EE386 Digital Signal Processing Lab **Jul-Dec 2021**

Experiment : 4

Author: Aditya Sriram Bhaskara *Email:* bhaskaraadityasriram.191ee209@nitk.edu.in

This lab experiment covers various practicality of Digital Signal Processing such as plotting signal spectra, extraction of fundamental frequency of a signal, design of a keylock system using fundamental frequencies, temporal variations etc. Along with Python, I have used libraries such as numpy, pandas, scipy etc. The code to my entire work in this lab experiment is here. And the input files and my output files can be viewed here.

Please Note : I have used $\alpha = 2$ because my registration number is 191910.

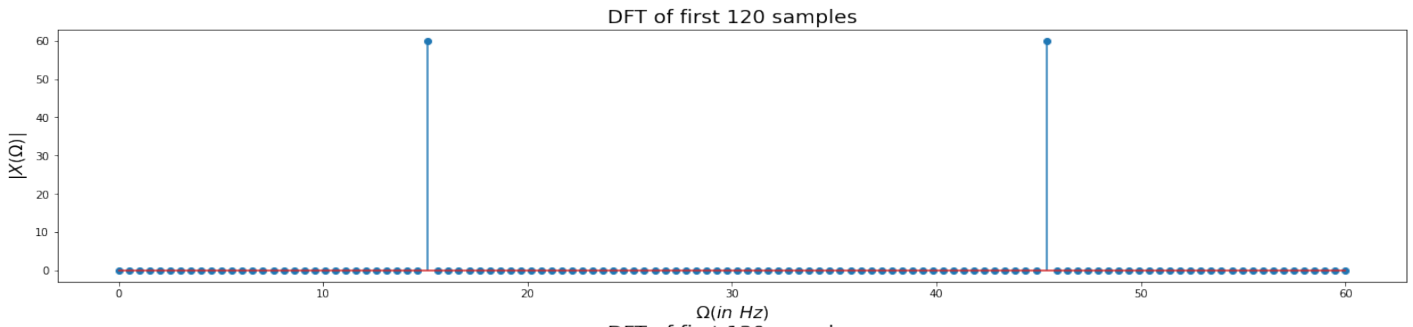
Question 1 - Computing DFT

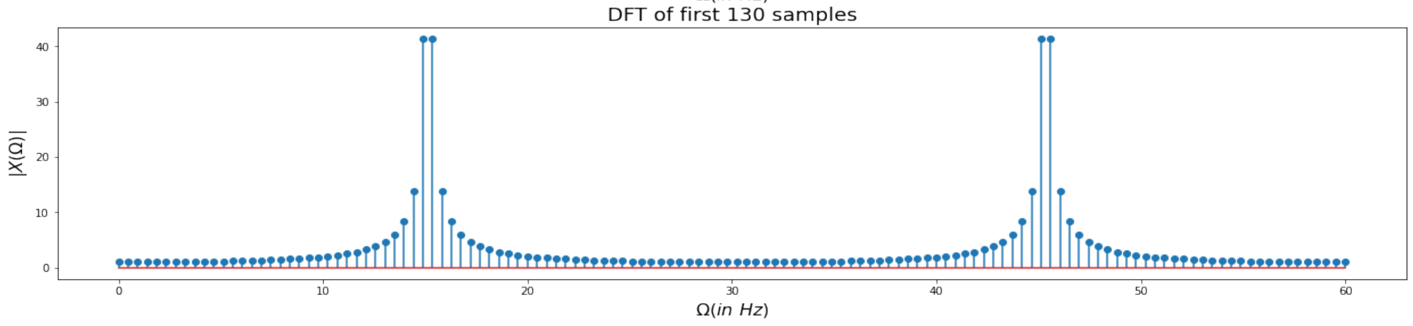
(Subproblem - 1 and 2)

This problem wants us to generate a unit amplitude sinusoidal signal of frequency 15 Hz for a duration of 2 seconds with a sampling rate of 120 samples/sec and plot magnitude of the DFT of the first 120 samples of the signal and plot the magnitude of the DFT of the first 130 samples of the signal against frequency in Hertz.

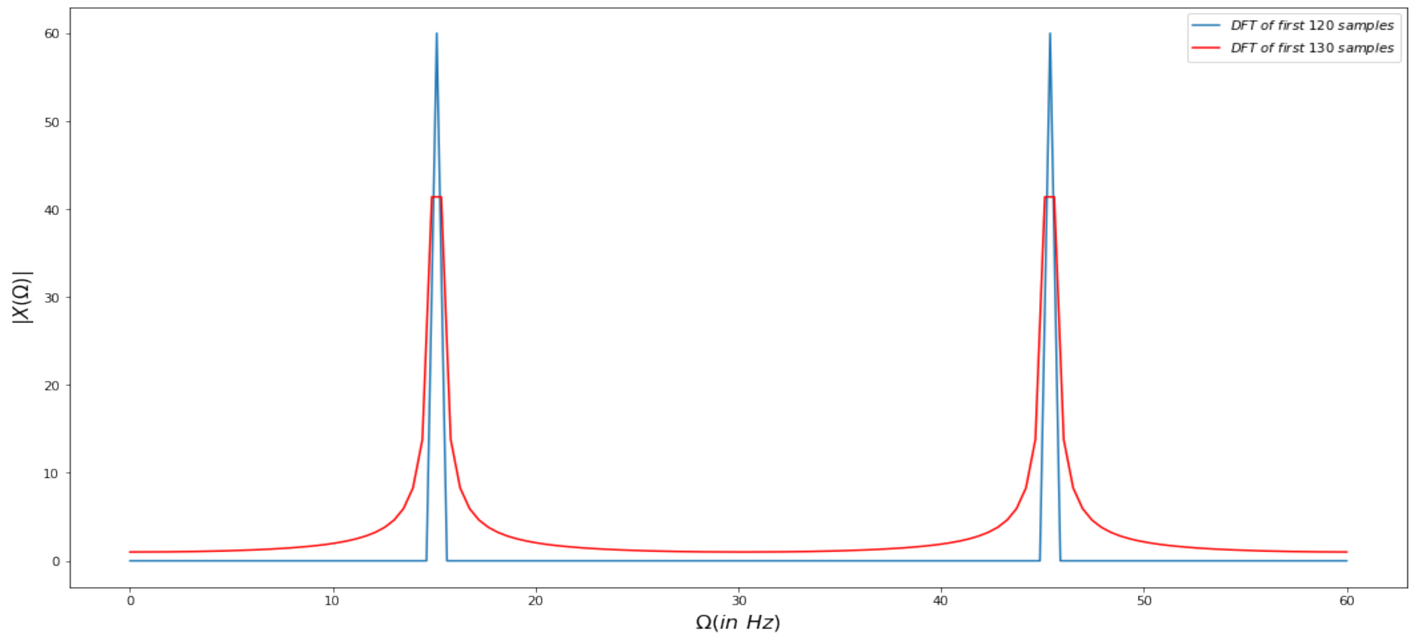
The code to this question can be found here

The stem plots are as follows :





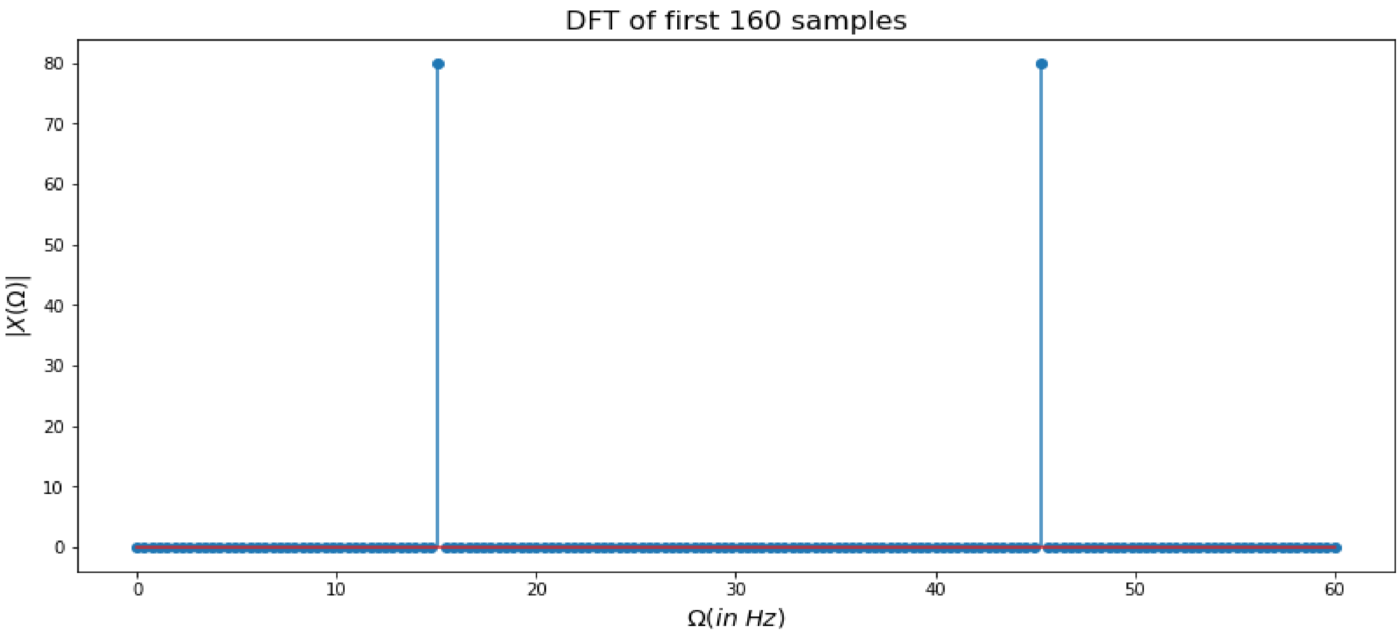
The plot of the two DFTs' being overlayed on each other can be observed as :



Analysing these plots, I observed that there is a spectral leakage in DFT plot for 130 samples but not for 120 samples and this is because in the later the DFT that is being caculated is over a period of time which is an integral multiple of the signal's fundamental period but for the former this is not the case.

(Subproblem - 3) This question asks to find out an N value such that

- The DFT of the first N points of the signal matches with the DFT of the first 120 samples of the signal
- $N \neq 120$
- Using trial and error I reckoned that $N = 160$
- So we can say that the DFT of the first 160 points of the signal matches with the DFT of the first 120 samples of the signal, this may be because it has no spectral leakage and is an integral multiple of the fundamental period.
- The code to this question can be found here
- The plot is as follows :



Question 2 - Resolution of DFT

The problem asks us to generate a signal

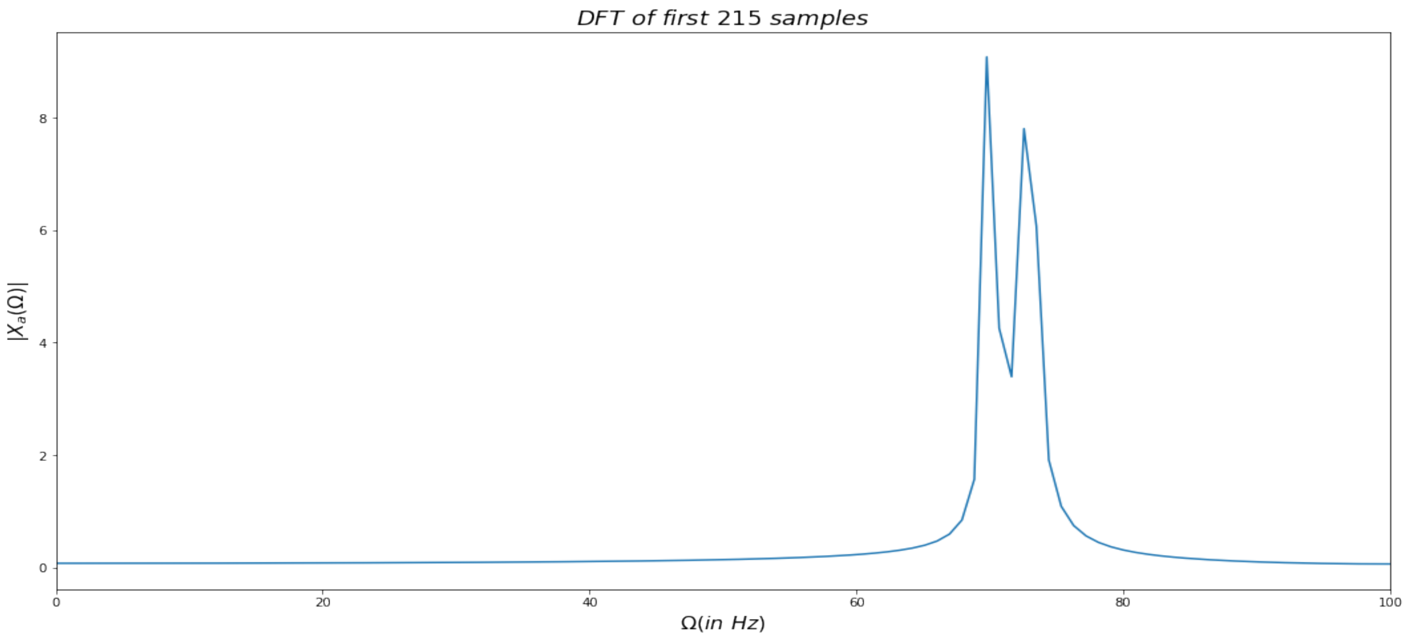
$$x_a(t) = 0.1\sin(A\pi t) + \cos(B\pi t)$$

and sample it with a rate of 200 samples/sec for a duration of 10 seconds and then plot the DFT of the signal for

- 215 Samples
- 415 Samples
- 1115 Samples
- 1515 Samples
- 1915 Samples
- The code to this question can be found here

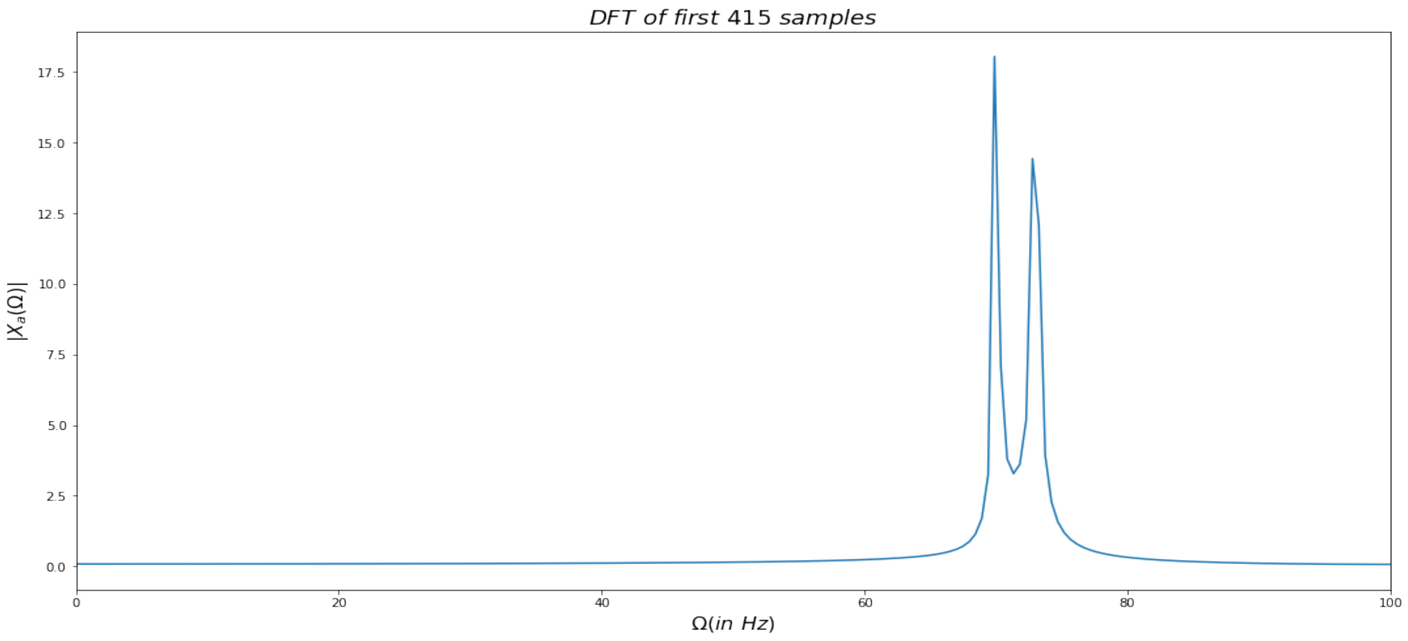
(Subproblem - (i))

For 215 Samples :



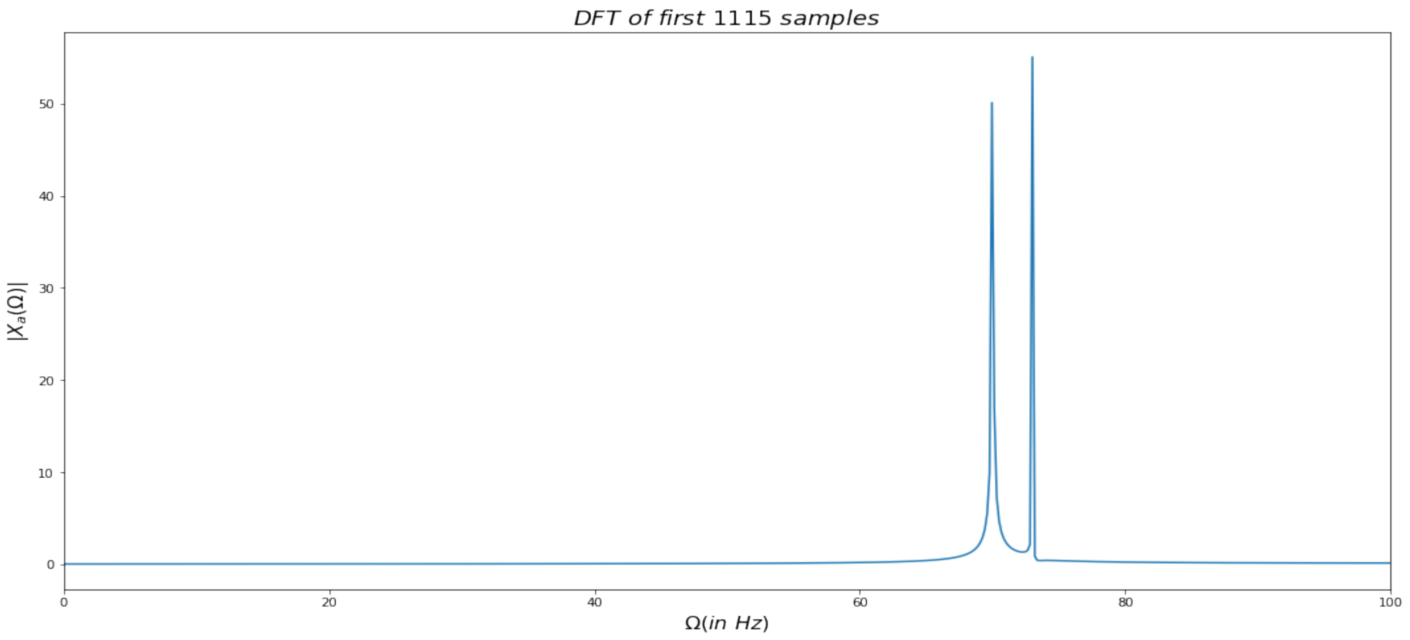
(Subproblem - (ii))

For 415 Samples :



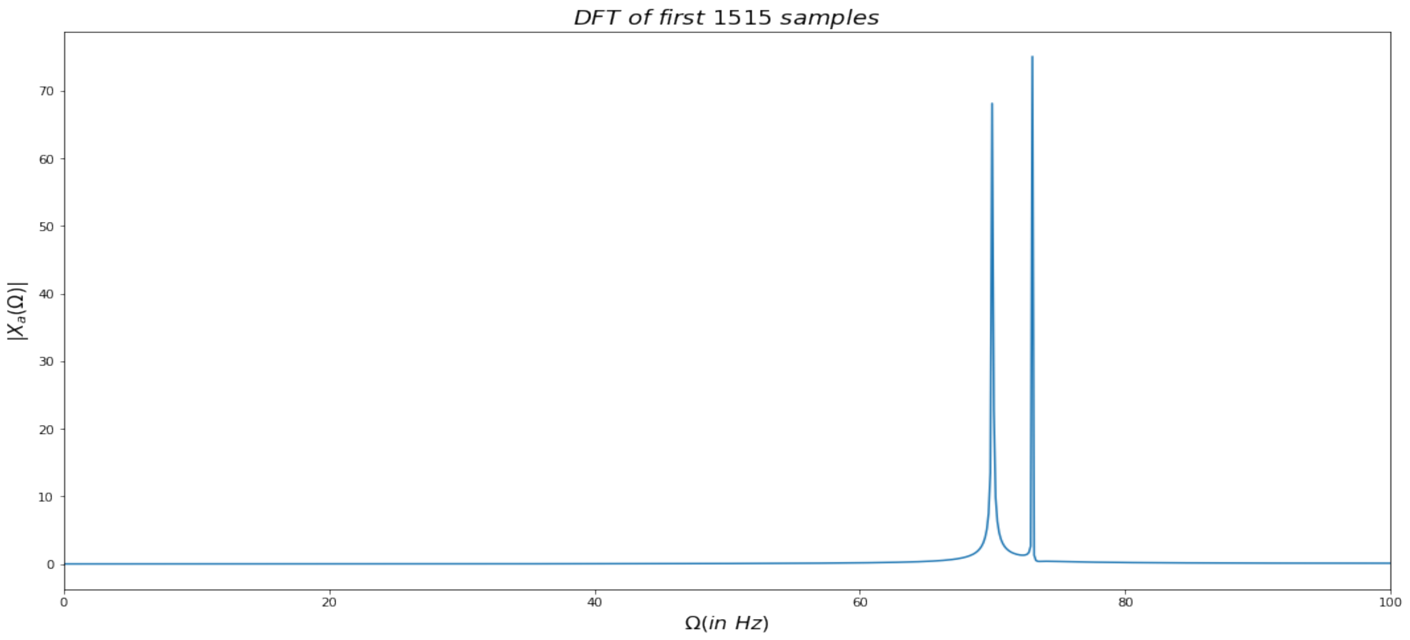
(Subproblem - (iii))

For 1115 Samples :



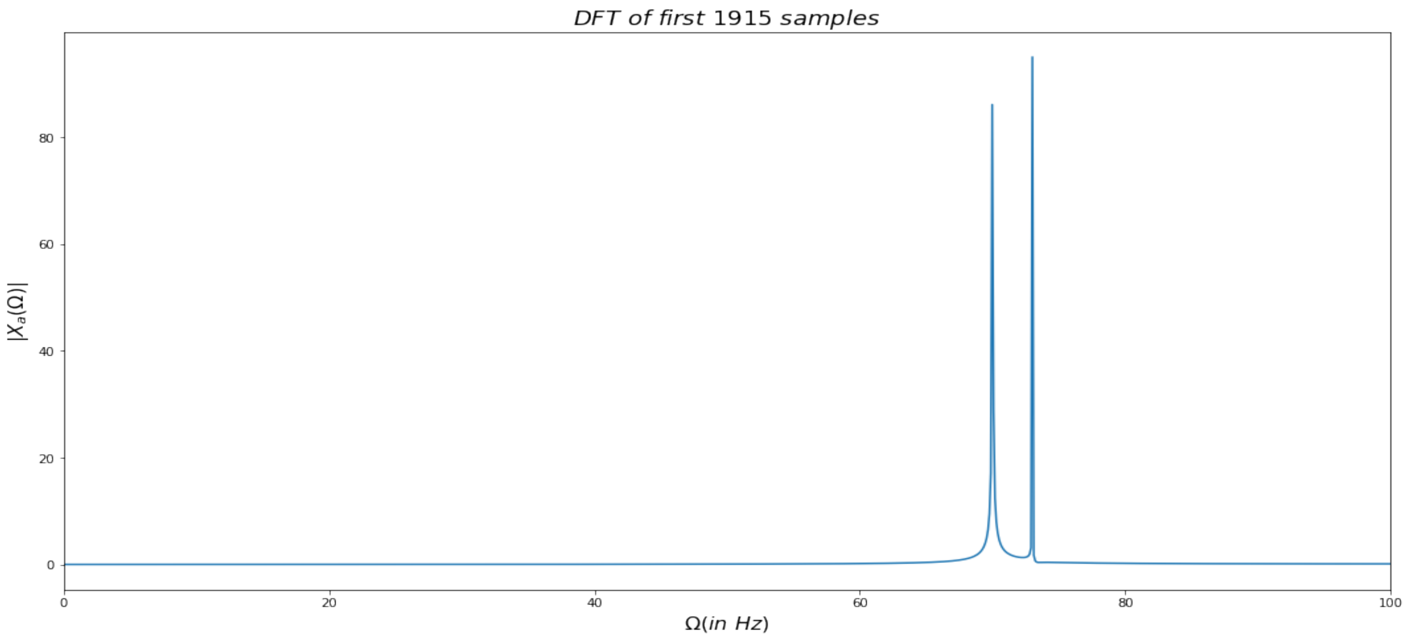
(Subproblem - (iv))

For 1515 Samples :



(Subproblem - (v))

For 1915 Samples :



- As we can see from above DFT plots, we can see many spectral leaks.
- One observation we can make is that more the samples in time domain, better the resolution in frequency domain.
- The code related to this question can be found here

Question 3 - Resolution of DFT with windowing

The problem asks us to generate a signal

$$x_a(t) = 0.1\sin(A\pi t) + \cos(B\pi t)$$

and repeat question 2 but by windowing the time-domain signal using $HanningWindow$ (because $\alpha = 2$) and also compare the results with the regular signals of question 2.

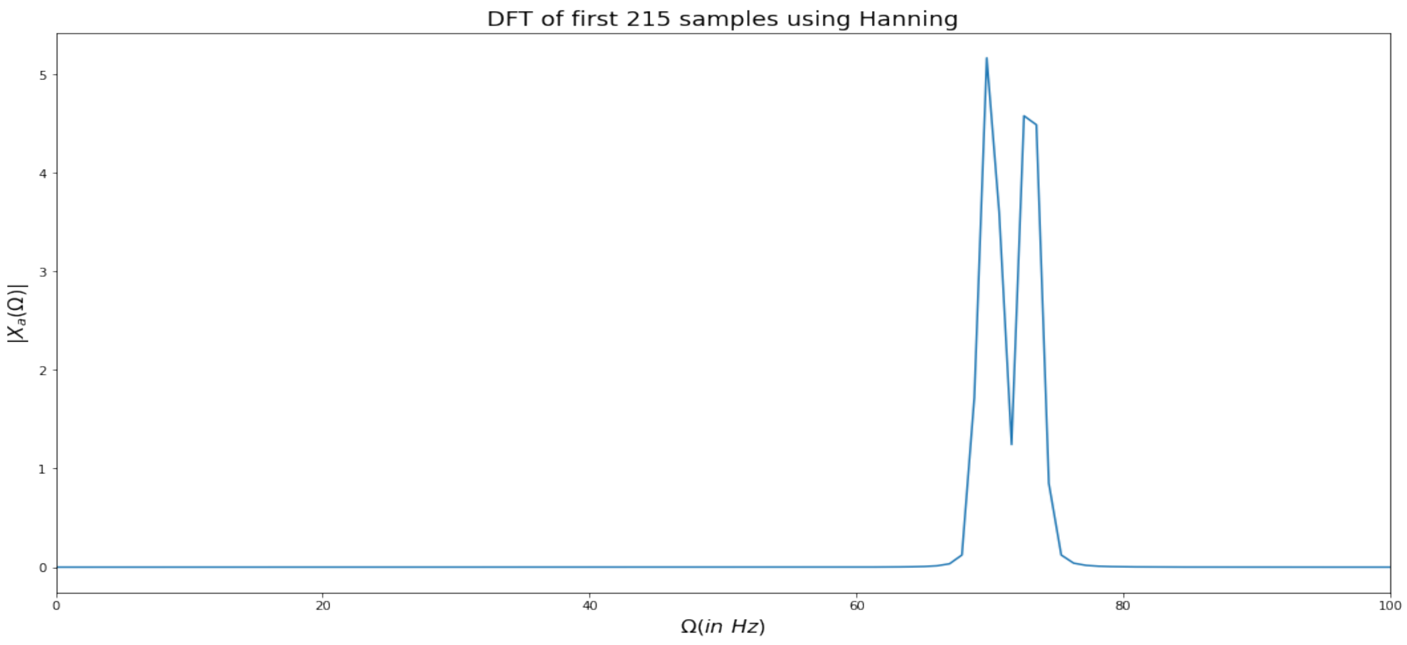
- To apply $HanningWindow$, I've used the hanning module present in NumPy.
- The code to this question can be found here

I observed that the **Effects of Windowing** are as following :

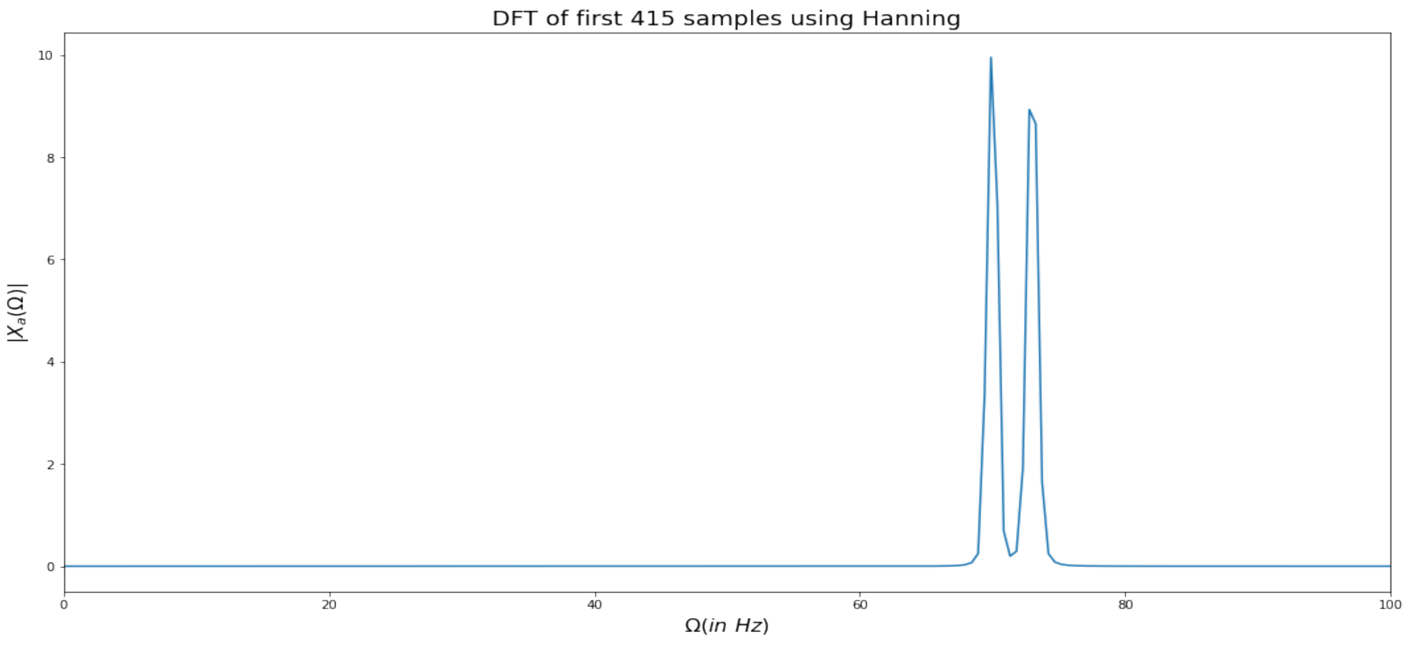
- Frequency domain plot gets smoothed out
- Spectral component gets stretched over all frequency components
- This produces convolution of signal with the window in frequency domain
- The peaks indicate the different frequencies the windowed or convoluted signal is made up of.

The plots of the different signals that were windowed are as follows :

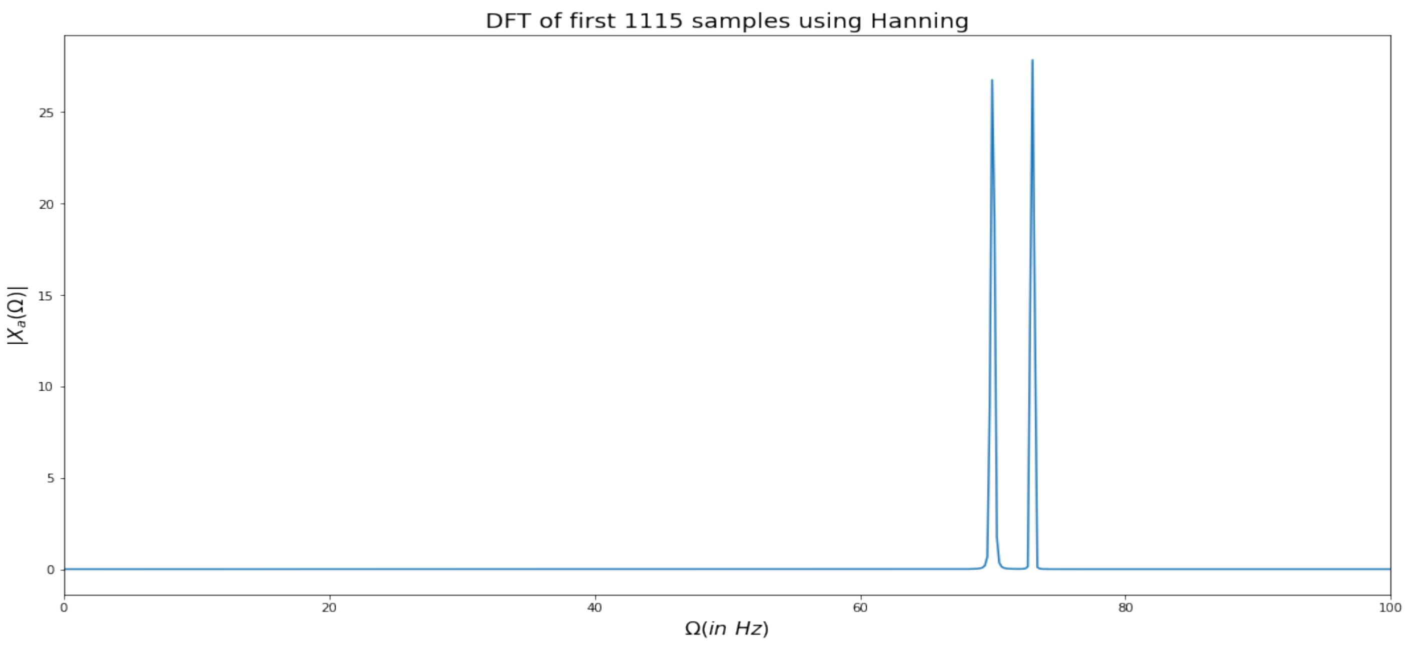
For 215 Samples :



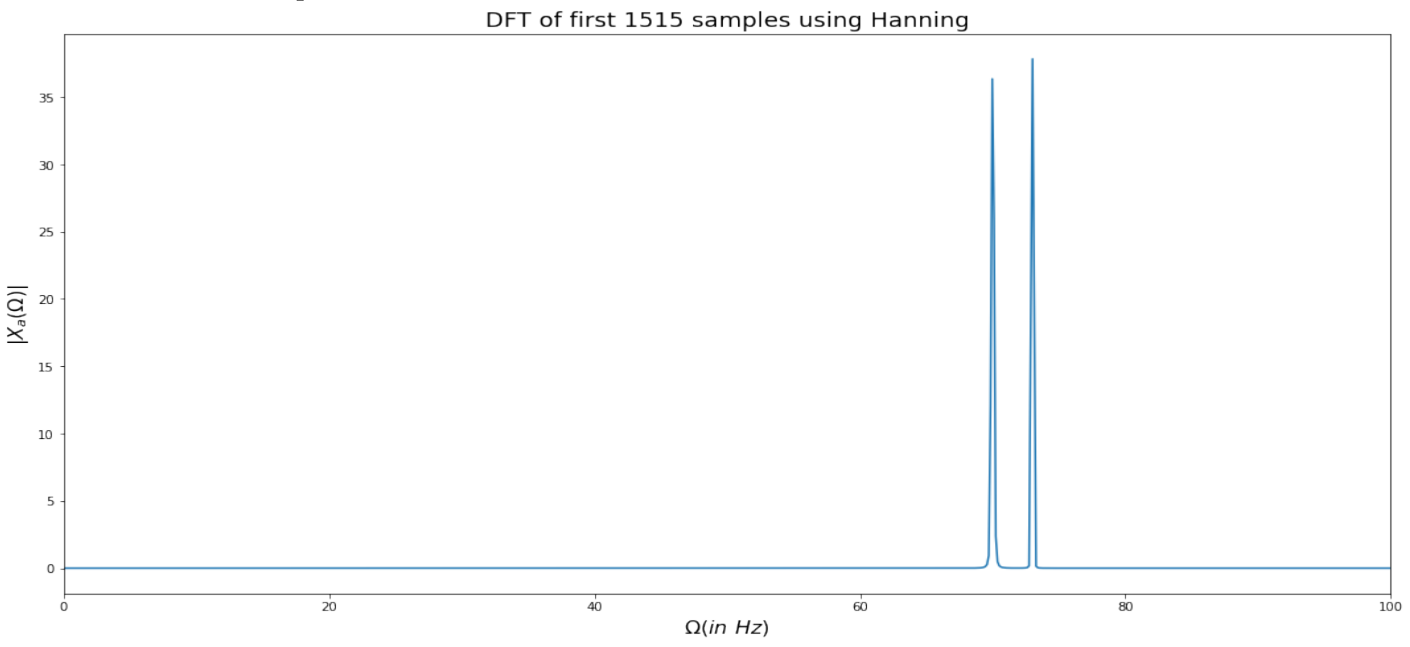
For 415 Samples :



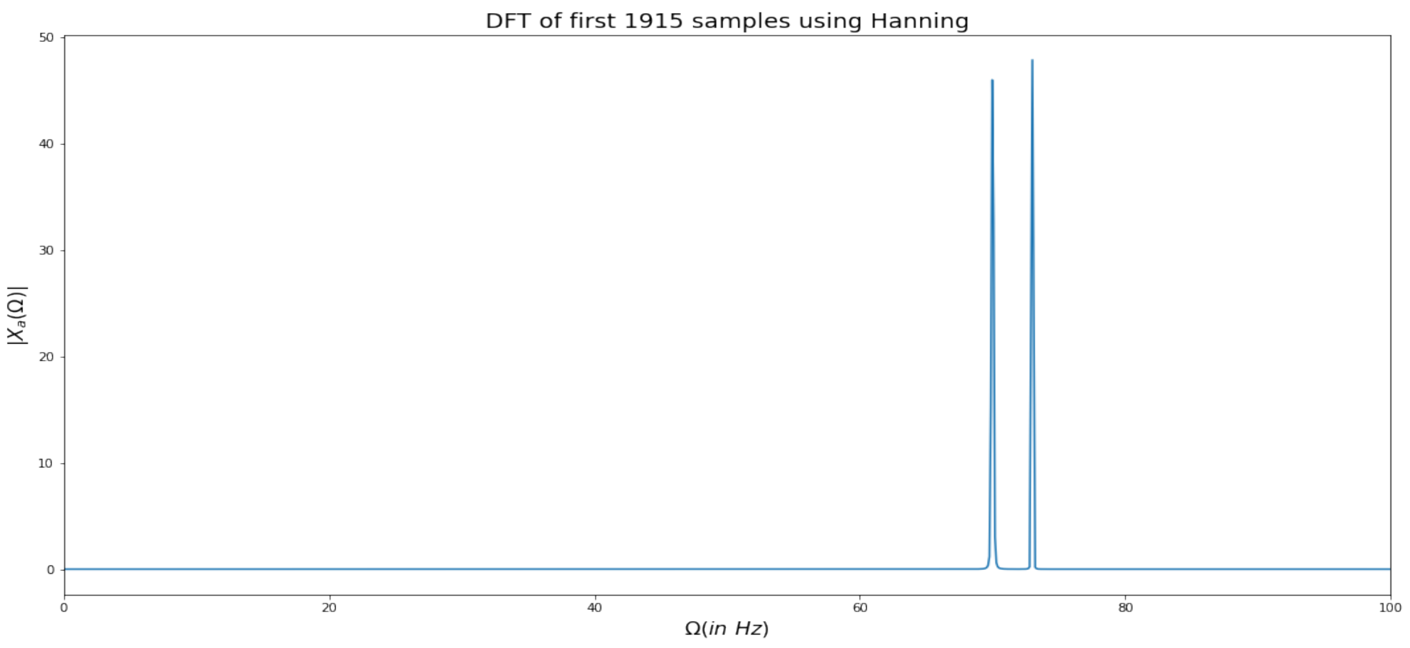
For 1115 Samples :



For 1515 Samples :



For 1915 Samples :



Question 4 - Frequency estimation using windowing

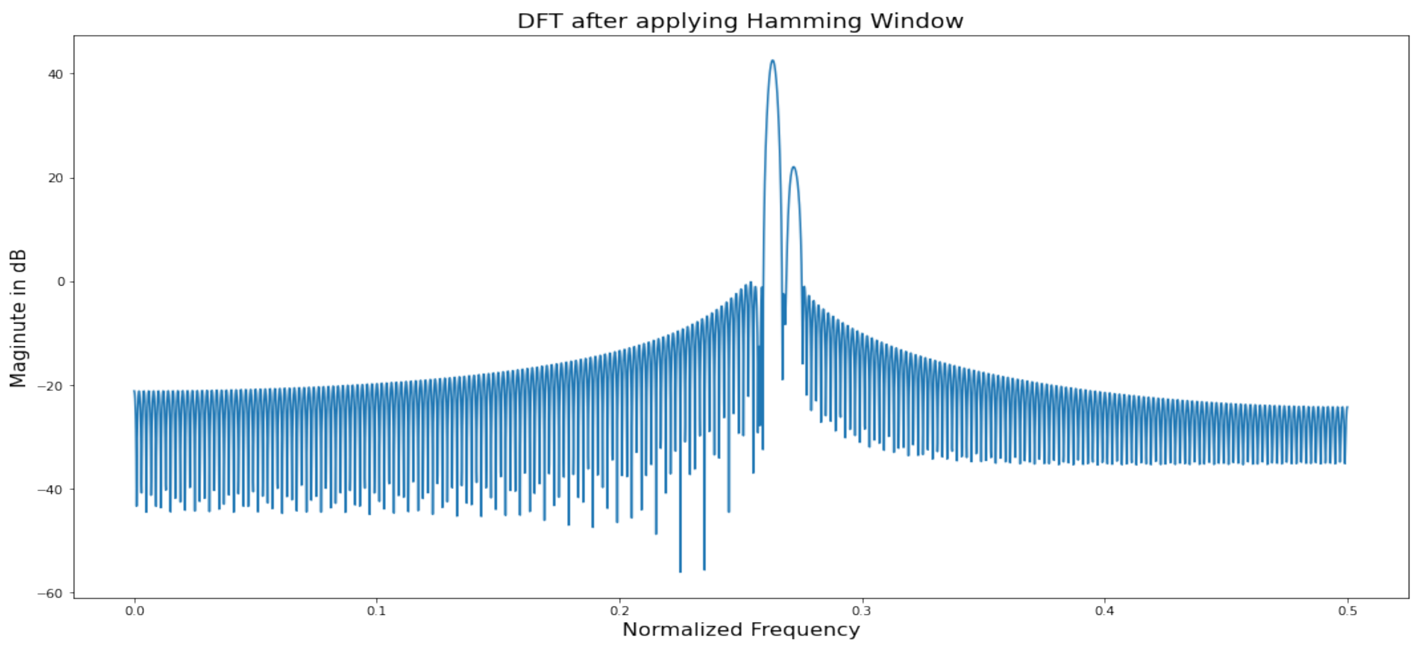
This question asked us to estimate the two frequency components using different windowing techniques.

(Subproblem - 1)

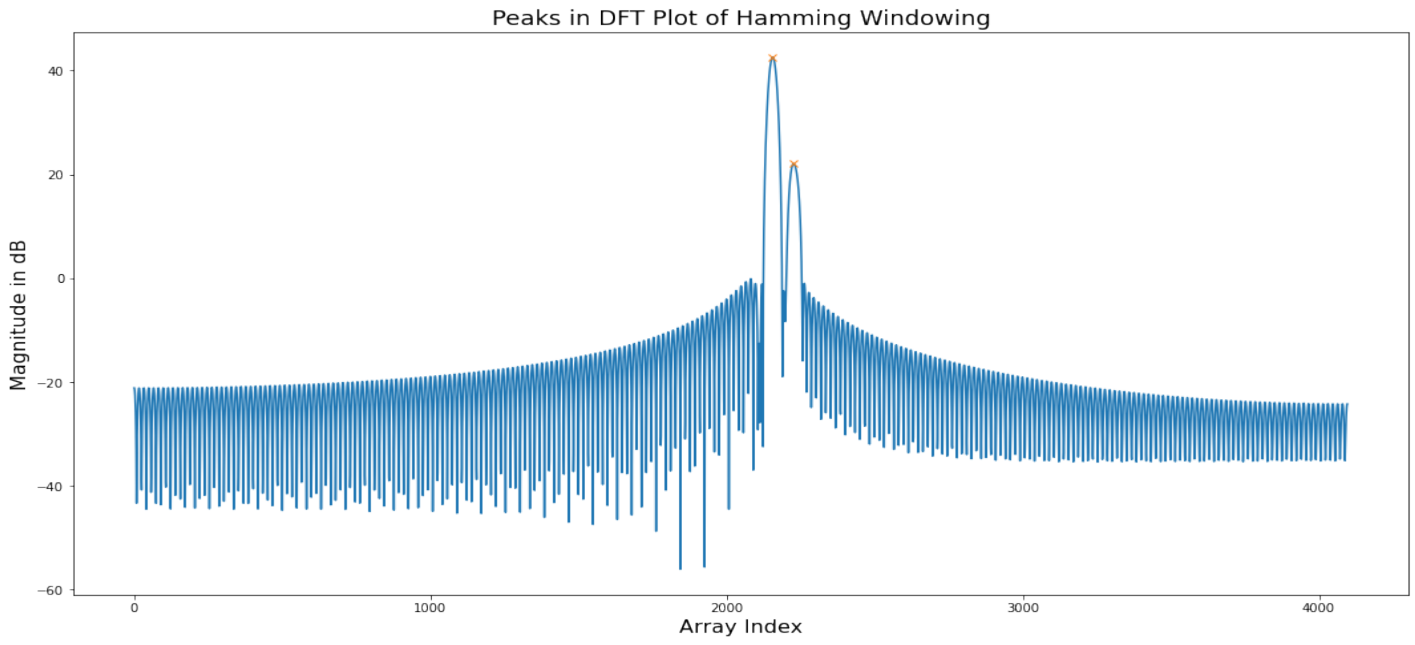
In this question, we are to use the $HammingWindow$ technique to estimate the two frequency components from the given 500 sampled dual tone input file.

- To apply $HammingWindow$, I've used the hamming module present in SciPy.
- Since zero padding does not affect the spectral resolution, while computing the DFT, I have zero-padded the signal for better interpolation.
- The code to this question can be found here

The DFT plot in decibels after applying Hamming windowing is :



The peaks plot looks like :

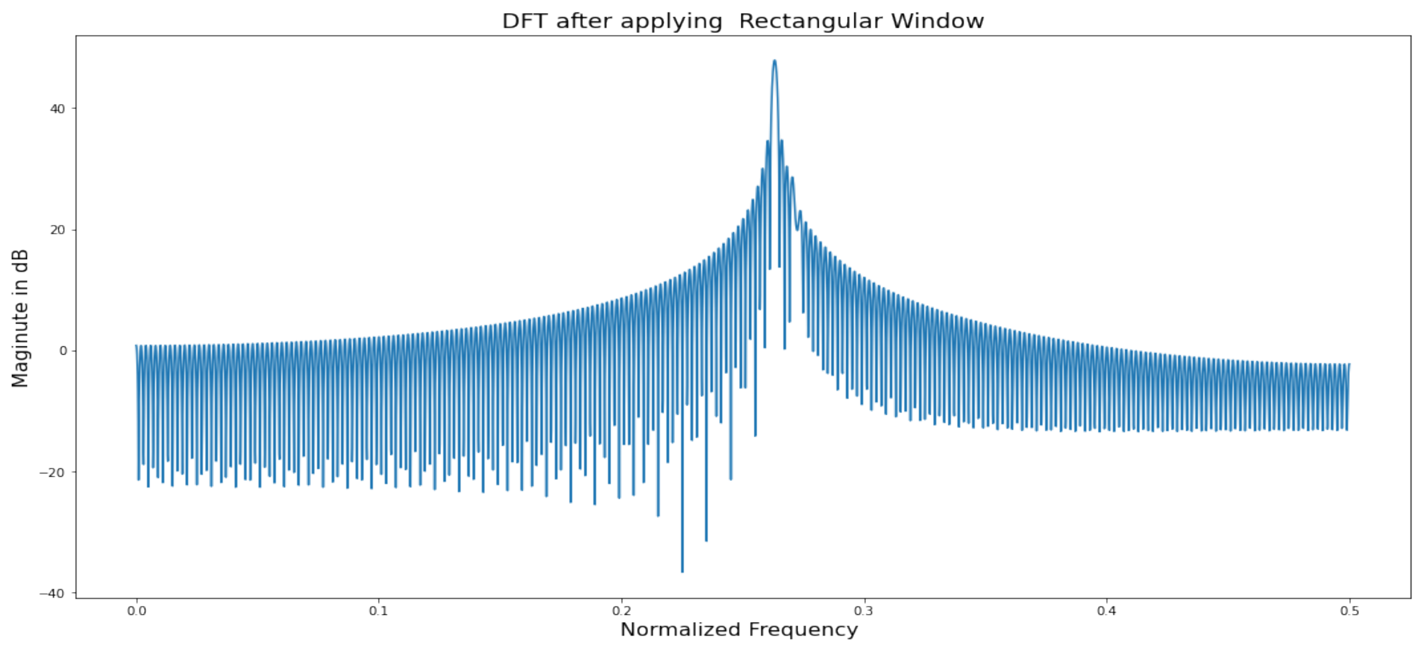


(Subproblem - 2)

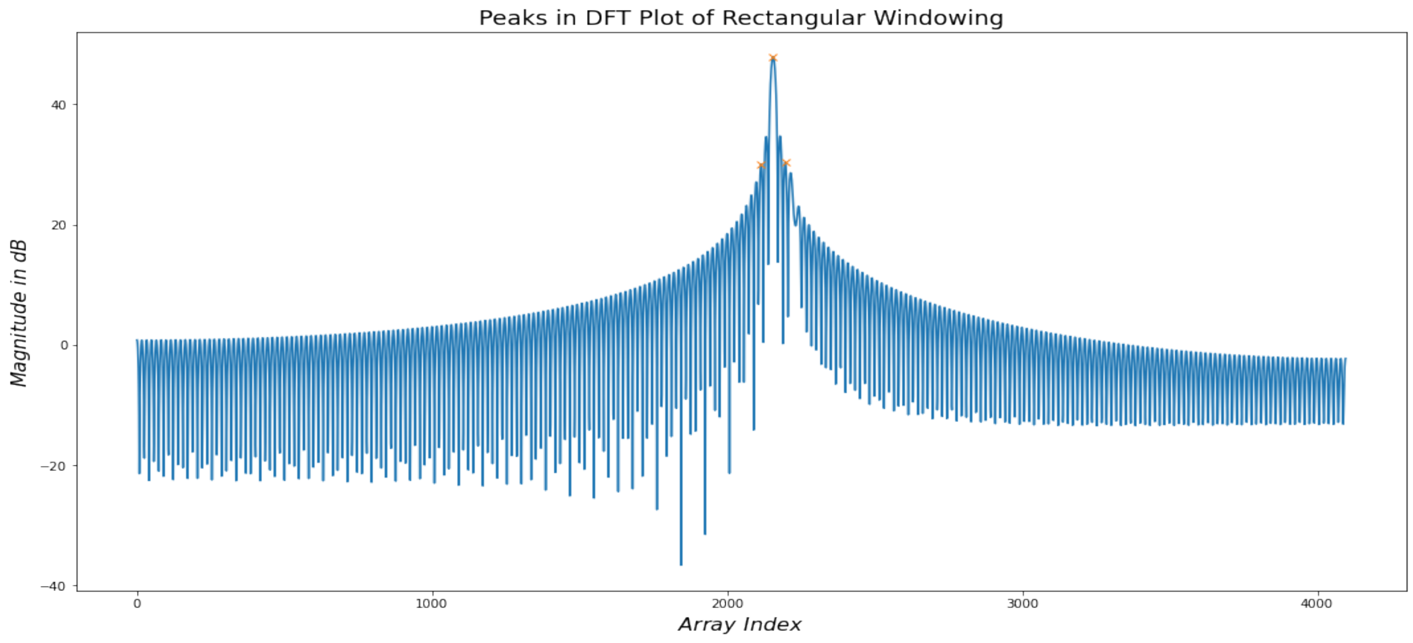
In this question, we are to use the $Rectangular Window$ technique to estimate the two frequency components from the given 500 sampled dual tone input file.

- To apply $Rectangular Window$, I've used the boxcar module present in SciPy.
- Since zero padding does not affect the spectral resolution, while computing the DFT, I have zero-padded the signal for better interpolation.
- The code to this question can be found here

The DFT plot in decibels after applying Rectangular windowing is :



The peaks plot looks like :



1 Appendix

- Note : I have used $\alpha = 2$ because my registration number is 191910. Since $\alpha = 1 +$ mod(910,3) $= 1 + 1$
- The link to all the code is here
- The link to all input and output files are here
- The Github repo to all code and previous experiments can be found here