EE 679: Speech Processing
Assignment 1

Kumar Ashutosh, 16D070043

August 31, 2019

Question 1

Reasoning

Given,
Formant Frequency = 900 Hz
Bandwidth = 200 Hz
Sampling Frequency = 16 kHz
We know from the Vocal Tract Model for Single Formant Resonator that roots of the transfer function is given by

$$r = e^{-B_i \pi T},\ \theta = 2\pi F_i T$$

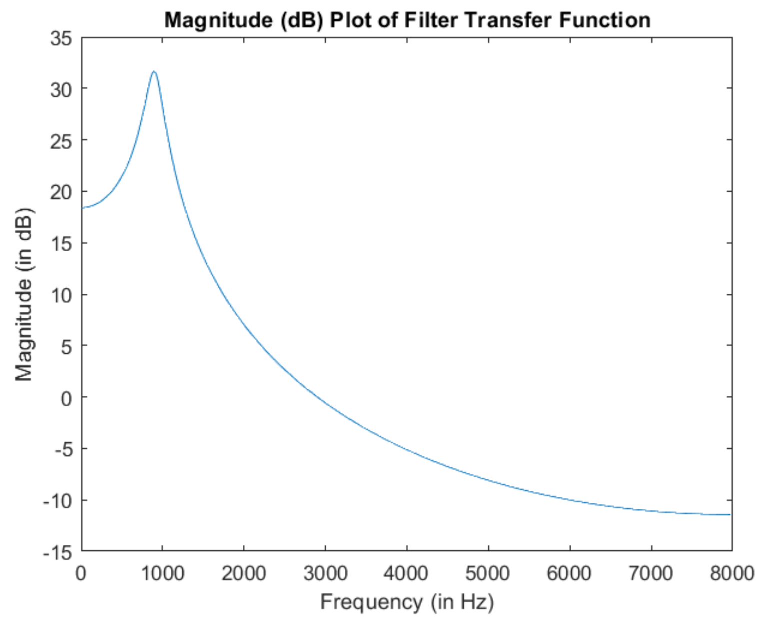
Using this and the given information we get the transfer function as

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

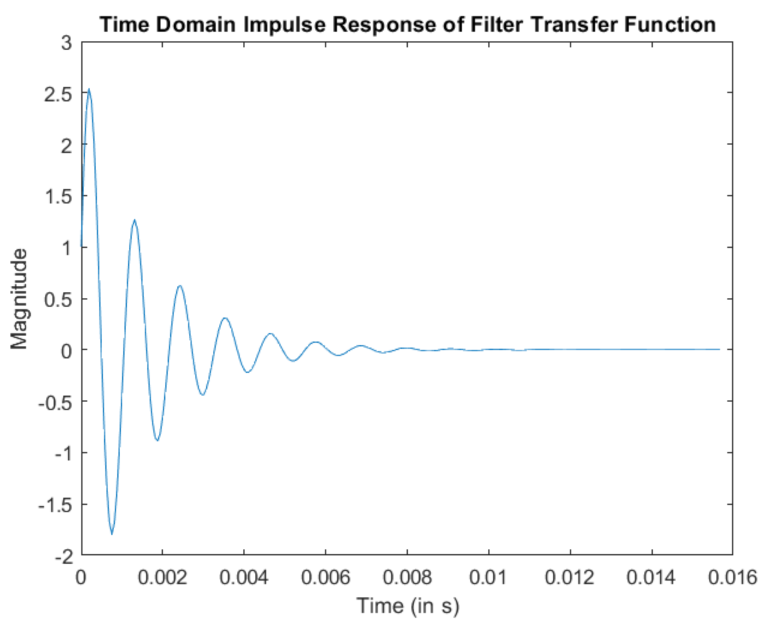
which on simplification becomes,

$$H(z) = \frac{1}{1-1.804z^{-1}+0.9245z^{-2}}$$

The magnitude plot of this transfer function and the impulse response is plotted below.



Magnitude Plot of the Transfer Function



Impulse Response of the Transfer Function

Code

design_tf.m
MATLAB function to design transfer function based on the roots of the filter. The roots are calculated as discussed in the above subsection.

```
function [Hz, num, den] = design_tf(formant_freq,
    bandwidth, samp_freq)

%% Finding roots of the Transfer Function
r = exp(-bandwidth*pi*(1/samp_freq));
theta = 2*pi*formant_freq*(1/samp_freq);

[~, num_formant] = size(formant_freq);

den = 1;
for iter =  1:num_formant
    den = conv(den, [1, -2*r*cos(theta(iter)), r*r]);
end

num = zeros(size(den));
num(1) = 1; %Multiplying num and den by the highest
    power of denominator

Hz = tf(num, den, 1/samp_freq);
end
```

question1.m MATLAB main script to call the above function with the desired parameters and obtain the transfer function.

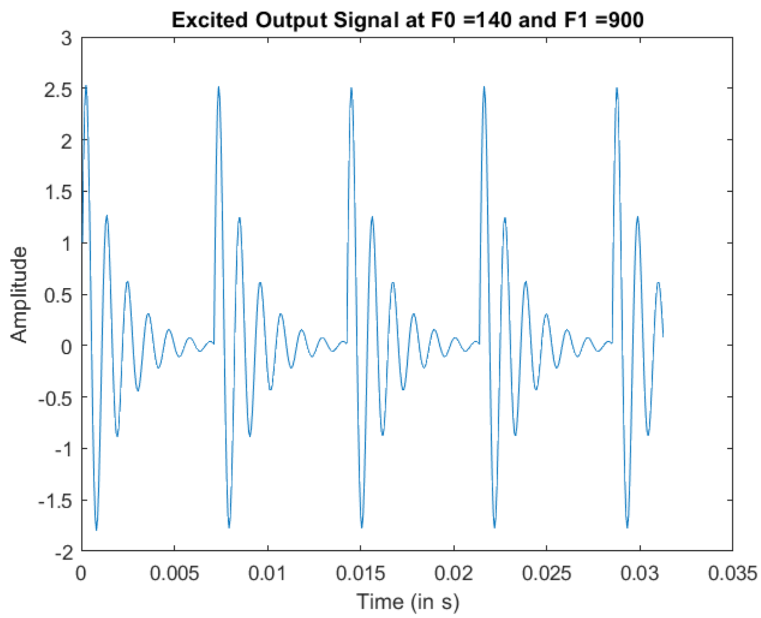
```
%% Data
%Frequencies in Hertz
formant_freq = 900;
bandwidth = 200;
samp_freq = 16e3;

%% Transfer Function
[Hz, num, den] = design_tf(formant_freq, bandwidth,
    samp_freq);
[h, w] = freqz(num, den);
```

```
figure
plot((w*samp_freq)/(2*pi), 20*log10(abs(h)))
title('Magnitude (dB) Plot of Filter Transfer Function'
    )
ylabel('Magnitude (in dB)')
xlabel('Frequency (in Hz)')


[val, index] = impz(num, den);

figure
plot(index/(samp_freq), val)
title('Time Domain Impulse Response of Filter Transfer
    Function')
ylabel('Magnitude')
xlabel('Time (in s)')
```

Question 2

Reasoning

The periodic source excitation is modelled as a train of impulse at 140 Hz. The linear convolution is then done iteratively to find the output of the filter operation. The time domain output waveform is as attached below.



Time Domain Filtering Output

0.1 Code

filter_input.m
MATLAB function to perform time domain convolution

```
function output = filter_input(f0, den, samp_freq, duration)
%% Input Data
input_data = zeros(1, samp_freq*duration); % Corresponding to 0.5s data
input_data(1:round(samp_freq/f0):end) = 1;

%% Convolution Parameters
filter_coeffs = -den;  %Coeffs are negative of each other except first one
filter_coeffs = filter_coeffs(2:end); %y[n-1] starts with second coeffs.
[~, filter_length] = size(filter_coeffs);

%% Time Domain Convolution
```

```
output = input_data;
[~, input_length] = size(input_data);

for iter=1:input_length
    for filter_iter=1:filter_length
        if iter - filter_iter < 1
            add_factor = 0; %y[n] = 0 for n < 1
        else
            add_factor = output(iter-filter_iter);
        end
        output(iter) = output(iter) + add_factor*
            filter_coeffs(filter_iter);
    end
end
end
```

question2.m

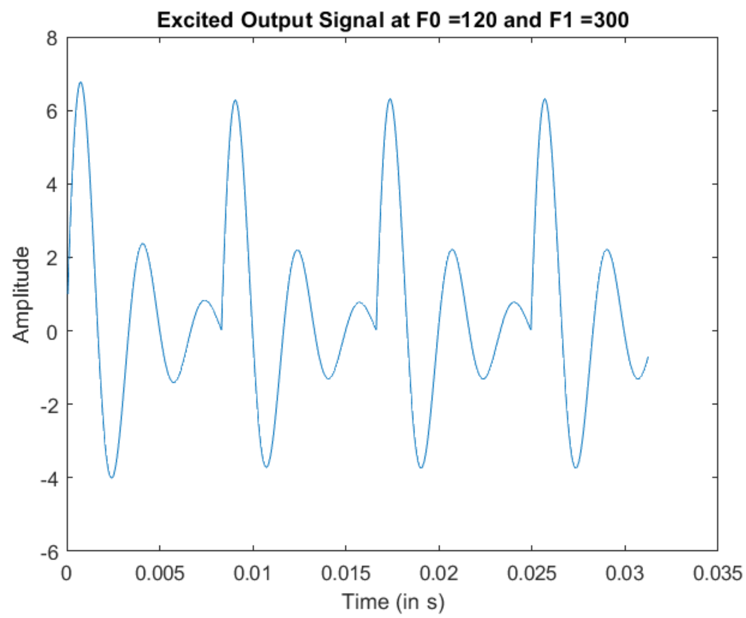
Main script to call the above function.

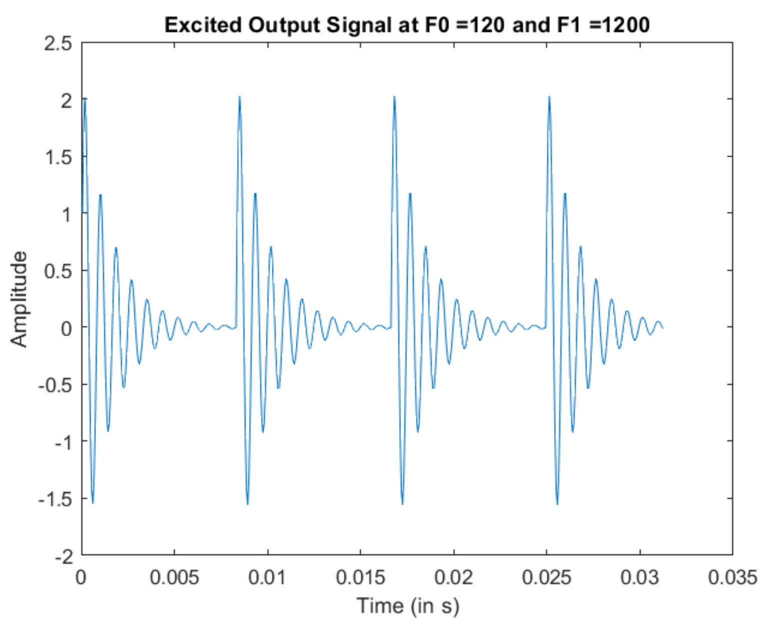
```
%% Parameters
%All frequencies in Hz
samp_freq = 16e3;
f0 = 140;
formant_freq = 900;
bandwidth = 200;
duration = 0.5; %in seconds

single_formant_analysis(formant_freq, f0, bandwidth,
    duration, samp_freq);
```

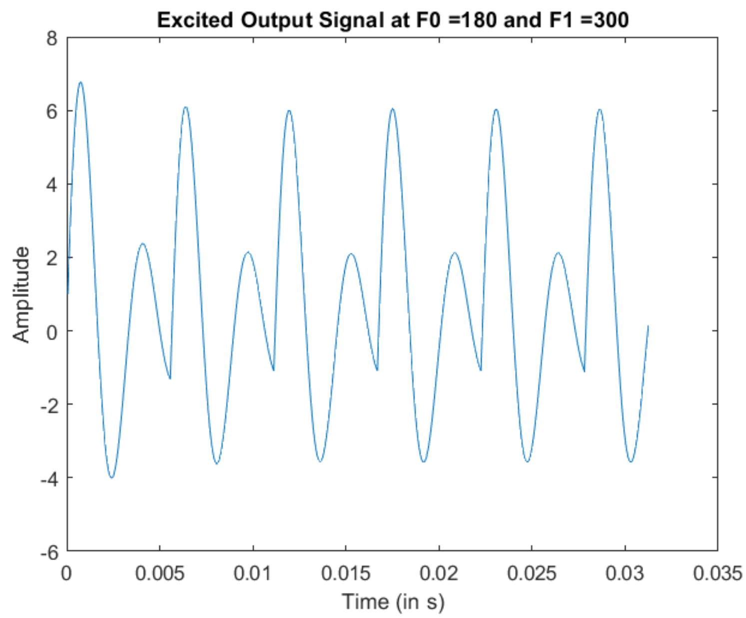
Question 3

Reasoning

For the three parts of this questions, the corresponding three waveform are as follows.


Excited Output Signal at F0 =120 and F1 =300
Amplitude
Time (in s)
8
6
4
2
0
-2
-4
-6
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035


Excited Output Signal at F0 =120 and F1 =1200
Amplitude
Time (in s)
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035



The three waveforms are also saved as .wav file and played to understand the sound. It is evident from the waveform as well that higher frequency of 1200 Hz results in faster oscillations in second waveform, hence the quality in this case decreases. (a) and (c) have the same formant frequency while F0 is same for (a) and (b). The waveforms in (a) and (c) looks similar due to tha same formant frequency.

The three parts are also saved as audio files. Names of the audio files for each part are as follows :-

- (a) - output_F0_120_F1_300.wav
- (b) - output_F0_120_F1_1200.wav
- (c) - output_F0_180_F1_300.wav

Code

single_formant_analysis.m

MATLAB function to do Q1 and Q2 simultaneously with formant frequency.

```
function output = single_formant_analysis(formant_freq,
    f0, bandwidth, duration, samp_freq)
```

```


%% Transfer Function and Time Domain Output Signal
[~, ~, den] = design_tf(formant_freq, bandwidth,
    samp_freq);
output = filter_input(f0, den, samp_freq, duration);

%% Plotting over a few pitch periods
plot_samples = 500; %Plot 500 samples. 500 samples is
    considerably good
figure
plot((1:plot_samples)/samp_freq, output(1:plot_samples)
    );
title(strcat('Excited Output Signal at F0 = ', num2str(
    f0), ' and F1 = ', num2str(formant_freq)));
ylabel('Amplitude')
xlabel('Time (in s)')

%% Saving half second audio output
scaled_output = (2/(max(output(:))-min(output(:))))*(
    output - min(output(:))) - 1;
audiowrite(strcat('output_F0_', num2str(f0), '_F1_',
    num2str(formant_freq),'.wav'), scaled_output,
    samp_freq);

end
```

question3.m Main code to run the above function with the three desired parameters

```
%% Parameters

%All frequencies in Hz
samp_freq = 16e3;
duration = 0.5; %in seconds

%(a) F0 = 120 Hz, F1 = 300 Hz, B1 = 100 Hz
f0 = 120;
formant_freq = 300;
```

```
bandwidth = 100;
single_formant_analysis(formant_freq, f0, bandwidth,
    duration, samp_freq);

%(b) F0 = 120 Hz, F1=1200 Hz, B1 = 200 Hz
f0 = 120;
formant_freq = 1200;
bandwidth = 200;
single_formant_analysis(formant_freq, f0, bandwidth,
    duration, samp_freq);

%(c) F0 = 180 Hz, F1 = 300 Hz, B1 = 100 Hz
f0 = 180;
formant_freq = 300;
bandwidth = 100;
single_formant_analysis(formant_freq, f0, bandwidth,
    duration, samp_freq);
```

Question 4

Reasoning

For multiple formant frequencies the transfer functions gets multiplied, or in other words, the roots is the combination of all the formant frequencies. The three vowels are saved as audio files. The name of the audio files for each vowel are as follows :-

- "a" for 120Hz - a-120.wav
- "a" for 220Hz - a-220.wav
- "i" for 120Hz - i-120.wav
- "i" for 220Hz - i-220.wav
- "u" for 120Hz - u-120.wav
- "u" for 220Hz - u-220.wav

Code

process_and_save_audio.m
MATLAB function to process the input as per the formant frequency, obtain the output, scale it accordingly and save it to a wav file.

```matlab
function scaled_output = process_and_save_audio(f0, formant_freq, bandwidth, samp_freq, save_audio, filename)

[~, ~, denominator] = design_tf(formant_freq, bandwidth, samp_freq);
output = filter_input(f0, denominator, samp_freq, 0.5);
%Scaling the output while saving to audio file
scaled_output = (2/(max(output(:))-min(output(:))))*(output - min(output(:))) - 1;
if save_audio == 1
    audiowrite(filename, scaled_output, samp_freq);
end
end
```

question4.m Main code to run the above frequencies for the required cases.

```matlab
%% Global Variables
samp_freq = 16e3;
bandwidth = 100;

%% /a/ at f0 = 120
f0 = 120;
formant_freq = [730, 1090, 2440];
process_and_save_audio(f0, formant_freq, bandwidth, samp_freq, 1, 'a-120.wav');

%% /a/ at f0 = 220
f0 = 220;
formant_freq = [730, 1090, 2440];
process_and_save_audio(f0, formant_freq, bandwidth, samp_freq, 1, 'a-220.wav');

```

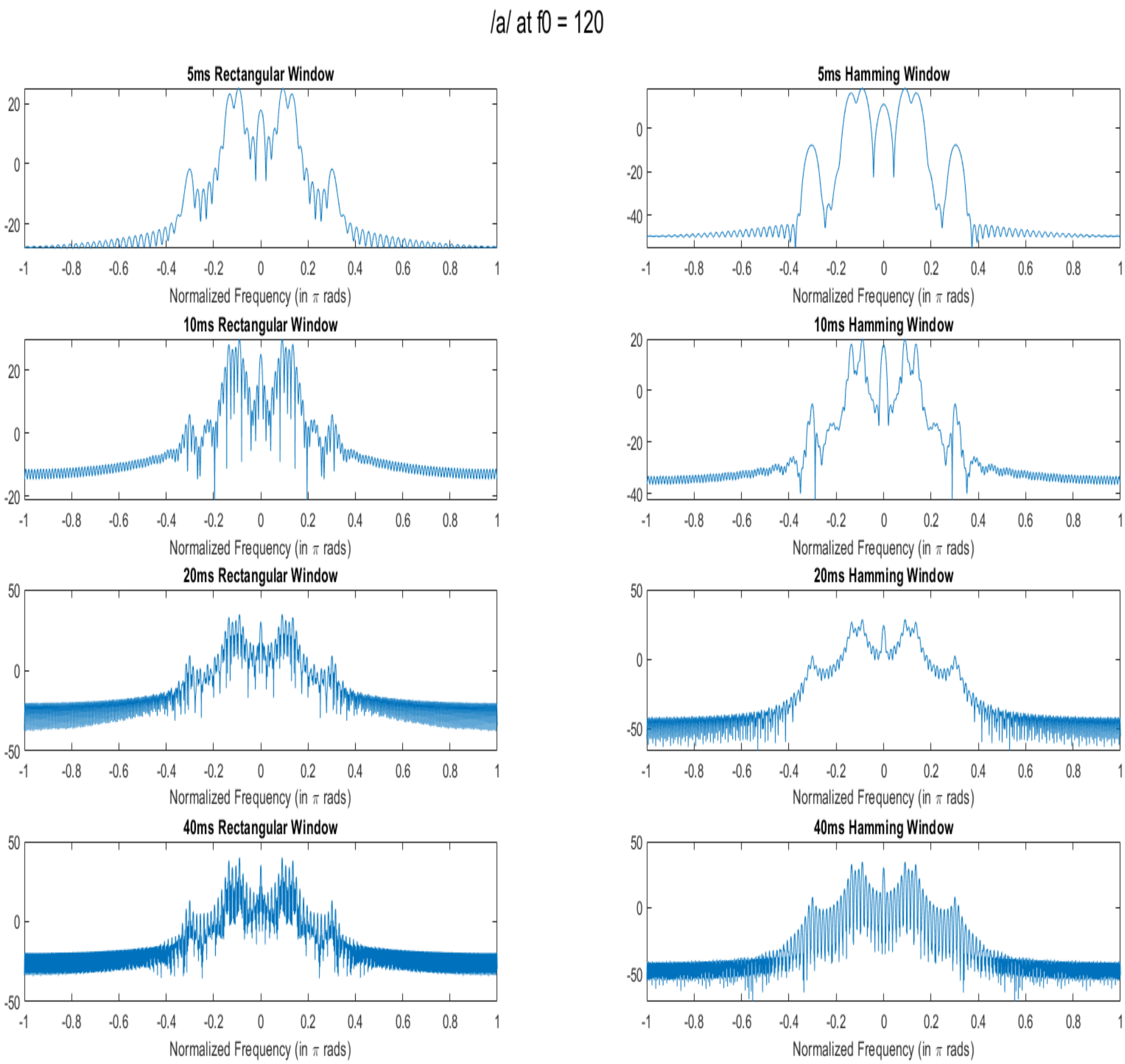
```
%% /i/ at f0 = 120
f0 = 120;
formant_freq = [270, 2290, 3010];
process_and_save_audio(f0, formant_freq, bandwidth,
    samp_freq, 1, 'i-120.wav');

%% /i/ at f0 = 220
f0 = 220;
formant_freq = [270, 2290, 3010];
process_and_save_audio(f0, formant_freq, bandwidth,
    samp_freq, 1, 'i-220.wav');

%% /u/ at f0 = 120
f0 = 120;
formant_freq = [300, 870, 2240];
process_and_save_audio(f0, formant_freq, bandwidth,
    samp_freq, 1, 'u-120.wav');

%% /u/ at f0 = 220
f0 = 220;
formant_freq = [300, 870, 2240];
process_and_save_audio(f0, formant_freq, bandwidth,
    samp_freq, 1, 'u-220.wav');
```

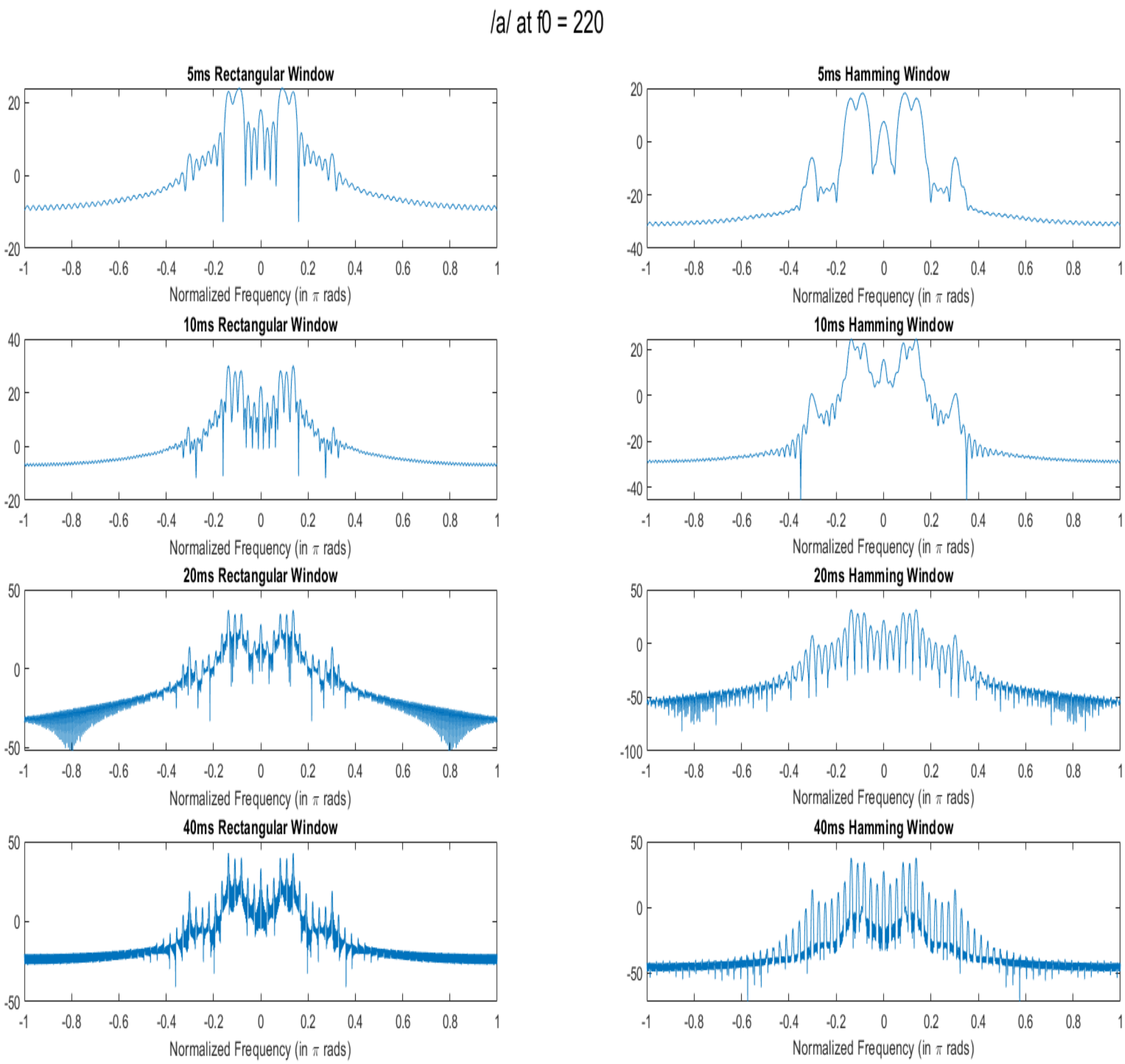
Question 5

Reasoning

For both the window examples, waveform in the particular window was multiplied and other samples were let to be zero. FFT of the resulting signal is calculated and plotted in the normalized angular angle case.

The plots for the two F0 case is attached below -


/a/ at f0 = 120
5ms Rectangular Window
Normalized Frequency (in π rads)
5ms Hamming Window
Normalized Frequency (in π rads)
10ms Rectangular Window
Normalized Frequency (in π rads)
10ms Hamming Window
Normalized Frequency (in π rads)
20ms Rectangular Window
Normalized Frequency (in π rads)
20ms Hamming Window
Normalized Frequency (in π rads)
40ms Rectangular Window
Normalized Frequency (in π rads)
40ms Hamming Window
Normalized Frequency (in π rads)



(i) Comparison/Observations in the plots

- Different spacing in DFTF with different fundamental frequency F0.
- Increasing window length makes the output more clear and observable. FOr low window length, the frequcies are packed and not clear.
- Hamming Window vs Rectangular Frequency. In rectangular window, only first formant peak is observable, while in Hamming window other peaks are also clear. This can be attributed to the Energy Leakage due to the use of Rectangular Window.

(ii) Estimation of Signal Parameters

- Estimated F0 - 117Hz and 223Hz (actual 120Hz and 220Hz)
- Estimated Formant Freq - 725Hz, 1095Hz and 2435Hz (actual 730Hz, 1090Hz, 2440Hz)

Code

compute_dtft.m MATLAB helper code to calculate the DTFT and return the log magnitude.

```
function dtft = compute_dtft(f0, formant_freq,
    bandwidth, samp_freq, window_size, window_type)
% Window type is 1 for rectangular and 2 for hamming

data = process_and_save_audio(f0, formant_freq,
    bandwidth, samp_freq, 0, '');
[~, data_points] = size(data);

if window_type == 1
    window = ones(1, window_size);
else
    window = hamming(window_size)';
end

windowed_data = zeros(size(data));
windowed_data(1:window_size) = window.*data(1:
    window_size);

```

```
dtft = fft(windowed_data);
dtft = circshift(20*log10(abs(dtft)), data_points/2);

end
```

question5.m Main Code to run the above funtions on all the sixteen combinations and plot the output in a single plot.

```
%% Global Variables
samp_freq = 16e3;
bandwidth = 100;

%% /a/ at f0 = 120
f0 = 120;
formant_freq = [730, 1090, 2440];

figure

x_points = (1:8000)/4000 - 1;

% 5 ms Rectangular Window
window_type=1;
window_size=5e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 1)
plot(x_points, dtft)
title('5ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 10 ms Rectangular Window
window_type=1;
window_size=10e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 3)
plot(x_points, dtft)
```

```
title('10ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 20 ms Rectangular Window
window_type=1;
window_size=20e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 5)
plot(x_points, dtft)
title('20ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 40 ms Rectangular Window
window_type=1;
window_size=40e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 7)
plot(x_points, dtft)
title('40ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 5 ms Hamming Window
window_type=2;
window_size=5e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 2)
plot(x_points, dtft)
title('5ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

% 10 ms Hamming Window
window_type=2;
```

```
window_size=10e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 4)
plot(x_points, dtft)
title('10ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

% 20 ms Hamming Window
window_type=2;
window_size=20e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 6)
plot(x_points, dtft)
title('20ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

% 40 ms Hamming Window
window_type=2;
window_size=40e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 8)
plot(x_points, dtft)
title('40ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

sgtitle('/a/ at f0 = 120')



%% /a/ at f0 = 220
f0 = 220;
formant_freq = [730, 1090, 2440];
```

```
figure

x_points = (1:8000)/4000 - 1;

% 5 ms Rectangular Window
window_type=1;
window_size=5e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 1)
plot(x_points, dtft)
title('5ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 10 ms Rectangular Window
window_type=1;
window_size=10e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 3)
plot(x_points, dtft)
title('10ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 20 ms Rectangular Window
window_type=1;
window_size=20e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 5)
plot(x_points, dtft)
title('20ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

```

```
% 40 ms Rectangular Window
window_type=1;
window_size=40e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 7)
plot(x_points, dtft)
title('40ms Rectangular Window')
xlabel('Normalized Frequency (in \pi rads)')

% 5 ms Hamming Window
window_type=2;
window_size=5e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 2)
plot(x_points, dtft)
title('5ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

% 10 ms Hamming Window
window_type=2;
window_size=10e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 4)
plot(x_points, dtft)
title('10ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

% 20 ms Hamming Window
window_type=2;
window_size=20e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);
```

```
subplot(4, 2, 6)
plot(x_points, dtft)
title('20ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

% 40 ms Hamming Window
window_type=2;
window_size=40e-3*samp_freq;
dtft = compute_dtft(f0, formant_freq, bandwidth,
    samp_freq, window_size, window_type);

subplot(4, 2, 8)
plot(x_points, dtft)
title('40ms Hamming Window')
xlabel('Normalized Frequency (in \pi rads)')

sgtitle('/a/ at f0 = 220')
```