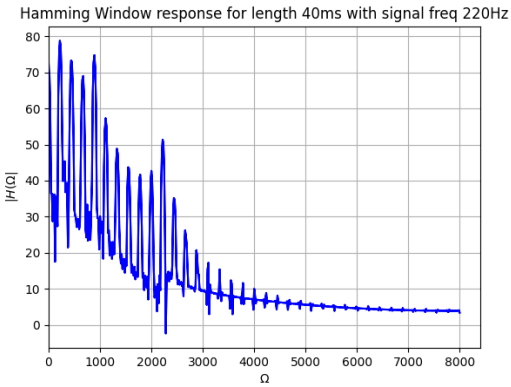
(a) 5 ms



(b) 10 ms

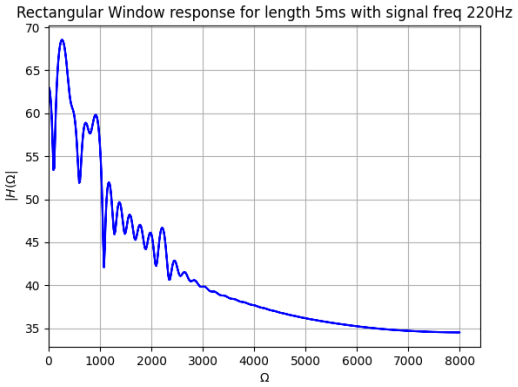


(c) 20 ms

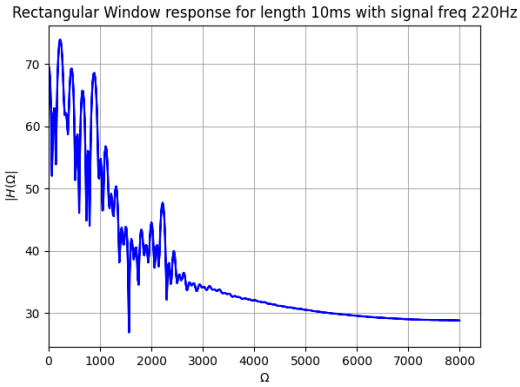


(d) 40 ms

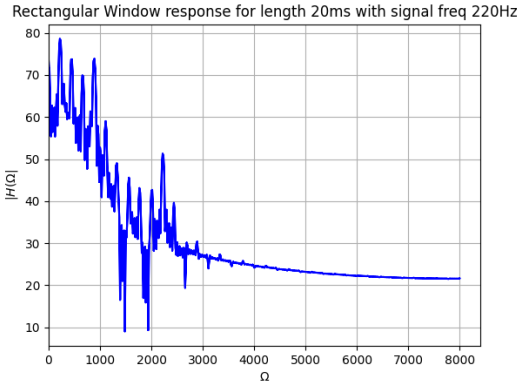
Figure 3: Hamming window outputs for 220 Hz F0 for different window lengths



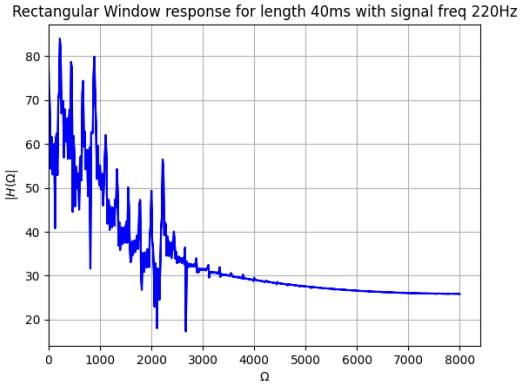
(a) 5 ms



(b) 10 ms



(c) 20 ms



(d) 40 ms

Figure 4: Rectangular window outputs for 220 Hz F0 for different window lengths

Window Length (ms)	F0 (Hz)	F1 (Hz)	F2 (Hz)	F3(Hz)
5	500	300	880	2250
10	143	230	860	2280
20	125	230	845	2280
40	125	230	844	2280

Table 1: Hamming Window at F0=120 Hz

Window Length (ms)	F0 (Hz)	F1 (Hz)	F2 (Hz)	F3(Hz)
5	333	297	845	2175
10	125	235	844	2172
20	125	235	844	2280
40	125	360	844	2280

Table 2: Rectangular Window at F0=120 Hz

Window Length (ms)	F0 (Hz)	F1 (Hz)	F2 (Hz)	F3(Hz)
5	500	290	890	2220
10	250	230	890	2220
20	250	220	890	2220
40	250	220	890	2220

Table 3: Hamming Window at F0=220 Hz

Window Length (ms)	F0 (Hz)	F1 (Hz)	F2 (Hz)	F3(Hz)
5	333	265	920	2200
10	250	220	890	2220
20	250	220	890	2220
40	250	220	890	2220

Table 4: Rectangular Window at F0=220 Hz

Ground Truth(Hz)	5ms	10ms	20ms	40ms
Hamming for 120 Hz F0				
F0: 120	380	23	5	5
F1: 300	0	-70	-70	-70
F2: 870	10	-10	-25	-26
F3: 2240	10	40	40	40
Hamming for 220 Hz F0				
F0: 220	380	30	30	30
F1: 300	-10	-70	-70	-70
F2: 870	20	20	20	20
F3: 2240	-20	-20	-20	-20

Table 5: Error in Hamming window = Obtained - ground truth

Ground Truth(Hz)	5ms	10ms	20ms	40ms
Rectangular for 120 Hz F0				
F0: 120	213	5	5	5
F1: 300	-3	-65	-65	60
F2: 870	-25	-26	-26	-26
F3: 2240	-65	-68	40	40
Rectangular for 220 Hz F0				
F0: 220	113	30	30	30
F1: 300	-35	-80	-80	-80
F2: 870	50	20	20	20
F3: 2240	-40	-20	-20	-20

Table 6: Error in Rectangular window = Obtained - ground truth

Observations

- The formant frequencies are easily calculated for small window sizes. The peaks get accumulated to give the exact formant frequency (wideband nature)

- on increasing the Window Size of both the windows, calculating the fundamental frequency F0 becomes a bit easy as peaks are clearly visible. (calculated by dividing 1 KHz frequency by the number of high peaks occurring from 0 to 1 KHz as mentioned before)
- on increasing the Window Size of both the windows, the peak magnitude is more clearly visible, but identifying the formant frequencies become challenging, due to peaks occurring in close proximity. (Increasing the window size implies going from wide-band to narrow-band and hence these changes are observed.)
- For a given window size, the peaks are more frequent and narrowly spaced in Rectangular window ones.
- As expected, the side lobes of the Hamming Windowed signal are lower than that in those where Rectangular window was applied. .
- As the fundamental frequency F0 increases, it affects the detection of the formant frequencies. For F1 we see that in both 120 and 220 Hz F0, it is not exactly obtained as it is not a multiple of the fundamental freq. And the error here is larger compared to that in F2 and F3 which are more near to multiples of fundamental freqency.
- For the same Window Function, if the fundamental frequency F0 is changed the peaks are more spaced out. The details of the waveform are more clear in the case of higher F0, however we see that error persists and increases for exact formant frequency value. But waveform is better.
- In case of error values, we see that increasing the window length does not have much effect on the values for formants especially F2 and F3. The largest change is visible when going from 5ms to 10ms window length; or changing the F0 or changing the window type.

Thus we can see distinctions of narrowband and wideband spectrogram very clearly in this assignment. One to be used for formant detection, other for fundamental freq detection. Distinction of hamming and recatngular window are also clear as the side lobes of hamming are much lower.

Note - Codes in .py file. Copied as such here

```
# -*- coding: utf-8 -*-
"""Copy of EE679_1b.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1tINW12bunqtkbVKDxDyMjHfUOdeEYNu7

Assignment 1b EE679

Shreya Laddha

Roll number - 180070054
"""

import numpy as np
from scipy.signal import zpk2tf,square
from scipy.fft import fft,fftfreq
from math import pi
from numpy import exp,zeros_like,log10,hamming
from numpy import convolve as conv
import matplotlib
import pylab as plt
from scipy import signal
from math import pi
import numpy as np
import matplotlib.pyplot as plt
```

```
from scipy.io.wavfile import write

def get_audio_waveform(f0, f1, f2, f3): ##taken from assignment 1
  #other parameters
  b1 = 100
  fs = 16000
  ts = 1/fs
  T = 0.5
  num_samples = int(fs*T)

  #getting transfer function for each frequency as a single formant resonator
  r = np.exp(-pi*b1*ts)
  theta = [2*pi*f1*ts, 2*pi*f2*ts, 2*pi*f3*ts]
  num1, den1 = signal.zpk2tf([0,0], [r*np.exp(1j*theta[0]) , r*np.exp(-1j*theta[0])],
    1)
  num2, den2 = signal.zpk2tf([0,0], [r*np.exp(1j*theta[1]) , r*np.exp(-1j*theta[1])],
    1)
  num3, den3 = signal.zpk2tf([0,0], [r*np.exp(1j*theta[2]) , r*np.exp(-1j*theta[2])],
    1)

  #for plotting magnitude response combine all poles and zeroes to get single transfer
     function of the cascade
  poles = [r*np.exp(1j*theta[0]) , r*np.exp(-1j*theta[0]), r*np.exp(1j*theta[1]) , r*
    np.exp(-1j*theta[1]), r*np.exp(1j*theta[2]) , r*np.exp(-1j*theta[2])]
  zeros = [0,0,0,0,0,0]
  num, den = signal.zpk2tf(zeros, poles, 1)

  #Plotting magnitude response
  omega, freq_resp = signal.freqz(num, den)
  # plt.figure()
  # plt.title('Filter magnitude response for '+phone+' at f0 = '+str(f0)+' Hz')
  f_hz = fs * omega/(2*pi)
  mag = 20*np.log10(abs(freq_resp))
  # plt.plot(f_hz,mag,'b')
  # plt.ylabel('Amplitude in dB')
  # plt.xlabel('Frequency in Hz')
  # plt.show()


  time = np.linspace(0, T, num_samples)
  x = np.zeros(num_samples)
  y = np.zeros(num_samples)
  state1 = np.zeros(num_samples)
  state2 = np.zeros(num_samples)

  #Impulse train
  for i in range(0, int(T*f0)):
    x[i*int(np.floor(fs/f0))] = 1

  #using differnce equations - since there are three cascade systems for three
    different formant freq, have to do three times
  #first system
  state1[0] = x[0]
  state1[1] = x[1] - den1[1].real*state1[0]
  for i in range(2, num_samples):
      state1[i] = x[i] - den1[1].real*state1[i-1] - den1[2].real*state1[i-2]

  #second system
  state2[0] = state1[0]
  state2[1] = state1[1] - den2[1].real*state2[0]
  for i in range(2, num_samples):
      state2[i] = state1[i] - den2[1].real*state2[i-1] - den2[2].real*state2[i-2]

  #Last third system
  y[0] = state2[0]
```

```
  y[1] = state2[1] - den3[1].real*y[0]
  for i in range(2, num_samples):
      y[i] = state2[i] - den3[1].real*y[i-1] - den3[2].real*y[i-2]

  # #plotting outputs
  # plt.figure()
  # plt.title('Excitation Response for '+phone+' at f0 = '+str(f0)+' Hz')
  # plt.plot(time, y, 'b')
  # plt.ylabel('Amplitude')
  # plt.xlabel('Time in sec')
  # plt.show()

  # plt.figure()
  # plt.title('Excitation Response Zoomed in for '+phone+' at f0 = '+str(f0)+' Hz')
  # plt.plot(time[0:1000], y[0:1000], 'b')
  # plt.ylabel('Amplitude')
  # plt.xlabel('Time in sec')
  # plt.show()
  # y = np.int16(y/np.max(np.abs(y))*32767)
  # write("audio_'"+phone[1]+"'_"+str(f0)+"Hz.wav",16000,y)

  return y

def get_window(window, win_length,fs,output,f0):
  window_size = int(win_length*fs/1000)
  if(window=='Hamming'):
    window_signal = output[:window_size] * hamming(window_size)
  else:
    window_signal = output[:window_size]
  dft = fft(window_signal, n=1024)
  freq = fftfreq(dft.shape[-1], 1/fs)

  plt.figure()

  plt.plot(abs(freq),20*log10(abs(dft)),'b')
  plt.xlim(xmin=0)
  plt.grid("True")
  plt.title("{} Window response for length {}ms with signal freq {}Hz".format(window,
    win_length,f0))
  plt.ylabel(r"$|H(\Omega|$")
  plt.xlabel(r"$\Omega$")
  plt.savefig("./plots/"+window+"_Window_Freq_resp_"+str(f0)+"_"+str(win_length)+".png
    ")
  plt.show()
  # plt.figure()
  # plt.plot(abs(freq[:200]),20*log10(abs(dft[:200])),'r') #magnified plot for F0
    calculation
  # plt.title("{} Window Magnified response for length {}ms with signal freq {}Hz".
    format(window, win_length,f0))
  # plt.ylabel(r"$|H(\Omega|$")
  # plt.xlabel(r"$\Omega$")
  #not saving this plot

def generate_vowels(formant_frequencies,bandwidth,signal_frequency,time,fs,window,
    win_length):
  f1 = formant_frequencies[0]
  f2 = formant_frequencies[1]
  f3 =formant_frequencies[2]

  response = get_audio_waveform(signal_frequency, f1,f2,f3)
  # get DFT and plot
  get_window(window, win_length,fs,response,signal_frequency)

f0 = [120,220]
formant_freq = [300,870,2240]
```

```
time = 0.5
fs = 16000
bw = 100
windows = ["Hamming","Rectangular"]
window_lengths = [5,10,20,40]

for freq in f0:
  for window in windows:
    for win_length in window_lengths:
      generate_vowels(formant_freq,bw,freq,time,fs,window,win_length)
      # break
```