JSS MAHAVIDYAPEETHA

JSS SCIENCE AND TECHNOLOGY UNIVERSITY

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

- Constituent College of JSS Science and Technology University
- Approved by A.I.C.T.E
- Governed by the Grant-in-Aid Rules of Government of Karnataka
- Identified as lead institution for World Bank Assistance under TEQIP Scheme

---

Department of Electronics and Communication

Digital Signal Processing Lab

Lab Manual

V Semester

Lab in-charge:

Prof. B A Sujathakumari
Associate Professor

Prof. R Shashidhar
Assistant Professor

Electronics and Communication Engineering

Sri Jayachamarajendra College of Engineering
JSS Science and Technology University
Mysore 570006

Preface

This laboratory manual is prepared by the Department of Electronics and communication engineering of Sri Jayachamarajendra College of Engineering for Digital signal processing Laboratory. This lab manual is vital for students to complement theory with practical software and hardware applications in their curriculum. This lab manual can also be used as instructional book by staff and instructors to assist in performing and understanding the experiments.

The course aims at practical experience with the simulation and application of basic signal processing algorithms, using standardized environments such as MATLAB and general-purpose DSP development kit such as TMS320C6713.

Experiments cover fundamental concepts of digital signal processing like sampling and aliasing, computation of discrete Fourier transform, verification of DFT properties, fast transforms, digital filter design and implementation, adaptive filtering, sampling-rate conversion and multi-rate processing.

The lab consists of 9 experiments of which 7 experiments are based on MATLAB simulation and two experiments on real time implementation using TMS Digital Signal Processing kit. Each lab session is of 3 hours duration. This lab manual also provides a uniform evaluation scheme to be followed by the staff members handling this lab.

List of Experiments

Lab No.	Experiment	Page No.
	Software Experiment Using MATLAB	
1.	Explore Digital Signal Processing Virtual Laboratory of Department of Electronics and Electrical Communication Engineering Indian Institute of Technology, Kharagpur http://www.digital.iitkgp.ernet.in/dsp/expts/index.php	
2.	a) Write a MATLAB code to illustrate the Nyquist sampling theorem. The program should illustrate the effects the sampling the signal at • Exactly the folding frequency • Frequency less than the folding frequency • Frequency greater than the folding frequency Plot the magnitude spectrum for all the above said cases b) Write a MATLAB code to compute the DTFT and DFT of a sequence x (n). Also plot the magnitude spectrum of both DTFT and DFT and provide the inference on the basis of results obtained. Further compute the IDTFT and IDFT.	
3.	Write a MATLAB code to verify the following properties of DFT a) Linearity b) Periodicity c) Circular shift and Circular symmetry of a sequence d) Symmetry property e) Circular convolution and multiplication of two sequences f) Time reversal of a sequence g) Circular time shift and Circular frequency shift of a sequence h) Parseval's theorem	
4.	a) Write a MATLAB code to compute the DFT of a sequence x (n) using DIT and DIF algorithm. Also indicate the speed improvement factor in calculating the DFT of a sequence using direct computation and FFT algorithm (Use the same sequence as used in Program 2). Further compute the IDFT using IDIT and IDIF algorithm. b) Write a MATLAB code to verify the Low pass and High Pass FIR linear phase filter design using Hamming and Hanning windows (with inbuilt	

	and without using inbuilt commands). Plot the magnitude and phase response. Also, Provide the inference on the basis of results obtained for the set of specifications. (To design should be verified by convolving the input signal with the designed filter coefficients)	
5.	a) Write a MATLAB code to verify the Band pass and Band reject FIR linear phase filter design using Hamming and Hanning windows (with inbuilt and without using inbuilt commands). Plot the magnitude and phase response. Also, Provide the inference on the basis of results obtained for the set of specifications. b) Write a MATLAB code to verify the Low pass Butterworth IIR filter design using bilinear transformation (BLT) method and Impulse Invariant Technique (IIT) method. c) Write a MATLAB code to implement the Low pass Chebyshev (Type 1) IIR filter design using bilinear transformation (BLT) method and Impulse Invariant Technique (IIT) method.	
6.	a) Write a MATLAB code to illustrate the effect of Decimation and Interpolation by an integer factor. Plot the magnitude spectrum. Design the necessary filter to overcome aliasing and image frequencies after decimating and interpolating the signal respectively. b) Write a MATLAB code to illustrate the effect of sampling rate conversion by a non-integer factor. Plot the magnitude spectrum. Design the necessary filter to overcome aliasing and image frequencies.	
7.	Read the data file named ecg2x60.dat from **http://people.ucalgary.ca/~ranga/enel563/SIGNAL_DATA_FILES/** that is corrupted with the 60Hz noise component. Write a MATLAB code to remove this 60Hz noise component from the signal using Notch filter and LMS adaptive filter. Plot the magnitude spectrum of the signal filtered using both Notch filter and LMS adaptive filter and provide the inference on the basis of results obtained.	
Hardware Experiment Using TMS320C6713 DSP Kit		
8.	a) Write a C code to obtain the impulse response of a given system and implement the same on TMS320C6713 DSK kit. b) Write a C code to compute the linear and circular convolution and implement the same on TMS320C6713 DSK kit.	
9.	a) Write a C code to compute the cross correlation and auto correlation and implement the same on TMS320C6713 DSK kit. b) Write a C code to compute N-point DFT and IDFT of a sequence and implement the same on TMS320C6713 DSK kit.	

Course Outcome

CO1: Verify signal processing concepts of time domain and frequency domain using computer.

CO2: To design, test and simulate FIR and IIR filters on computer.

CO3: To design, test and simulate adaptive filters on computer.

CO4: To verify the concepts of Multi rate DSP on computer.

CO5: To verify the DSP concepts in real time using DSP Processor.

Evaluation Scheme

Sl.No	Scheme type	Weightage
1	Continuous Internal Evaluation	40%
2	Assessment-1	5%
3	Assessment-2	10%
4	Assessment-3	5%
5	Semester end test	40%

Continuous Internal Evaluation Scheme (CIE)

Sl.No	Evaluation Component	Duration	Marks
1	Preparedness	Continuous	10M
2	Implementation	Continuous	10M
3	Inference Drawn	Continuous	5M
4	Question Answer/Viva	Continuous	10M
5	Record	Continuous	5M

Assessment-1: 5 Marks [Quiz] (To be done before 5th week)

Assessment-2: 10 Marks [Mini project based on filtering] (To be done before 10th week)

Assessment-3: 5 Marks [Assignment based on DSP Kit] (To be done before SET)

Semester End test Evaluation Scheme (SET)

Sl.No	Evaluation Component	Marks
1	Program Writing (Two Programs to be asked: One based on MATLAB and One based on DSP Kit	10M
2	Implementation	20M
4	Question Answer/Viva	10M

Total Marks= (CIE+Assessment1+Assessment2+Assessment3+SET)/2

CIE Format

JSS MAHAVIDYAPEETHA: SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING, MYSURU

RECORD OF PERFORMANCE IN THE DSP LAB FOR CALCULATING CIE

Section		Batch		Group Number	
Staff in Charge		Day		Timings	

Sl. No.	USN	Name of the Student	AC1: Preparedness (10M) AC2: Implementation (10M) AC3: Viva (10M) AC4: Report Writing (5M) AC5: Result Interpretation (5M) T: Total	EC59L
1.				
2.				
3.				
4.				

Date	Experiment	Student-1						Student-2						Student-3						Student-4					
		AC 1	AC 2	AC 3	AC 4	AC 5	T	AC 1	AC 2	AC 3	AC 4	AC 5	T	AC 1	AC 2	AC 3	AC 4	AC 5	T	AC 1	AC 2	AC 3	AC 4	AC 5	T
	Remarks:																								
	Remarks:																								
	Remarks:																								
	Remarks:																								

Experiment-1

Explore Digital Signal Processing Virtual Laboratory of Department of Electronics and Electrical Communication Engineering Indian Institute of Technology, Kharagpur

http://www.digital.iitkgp.ernet.in/dsp/expts/index.php

Outcome: This experiment is to make student understand what virtual laboratory is and make them comfortable with the basic DSP concepts. The content of this website aims to provide a virtual laboratory platform for undergraduate Engineering students studying the course of Digital Signal Processing. The experiments designed will be two paced. For the relatively weaker students, it will dwell more on analysis part. For relatively stronger students, more challenging synthesis/design related experiments is made available.

Within each experiment, there will be many sub-experiments.

1. Study of sampling theorem, effect of under sampling.
2. Study of Quantization of continuous-amplitude, discrete-time analog signals.
3. Study of different types of Companding Techniques.
4. Study of properties of Linear time-invariant (LTI) system.
5. Study of convolution: series and parallel system.
6. Study of Discrete Fourier Transform (DFT) and its inverse.
7. Study of Transform domain properties and its use.
8. Study of FIR filter design using window method: Lowpass and highpass filter.
9. Study of FIR filter design using window method: Bandpass and Bandstop filter.
10. Study of Infinite Impulse Response (IIR) filter.

Experiment-2

a) Write a MATLAB code to illustrate the Nyquist sampling theorem. The program should illustrate the effects the sampling the signal at

- Frequency greater than or equal to the folding frequency
- Frequency less than the folding frequency
- Frequency greater than the folding frequency

Plot the magnitude spectrum for all the above said cases

Outcome: The student will be able to practically verify the concept of sampling theorem and apply sampling theorem on any given continuous input signal in order to reconstruct the signals exactly from its samples.

Theory Background:

Sampling theorem is a fundamental bridge between continuous-time signals (often called "analog signals") and discrete-time signals (often called "digital signals"). It establishes a sufficient condition for a sample rate that permits a discrete sequence of *samples* to capture all the information from a continuous-time signal of finite bandwidth.

A bandlimited signal can be reconstructed exactly if it is sampled at a rate atleast twice the maximum frequency component in it - Nyquist

Alternatively, the maximum frequency content of a signal (f_{max}) should be less than or equal to the folding frequency ($f_{s/2}$)

$$\mathbf{f_{max} \leq f_{s/2}}$$

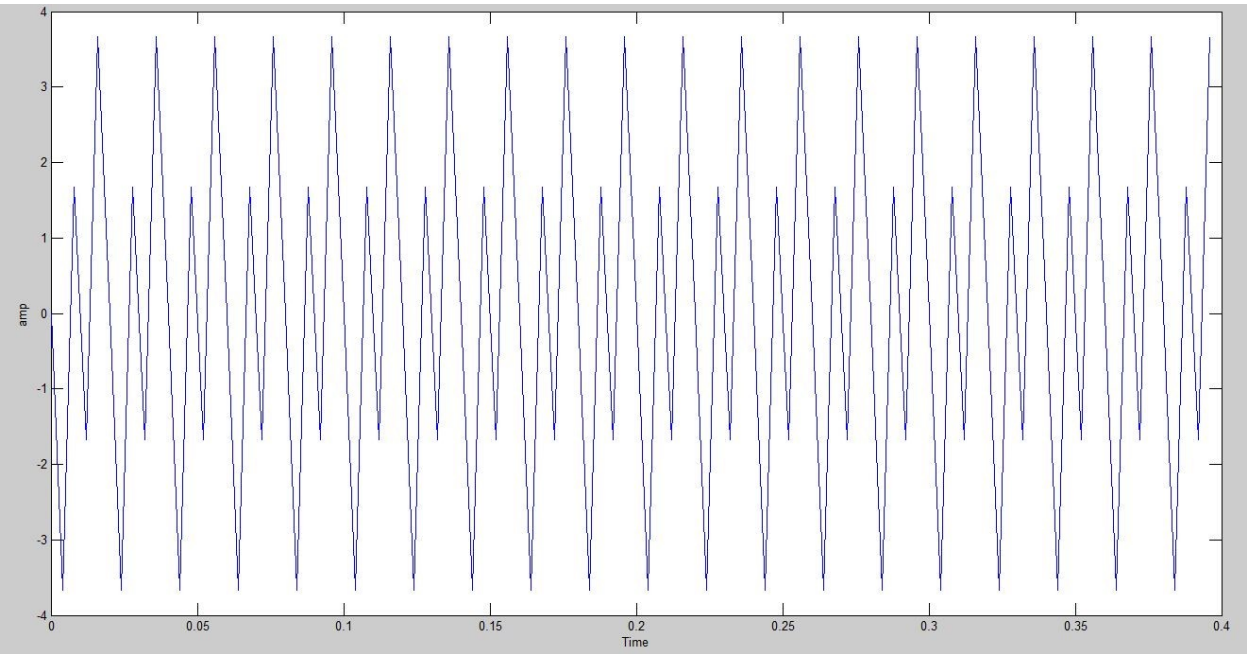
To be demonstrated:

Input:

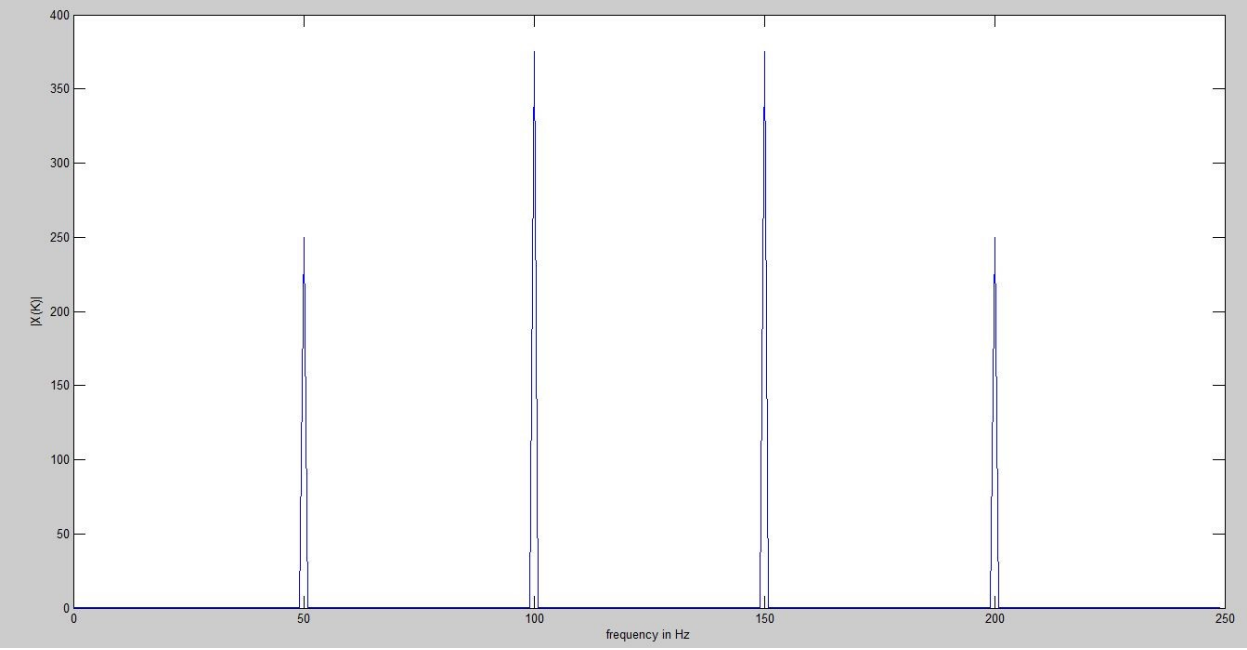
For the input signal **x (t) =2*sin (2*pi*f1*t) +3*sin (2*pi*f2*t) with f1=200Hz and f2=400Hz apply the sampling theorem to the signal to obtain each of the following outputs and provide your inference for each case.**

Case-1: Both the frequency components of the input signal are greater than folding frequency. (Case of aliasing)

Time domain plot

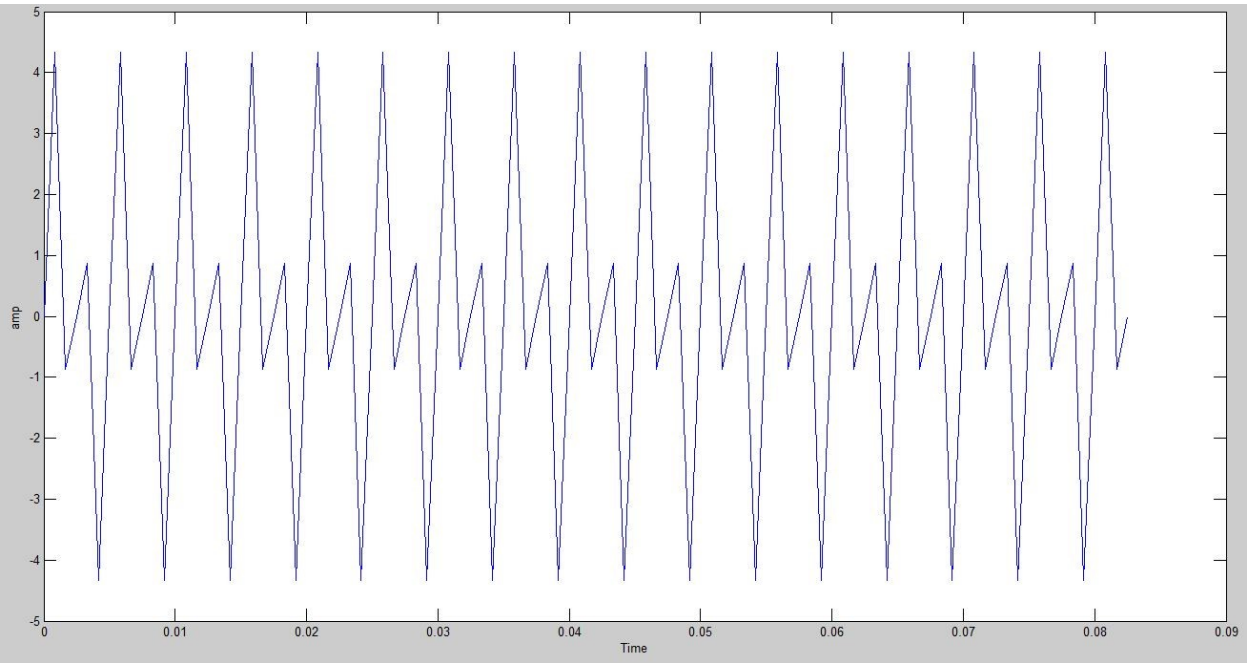


Frequency Plot

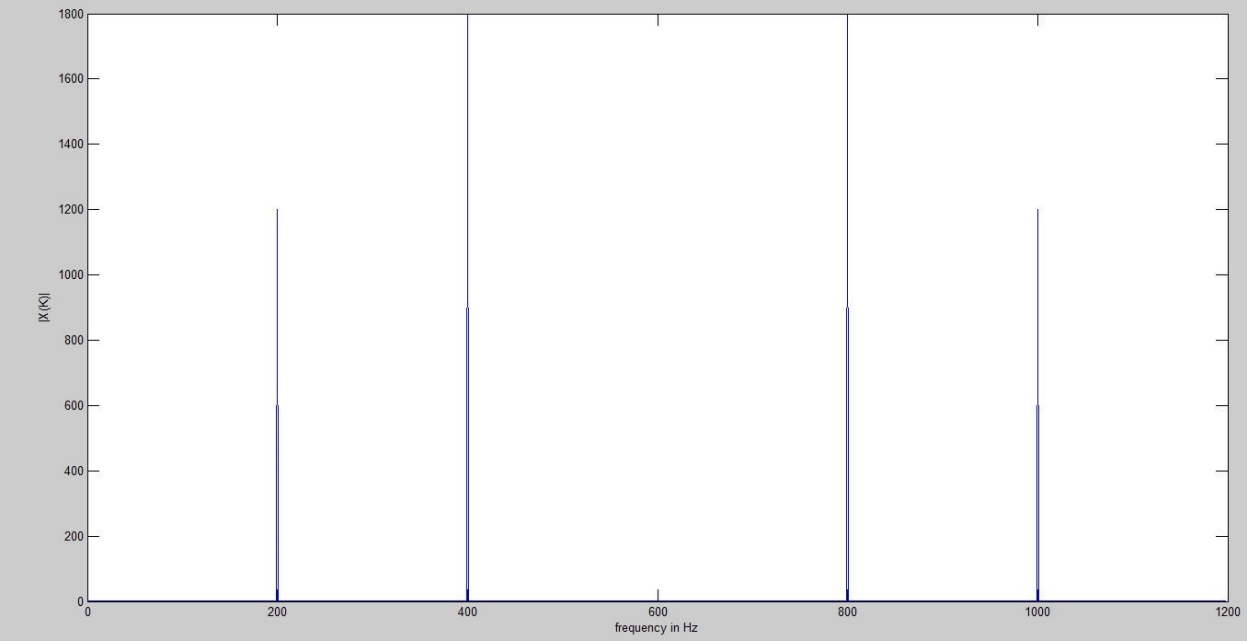


Case-2: Both the frequency components of the input signal are less than folding frequency. (Case of perfect reconstruction)

Time domain plot

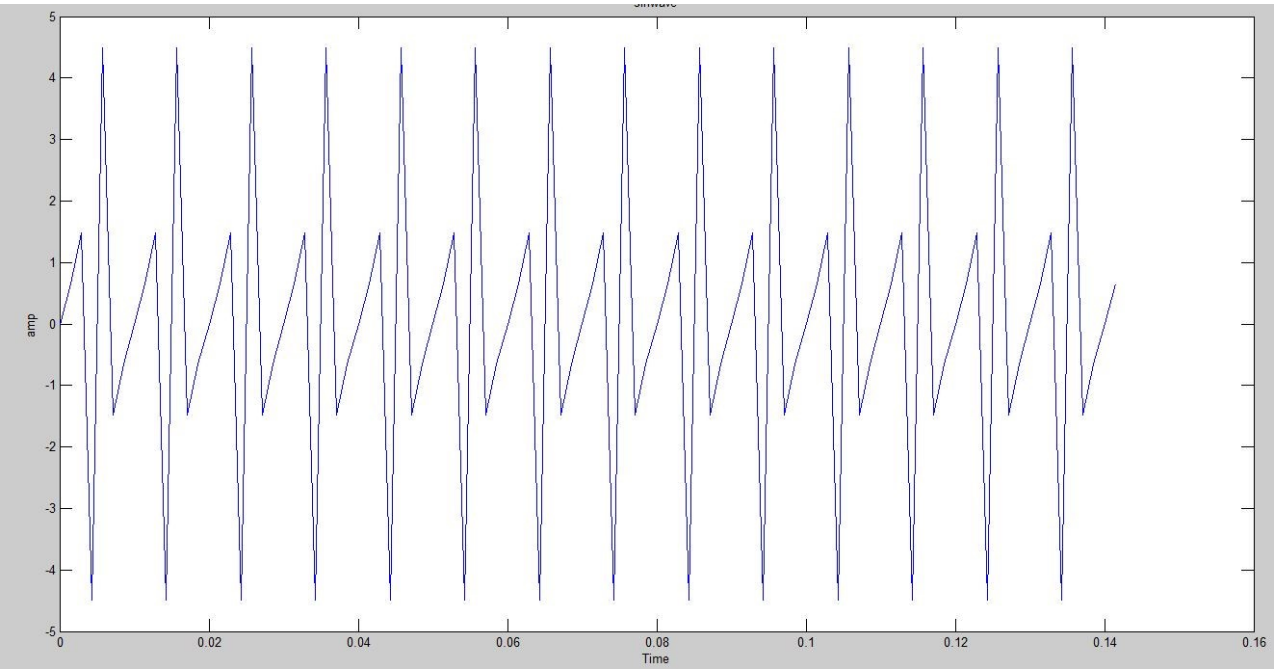


Frequency Plot

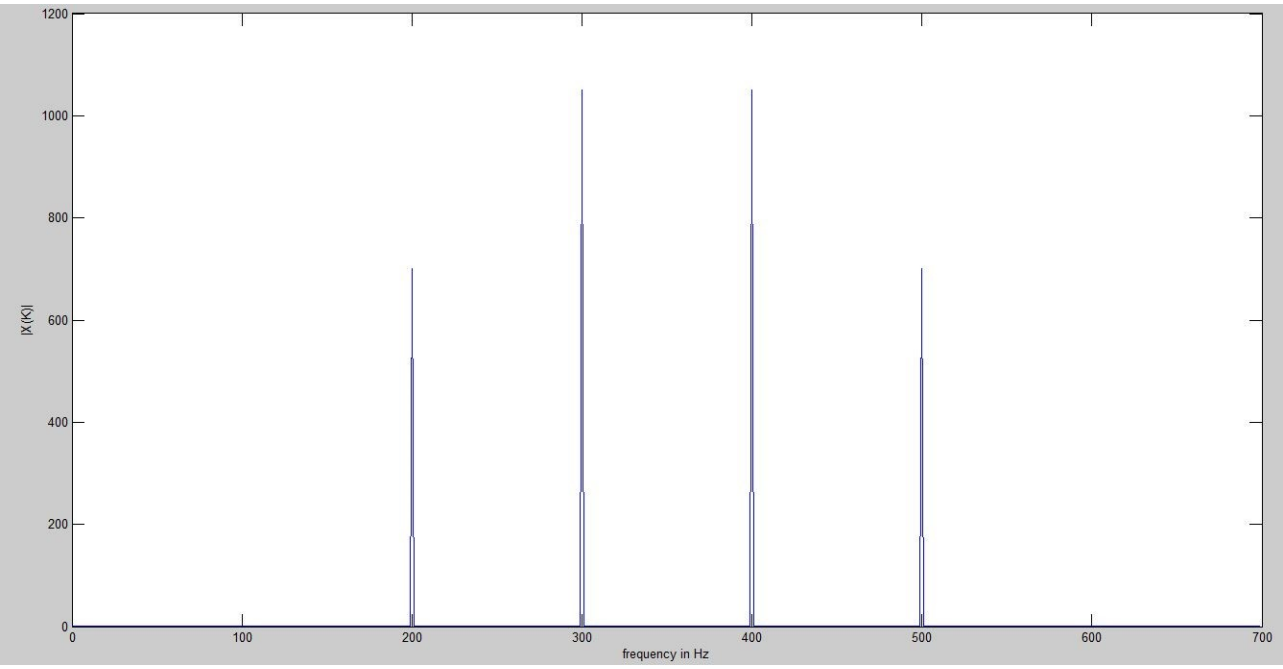


Case-3: One of the input frequency component is greater than folding frequency and one less than folding frequency.

Time domain plot



Frequency Plot



b) Write a MATLAB code to compute the DTFT and DFT of a sequence x (n). Also plot the magnitude spectrum of both DTFT and DFT and provide the inference on the basis of results obtained. Further compute the IDTFT and IDFT.

Outcome: The student will be able to practically compute the DFT and IDFT of a given sequence.

Theory Background:

Most time signals in practice are continuous and non-periodic, and their analytical expressions are not available in general. The spectrum of such a non-periodic and continuous signal can only be obtained numerically by a digital computer. To do so, the signal needs to be modified in two ways:

- First, we need to truncate the signal so that it has a finite time duration from 0 to T, with the underlying assumption that the signal repeats itself outside the interval 0<t<T, i.e., it becomes a periodic with period T. Correspondingly, its spectrum becomes discrete.
- Second, we need to sample the signal with some sampling frequency F so that it becomes discrete to be processed by a digital computer. Correspondingly, the spectrum of the signal becomes periodic.

The order can be reversed so that the continuous signal is sampled first and then truncated. In either case, only when the signal is both finite and discrete, can we apply the *discrete Fourier transform (DFT)* to find its spectrum, which is also discrete and finite.

Given a sequence of N samples $f(n)$, indexed by $n = 0..N\text{-}1$, the forward Discrete Fourier Transform (DFT) is defined as $F(k)$, where $k=0..N\text{-}1$:

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n)\, e^{-j2\pi kn/N}$$

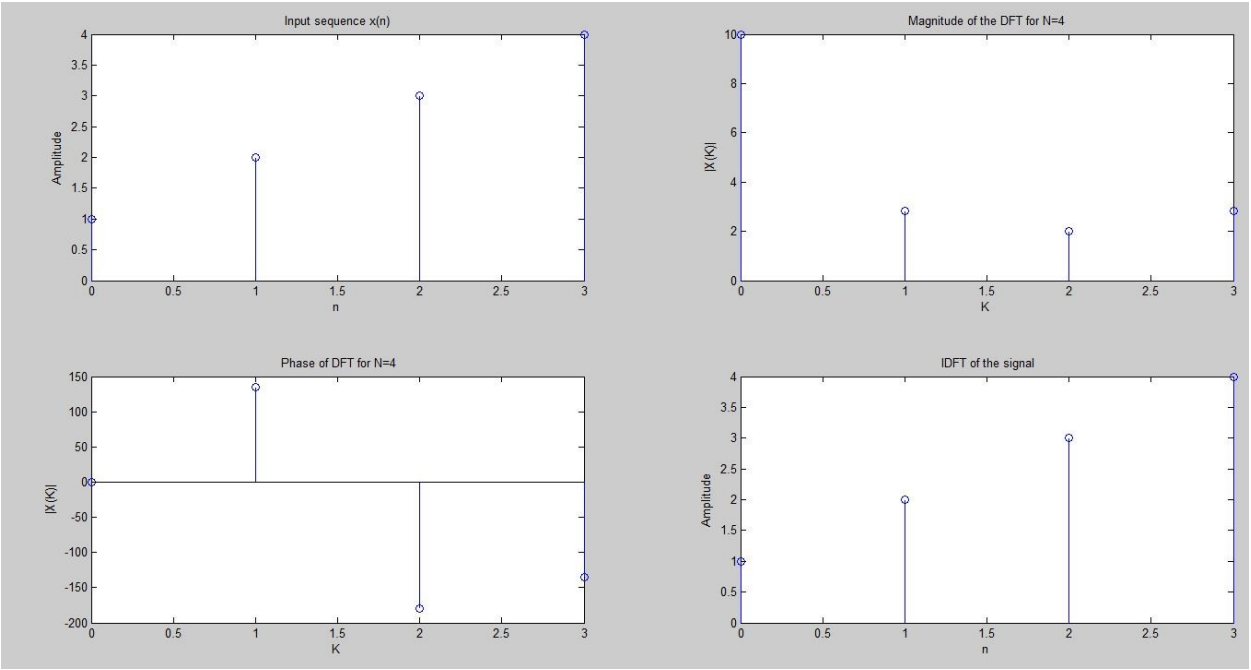
$F(k)$ are often called the 'Fourier Coefficients' or 'Harmonics'.

The sequence $f(n)$ can be calculated from $F(k)$ using the Inverse Discrete Fourier Transform (IDFT):

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k)\, e^{+j2\pi nk/N}$$

To be demonstrated:

Input: For the input sequence x(n) = [1 2 3 4] compute the DFT of the sequence and then obtain the magnitude and phase plot as shown below. Also compute the IDFT of the DFT sequence. Compute the DFT for N=4, 8 & 16 and provide your inference.



Experiment-3

Write a MATLAB code to verify the following properties of DFT

Outcome: The student will be able to appreciate the properties of the DFT and apply it to the practical problems.

a) Linearity

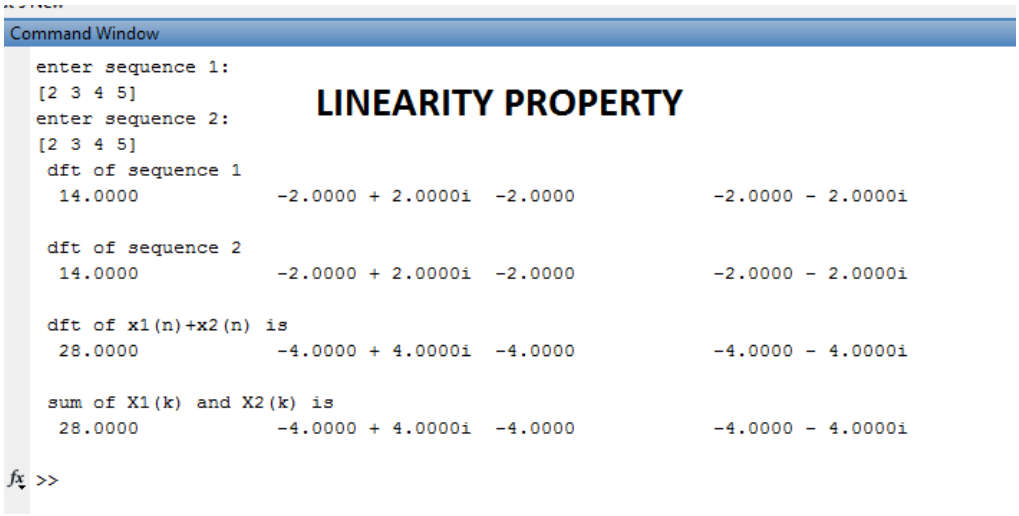
If x1(n) and x2(n) denote two periodic signals with period N and its Fourier Transform coefficients denoted by X1(k) and X2(k) respectively, then by linearity, the Fourier transform of x1(n)+x2(n) is given by X1(k)+X2(k).

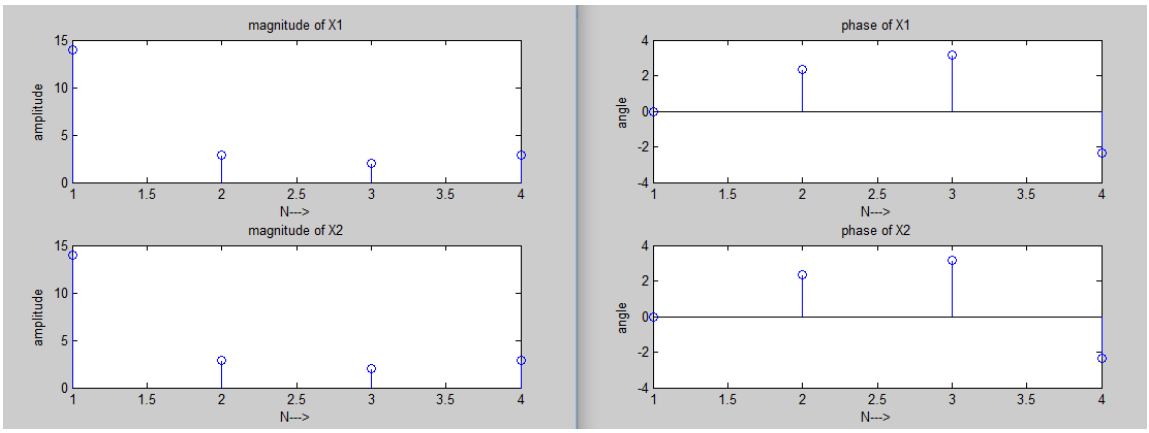
To demonstrate:

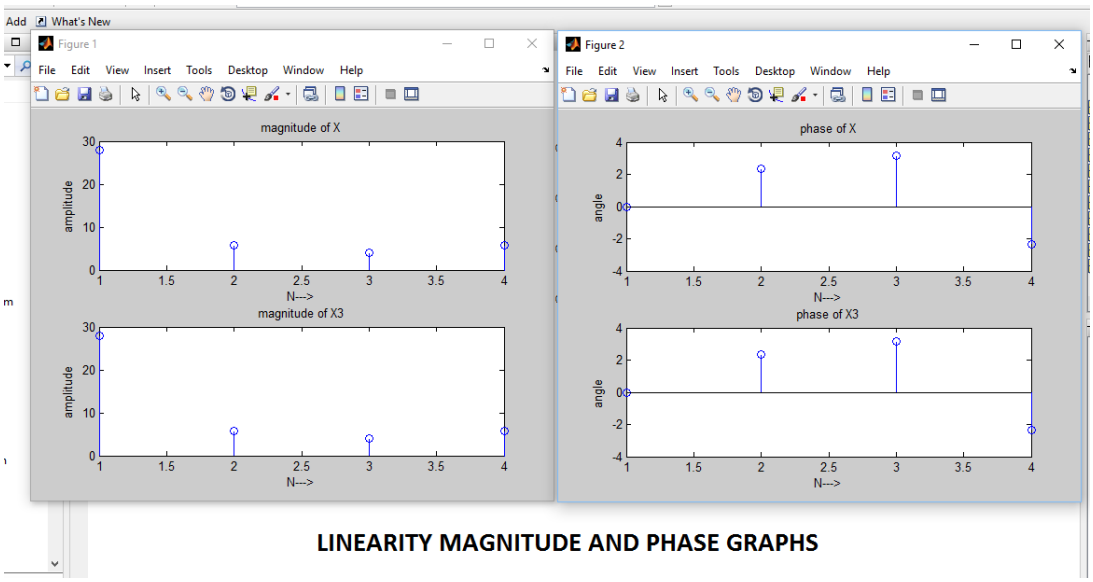
Verify the property of linearity for the given input sequences:

Sequence 1: [2 3 4 5]

Sequence 2: [2 3 4 5]







LINEARITY MAGNITUDE AND PHASE GRAPHS

b) Periodicity

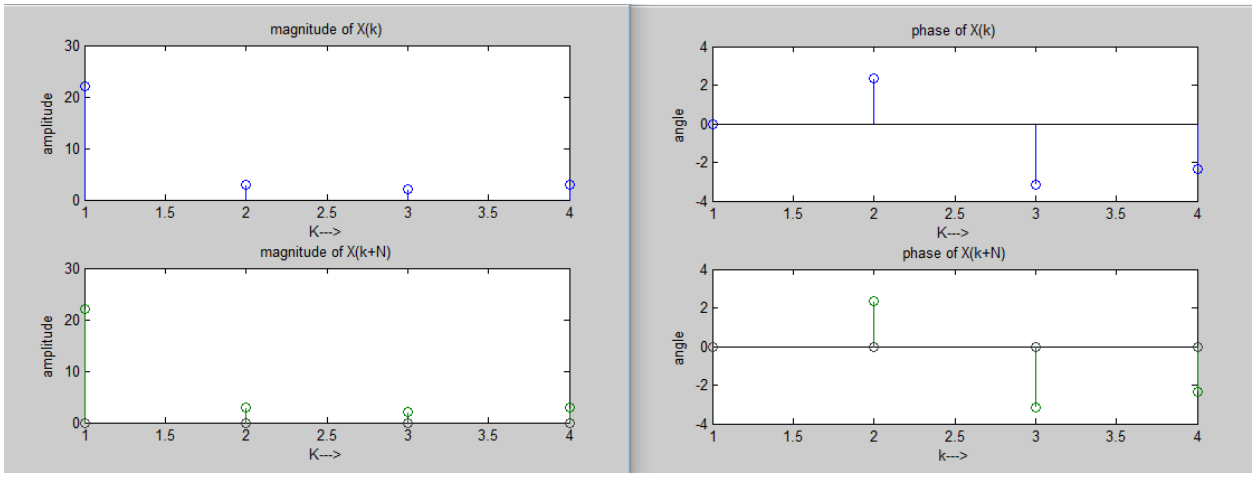
This property states that if X(k) is the DFT of the sequence x(n) then if x(n+N)=x(n) for all n, then ‘ X(k) = X(k + N) for all k.

To demonstrate:

Verify the property of periodicity for the given input sequence: [4 5 6 7]

```
periodicity property
enter the sequence [ 4 5 6 7]
 sequence you entered is x(n):
     4     5     6     7

 dft of x(n) is X(k):
  22.0000            -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 - 2.0000i

 X(N+k):
  Columns 1 through 6

        0           22.0000 + 0.0000i        0                0                0                0
        0           -2.0000 + 2.0000i        0                0                0                0
        0           -2.0000 - 0.0000i        0                0                0                0
        0           -2.0000 - 2.0000i        0                0                0                0

  Column 7

        0
        0
        0
        0
```

Periodicity property



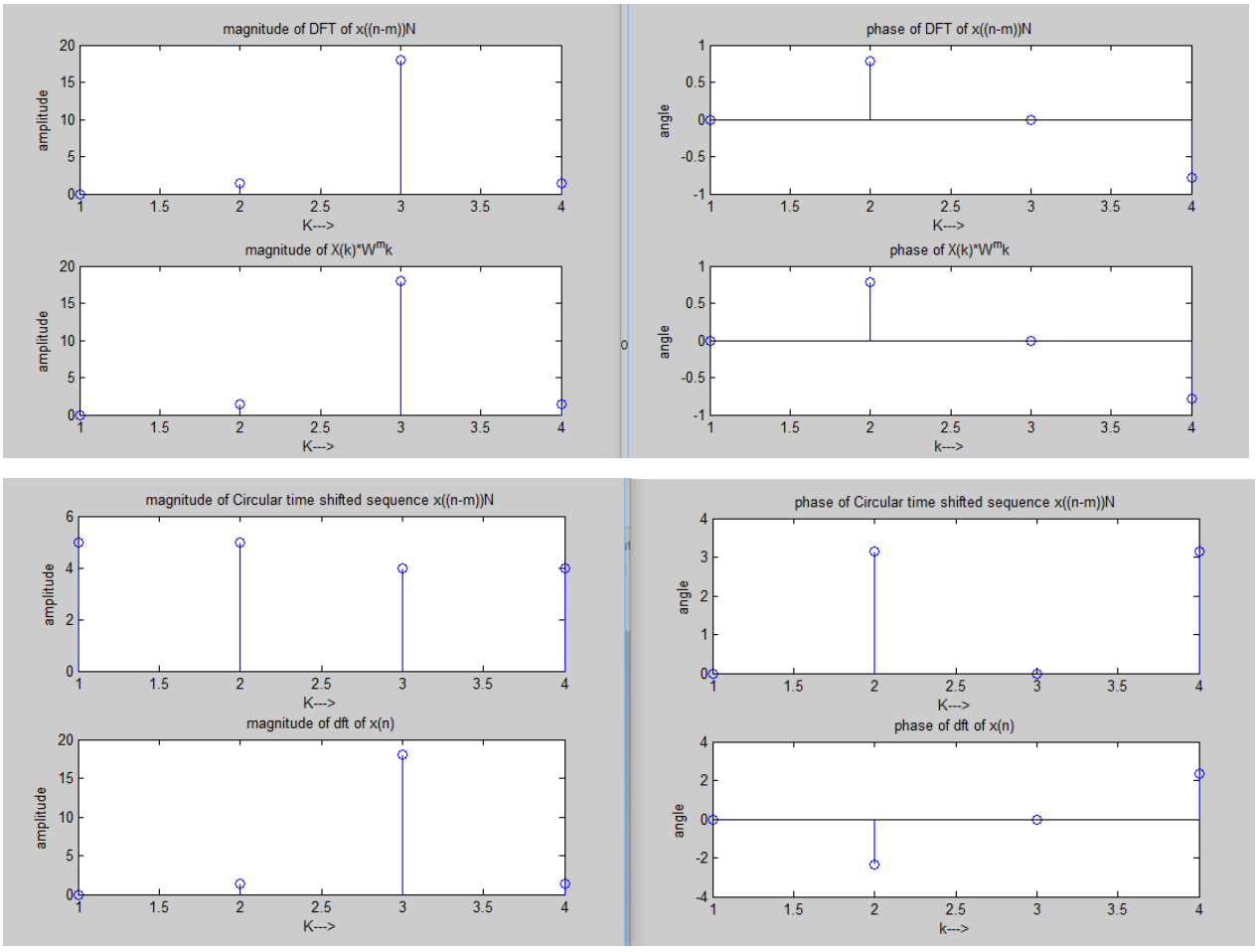
c) Circular time shift property of a sequence

Circular time shift property states that if X(k) is the DFT of the sequence x(n) then shifting the sequence circularly by 'm' samples is equivalent to multiplying its DFT by e^(-j*2*pi*k*m / N)

To demonstrate:

Verify the property of Circular time shift for the given input sequence: [4 -4 5 -5] with number of shifts equal to 2.

```
Enter the sequence
[4 -4 5 -5]
Enter the number of shifts
2
DFT of the input sequence x(n) is
         0            -1.0000 - 1.0000i  18.0000            -1.0000 + 1.0000i

Circular time shifted sequence x((n-m))N
     5    -5     4    -4

DFT of x((n-m))N
         0             1.0000 + 1.0000i  18.0000             1.0000 - 1.0000i

X(k)*W^mk
         0             1.0000 + 1.0000i  18.0000 + 0.0000i   1.0000 - 1.0000i

Thus Circular-Time Shift Property is verified
fx >>
```



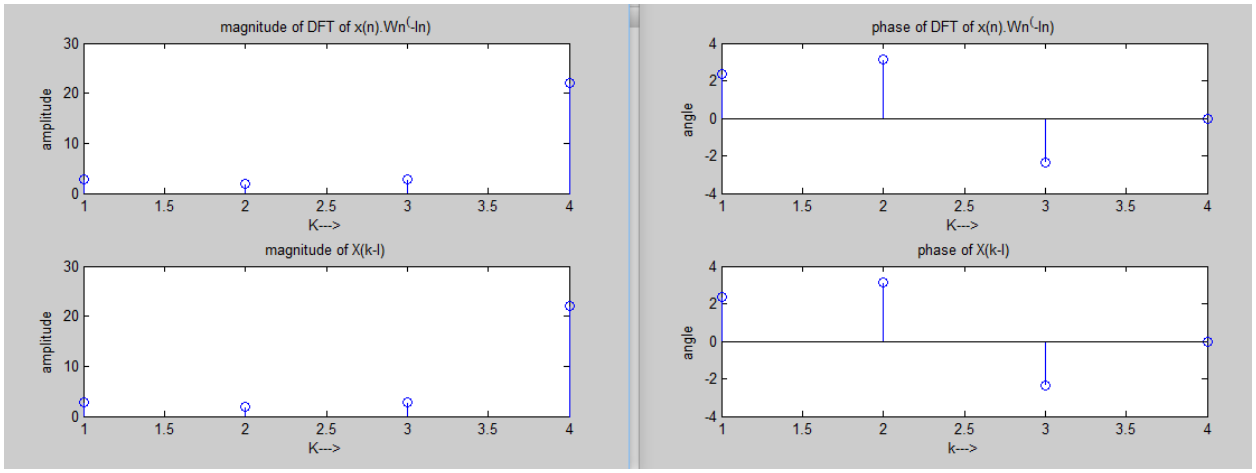
d) Circular frequency shift property of a sequence

Circular frequency shift property states that if X(k) is the DFT of the sequence x(n) then multiplying x(n) by e^(j*2*pi*l*n / N) is equivalent to shifting the DFT of the sequence x(n) circularly by 'l' samples as X((k − l))N.

To demonstrate:

Verify the property of Circular frequency shift for the given input sequence: [4 5 6 7] with number of shifts equal to 3.

```
Enter the sequence
[4 5 6 7]
Enter the delay
3
 DFT of the input sequence x(n)
  22.0000            -2.0000 + 2.0000i  -2.0000            -2.0000 - 2.0000i

Circular frequency shift of X(K) i.e DFT of x(n).e^(j*2*pi*l*n / N)
  -2.0000 + 2.0000i  -2.0000            -2.0000 - 2.0000i  22.0000

Resultant circular frequency shift i.e X((k – l))N
  -2.0000 - 2.0000i  22.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 - 0.0000i

Thus Circular-Frequency Shift Property is verified
>>
```



e) Conjugate Symmetry property

The Conjugate symmetry property states that the DFT of '$x^*(n)$ ' is given by ' $X^*(N-k)$ ' which is also equal to $X^*((-k))N$.

To demonstrate:

Verify the property of Conjugate Symmetry for the given input sequence:

Real input sequence: [4 5 6 7]

Imaginary input sequence: [2i 3 4 5]

```
 Symmetric property
enter the real input sequence [4 5 6 7]

N =

     4

 it has symmetry
  22.0000            -2.0000 + 2.0000i  -2.0000            -2.0000 - 2.0000i

fx >>
```

Conjugate symmetry property for a real sequence

```
 Symmetric property
enter the real input sequence [2i 3 4 5]

N =

     4

 it doesnt have symmetry
  12.0000 + 2.0000i  -4.0000 + 4.0000i  -4.0000 + 2.0000i  -4.0000

fx >>
```

f) Circular convolution of two sequences

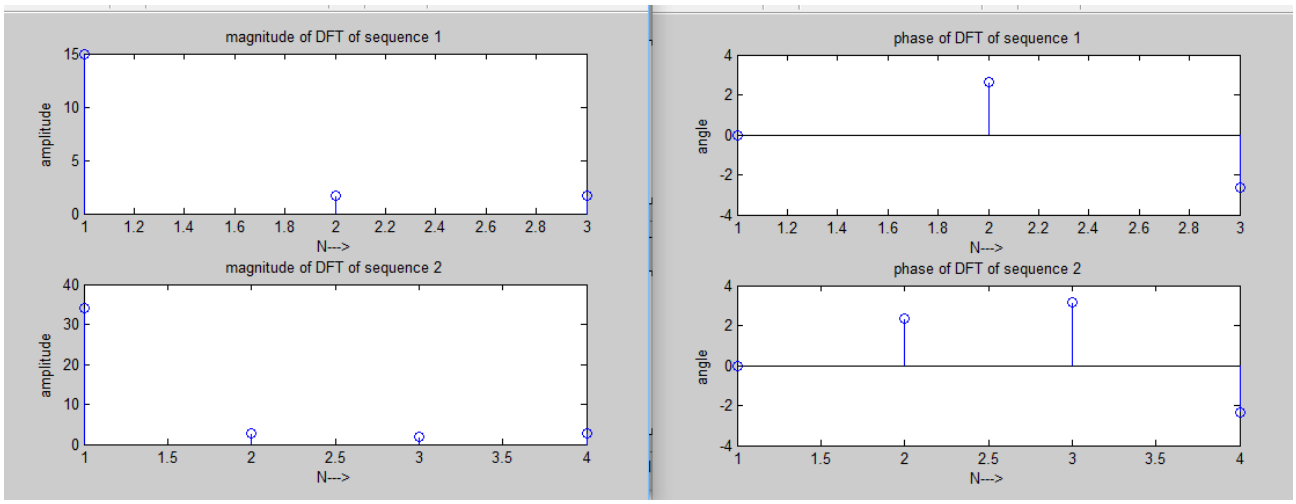
The circular convolution of two sequences x1(n) and x2(n) in time domain i.e. 'x1(n) ⊗ x2(n) ' is represented as multiplication i.e. ' X1(k).X2(k) ' in frequency domain.

To demonstrate:

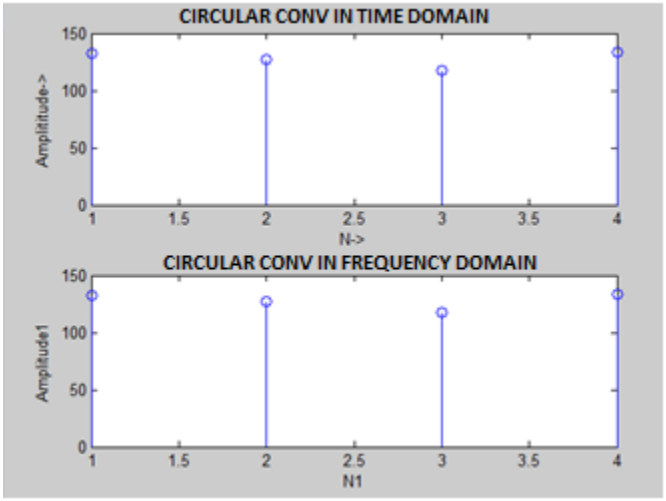
Verify the property of circular convolution for the given input sequences:

Sequence 1: [4 5 6]

Sequence 2: [7 8 9 10]




```
Enter the sequence 1:[4 5 6]
Enter the sequence 2:[7 8 9 10]
The resultant time domain convolution is
   132   127   118   133

conv in freq domain:
   132   127   118   133

DFT of sequence 1:
  15.0000            -1.5000 + 0.8660i  -1.5000 - 0.8660i

DFT of sequence 2:
  34.0000            -2.0000 + 2.0000i  -2.0000            -2.0000 - 2.0000i

fx >>
```



g) Time reversal of a sequence

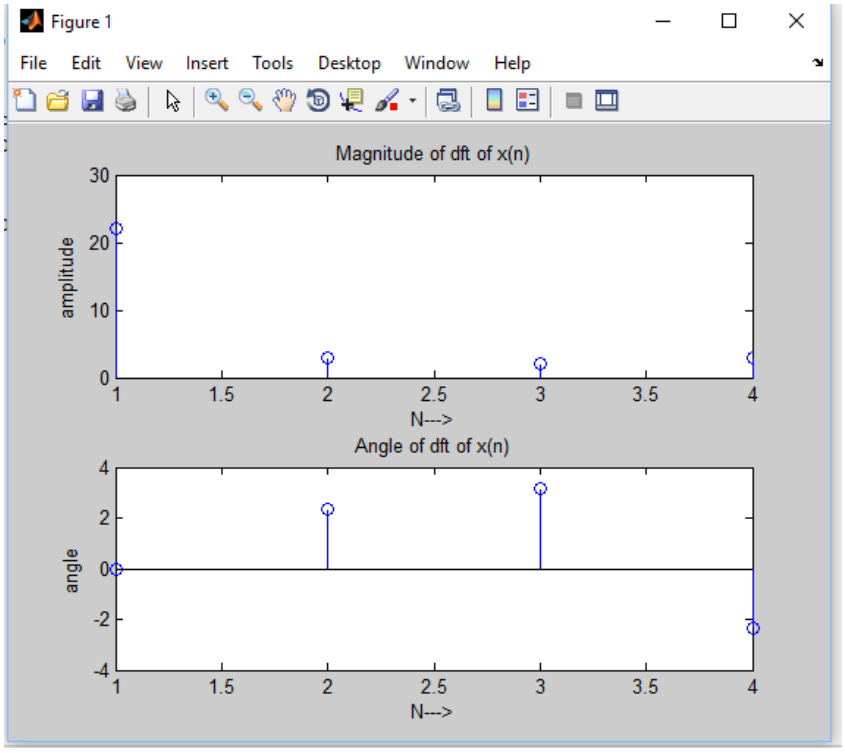
Time reversal of a sequence is demonstrated as: DFT of 'x(N − n) ' is given by ' X(N − k) '.

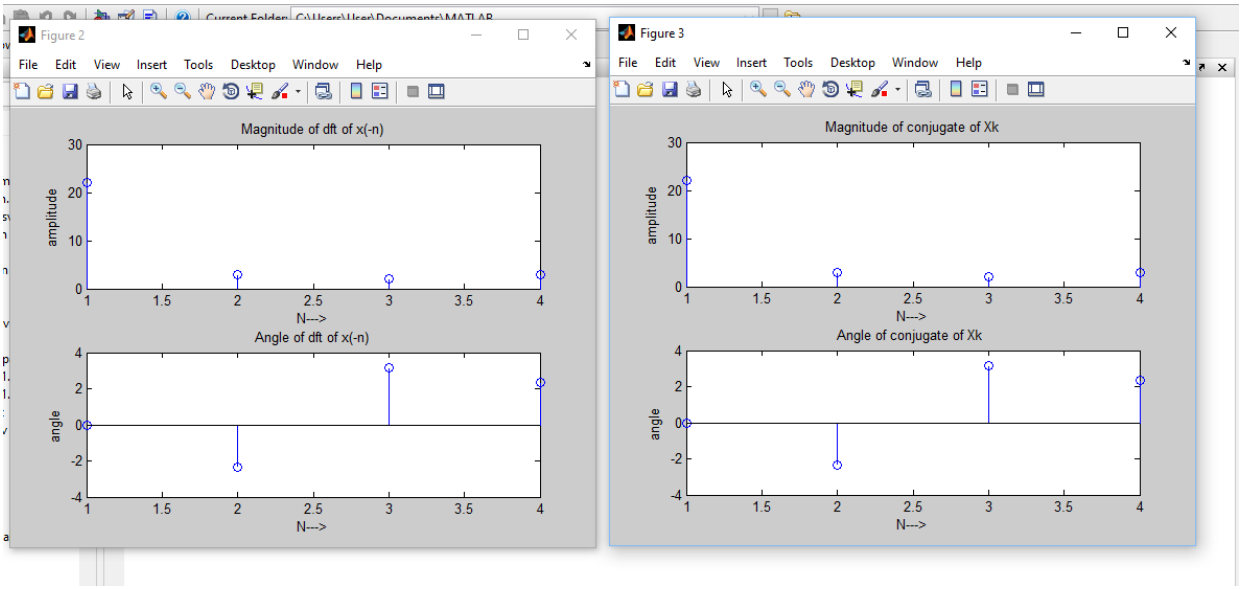
To demonstrate:

Verify the property of Time reversal for the given input sequence: [4 5 6 7]

```
Enter the sequence
[4 5 6 7]
DFT of the sequence x(n)
  22.0000            -2.0000 + 2.0000i  -2.0000            -2.0000 - 2.0000i

Time reversed sequence x(-n)
     4     7     6     5

DFT of time reversed sequence
  22.0000            -2.0000 - 2.0000i  -2.0000            -2.0000 + 2.0000i

Conjugate of X(k)
  22.0000            -2.0000 - 2.0000i  -2.0000            -2.0000 + 2.0000i

Thus Time-reversal Property is verified
>>
```

Time reversal property





h) Parseval's theorem

Parsevals theorem defines energy in time domain in the form $\sum_{n=0}^{N-1} x(n)y^*(n)$ which is equivalently represented in the frequency domain as $\frac{1}{N}\sum_{k=0}^{N-1} X(k)Y^*(k)$.

To demonstrate:

Verify Parseval's theorem for the given input sequence: [0 0.707 1 0.707]

```
Enter the sequence
[0 0.707 1 0.707]
Energy in time domain
    1.9997

Energy in frequency domain
    1.9997

Thus Parsevals theorem is verified
>>
```

Experiment-4

a) Write a MATLAB code to compute the DFT of a sequence x (n) using DIT and DIF algorithm. Also indicate the speed improvement factor in calculating the DFT of a sequence using direct computation and FFT algorithm .Further compute the IDFT using IDIT and IDIF algorithm.

Outcome: The student will be able to apply computationally efficient algorithms for evaluating the DFT.

A fast Fourier transform (FFT) is any fast algorithm for computing the DFT. The development of FFT algorithms had a tremendous impact on computational aspects of signal processing and applied science. The DFT of an N-point signal

$$\{x[n], 0 \leq n \leq N-1\}$$

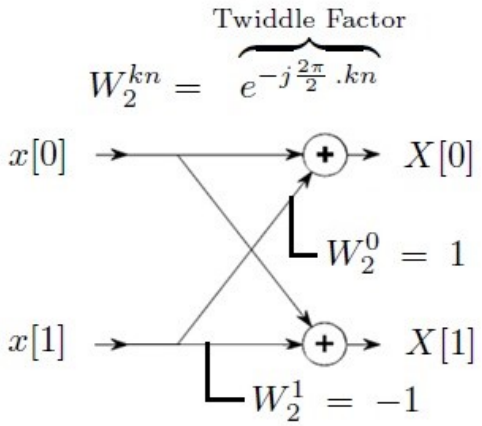
is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{-kn}, \qquad 0 \leq k \leq N-1$$

where

$$W_N = e^{j\frac{2\pi}{N}} = \cos\left(\frac{2\pi}{N}\right) + j\,\sin\left(\frac{2\pi}{N}\right)$$

is the principal N-th root of unity.



I. DFT using DIT Algorithm

Consider an N-point signal $x[n]$ of even length. The derivation of the DIT radix-2 FFT begins by splitting the sum into two parts — one part for the even-indexed values $x[2n]$ and one part for the odd-indexed values $x[2n+1]$. Define two $N/2$-point signals $x_1[n]$ and $x_2[n]$ as

$$x_0[n] = x[2n]$$
$$x_1[n] = x[2n+1]$$

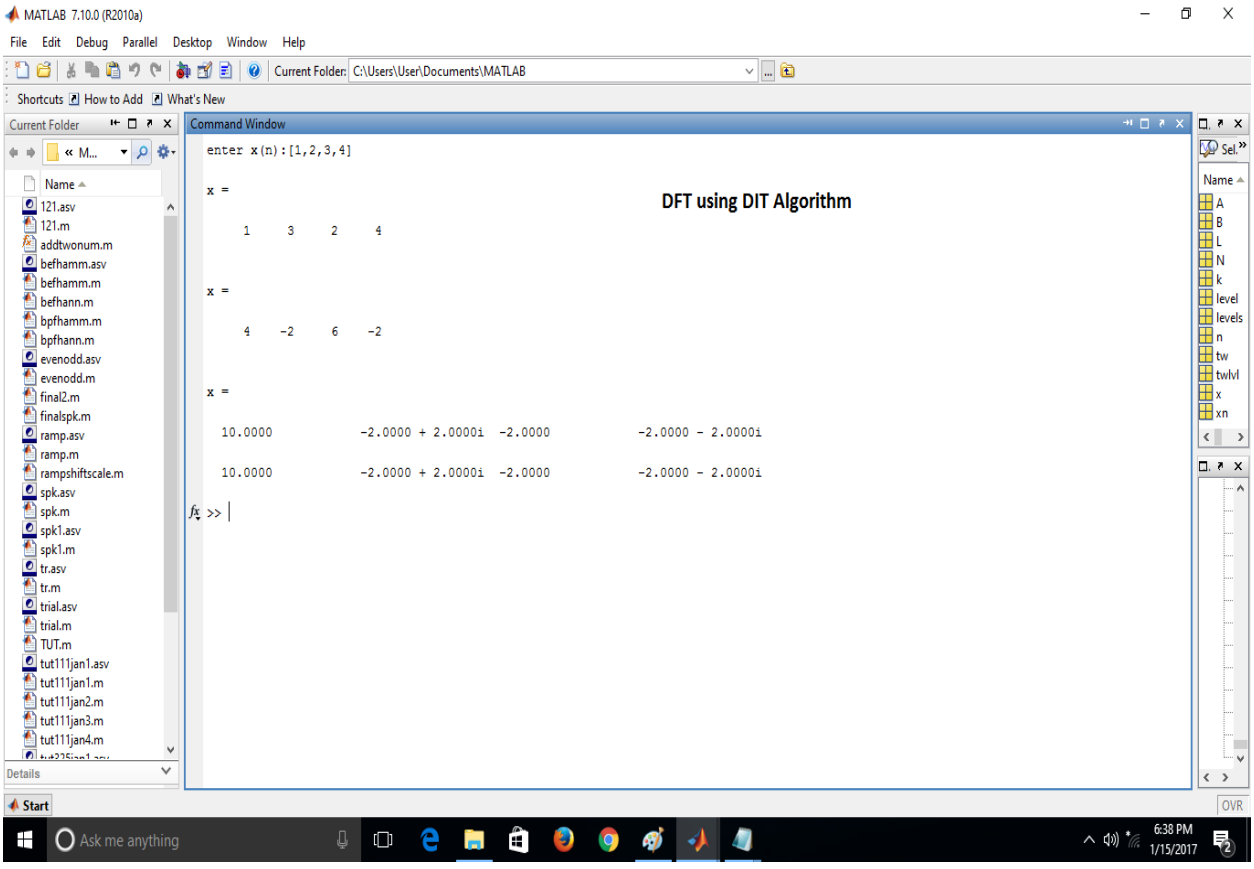
for $0 \leq n \leq N/2 - 1$. The DFT of the N-point signal $x[n]$ can be written as

$$X[k] = \sum_{\substack{n=0 \\ n\ even}}^{N-1} x[n]\, W_N^{-nk} + \sum_{\substack{n=0 \\ n\ odd}}^{N-1} x[n]\, W_N^{-nk}$$

To demonstrate:

To find the DFT of the given input sequence using DIT Algorithm

Input : [1 2 3 4]



II. DFT using DIF Algorithm

Consider an N-point signal $x[n]$ of even length. The derivation of the DIF radix-2 FFT begins by splitting the DFT coefficients $X[k]$ in to even- and odd- indexed values. The even values $X[2k]$ are given by:

$$X[2k] = \sum_{n=0}^{N-1} x[n]\, W_N^{-2kn}$$
$$= \sum_{n=0}^{N-1} x[n]\, W_{N/2}^{-kn}.$$

Splitting this sum into the first $N/2$ and second $N/2$ terms gives

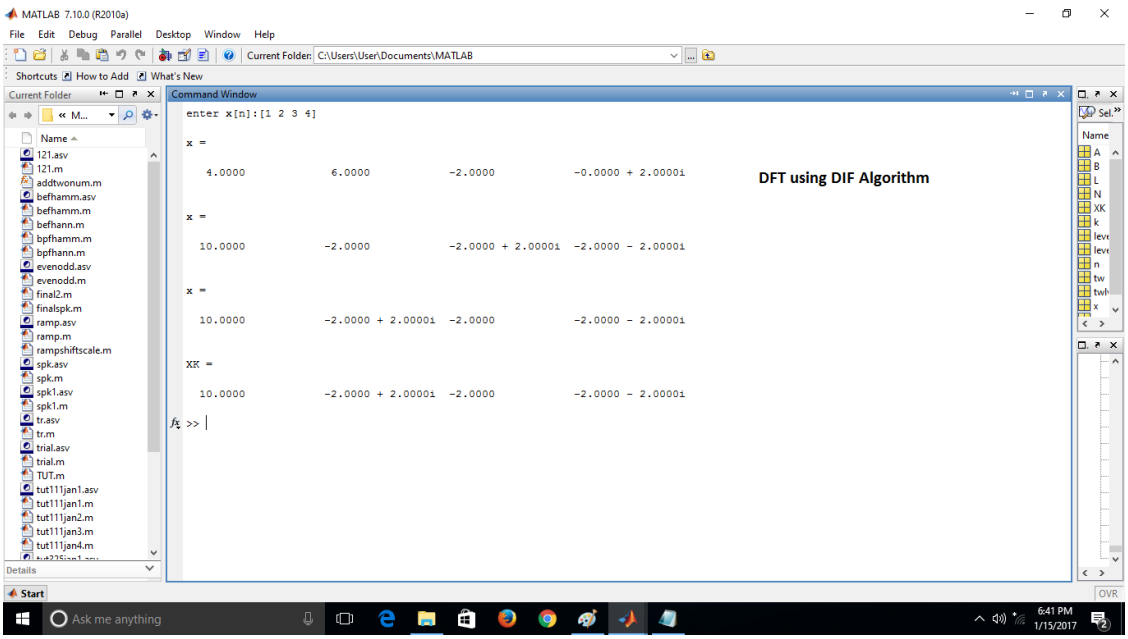
$$X[2k] = \sum_{n=0}^{\frac{N}{2}-1} x[n]\, W_{N/2}^{-kn} + \sum_{n=\frac{N}{2}}^{N-1} x[n]\, W_{N/2}^{-kn}$$
$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, W_{N/2}^{-kn} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}]\, W_{N/2}^{-k(n+\frac{N}{2})}$$
$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, W_{N/2}^{-kn} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}]\, W_{N/2}^{-kn}$$
$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + x[n+\frac{N}{2}] \right) W_{N/2}^{-kn}$$
$$= \mathrm{DFT}_{\frac{N}{2}} \left\{ x[n] + x[n+\frac{N}{2}] \right\}.$$

That is, the even DFT values $X[2k]$ for $0 \leq 2\,k \leq N-1$ are given by the $\frac{N}{2}$-point DFT of the $\frac{N}{2}$-point signal $x[n] + x[n+N/2]$.

To demonstrate:

To find the DFT of the given input sequence using DIF Algorithm

Input: [1 2 3 4]

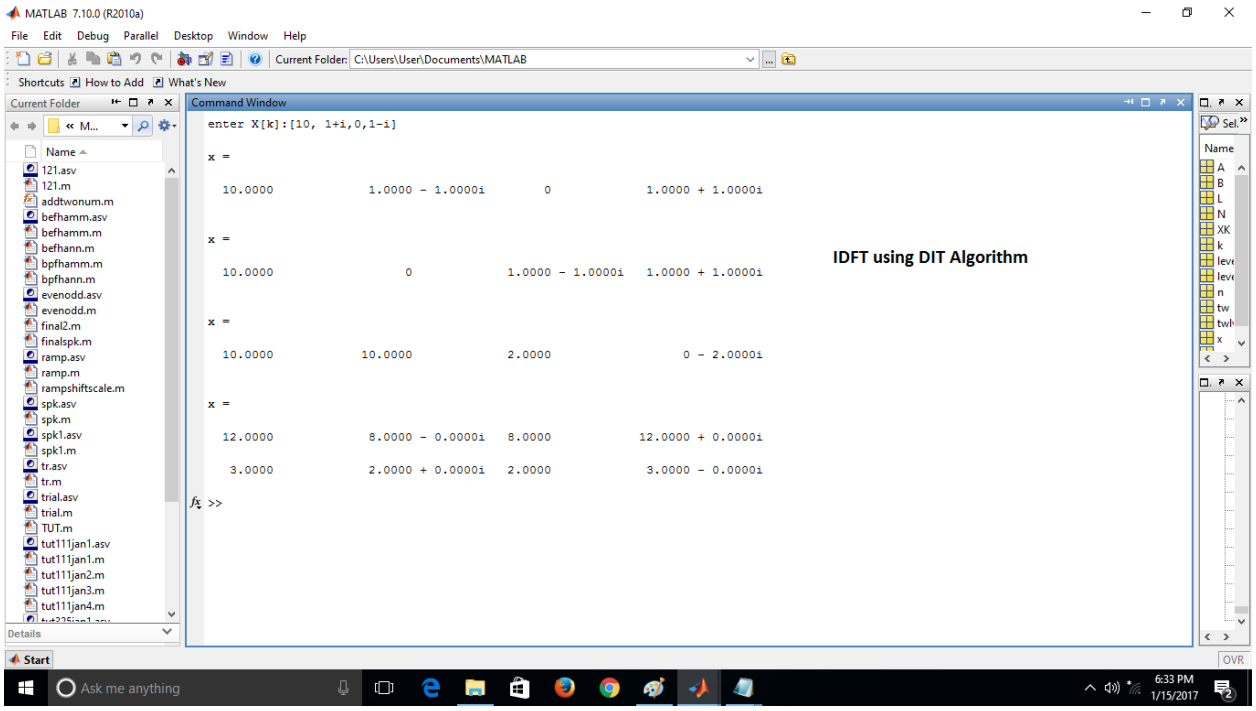


III. IDFT using DIT Algorithm

To demonstrate:

To find the IDFT of the given input sequence using DIT Algorithm

Input : [10 1+i 0 1-i]

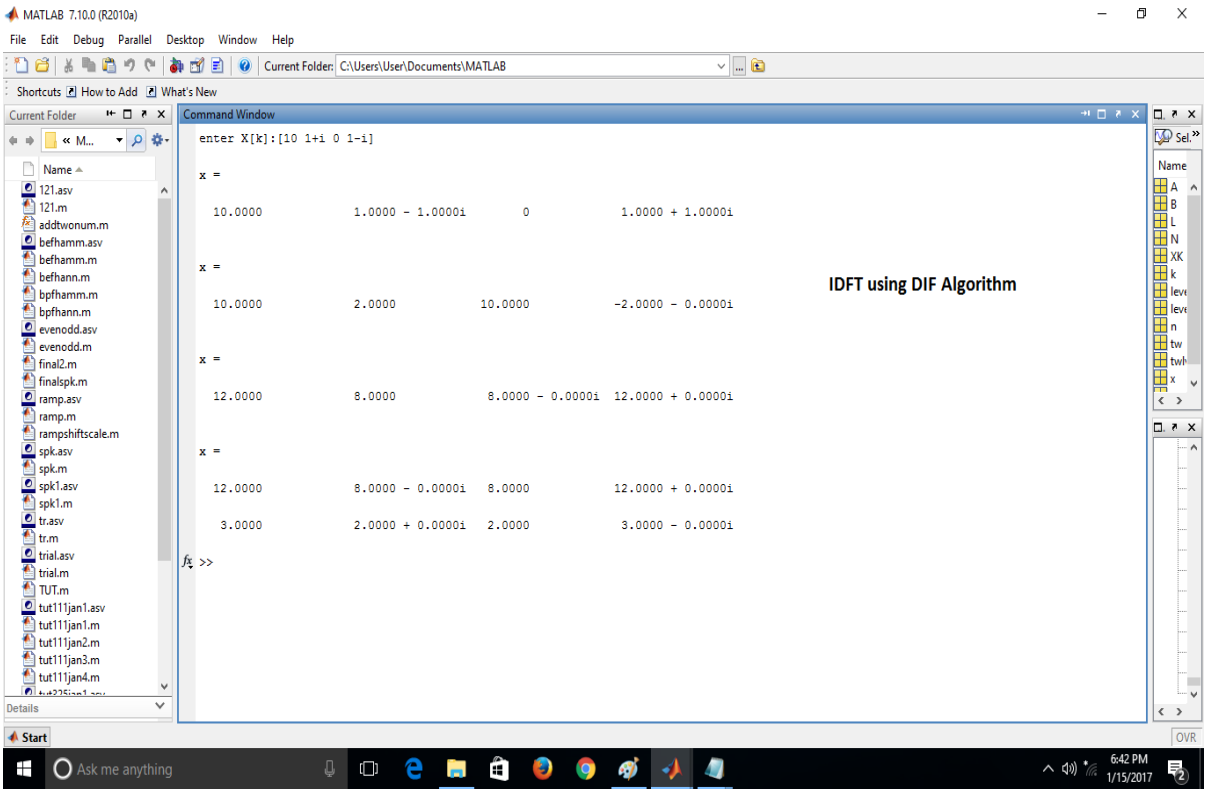


IV. IDFT using DIF Algorithm

To demonstrate:

To find the IDFT of the given input sequence using DIF Algorithm

Input : [10 1+i 0 1-i]



b) Write a MATLAB code to verify the Low pass and High Pass FIR linear phase filter design using Hamming and Hanning windows. Plot the magnitude and phase response. Also, Provide the inference on the basis of results obtained for the set of specifications. (To design should be verified by convolving the input signal with the designed filter coefficients).

Outcome: The student will be able to design and implement a linear phase FIR filter that best matches the application and satisfies the design requirements.

THEORY: A digital filter is a system which passes some desired signals more than others to reduce or enhance certain aspects of that signal. It can be used to pass the signals according to the specified frequency pass-band and reject the other frequency than the pass-band specification. The basic filter types can be divided into four categories:

1. Low-pass
2. High-pass
3. Band-pass
4. Band-stop.

On the basis of impulse response, there are two fundamental types of digital Filters:

1. Infinite Impulse Response (IIR) filters
2. Finite Impulse Response (FIR) filters.

FIR filter is described by the difference equation

$$y(n) = h(k)x(n - k)$$

Where, x(n) is input signal and h(n) is impulse response of FIR filter. The transfer function of a causal FIR filter is described by

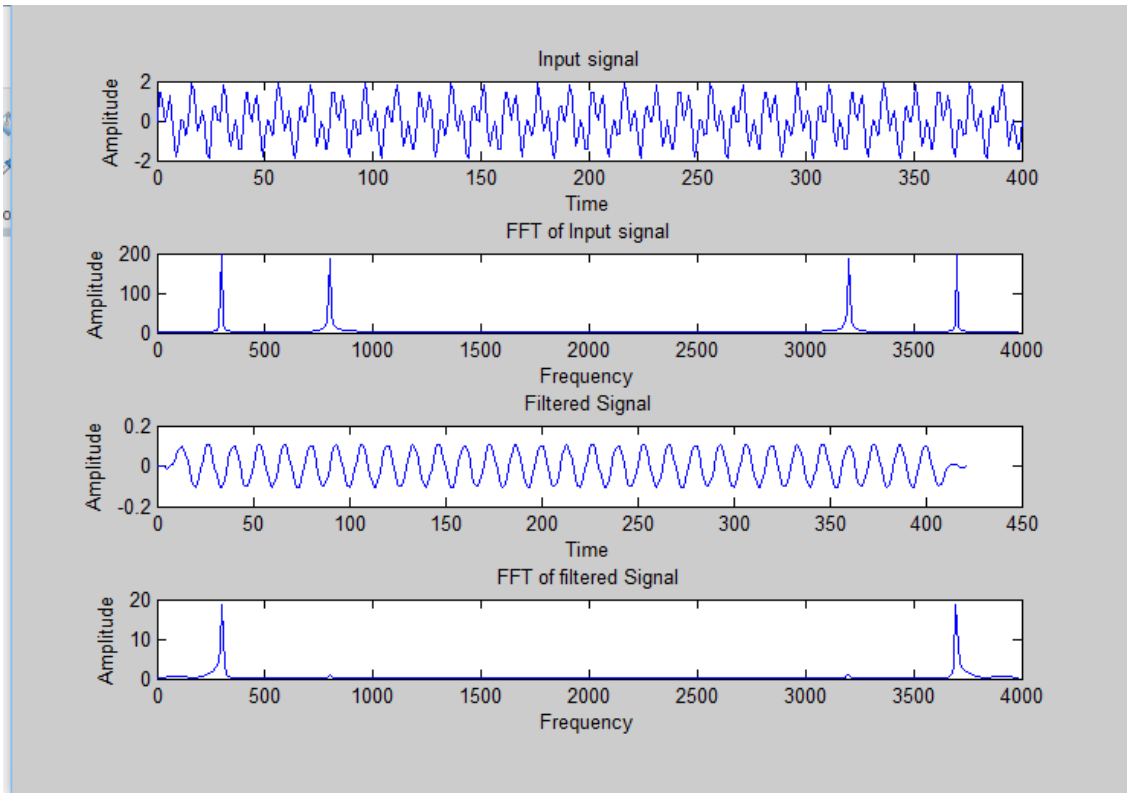
$$H(z) = \sum_{k=0}^{N-1} h(k)z^{-k}$$

A simple and effective way to design digital FIR filter is window method.
Window functions are used for designing a FIR filter.
(i). Hanning window function w (n) =0.5 − 0.5 cos (2nπ/N-1) , 0 ≤ n ≤ N – 1 0, otherwise
(ii). Hamming window function w (n) =0.54 − 0.46 cos (2nπ/N−1), 0 ≤ n ≤ N − 1 0, otherwise

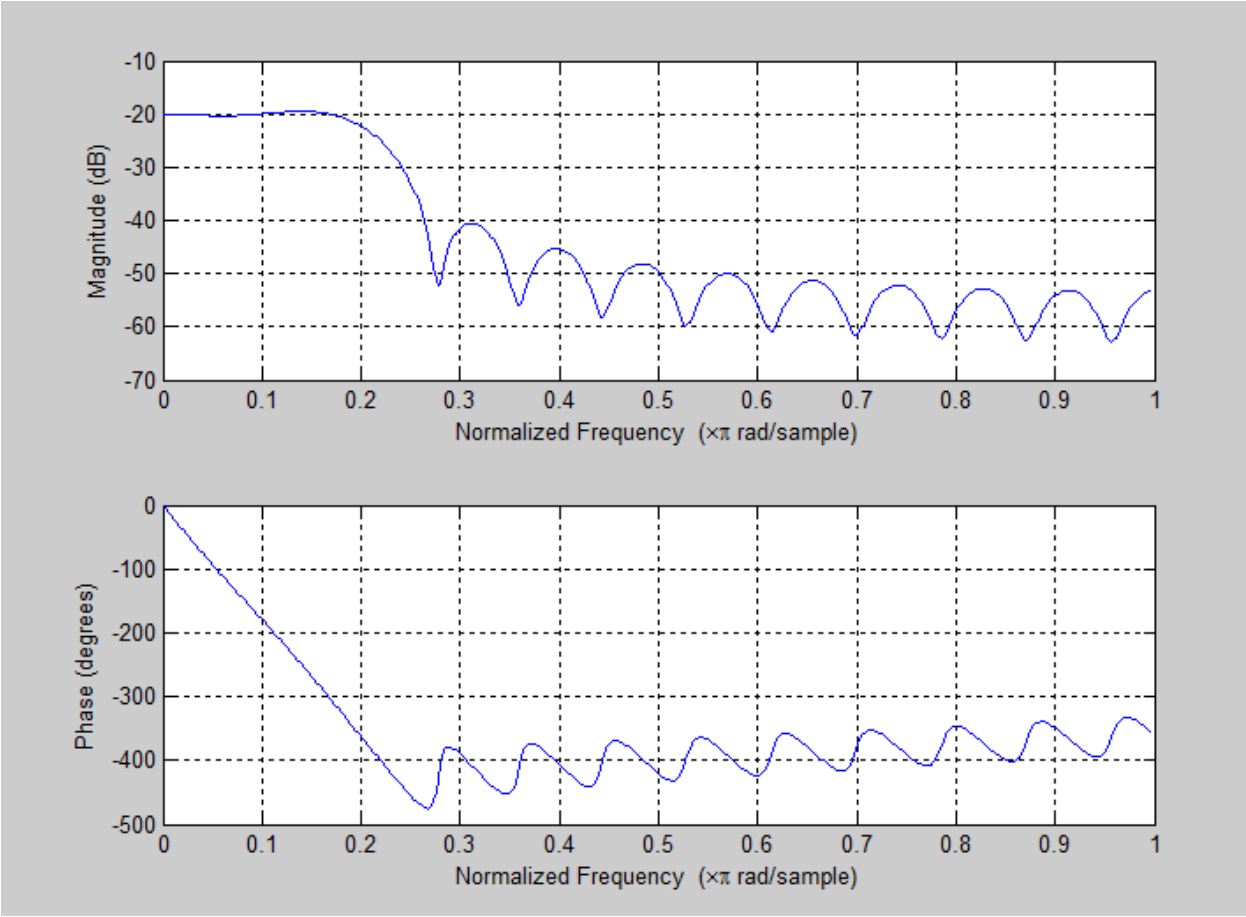
I. HAMMING LOW PASS FILTER

To be demonstrated:

For the given data below, construct a Hamming low pass filter and filter the input signal with frequencies as below:
Filter specifications:
Cut-off frequency – 450 Hz
Order of the filter – 23
Input signal:
First input frequency – 300 Hz
Second input frequency – 800 Hz



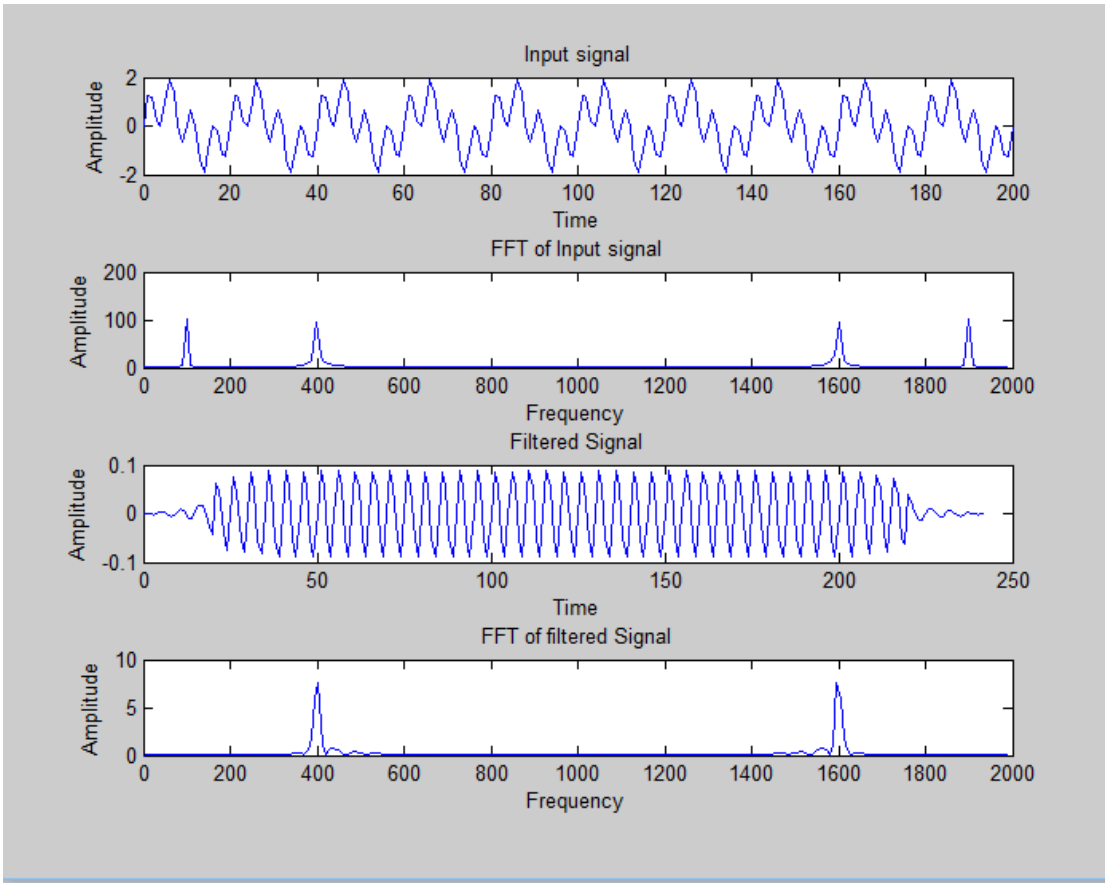

```
enter the order : 23
enter f1 : 300
enter f2 : 800
enter cut-off freq : 450
  Columns 1 through 6

    0.0022    0.0003   -0.0023   -0.0044   -0.0047   -0.0024

  Columns 7 through 12

    0.0024    0.0089    0.0155    0.0204    0.0222    0.0204

  Columns 13 through 18

    0.0155    0.0089    0.0024   -0.0024   -0.0047   -0.0044

  Columns 19 through 23

   -0.0023    0.0003    0.0022    0.0028    0.0021

fx >>
```

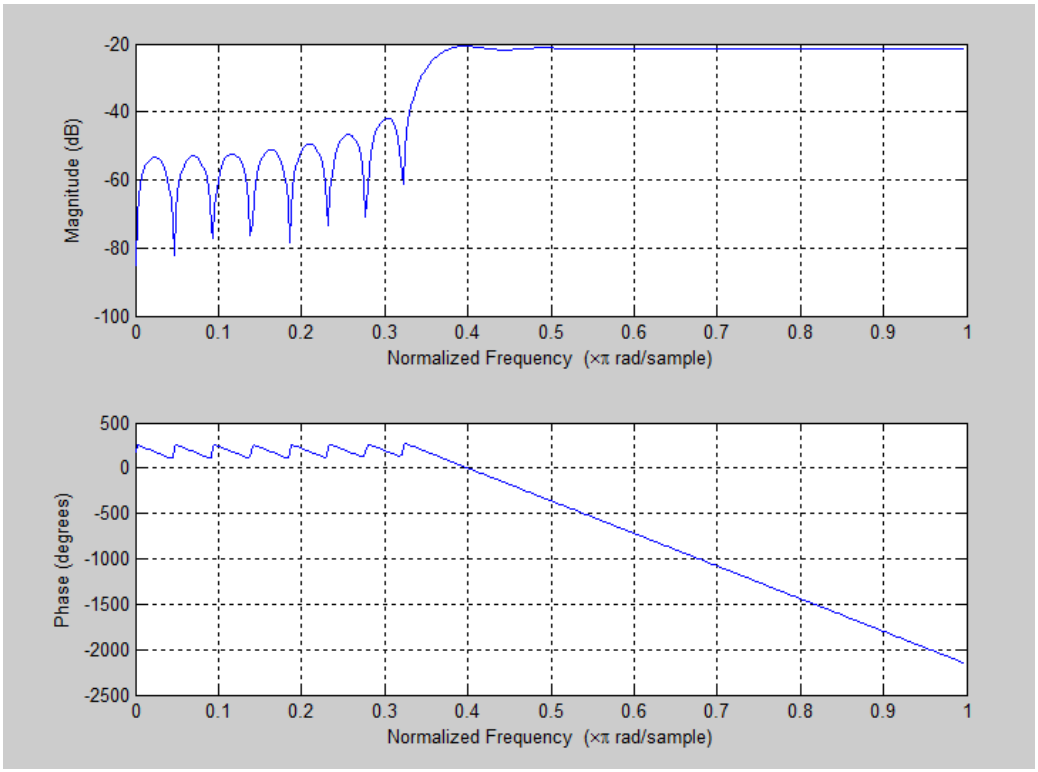


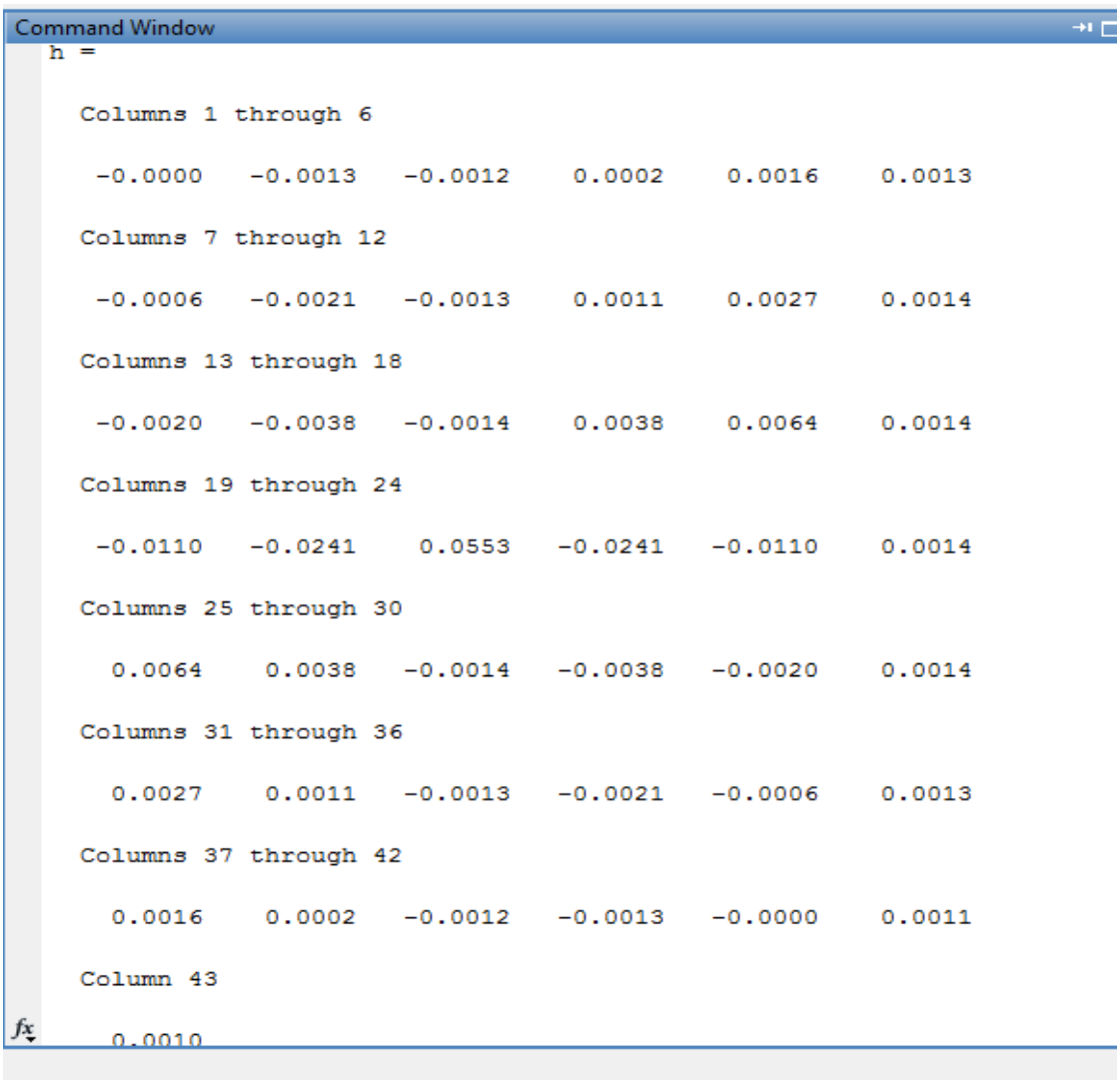
II. HAMMING HIGH PASS FILTER

To be demonstrated:

For the given data below, construct a Hamming high pass filter and filter the input signal with frequencies as below:
Filter specifications:
Cut-off frequency – 350 Hz
Order of the filter – 43
Input signal:
First input frequency – 100 Hz
Second input frequency – 400 Hz


Input signal
Amplitude
Time
FFT of Input signal
Amplitude
Frequency
Filtered Signal
Amplitude
Time
FFT of filtered Signal
Amplitude
Frequency

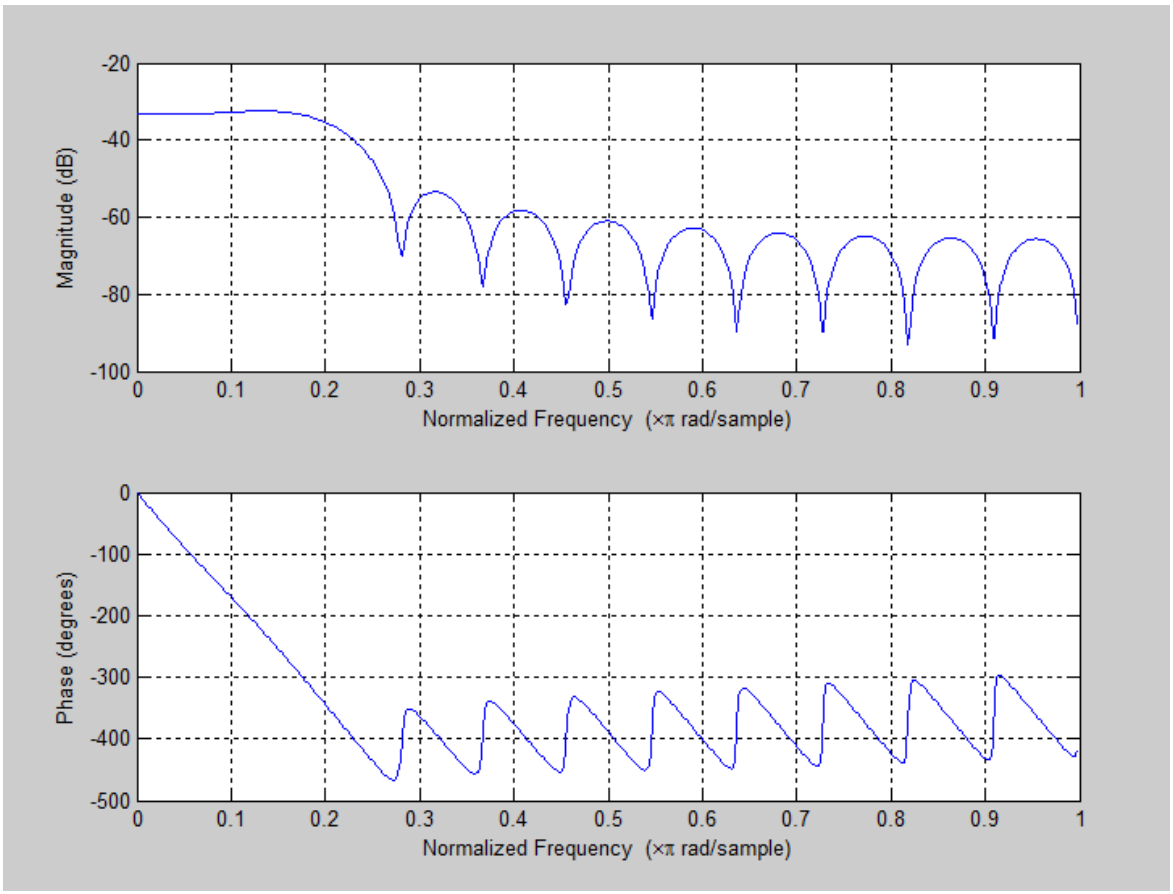

Magnitude (dB)
Normalized Frequency (×π rad/sample)
Phase (degrees)
Normalized Frequency (×π rad/sample)

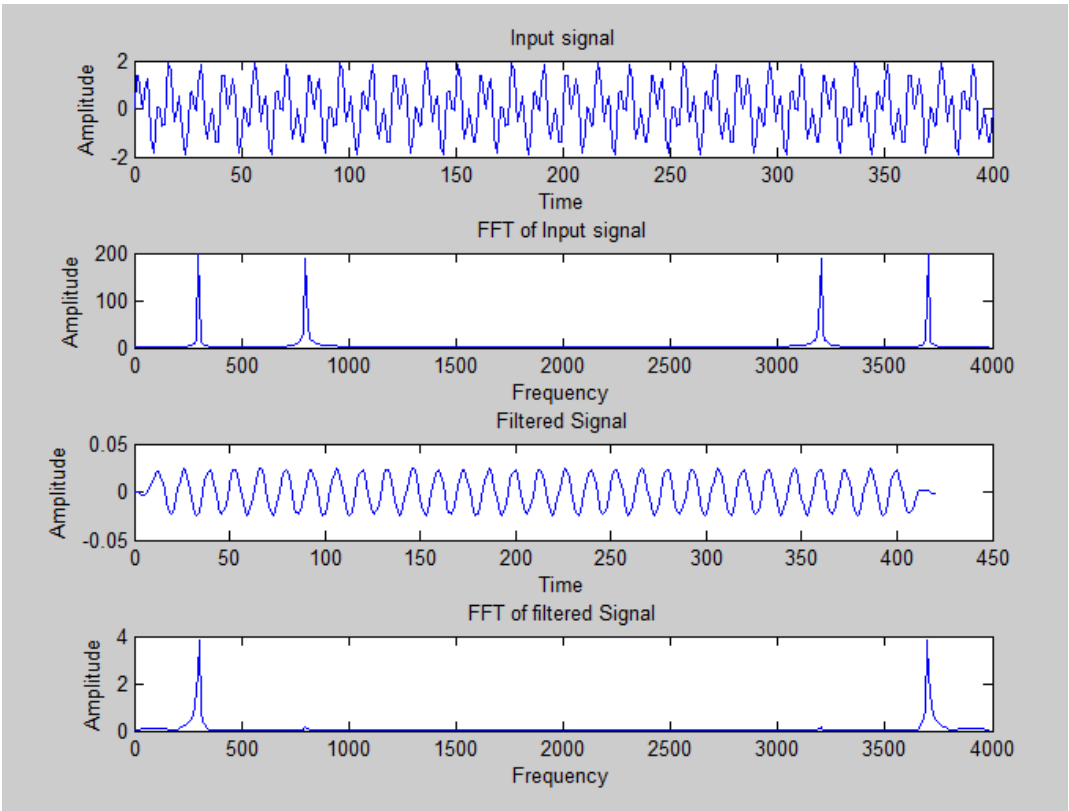


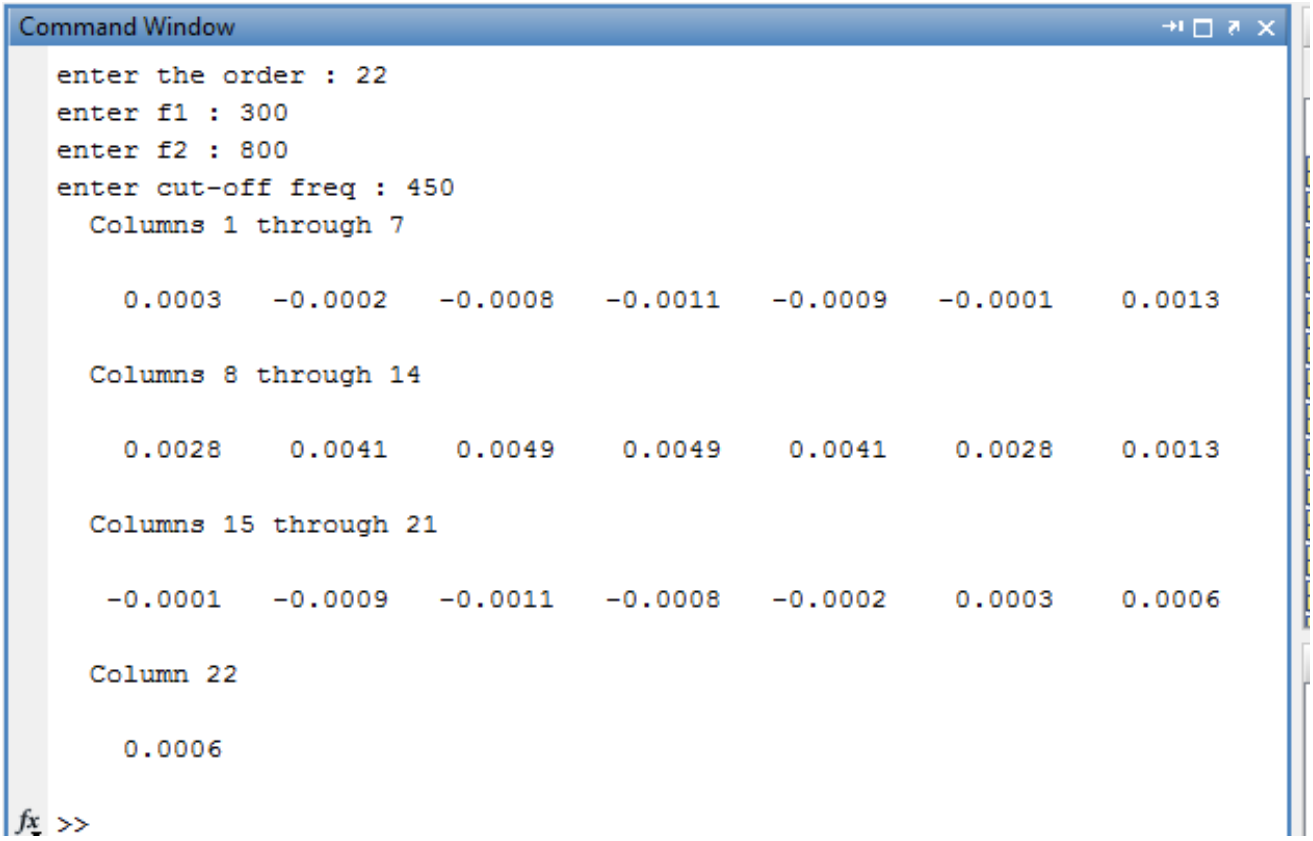
III. HANNING LOW PASS FILTER

To be demonstrated:

For the given data below, construct a Hanning low pass filter and filter the input signal with frequencies as below:
Filter specifications:
Cut-off frequency – 450 Hz
Order of the filter – 22
Input signal:
First input frequency – 300 Hz
Second input frequency – 800 Hz


Magnitude (dB)
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-100
-200
-300
-400
-500
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)

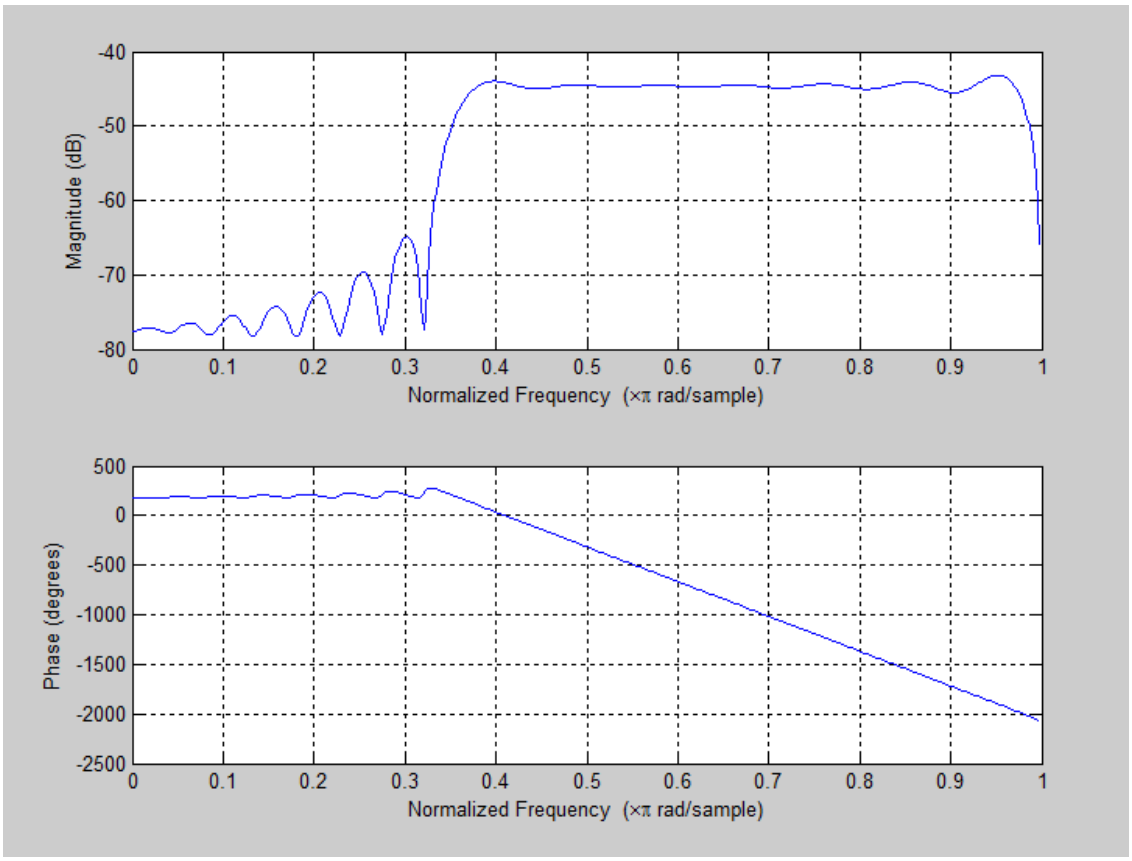

Input signal
Amplitude
2
0
-2
0
50
100
150
200
250
300
350
400
Time
FFT of Input signal
Amplitude
200
100
0
0
500
1000
1500
2000
2500
3000
3500
4000
Frequency
Filtered Signal
Amplitude
0.05
0
-0.05
0
50
100
150
200
250
300
350
400
450
Time
FFT of filtered Signal
Amplitude
4
2
0
0
500
1000
1500
2000
2500
3000
3500
4000
Frequency

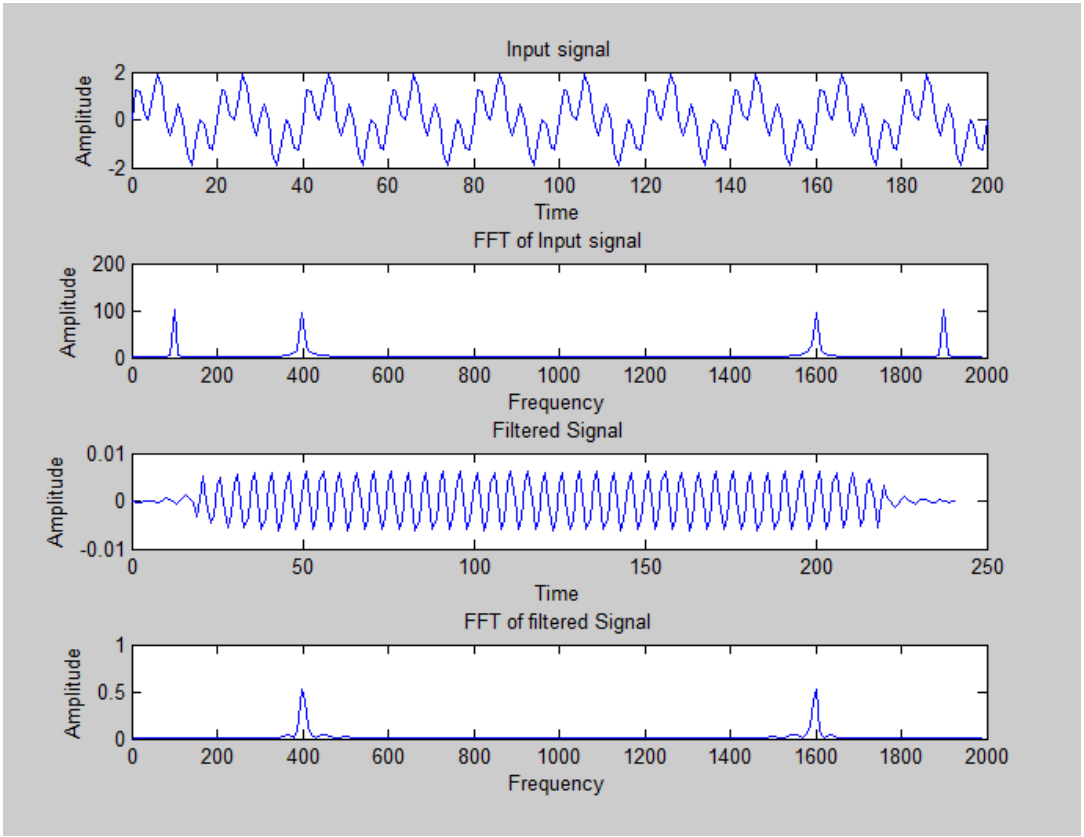


IV. HANNING HIGH PASS FILTER

To be demonstrated:

For the given data below, construct a Hanning high pass filter and filter the input signal with frequencies as below:
Filter specifications:
Cut-off frequency – 350 Hz
Order of the filter – 42
Input signal:
First input frequency – 100 Hz
Second input frequency – 400 Hz


Magnitude (dB)
-40
-50
-60
-70
-80
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)
Phase (degrees)
500
0
-500
-1000
-1500
-2000
-2500
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (×π rad/sample)


Input signal
Amplitude
2
0
-2
0
20
40
60
80
100
120
140
160
180
200
Time
FFT of Input signal
Amplitude
200
100
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Frequency
Filtered Signal
Amplitude
0.01
0
-0.01
0
50
100
150
200
250
Time
FFT of filtered Signal
Amplitude
1
0.5
0
0
200
400
600
800
1000
1200
1400
1600
1800
2000
Frequency

```
enter the order : 42
enter f1 : 100
enter f2 : 400
enter cut-off freq : 350

h =

  Columns 1 through 7

   -0.0001    0.0000   -0.0001    0.0002   -0.0000    0.0002   -0.0002

  Columns 8 through 14

    0.0000   -0.0002    0.0003   -0.0000    0.0002   -0.0005    0.0001

  Columns 15 through 21

   -0.0003    0.0008   -0.0002    0.0005   -0.0025    0.0018    0.0018

  Columns 22 through 28

   -0.0025    0.0005   -0.0002    0.0008   -0.0003    0.0001   -0.0005

  Columns 29 through 35

    0.0002   -0.0000    0.0003   -0.0002    0.0000   -0.0002    0.0002

  Columns 36 through 42

   -0.0000    0.0002   -0.0001    0.0000   -0.0001    0.0001   -0.0000

fx >>
```

Experiment-5

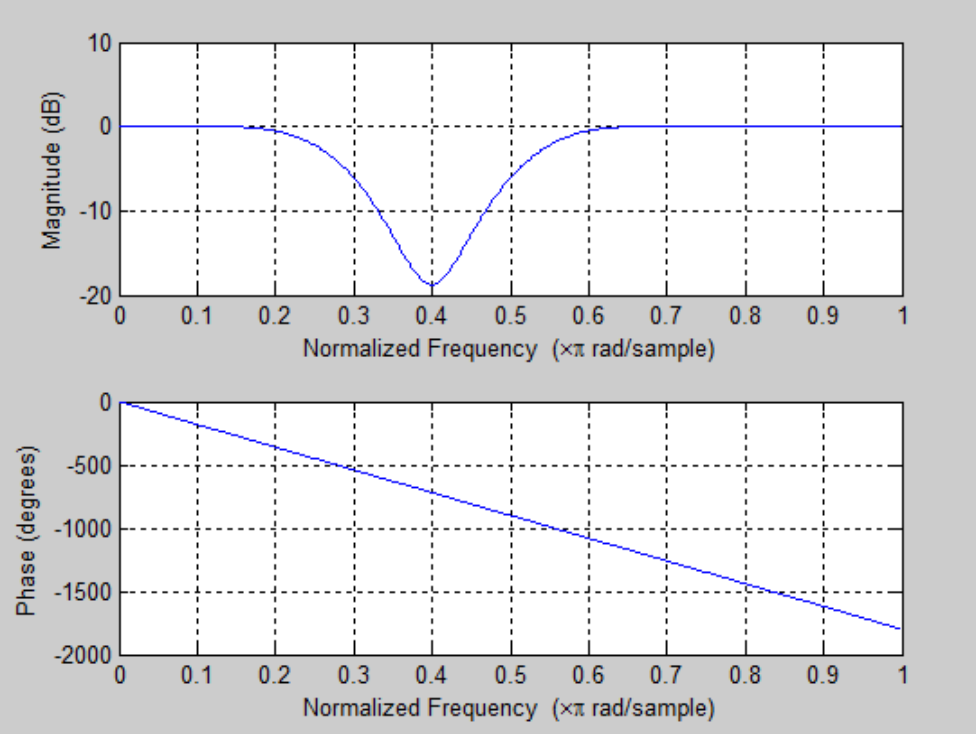
a) Write a MATLAB code to verify the Band pass and Band reject FIR linear phase filter design using Hamming and Hanning windows (with inbuilt and without using inbuilt commands). Plot the magnitude and phase response. Also, Provide the inference on the basis of results obtained for the set of specifications.

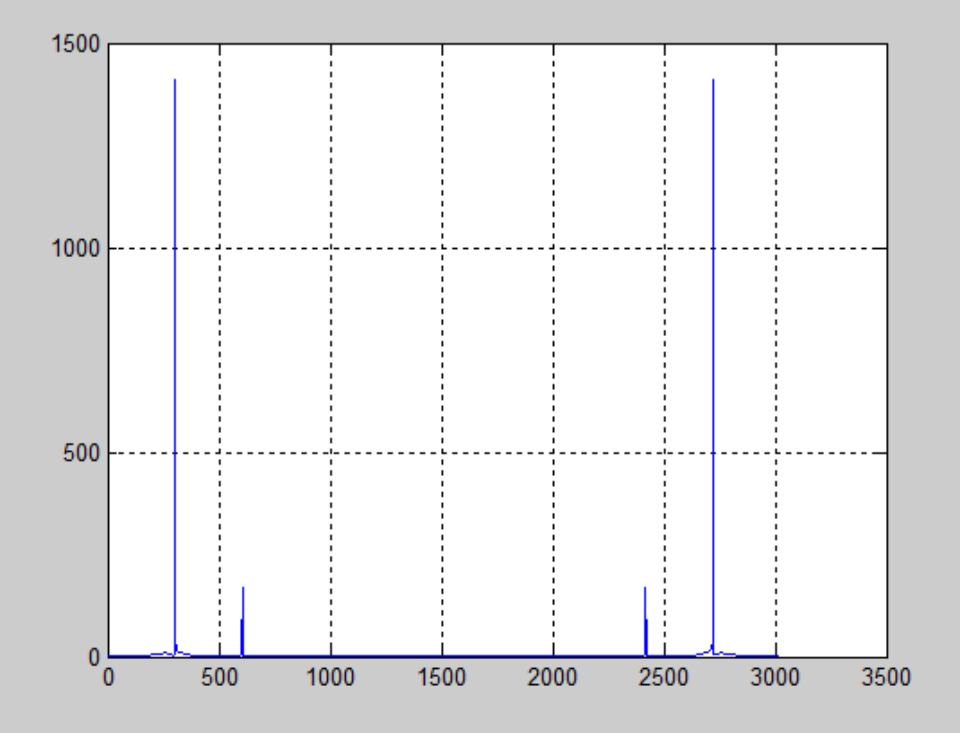
Outcome: The student will be able to design and implement a linear phase FIR filter that best matches the application and satisfies the design requirements.

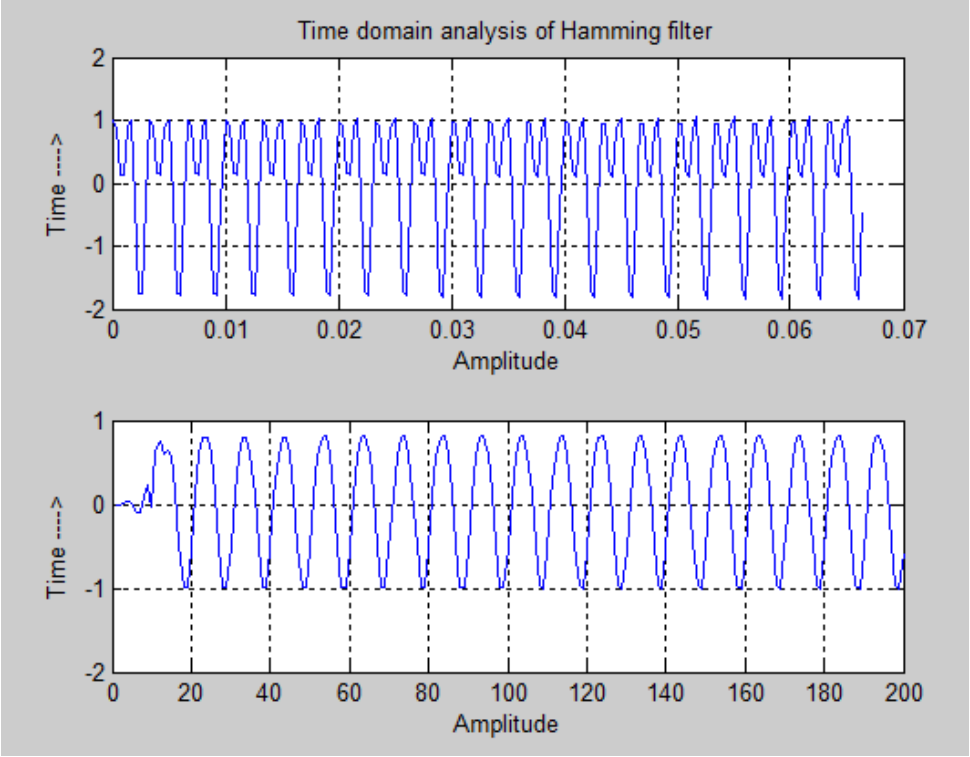
I. HAMMING BAND ELIMINATION FILTER

To be demonstrated:

For the given data below, construct a Hamming band elimination filter and filter the input signal with frequencies as below:
Filter specifications:
First cut-off frequency – 450 Hz
Second cut-off frequency – 750 Hz
Order of the filter – 23
Input signal:
First input frequency – 300 Hz
Second input frequency – 600 Hz




1500
1000
500
0
0
500
1000
1500
2000
2500
3000
3500


Time domain analysis of Hamming filter
Time --->
Amplitude
Time --->
Amplitude

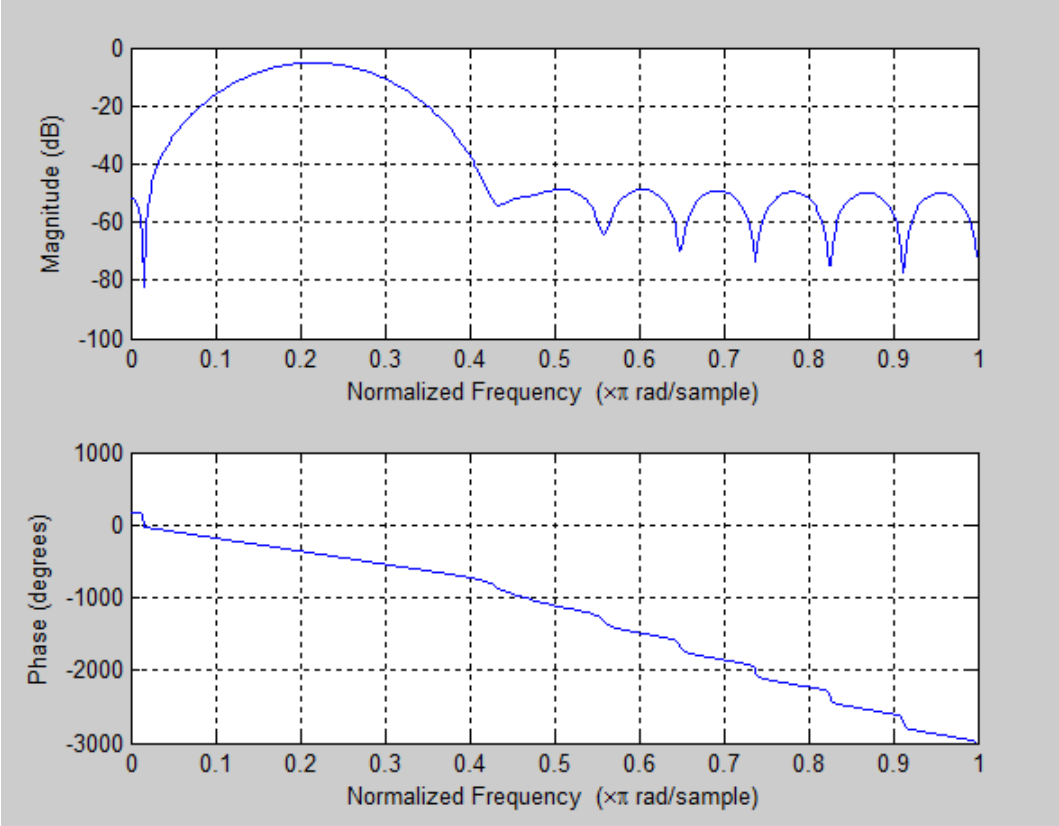
```
Command Window
Enter the 1st input frequency :300
Enter the 2nd input frequency :600
Enter the cutoff first frequency :450
Enter the cutoff second frequency :750
Enter the Order of the filter :23
  Columns 1 through 7

   -0.0000   -0.0010    0.0091    0.0207   -0.0150   -0.0771   -0.0339

  Columns 8 through 14

    0.1170    0.1401   -0.0598    0.8000   -0.0598    0.1403    0.1171

  Columns 15 through 21

   -0.0340   -0.0773   -0.0150    0.0208    0.0091   -0.0010   -0.0000

  Columns 22 through 23

    0.0004   -0.0025
```

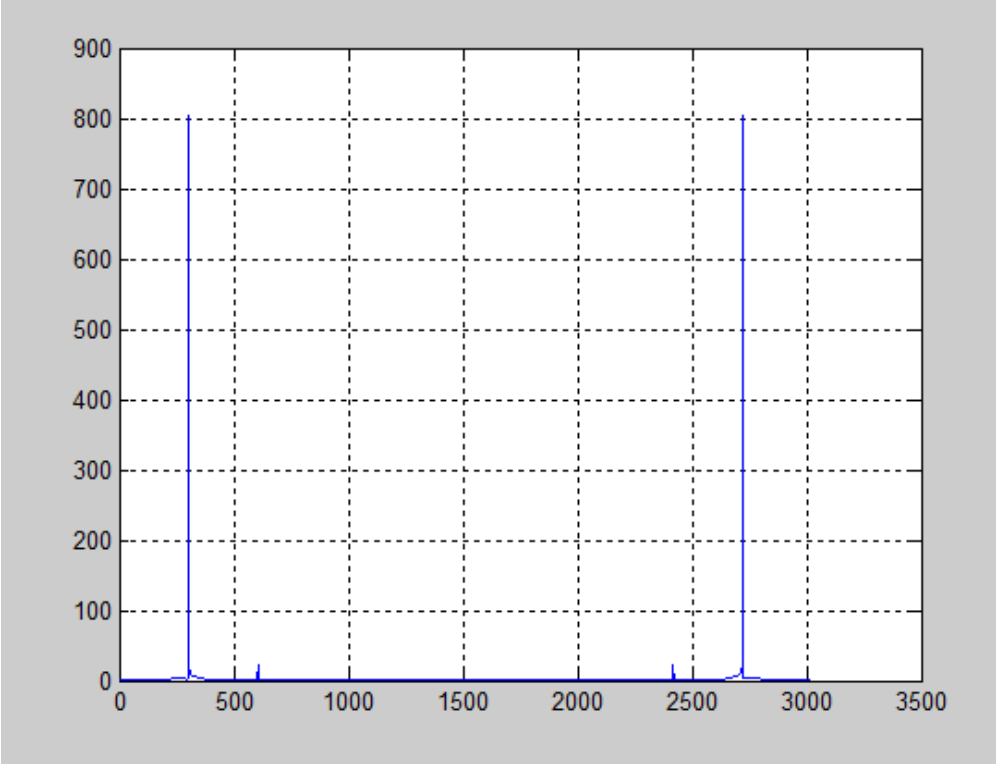
II. HAMMING BAND PASS FILTER

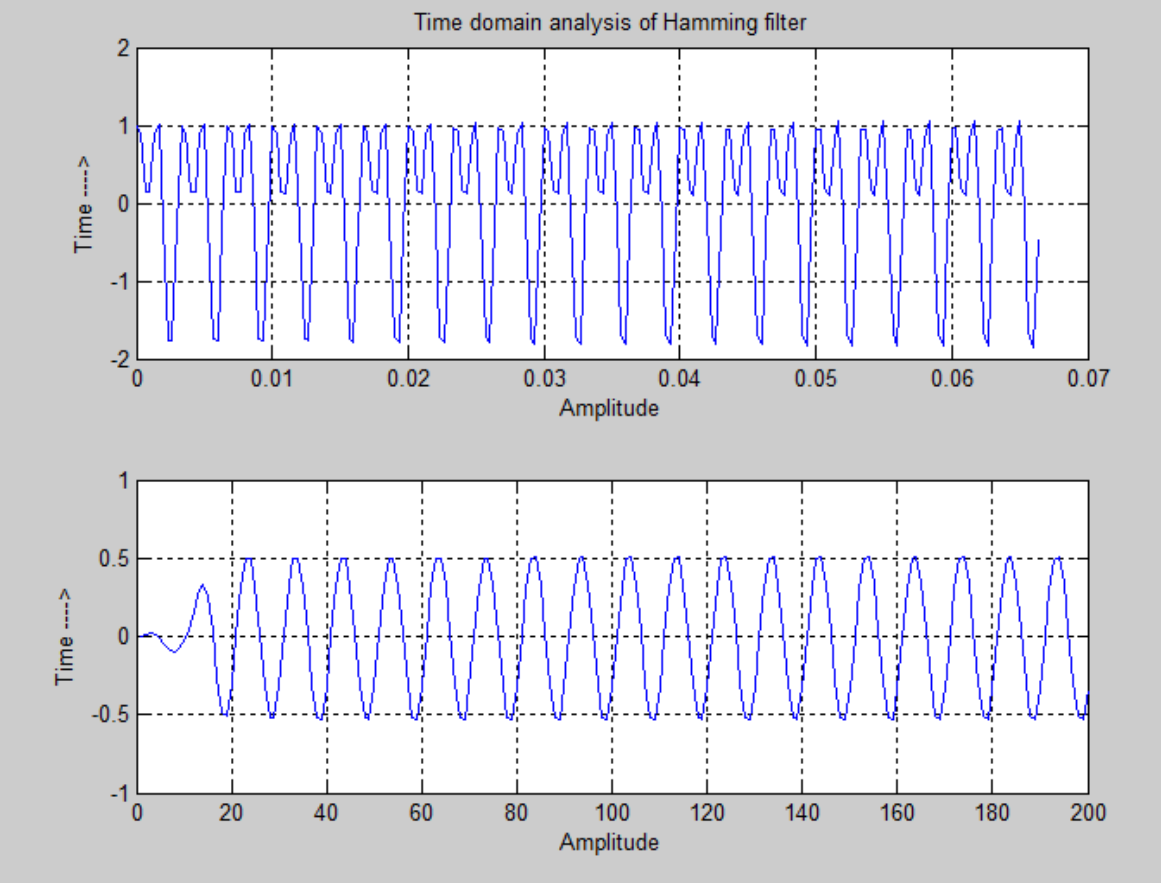
To be demonstrated:

For the given data below, construct a Hamming band pass filter and filter the input signal with frequencies as below:

Filter specifications:
First cut-off frequency – 250 Hz
Second cut-off frequency – 400 Hz
Order of the filter – 23
Input signal:
First input frequency – 300 Hz
Second input frequency – 600 Hz


Magnitude (dB)
0
-20
-40
-60
-80
-100
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
1000
0
-1000
-2000
-3000
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)


900
800
700
600
500
400
300
200
100
0
0 500 1000 1500 2000 2500 3000 3500



Command Window

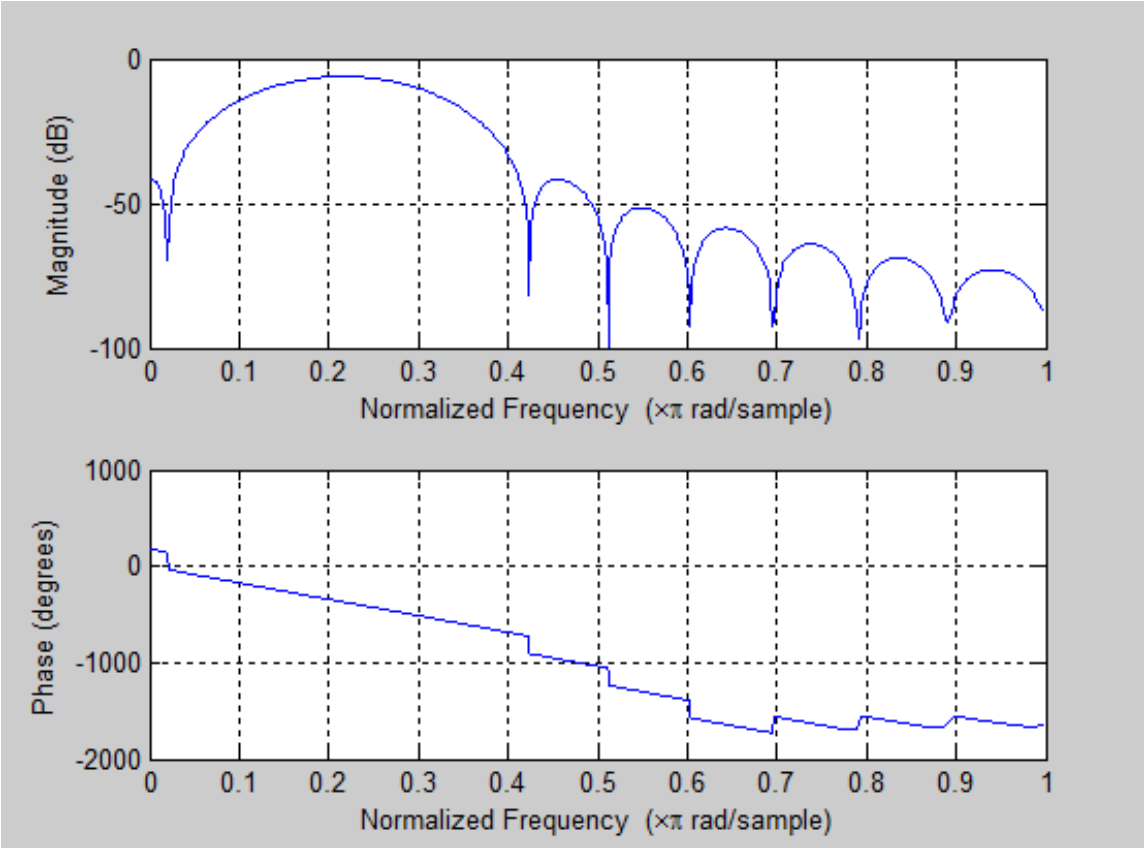
```
Enter the 1st input frequency :300
Enter the 2nd input frequency :600
Enter the cutoff first frequency :250
Enter the cutoff second frequency :400
Enter the Order of the filter :23
  Columns 1 through 7

    0.0055    0.0106    0.0121    0.0014   -0.0240   -0.0526   -0.0624

  Columns 8 through 14

   -0.0367    0.0190    0.0760    0.1000    0.0760    0.0190   -0.0368

  Columns 15 through 21

   -0.0625   -0.0528   -0.0241    0.0014    0.0121    0.0106    0.0055

  Columns 22 through 23

    0.0017   -0.0015

fx >>
```

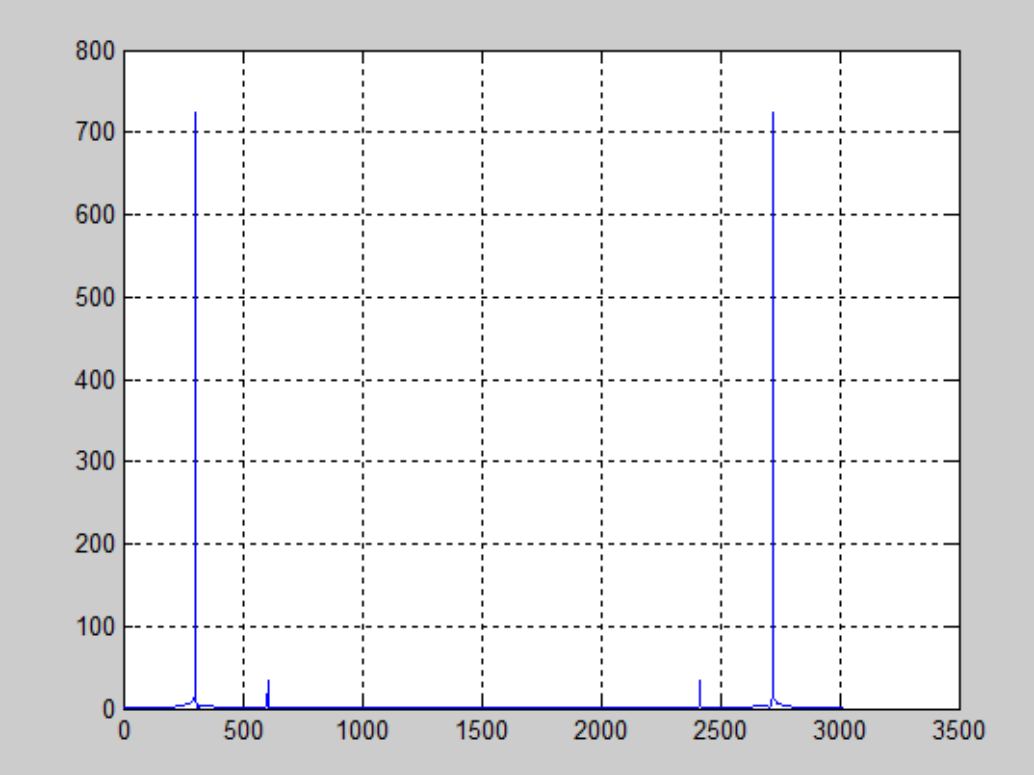
III. HANNING BAND PASS FILTER

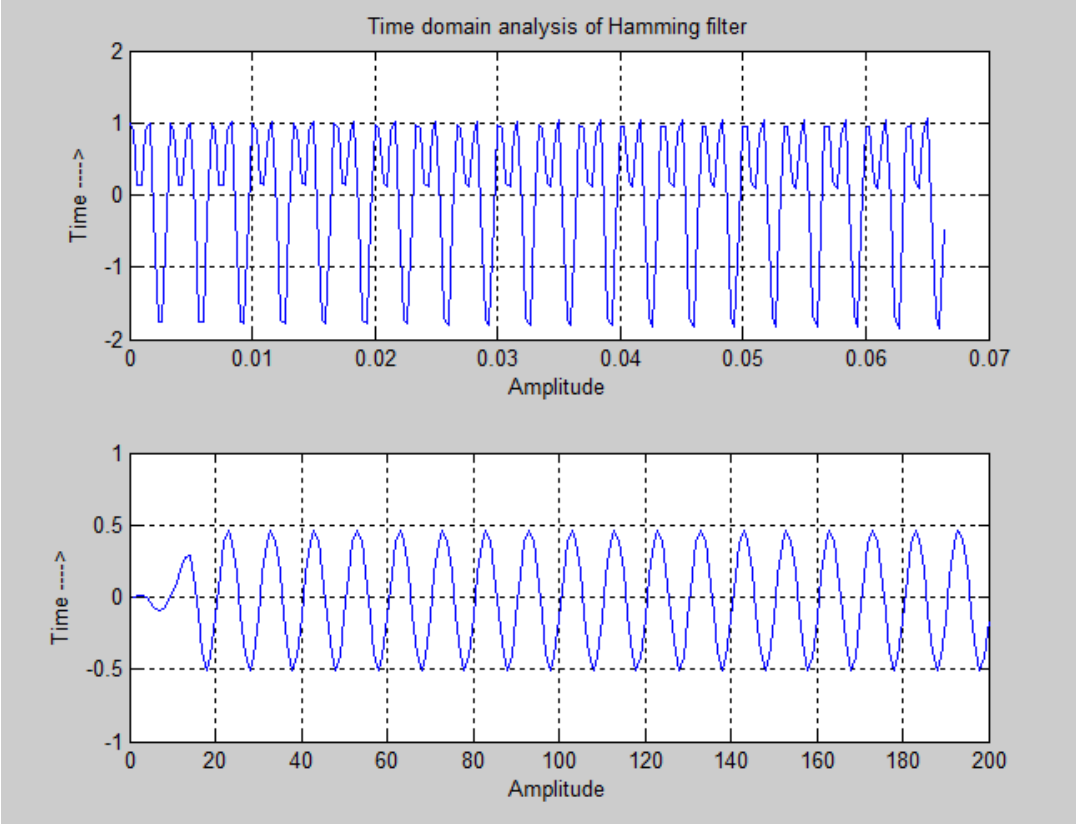
To be demonstrated:

For the given data below, construct a Hanning band pass filter and filter the input signal with frequencies as below:

Filter specifications:
First cut-off frequency – 250 Hz
Second cut-off frequency – 400 Hz
Order of the filter – 22
Input signal:
First input frequency – 300 Hz
Second input frequency – 600 Hz



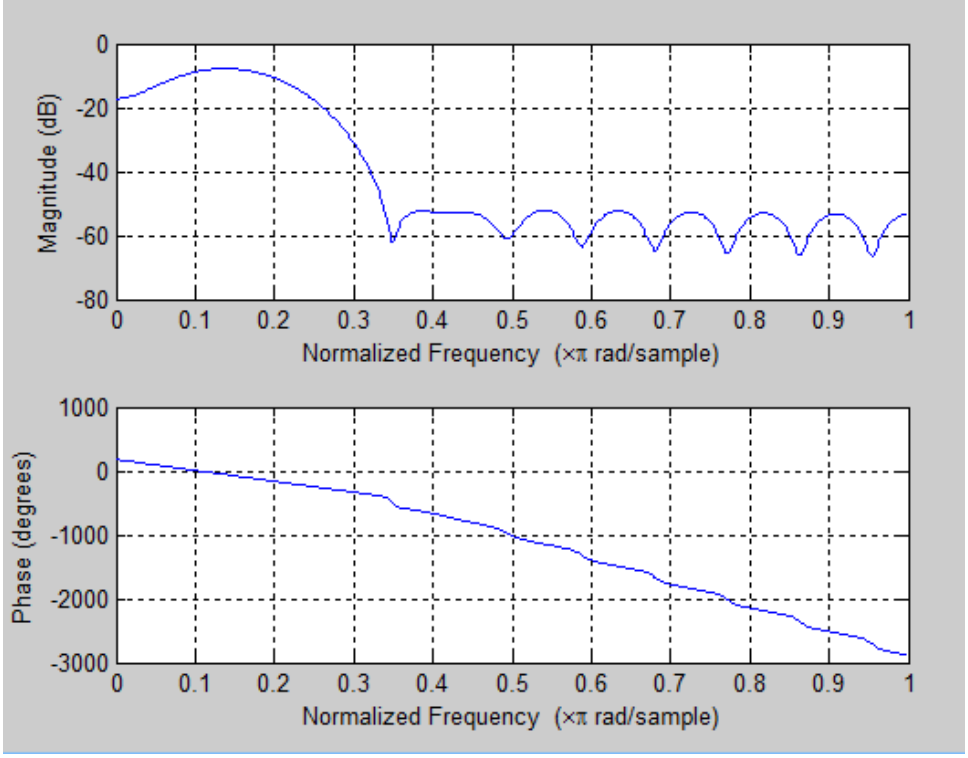

800
700
600
500
400
300
200
100
0
0
500
1000
1500
2000
2500
3000
3500

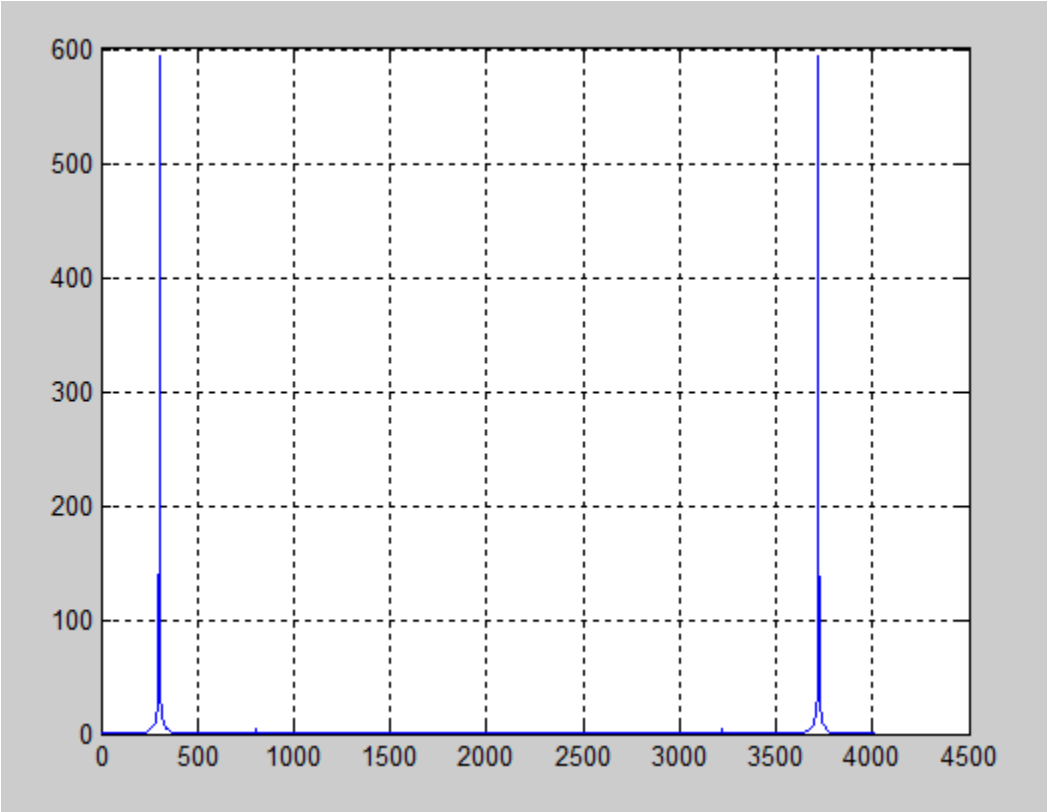

Time domain analysis of Hamming filter
Time ---->
Amplitude
Time ---->
Amplitude

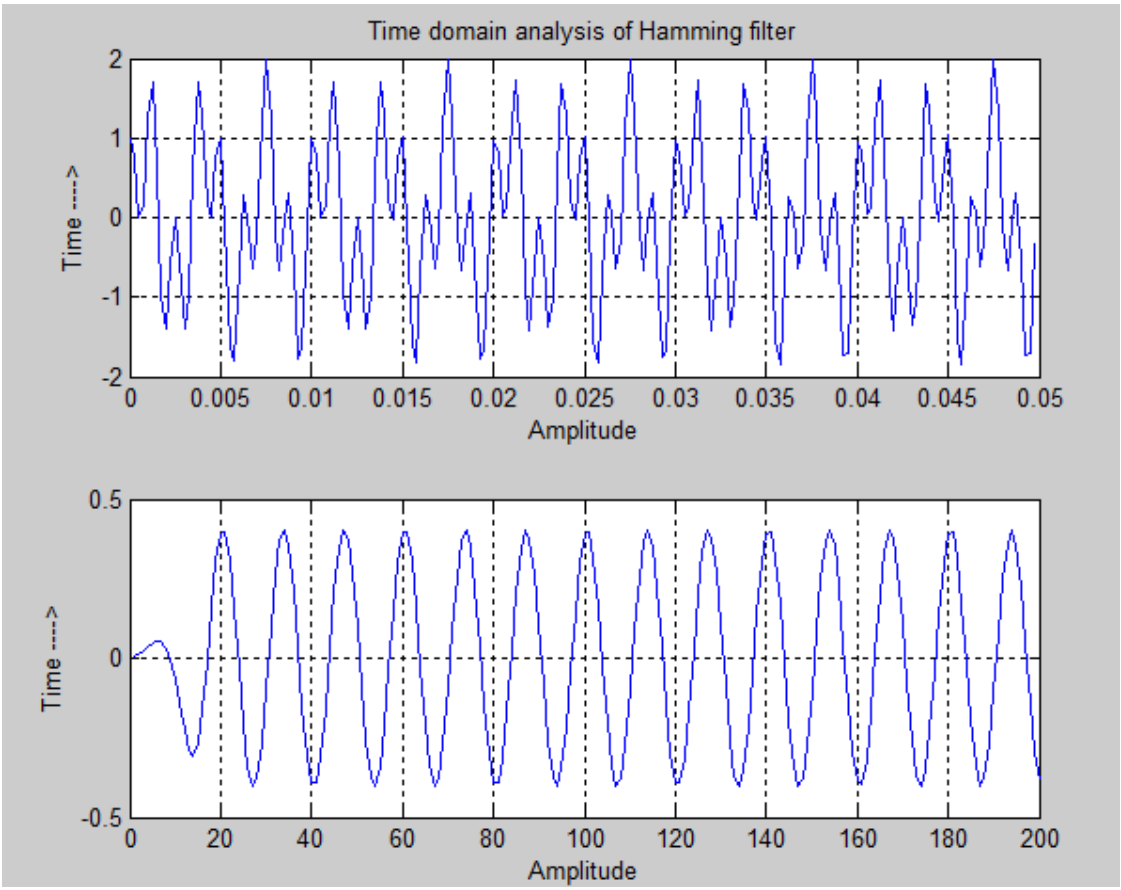
```
Command Window
Enter the 1st input frequency :300
Enter the 2nd input frequency :600
Enter the cutoff first frequency :250
Enter the cutoff second frequency :400
Enter the Order of the filter :22
  Columns 1 through 7

    0.0015    0.0055    0.0056   -0.0076   -0.0336   -0.0559   -0.0516

  Columns 8 through 14

   -0.0109    0.0492    0.0936    0.0937    0.0493   -0.0110   -0.0517

  Columns 15 through 21

   -0.0561   -0.0337   -0.0076    0.0056    0.0056    0.0015    0.0000

  Column 22

    0.0000
```

IV. HANNING BAND ELIMINATION FILTER

To be demonstrated: For the given data below, construct a Hanning band elimination filter and filter the input signal with frequencies as below:
Filter specifications:
First cut-off frequency – 200 Hz
Second cut-off frequency – 350 Hz
Order of the filter – 22
Input signal:
First input frequency – 300 Hz
Second input frequency – 800 Hz


Magnitude (dB)
0
-20
-40
-60
-80
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
1000
0
-1000
-2000
-3000
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency (×π rad/sample)


600
500
400
300
200
100
0
0 500 1000 1500 2000 2500 3000 3500 4000 4500



Command Window

```
Enter the 1st input frequency :300
Enter the 2nd input frequency :800
Enter the cutoff first frequency :200
Enter the cutoff second frequency :350
Enter the Order of the filter :22
  Columns 1 through 7

    0.0035    0.0087    0.0165    0.0238    0.0254    0.0167   -0.0033

  Columns 8 through 14

   -0.0306   -0.0568   -0.0728   -0.0728   -0.0568   -0.0306   -0.0034

  Columns 15 through 21

    0.0167    0.0255    0.0239    0.0166    0.0088    0.0035    0.0008

  Column 22

   -0.0014
```

b) Write a MATLAB code to verify the Low pass Butterworth IIR filter design using bilinear transformation (BLT) method and Impulse Invariant Technique (IIT) method.

Outcome: The student will be able to design and implement an IIR filter that best matches the application and satisfies the design requirements.

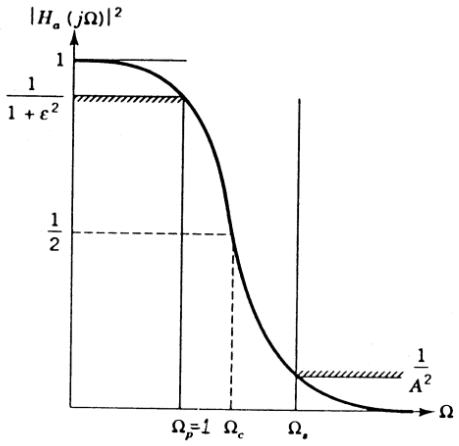
In the impulse invariance technique, the frequency response of analog and digital filters are similar only at low frequencies. This limits the application of impulse invariance method to only low pass filters. A better transformation that preserves frequency response and overcomes the limitations of impulse invariance method is called bilinear transformation method. The warping effect in impulse invariance between the analog and digital frequencies is eliminated by pre-warping the analog frequencies.

BUTTERWORTH FILTERS

- For the Nth-order Butterworth filter the squared-magnitude function is given by

$$|H_a(j\Omega)|^2 = \frac{1}{1+\epsilon^2\Omega^{2N}}.$$

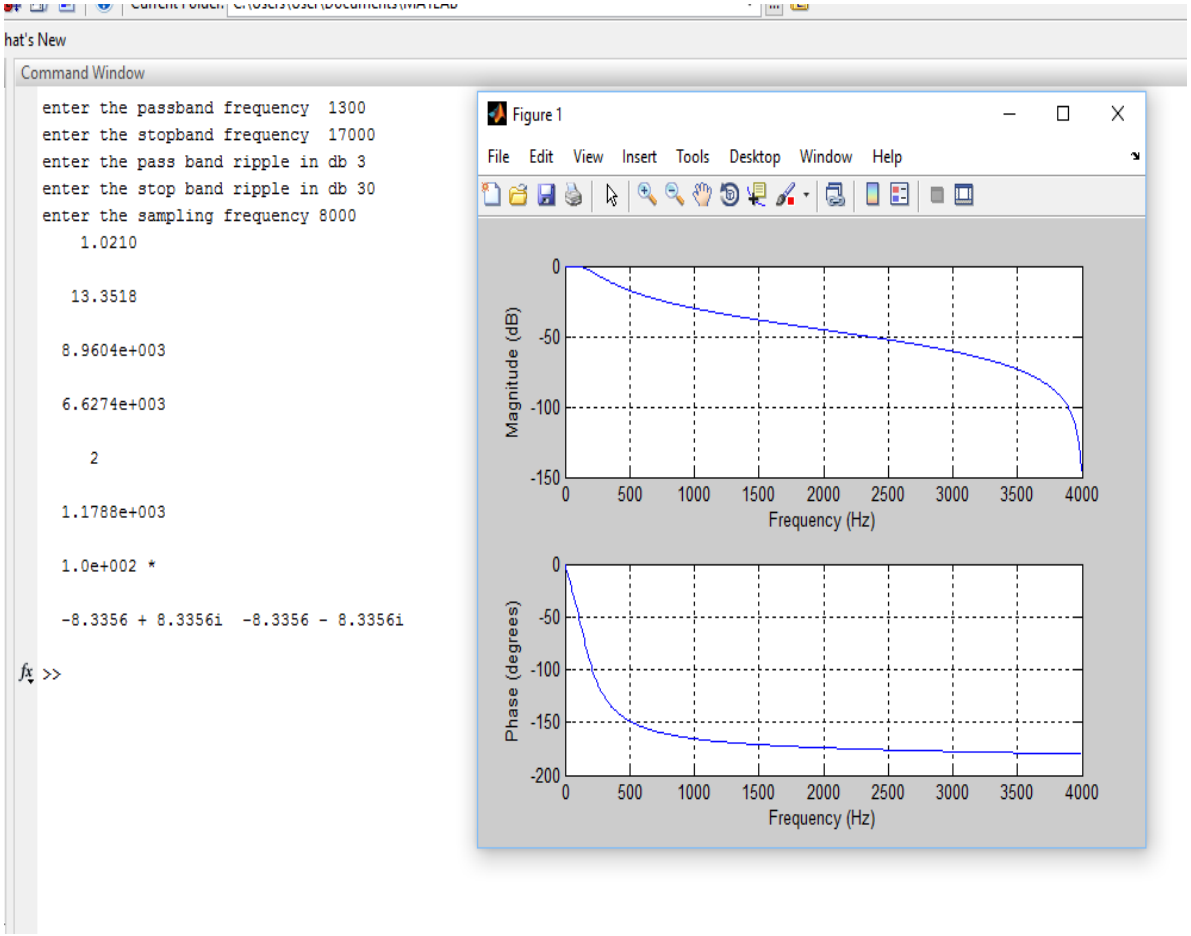
- This function achieves the value of unity at $\Omega = 0$ with the first $2N-1$ derivatives being zero at this point (**maximally flat passband**).
- At infinity, the value is zero and the first $2N-1$ derivatives are zero (**maximally flat stopband**).
  - The following figure gives the normalized specifications, where $\Omega_p = 1$, $|H_a(j\Omega_p)|^2 = 1/(1+\epsilon^2)$, and it is required that $|H_a(j\Omega_s)|^2 \leq 1/A^2$.



I. LOW PASS BUTTERWORTH IIR FILTER DESIGN USING BILINEAR TRANSFORMATION TECHNIQUE (BLT) METHOD

To be demonstrated: For the given data below, construct a Low pass Butterworth IIR filter design using Bilinear Transformation (BLT) method and filter the input signal with filter specifications as below:

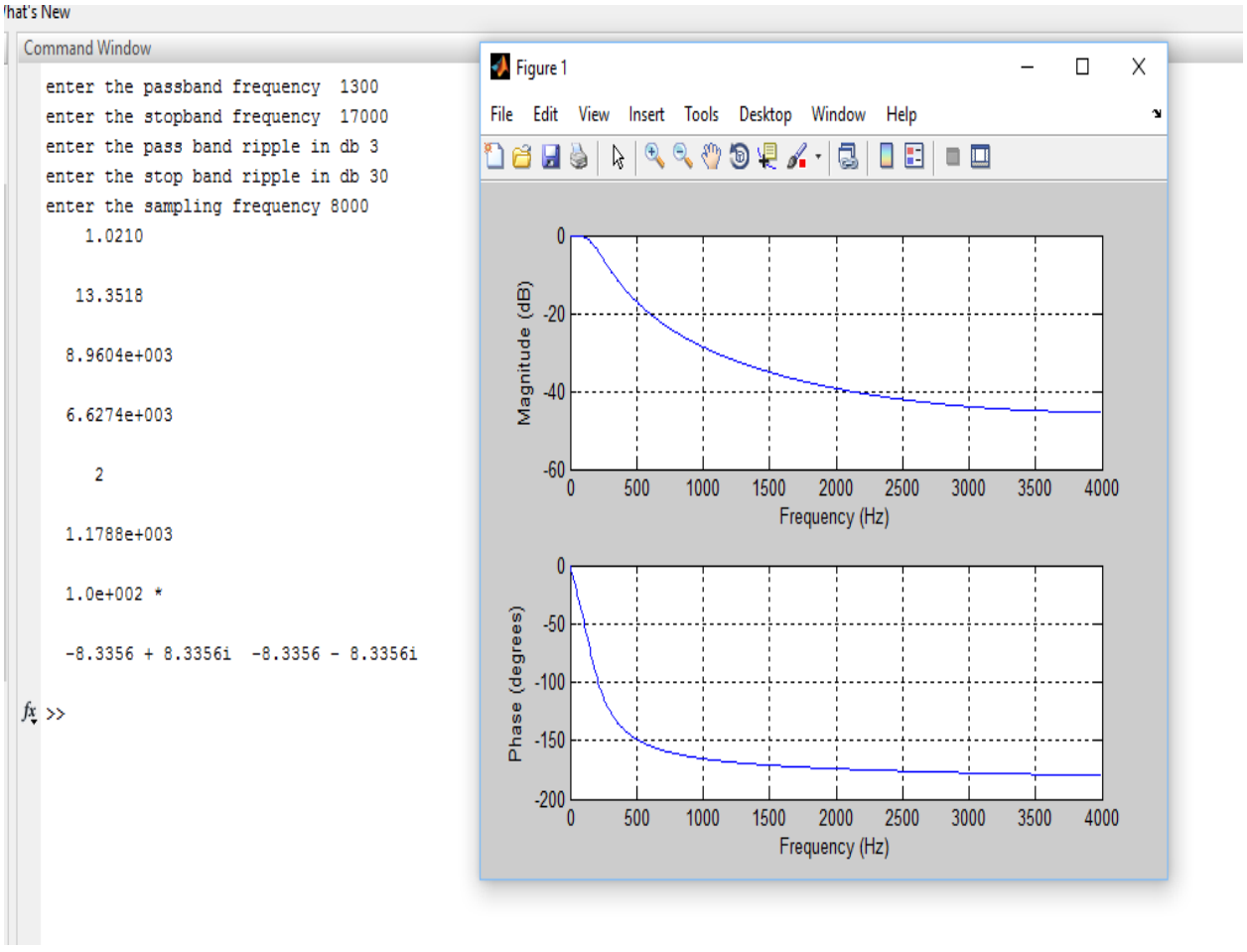
Filter specifications:
Pass band edge frequency – 1300 Hz
Stop band edge frequency – 17000 Hz
Pass band attenuation in db - 3
Stop band attenuation in db – 30
Sampling frequency – 8000 Hz



II. LOW PASS BUTTERWORTH IIR FILTER DESIGN USING IMPULSE INVARIANT TECHNIQUE (IIT) METHOD

To be demonstrated: For the given data below, construct a Low pass Butterworth IIR filter design using Impulse Invariant Technique (IIT) method and filter the input signal with filter specifications as below:

Filter specifications:
Pass band edge frequency – 1300 Hz
Stop band edge frequency – 17000 Hz
Pass band attenuation in db - 3
Stop band attenuation in db – 30
Sampling frequency – 8000 Hz



c) Write a MATLAB code to implement the Low pass Chebyshev (Type 1) IIR filter design using bilinear transformation (BLT) method and Impulse Invariant Technique (IIT) method.

CHEBYSHEV FILTERS OR CHEBYSHEV TYPE I FILTERS

- The squared-magnitude function of this filter of order N is given by

$$|H_a(j\Omega)|^2 = \frac{1}{1+\epsilon^2 T_N^2(\Omega)}, \qquad (16a)$$

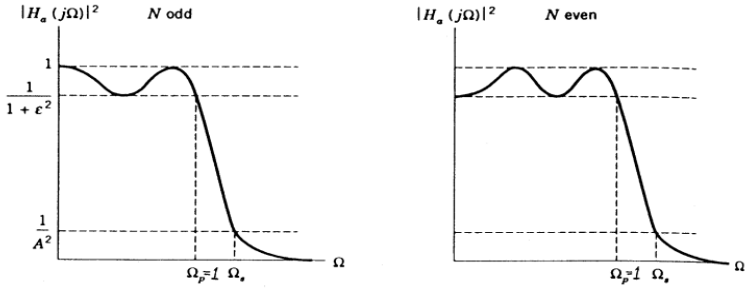
where

$$T_N(\Omega) = \begin{cases} \cos(N\cos^{-1}\Omega), & |\Omega| \le 1 \\ \cosh(N\cosh^{-1}\Omega) & |\Omega| > 1 \end{cases} \qquad (16b)$$

is the Nth-degree Chebyshev polynomial.

- In the normalized passband $0 \le \Omega \le \Omega_p = 1$, this function alternatingly achieves the values of 1 and $1/(1+\epsilon^2)$ at $N+1$ points such that $|H_a(j\Omega_p)|^2 = 1/(1+\epsilon^2)$. For N even, $|H_a(j0)|^2 = 1/(1+\epsilon^2)$ and for N odd, $|H_a(j0)|^2 = 1$ (**equiripple passband**).

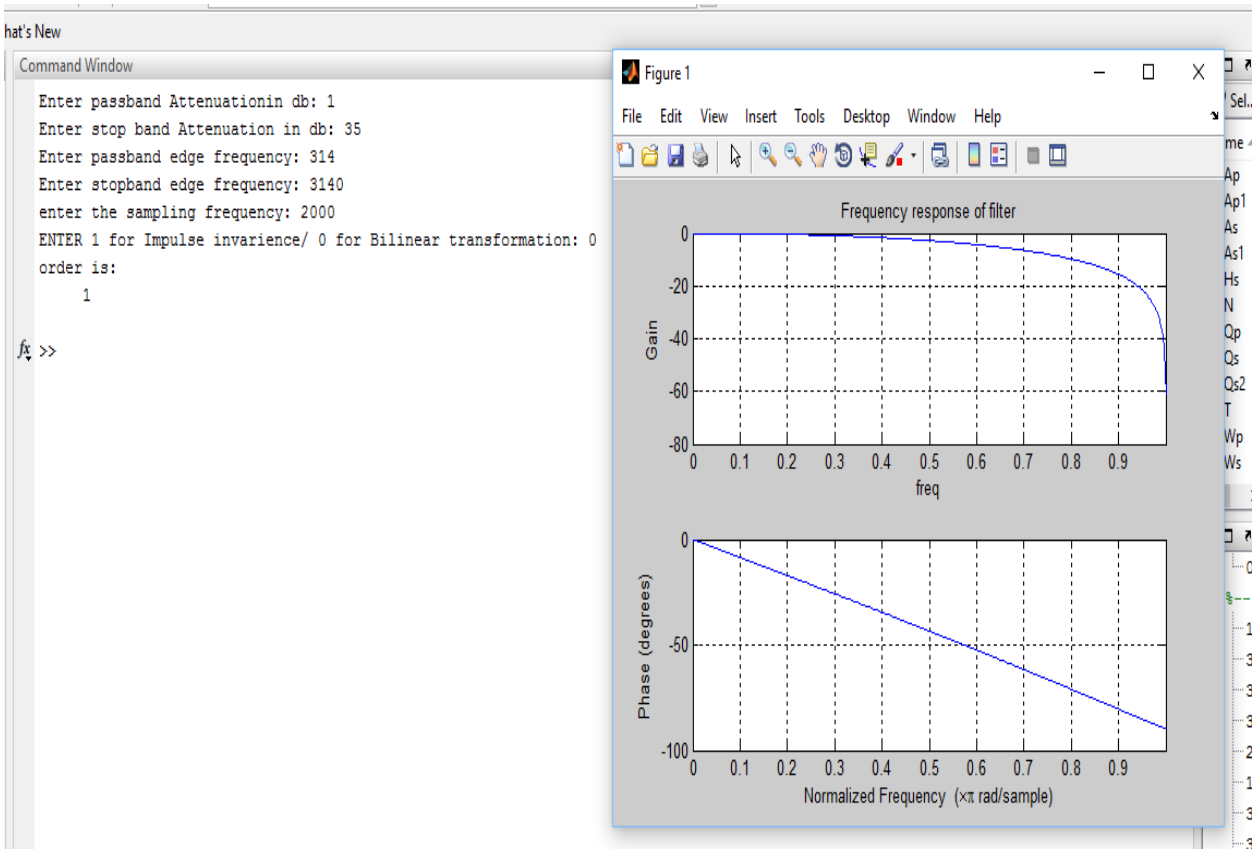
- At infinity, the value of $|H_a(j\Omega)|^2$ is zero and the first $2N-1$ derivatives are zero (**maximally flat stopband**) (see the figure shown below).



I. LOW PASS CHEBYSHEV IIR FILTER DESIGN USING BILINEAR TRANSFORMATION (BLT) METHOD

To be demonstrated: For the given data below, construct a Low pass Butterworth IIR filter design using bilinear transformation (BLT) method and filter the input signal with filter specifications as below:

Filter specifications:
Pass band edge frequency – 314 Hz
Stop band edge frequency – 3140 Hz
Pass band attenuation in db - 1
Stop band attenuation in db – 35
Sampling frequency – 2000 Hz



II. LOW PASS CHEBYSHEV IIR FILTER DESIGN USING IMPULSE INVARIANT TECHNIQUE (IIT) METHOD

To be demonstrated: For the given data below, construct a Low pass Butterworth IIR filter design using Impulse Invariant Technique (IIT) method and filter the input signal with filter specifications as below:

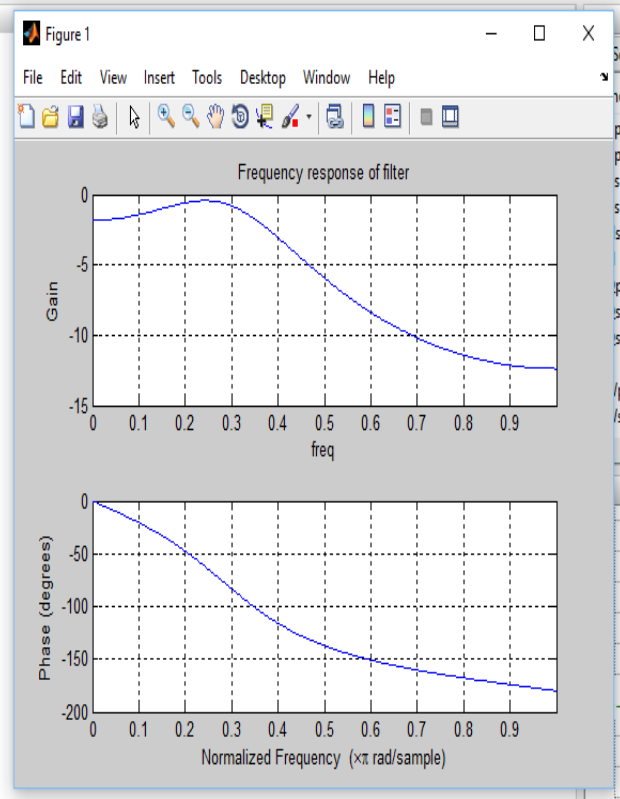
Filter specifications:
Pass band edge frequency – 314 Hz
Stop band edge frequency – 3140 Hz
Pass band attenuation in db - 1
Stop band attenuation in db – 35
Sampling frequency – 2000 Hz

Current Folder: C:\Users\User\Documents\MATLAB

at's New

Command Window

```
Enter passband Attenuationin db: 1
Enter stop band Attenuation in db: 35
Enter passband edge frequency: 314
Enter stopband edge frequency: 3140
enter the sampling frequency: 2000
ENTER 1 for Impulse invarience/ 0 for Bilinear transformation: 1
order is:
     2

>>
```



Experiment-6

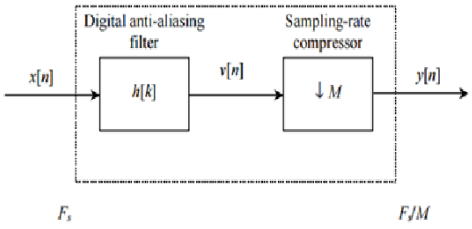
Write a MATLAB code to illustrate the effect of Decimation and Interpolation by an integer factor. Plot the magnitude spectrum. Design the necessary filter to overcome aliasing and image frequencies after decimating and interpolating the signal respectively.

Outcome: The student will be able to implement fundamental operations in multirate DSP.

I. DECIMATION

Decimation can be regarded as the discrete-time counterpart of sampling. Whereas in sampling we start with a continuous-time signal x(t) and convert it into a sequence of samples x[n], in decimation we start with a discrete-time signal x[n] and convert it into another discrete-time signal y[n], which consists of sub-samples of x[n]. Thus, the formal definition of M-fold decimation, or down-sampling, is defined by Equation 9.1. In decimation, the sampling rate is reduced from Fs to Fs/M by discarding M – 1 samples for every M samples in the original sequence.

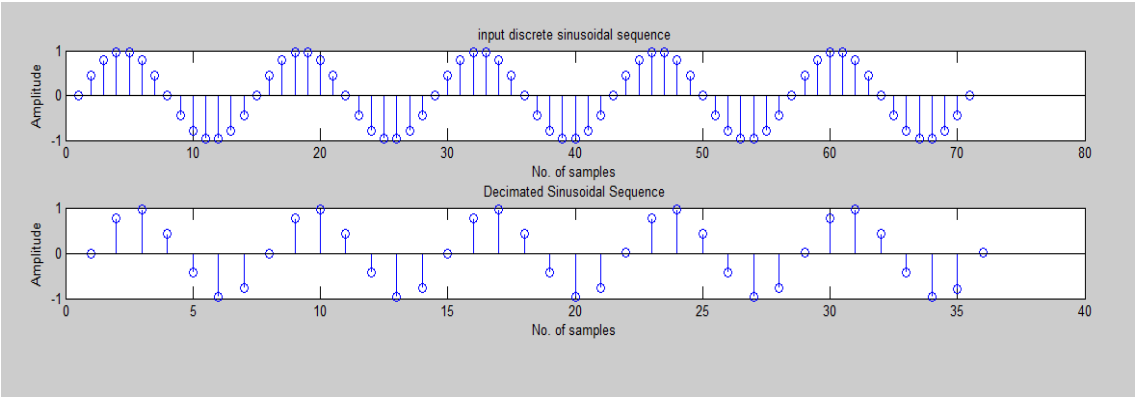
$$y[n] = v[nM] = \sum_{k=-\infty}^{\infty} h[k]x[nM-k]$$



Block diagram notation of decimation, by a factor of M.

To be demonstrated: Generate a sinusoidal signal of specifications given below and decimate it by the specified factor.

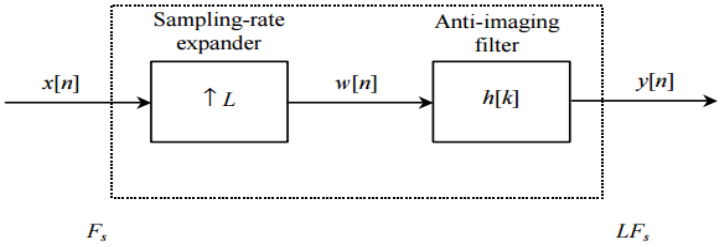
Signal specifications:
Input frequency – 10 Hz
Sampling frequency – 140 Hz
Decimation factor – 2



I. INTERPOLATION

Interpolation is the exact opposite of decimation. It is an information preserving operation, in that all samples of x[n] are present in the expanded signal y[n]. Interpolation works by inserting (L–1) zero-valued samples for each input sample. The sampling rate therefore increases from Fs to LFs. The expansion process is followed by a unique digital low-pass filter called an anti-imaging filter. Although the expansion process does not cause aliasing in the interpolated signal, it does however yield undesirable replicas in the signal's frequency spectrum.

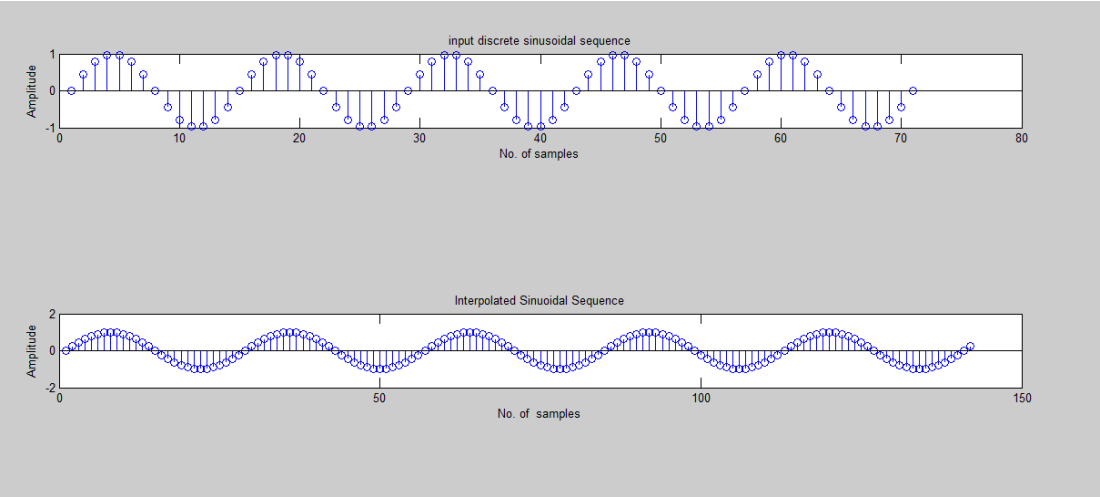
$$y[n] = L\sum_{k=-\infty}^{\infty} h[k]w[n-k]$$



Block diagram notation of interpolation, by a factor of L.

To be demonstrated: Generate a sinusoidal signal of specifications given below and decimate it by the specified factor.

Signal specifications:
Input frequency – 10 Hz
Sampling frequency – 140 Hz
Interpolation factor – 2

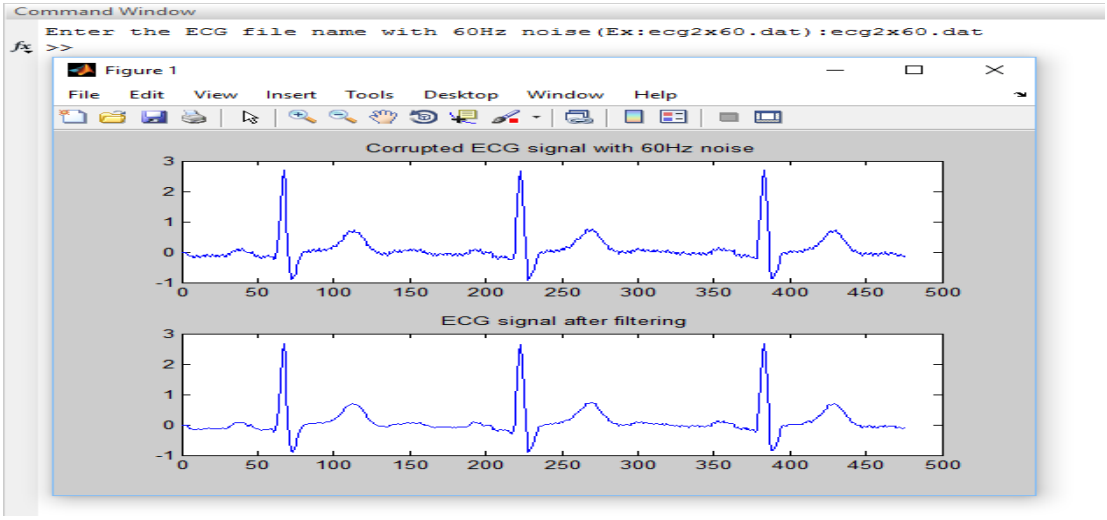


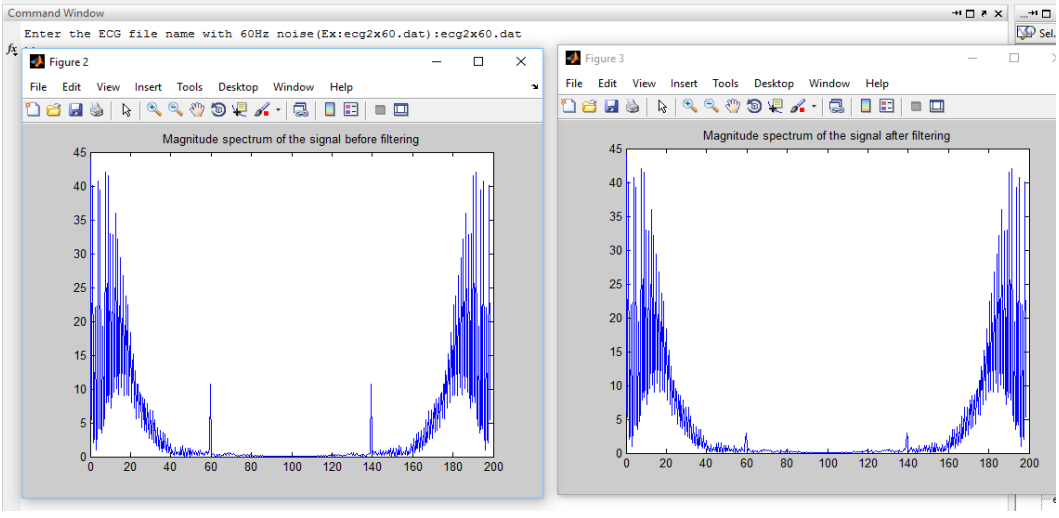
Experiment-7

Read the data file named ecg2x60.dat from http://people.ucalgary.ca/~ranga/enel563/SIGNAL_DATA_FILES/) that is corrupted with the 60Hz noise component. Write a MATLAB code to remove this 60Hz noise component from the signal using Notch filter and LMS adaptive filter. Plot the magnitude spectrum of the signal filtered using both Notch filter and LMS adaptive filter and provide the inference on the basis of results obtained.

Outcome: The student will be able to apply adaptive filtering concept to remove the noise from the signal, whose statistics are not known in prior.

Least mean squares (LMS) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time. It was invented in 1960 by Stanford University professor Bernard Widrow and his first Ph.D. student, Ted Hoff.

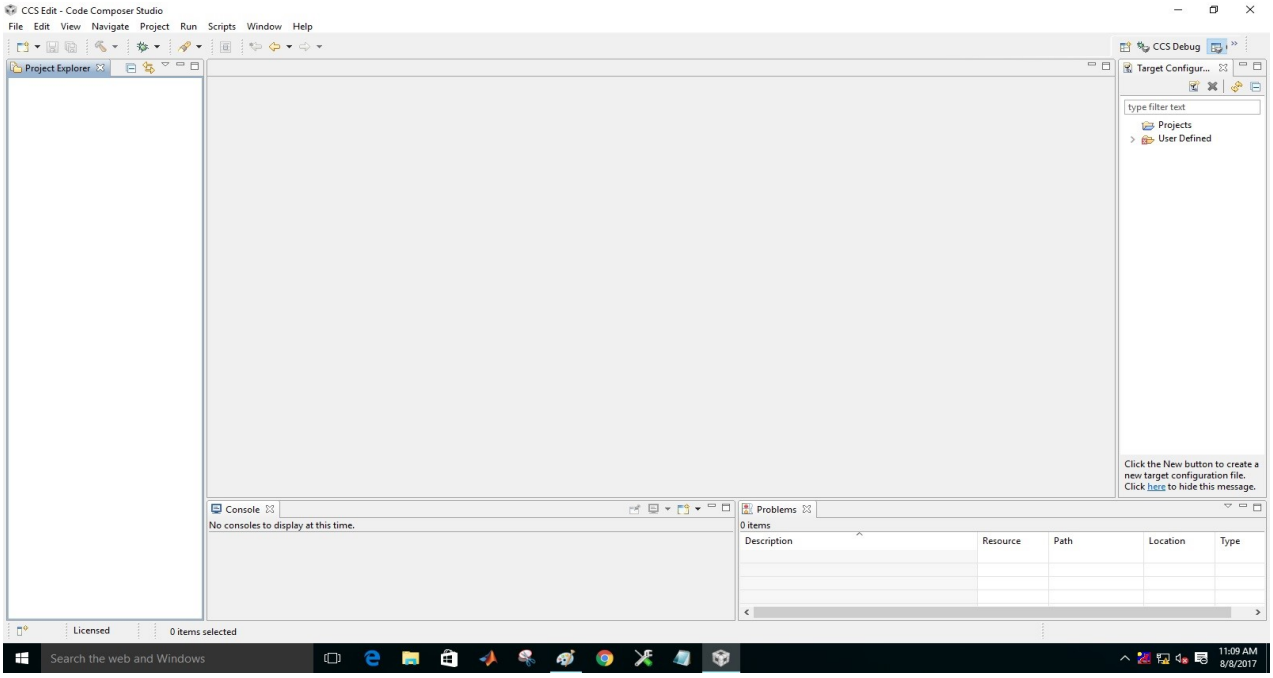




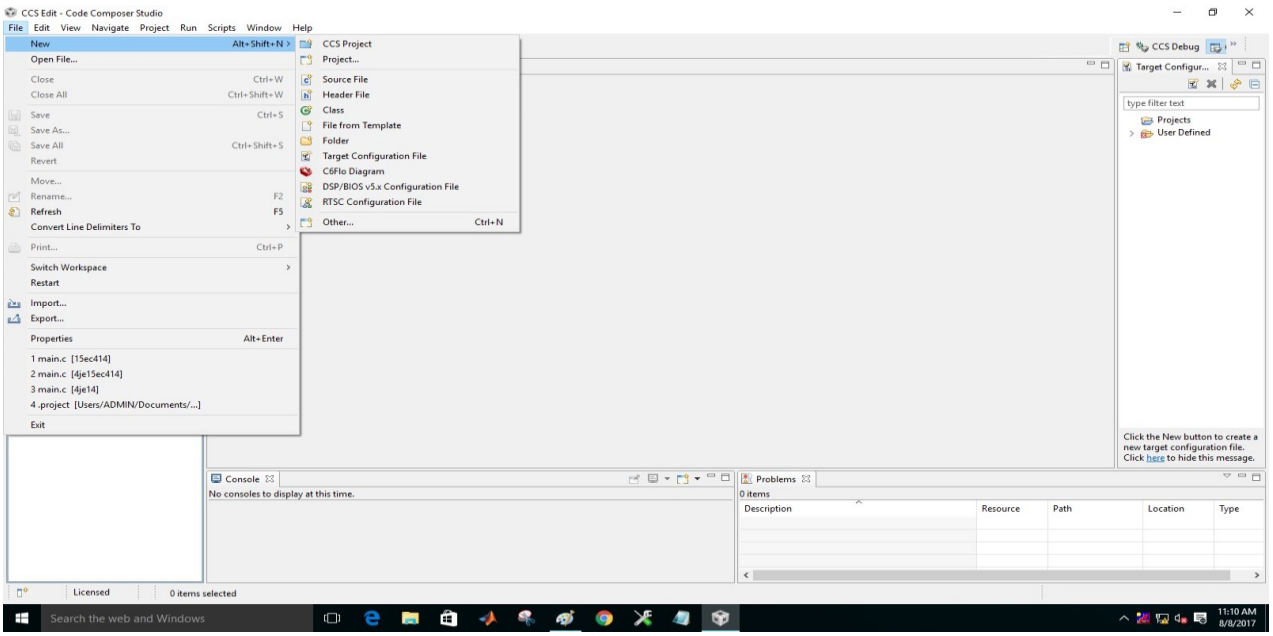
Experiment-8

Procedure to use CCS Studio Version 5.0

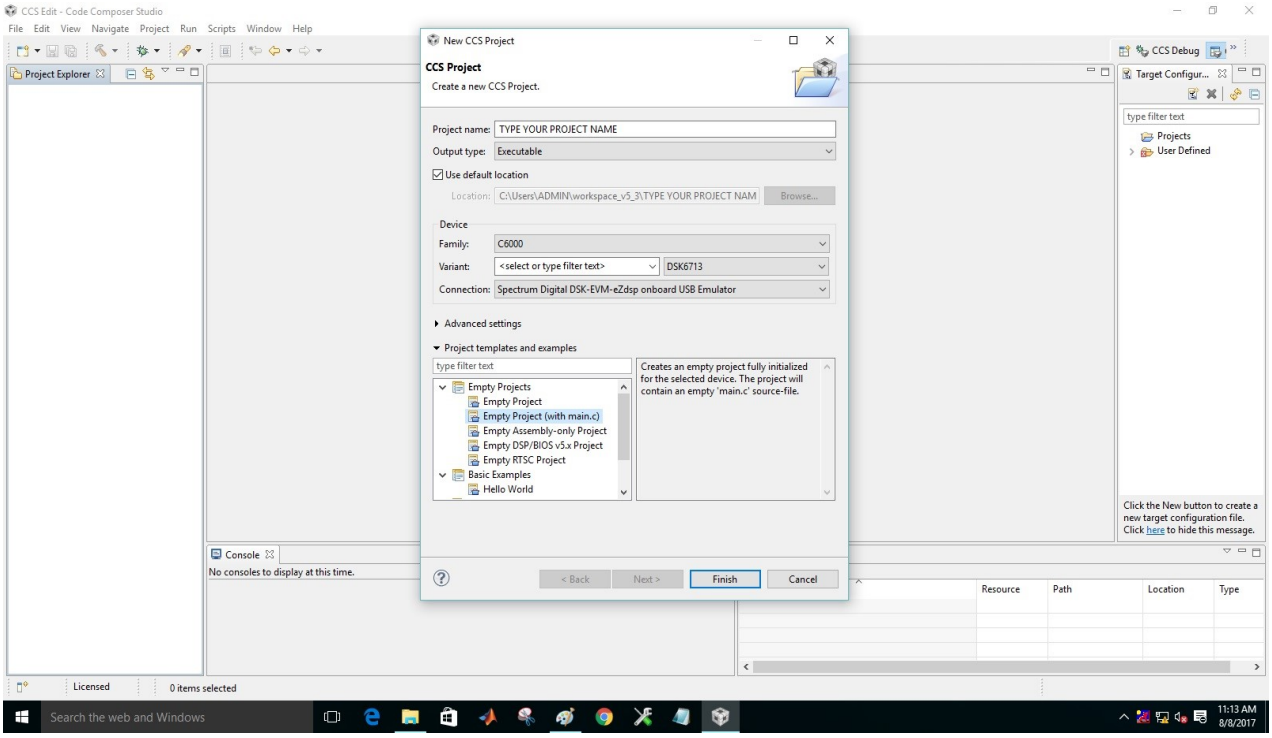
Step-1: Double click on CCS Studio icon and the following screen will appear. We can see the project explorer window towards the left, console at the middle and target configuration files towards the right side of the screen.



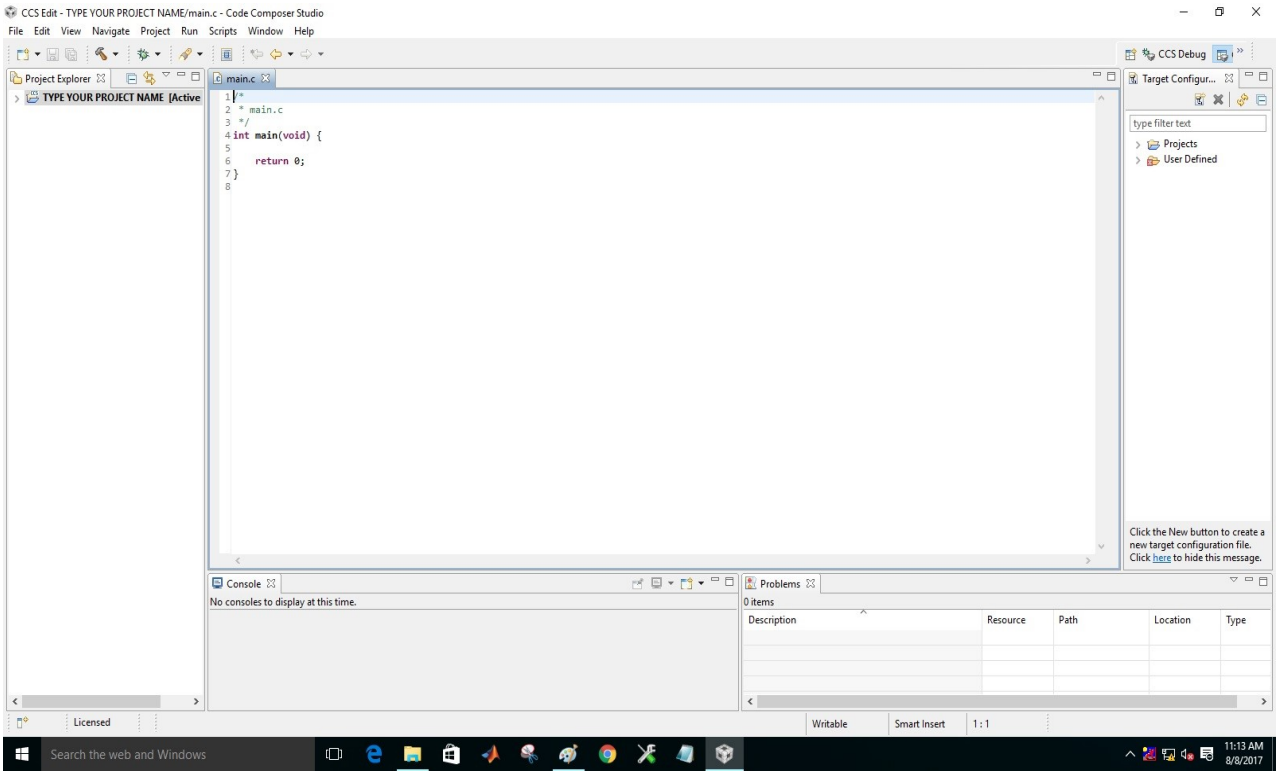
Step-2: Click on File → New → CCS Project



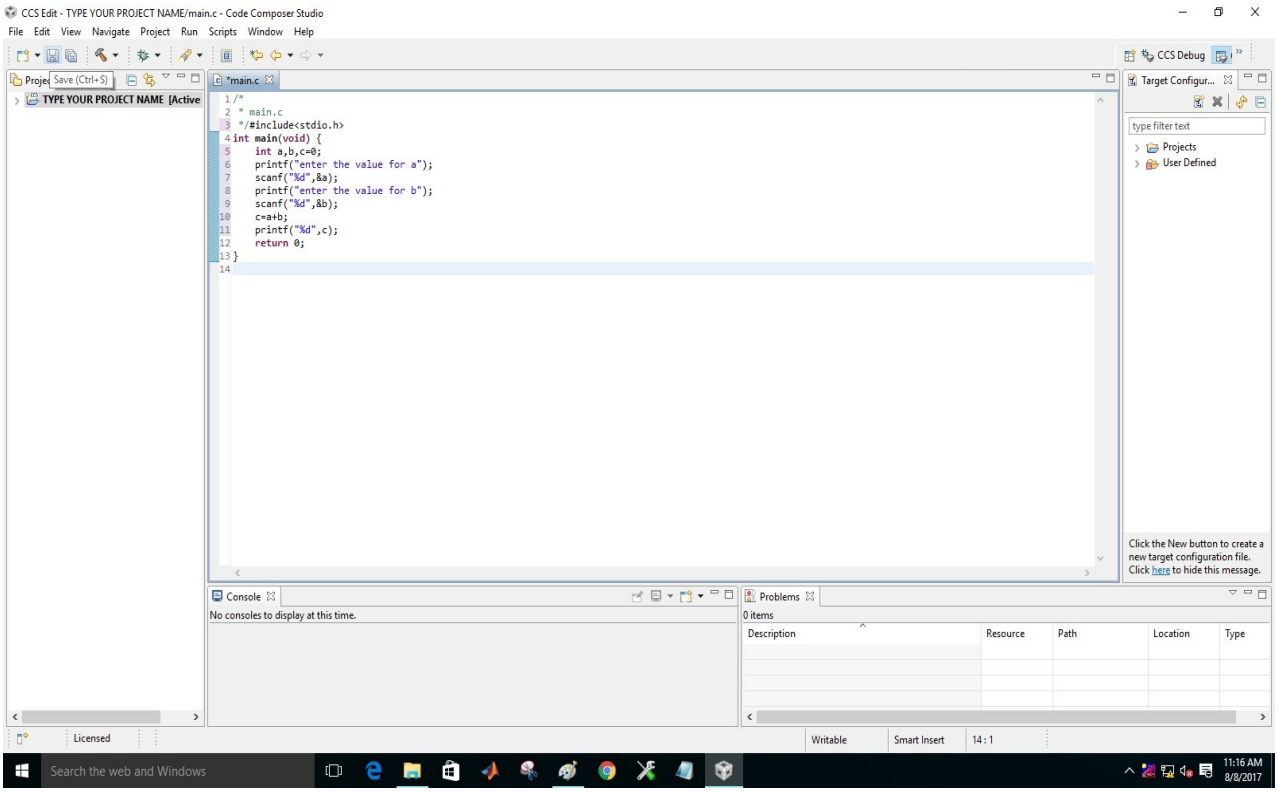
Step-3: Type your Project name, select Output type as Executable from the dropdown menu, Device Family as C6000, Variant as DSK6713, Connection as Spectrum Digital DSK EVM eZdsp USB Emulator, click on Empty projects (with main.c) & finish.



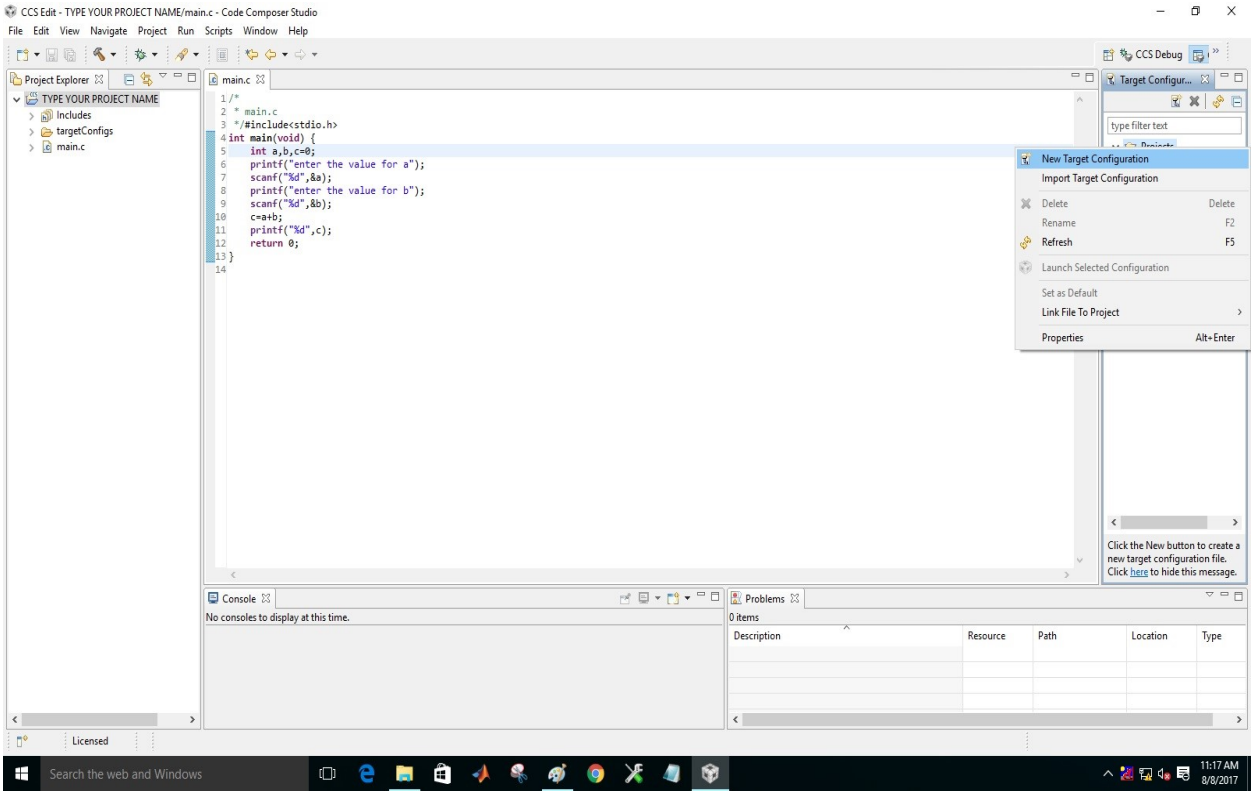
Step-4: Screen as shown below will appear with the Project Name.



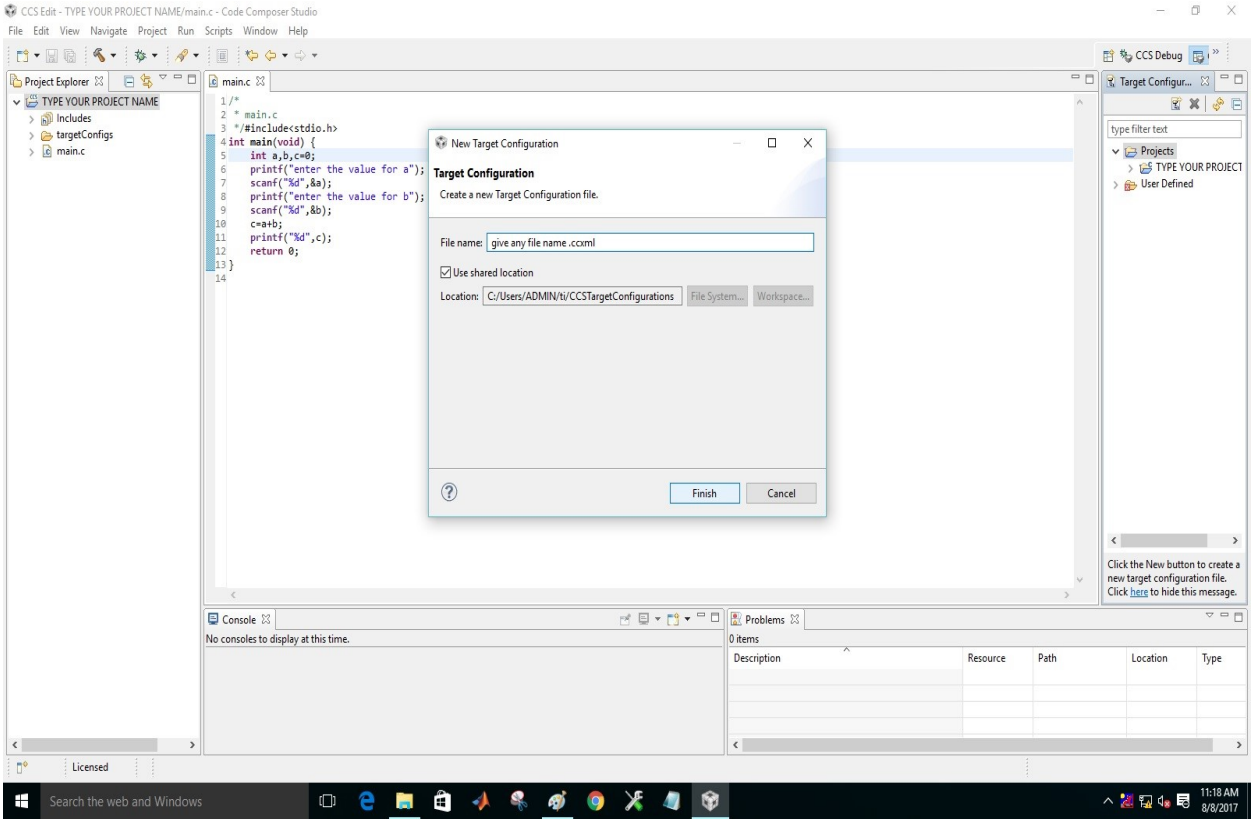
Step-5: Write your C code and then click on save.



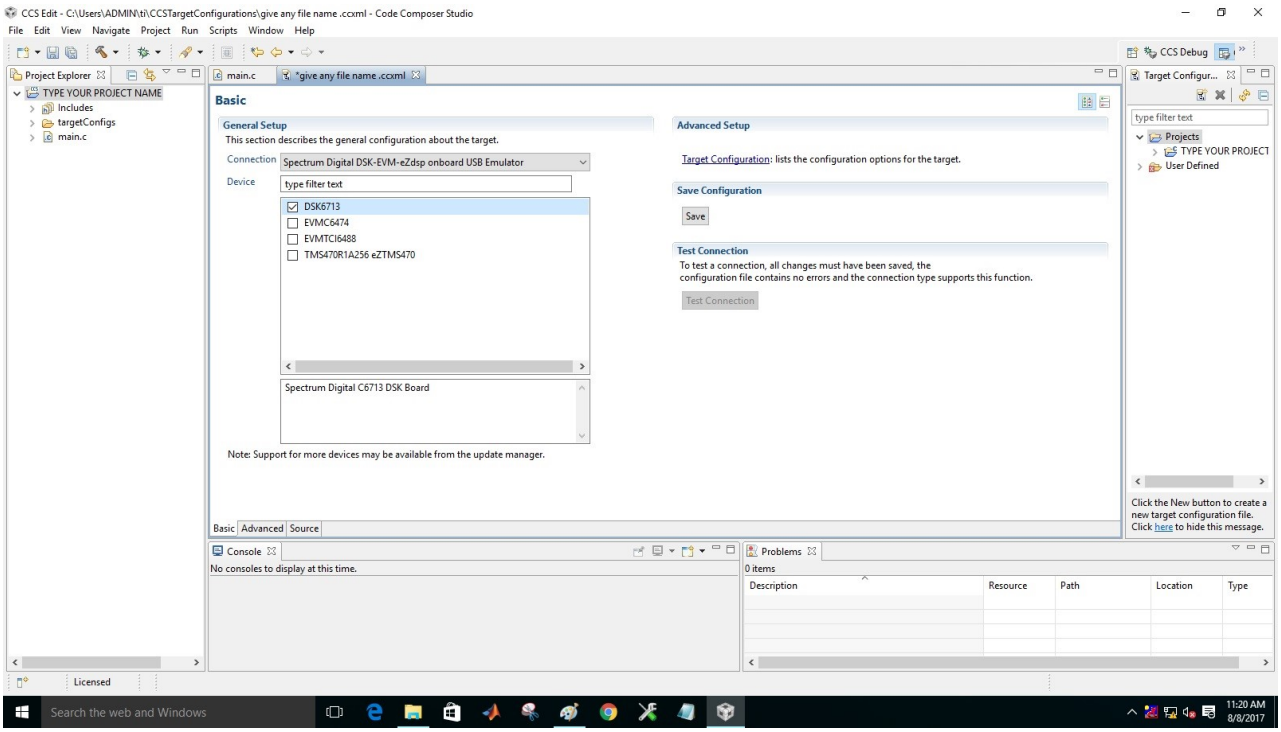
Step-6: Right click on projects and select New Target Configuration.



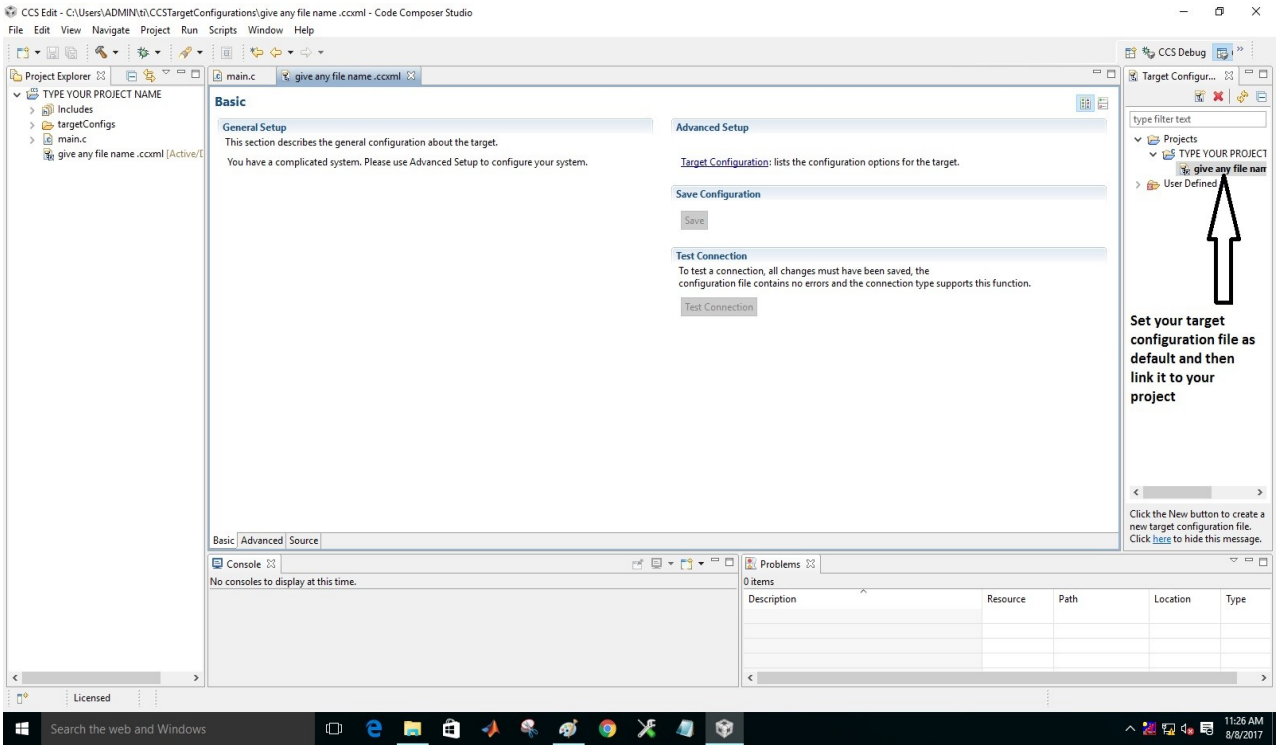
Step-7: Give the file name with .ccxml extension and click finish.



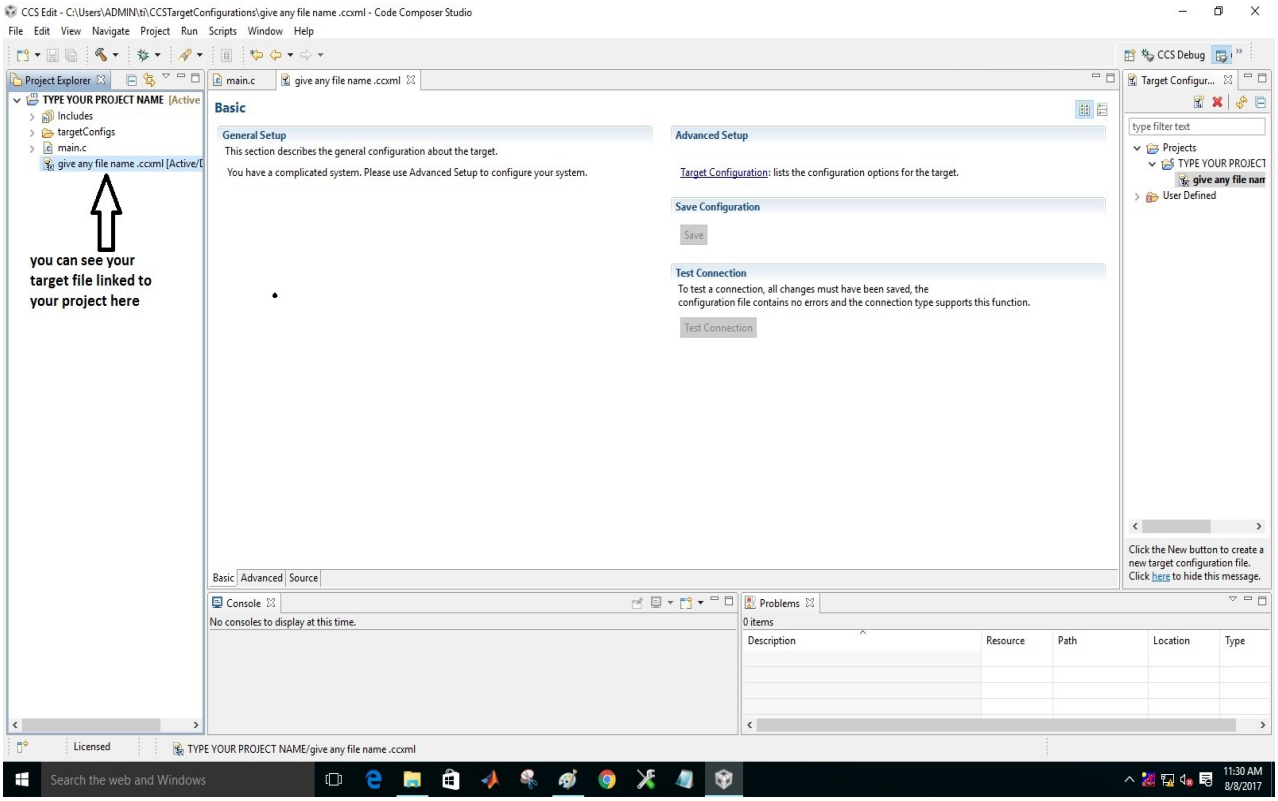
Step-8: Select the connection as Spectrum Digital DSK EVM eZdsp onboard USB Emulator and then select device as DSK6713 and click save.



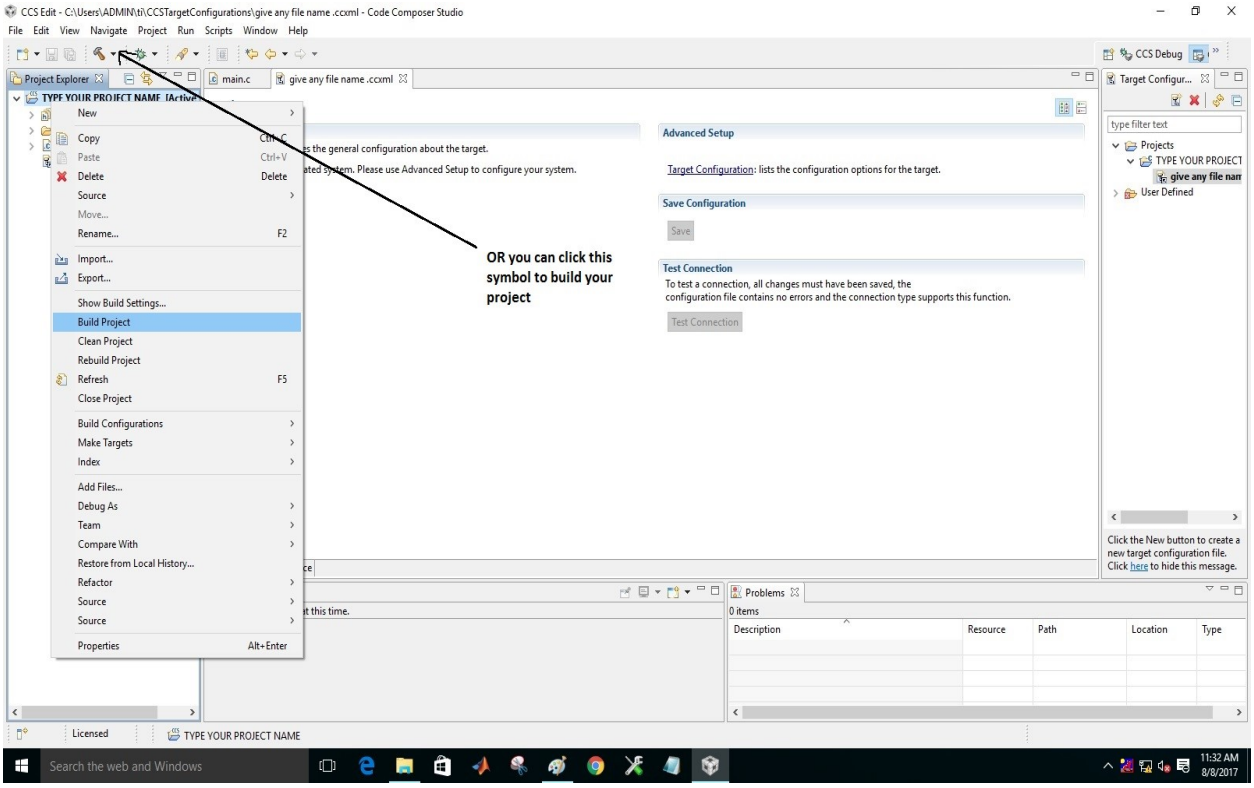
Step-9: Target Configuration file will appear as shown using the arrow symbol.

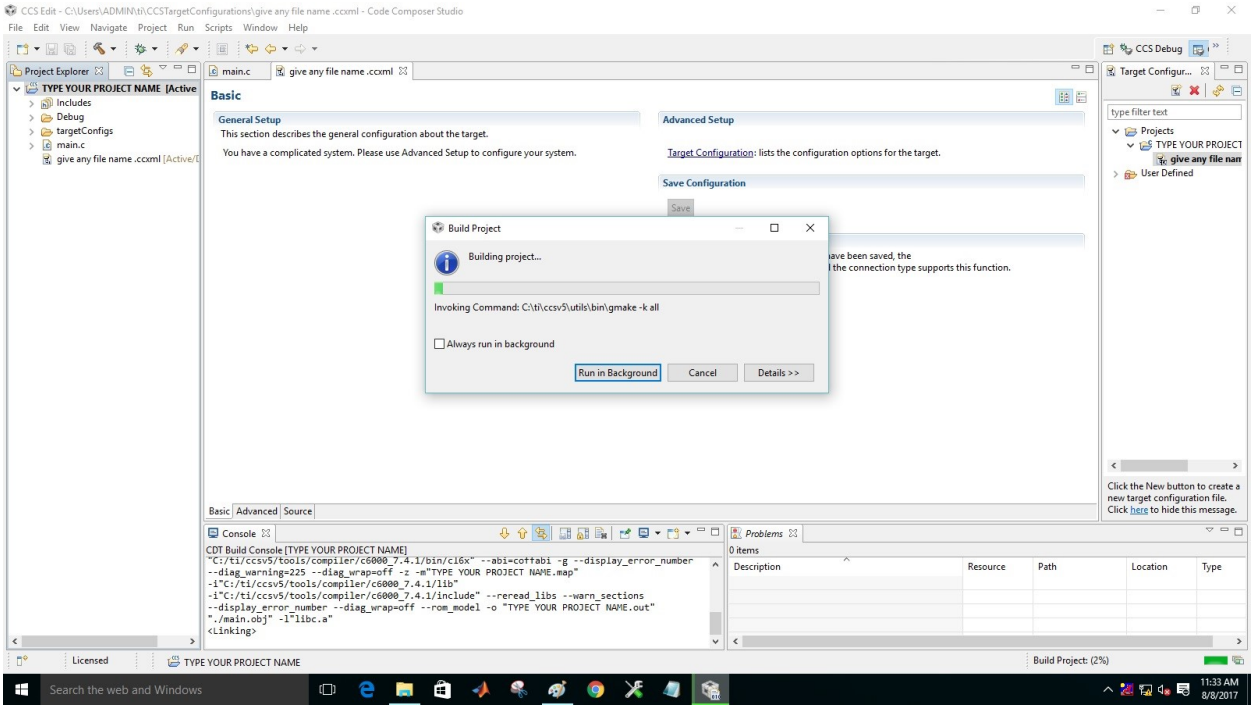


Step-10: Right click on the target configuration file, select link file to project and now you can see the target configuration file linked to your project as shown using arrow symbol

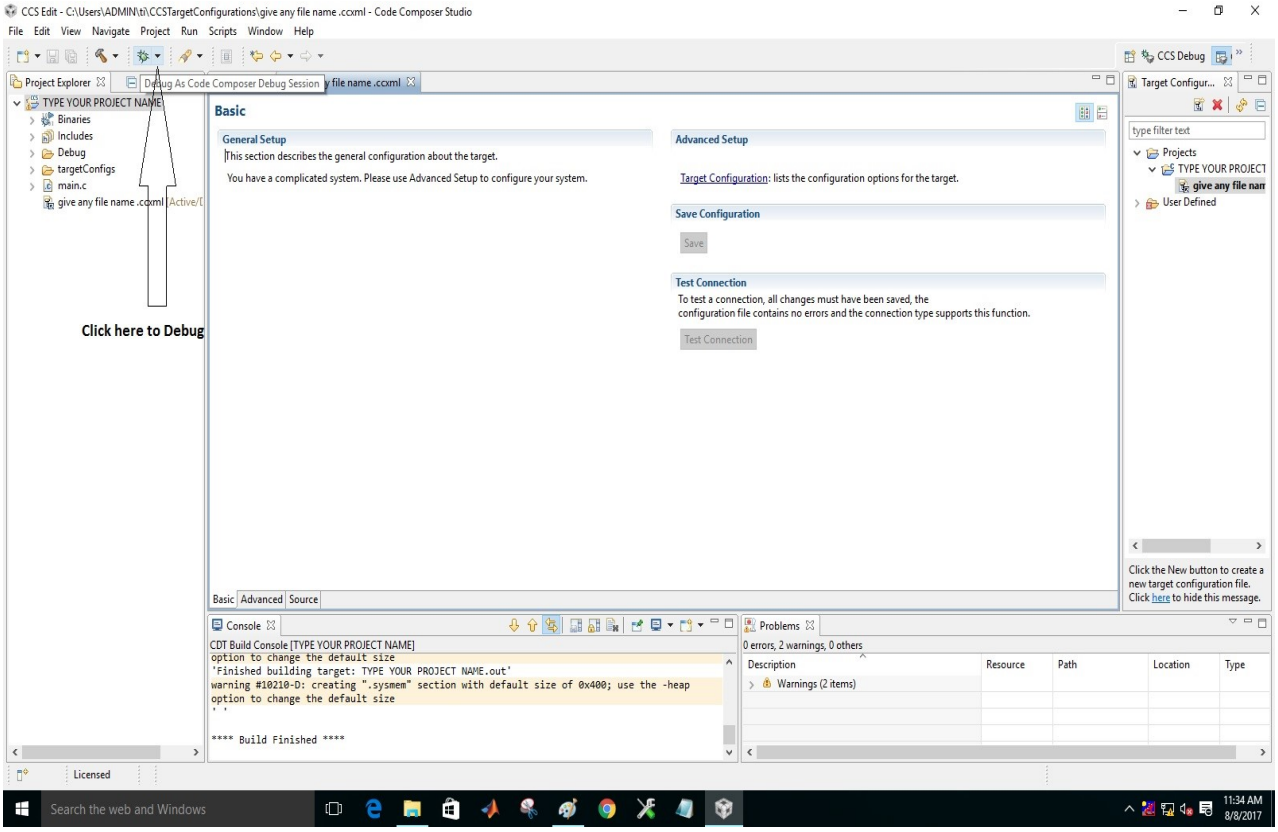


Step-11: Right click on the Project name and select Build project or click on the symbol as indicated by the arrow mark.

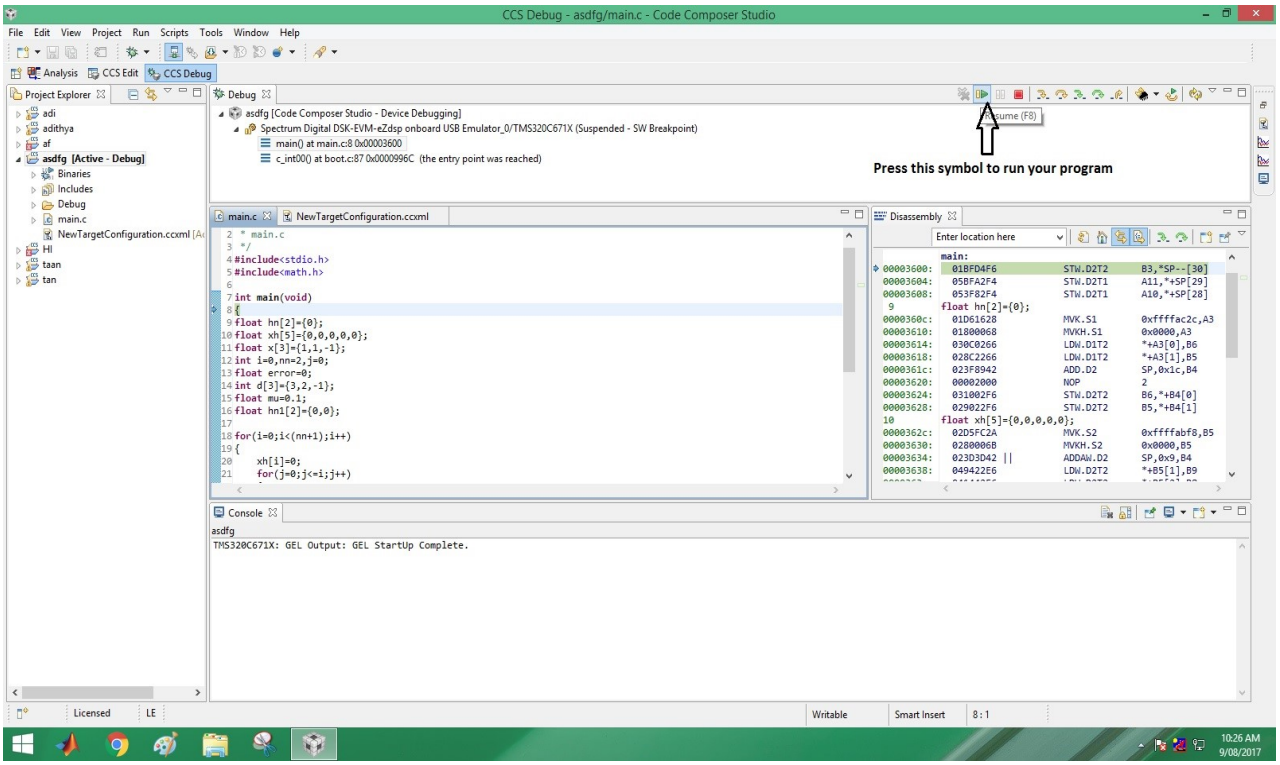




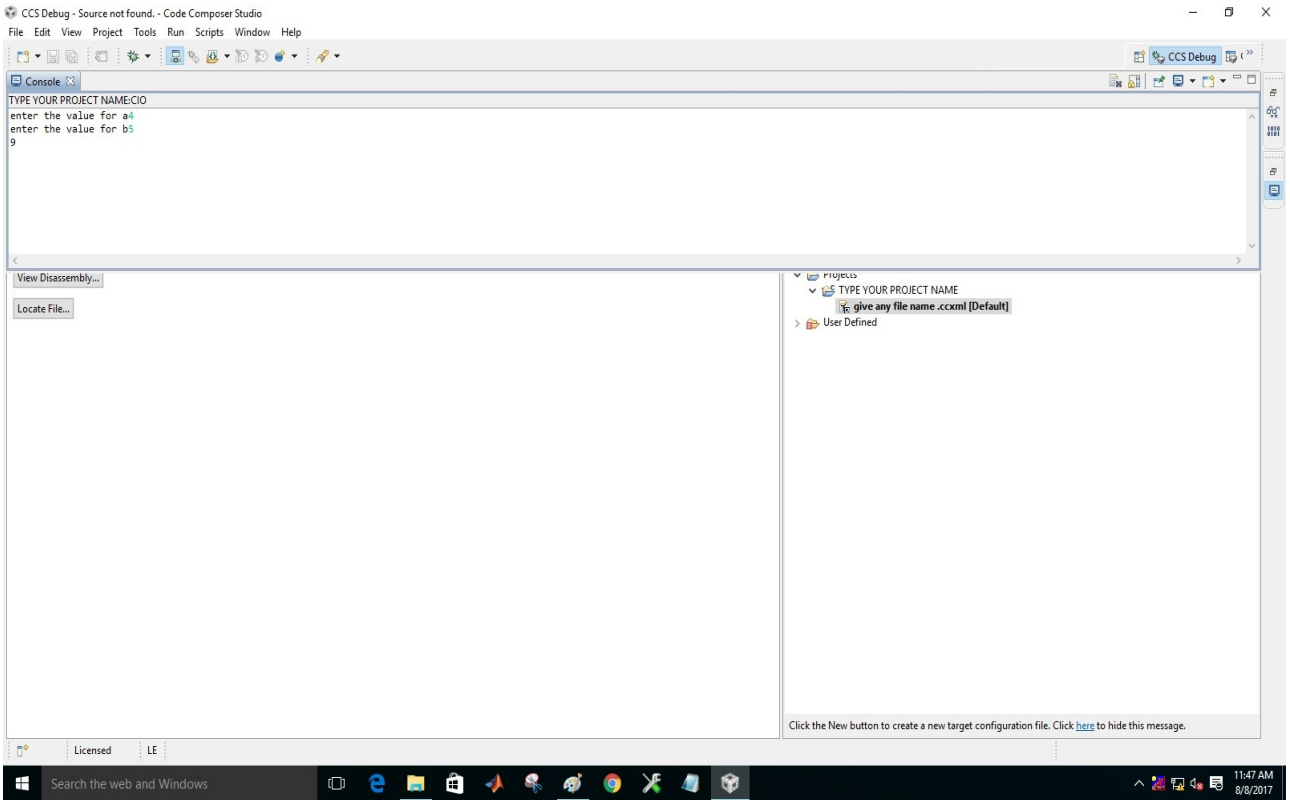
Step-12: Click on the symbol shown to debug your code.

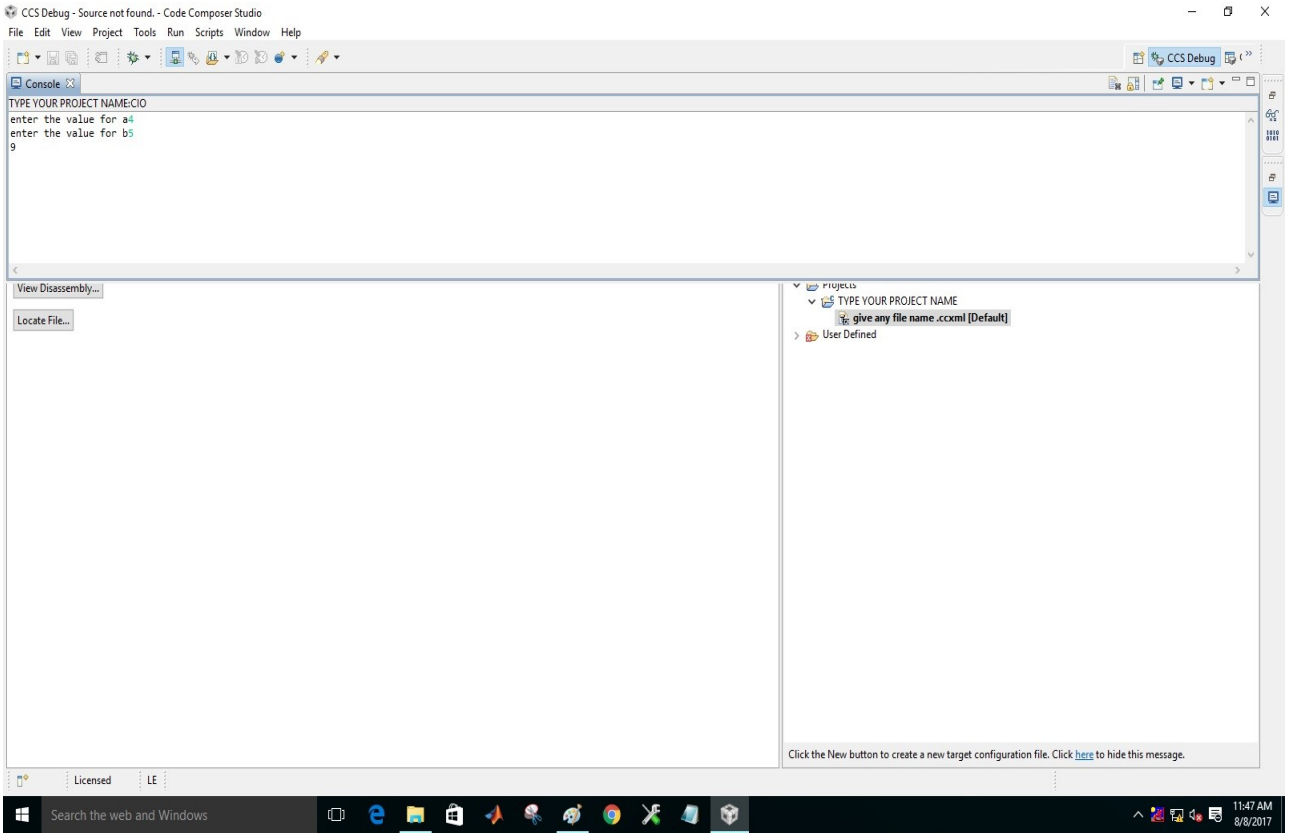


Step-13: Click on the Run symbol as depicted below to execute your code.



Step-13: After successful execution, output can be viewed in the console window as shown.





Outcome: The student will be able to implement the fundamental concepts of DSP using TMS320C6713 DSK.

For more information on 6713 DSK click here:
http://my.fit.edu/~vkepuska/ece3551/DSP%20Applications%20with%20the%20C6713%20and%20C6416%20DSK/ChassaingBook

8. a) Impulse Response: The response of a system to unit impulse input is called Impulse response, which can completely characterize a system.

To be demonstrated:

A discrete system LTI system is described by the following difference equation. Determine the first six samples of the system's impulse response.

$$y(n) + y(n-1) = (\frac{1}{3})x(n) + (\frac{1}{3})x(n-3)$$

Output: h (n) = [1/3, 1/3, 1/3, 2/3, 2/3, 2/3]

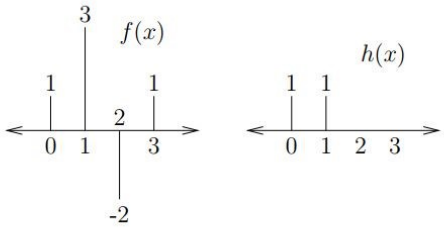
b) Convolution: It is the process through which we can obtain the output of an LTI system.

Linear Convolution

One dimensional linear discrete convolution is defined as:

$$g(x) = \sum_{s=-\infty}^{\infty} f(s)\,h(x-s) = f(x) * h(x)$$

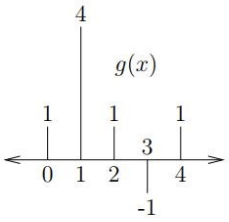
For example, consider the convolution of the following two functions:



This convolution can be performed graphically by reflecting and shifting $h(x)$, as shown in Figure 1. The samples of $f(s)$ and $h(s-x)$ that line up vertically are multiplied and summed:

$$g(0) = f(-1)h(1) + f(0)h(0) = 0 + 1 = 1$$
$$g(1) = f(0)h(1) + f(1)h(0) = 1 + 3 = 4$$
$$g(2) = f(1)h(1) + f(2)h(0) = 3 + -2 = 1$$
$$g(3) = f(2)h(1) + f(3)h(0) = -2 + 1 = -1$$
$$g(4) = f(3)h(1) + f(4)h(0) = 1 + 0 = 1$$

The result of the convolution is as shown below:



Notice that when $f(x)$ is of length 4, and $h(x)$ is of length 2, the linear convolution is of length $4 + 2 - 1 = 5$.

Circular Convolution

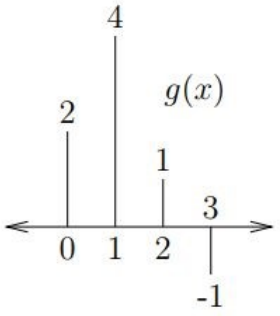
One dimensional circular discrete convolution is defined as:

$$g(x) = \sum_{s=0}^{M-1} f(s)\, h((x - s) \bmod M) = f(x) \circledast h(x)$$

For $M = 4$, the convolution can be performed using circular reflection and shifts of $h(x)$, as shown in Figure 2. The samples of $f(s)$ and $h((s - x) \bmod M)$ that line up vertically are multiplied and summed:

$$\begin{aligned}
g(0) &= f(3)h(1) + f(0)h(0) = 1 + 1 = 2 \\
g(1) &= f(0)h(1) + f(1)h(0) = 1 + 3 = 4 \\
g(2) &= f(1)h(1) + f(2)h(0) = 3 + -2 = 1 \\
g(3) &= f(2)h(1) + f(3)h(0) = -2 + 1 = -1
\end{aligned}$$

The result of the convolution is as shown below:



Notice that $f(x)$ and $h(x)$ are both treated as if they are of length 4, and the circular convolution is also of length 4.

9. a) Cross-correlation of two discrete-time sequences, x and y measures the similarity between x and shifted (lagged) copies of y as a function of the lag.

Auto-correlation is the linear dependence of a variable with itself at two points in time.

The cross correlation between $x(n)$ and $y(n)$ is defined as

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n-l) \qquad l = 0,\pm1,\pm2,.....$$

$$r_{xy}(l) = \sum_{n=-\infty}^{\infty} x(n+l)y(n) \qquad l = 0,\pm1,\pm2,.....$$

---- (1)

The cross correlation between $y(n)$ and $x(n)$ is defined as

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} x(n-l)y(n) \qquad l = 0,\pm1,\pm2,.....$$

$$r_{yx}(l) = \sum_{n=-\infty}^{\infty} x(n)y(n+l) \qquad l = 0,\pm1,\pm2,.....$$

---- (2)

From (1) and (2)

$$r_{xy}(l) = r_{yx}(-l)$$

To be demonstrated:

Input signal: Obtain the cross correlation of the two sequences x(n)= [2 1 2 4] and y(n)= [2 1 2 4] and plot it.

Output: $r_{xy}(l) = [8, 8, 12, 25, 12, 8, 8]$

Correlation Coefficient=1

The autocorrelation of $x(n)$ is defined as

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n+l)x(n) \qquad l = 0,\pm1,\pm2,.....$$

$$r_{xx}(l) = \sum_{n=-\infty}^{\infty} x(n-l)x(n) \qquad l = 0,\pm1,\pm2,.....$$

Input signal: Obtain the auto correlation of the sequence x(n)= [2 -1 1 3 5] and plot it.

Output: $r_{xx}(l) = [10, 1, 4, 15, 40, 15, 4, 1, 10]$

(b) As defined in Experiment 4 (a)