SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyalurmetta, Nandyal – 518503. Website: www.svrec.ac.in

Department of Electronics and Communication Engineering

DIGITAL SIGNAL PROCESSING LABORATORY (R 15)

III B.Tech (ECE) II Semester

A Y - 2020-21

STUDENT NAME	
ROLL NUMBER	
SECTION	

SVR ENGINEERING COLLEGE

Approved by AICTE & Permanently Affiliated to JNTUA

Ayyalurmetta, Nandyal – 518503. Website: www.svrec.ac.in

Department of Electronics and Communication Engineering

DEPARTMENT OF

ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

ACADEMIC YEAR: 2020-21

This is to certify that the bonafide record work done by Mr./Ms.____ bearing H.T.No._________________ of II B.Tech II Semester in the **Digital signal processing Laboratory**

Faculty In-Charge **Head of the Department**

INDEX

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

B. Tech III-II Sem. (ECE) L T P C 0 0 4 2

(15A04608) DIGITAL SIGNAL PROCESSING LABORATORY

Course Outcomes:

• Able to design real time DSP systems and real world applications.

• Able to implement DSP algorithms using both fixed and floating point processors.

List of Experiments: (Minimum of 5 experiments are to be conducted from each part)

Software Experiments (PART – A)

1. Generation of random signal and plot the same as a waveform showing all the specifications.
2. Finding Power and (or) Energy of a given signal.
3. Convolution and Correlation (auto and cross correlation) of discrete sequences without using built in functions for convolution and correlation operations.
4. DTFT of a given signal
5. N – point FFT algorithm
6. Design of FIR filter using windowing technique and verify the frequency response of the filter.
7. Design of IIR filter using any of the available methods and verify the frequency response of the filter.
8. Design of analog filters.

Using DSP Processor kits (Floating point) and Code Composure Studio (CCS) (PART – B)

1. Generation of random signal and plot the same as a waveform showing all the specifications.
2. Finding Power and (or) Energy of a given signal.
3. Convolution and Correlation (auto and cross correlation) of discrete sequences without using built in functions for convolution and correlation operations.
4. DTFT of a given signal
5. N – point FFT algorithm
6. Design of FIR filter using windowing technique and verify the frequency response of the filter.
7. Design of IIR filter using any of the available methods and verify the frequency response of the filter.
8. Design of analog filters

Equipment/Software Required:

1. Licensed MATLAB software with required tool boxes for 30 users.

2. DSP floating Processor Kits with Code Composure Studio (8 nos.),Function generators,CROs,Regulated Power Supplies

ECE DEPT VISION & MISSION PEOs and PSOs

Vision

To produce highly skilled, creative and competitive Electronics and Communication Engineers to meet the emerging needs of the society.

Mission

- Impart core knowledge and necessary skills in Electronics and Communication Engineering through innovative teaching and learning.
- Inculcate critical thinking, ethics, lifelong learning and creativity needed for industry and society
- Cultivate the students with all-round competencies, for career, higher education and self-employability

I. PROGRAMME EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Graduates apply their knowledge of mathematics and science to identify, analyze and solve problems in the field of Electronics and develop sophisticated communication systems.

PEO2: Graduates embody a commitment to professional ethics, diversity and social awareness in their professional career.

PEO3: Graduates exhibit a desire for life-long learning through technical training and professional activities.

II. PROGRAM SPECIFIC OUTCOMES (PSOS)

PSO1: Apply the fundamental concepts of electronics and communication engineering to design a variety of components and systems for applications including signal processing, image processing, communication, networking, embedded systems, VLSI and control system

PSO2: Select and apply cutting-edge engineering hardware and software tools to solve complex Electronics and Communication Engineering problems.

III. PROGRAMME OUTCOMES (PO'S)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

IV. COURSE OBJECTIVES

✧ To provide background and fundamental material for the analysis and processing of

digital signals.

✧ To familiarize the relationships between continuous-time and discrete time signals

and systems.

✧ To study fundamentals of time, frequency and Z-plane analysis and to discuss the

inter-relationships of these analytic method.

✧ To study the designs and structures of digital (IIR and FIR) filters from analysis to

synthesis for a given specifications.

✧ To introduce a few real-world signal processing applications.

✧ To acquaint with DSP processor

V. COURSE OUTCOMES

After the completion of the course students will be able to

Course Outcomes	Course Outcome statements	BTL
CO1	To Review of discrete-time signals and systems and DFT	L1
CO2	Computation of DFT & FFT	L2
CO3	To study the realization of digital filters	L3
CO4	To study the IIR &FIR filters	L4
CO5	To know and study DSP Processors	L5

VI.COURSE MAPPING WITH PO'S AND PEO'S

Course Title	P0 1	P 02	P0 3	P0 4	P 0 5	P 0 6	P 0 7	P 0 8	P 0 9	P0 10	P 0 11	P 0 12	P E 0 1	P E 0 2
DSP Lab	**2.2**	**2.6**	**2.2**	**2.4**	**2.6**	**2.2**	**1.4**	**2**	**2.0**	**2.2**	**2.2**	**2.6**	**2.6**	**2.4**

V MAPPING OF COURSE OUTCOMES WITH PEO'S AND PO'S

Course Title	P 0 1	P 0 2	P 0 3	P 0 4	P 0 5	P 0 6	P 0 7	P 0 8	P 0 9	P0 10	P 0 11	P 0 12	P E 0 1	P E 0 2
CO1	3	3			2		1	3	3	3	3	1	3	2
CO2	3	3		3	3	2	3		1	2	1	1	3	2
CO3	3	3	2	3	3	2	2	1	2	2		2	3	2
CO4	3	1	2	3	3	2	3	1	2	2	1	2	3	2
CO5	3	3		2			2	3	3		3	1	3	2

LABORATORY INSTRUCTIONS

1. While entering the Laboratory, the students should follow the dress code. (Wear shoes and White apron, Female Students should tie their hair back).

2. The students should bring their observation book, record, calculator, necessary stationery items and graph sheets if any for the lab classes without which the students will not be allowed for doing the experiment.

3. All the Equipment and components should be handled with utmost care. Any breakage or damage will be charged.

4. If any damage or breakage is noticed, it should be reported to the concerned in charge immediately.

5. The theoretical calculations and the updated register values should be noted down in the observation book and should be corrected by the lab in-charge on the same day of the laboratory session.

6. Each experiment should be written in the record note book only after getting signature from the lab in-charge in the observation notebook.

7. Record book must be submitted in the successive lab session after completion of experiment.

8. 100% attendance should be maintained for the laboratory classes.

Precautions.

1. Check the connections before giving the supply
2. Observations should be done carefully

Day to Day Observations

S.NO	Name of the experiment	Page No.	Performed Date	Date of submission	Marks	Faculty Sgnature
1	Generation of random signal and plot the same as a waveform showing all the specifications	16-17				
2	Finding Power and (or) Energy of a given signal.	18-19				
3	Convolution and Correlation (auto and cross correlation) of discrete sequences without using built in functions for convolution and correlation operations.	20-28				
4	DTFT of a given signal	29-44				
5	N – point FFT algorithm	45-46				
6	Design of FIR filter using windowing technique and verify the frequency response of the filter	47-53				
7	Design of IIR filter using any of the available methods and verify the frequency response of the filter.	54-56				
8	Design of analog filters.	57-58				
9	Generation of random signal and plot the same as a waveform showing all the specifications.	71-72				
10	Finding Power and (or) Energy of a given signal. Design of FIR filter using windowing technique and verify the frequency response of the filter.	73-74				
11	Convolution and Correlation (auto and cross correlation) of discrete sequences without using built in functions for convolution	75-78				

	and correlation operations.					
12	DTFT of a given signal	79				
13	N – point FFT algorithm	80-82				
14	Design of FIR filter using windowing technique and verify the frequency response of the filter	83-84				
15	Design of IIR filter using any of the available methods and verify the frequency response of the filter.	85-86				
16	Design of analog filters.	87-88				

WORKING PROCEDURE WITH MATLAB:

1) Double click on Matlab icon. -> Then Matlab will be opened

2) To write the Matlab Program

Goto file menu-> New -> Script(Mfile) -> In the opened Script file write the Matlab code and save the file with an extension of .m

Ex: "linear.m"

3)To execute Matlab Program Select the all lines in matlab program(ctrl+A) of mfile and press "F9"

to execute the matlab code

4)Entering the inputs in command window

 If the command window is displaying the message like "enter the input sequence" then

enter the sequence with square brackets and each sample values is spaced with single space

Ex: Enter input sequence [1 2 3 4] If it is asking a value input write the value without brackets

Ex: "enter length of sequence 4" After entering inputs It displays the Output Graphs.

PROCEDURE TO WORK ON CODE COMPOSER STUDIO PROCEDURE FOR EXECUTING NON REAL TIME PROGRAMS (EX: LINEAR & CIRCULAR CONVOLUTION, FFT, PSD)

Test the USB port by running DSK Port test from the start menu

Use StartProgramsTexas InstrumentsCode Composer StudioCode Composer Studio

CDSK6713 Tools DSK6713 Diagnostic Utilities

 *Select Start Select DSK*6713 *Diagnostic Utility Icon from Desktop*

 *Select **Start** Option*

 Utility Program will test the board

 *After testing Diagnostic Status you will get **PASS***

To create the New Project

Project New (File Name. pjt , Eg: **Vectors.pjt**)

To Create a Source file -- File New Type the code (Save & give file name, Eg: **sum.c**).

To Add Source files to Project

Project Add files to Project c/ccs studio3.1/my projects/your project name/

sum.c(select the file type as c/c++ source files)

To Add rts.lib file & hello.cmd:

Project Add files to Project rts6700.lib

(Path:c/ccs studio3.1/cg tools/c6000/lib/ rts6700.lib**)**

Note: Select Object & Library in(*.o,*.l) in Type of files

Project Add files to Project hello.cmd

CMD file – Which is common for all non real time programs.

(Path: c/ccs studio3.1\tutorial\dsk 6713 \hello1\hello.cmd)

Note: Select Linker Command file(*.cmd) in Type of files

Compile:

To Compile: Project Compile project

To Build: Project build project,

To Rebuild: Project rebuild,

Which will create the final .out executable file.(Eg. Vectors.out).

Procedure to Load and Run program:

Load the program to DSK: File Load program Vectors. out

To Execute project: Debug Run.

1. Execution should halt at break point.

2. Now press F10. See the changes happening in the watch window.

3. Similarly go to view & select CPU registers to view the changes happening in CPU registers.

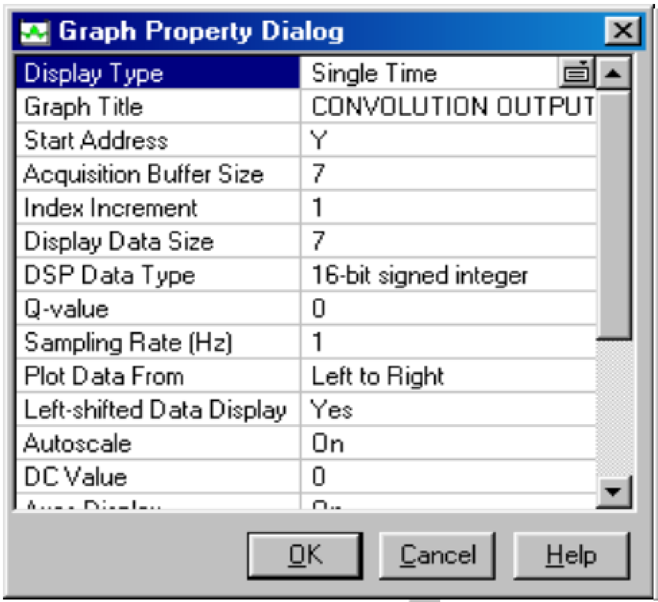
Configure the graphical window as shown below

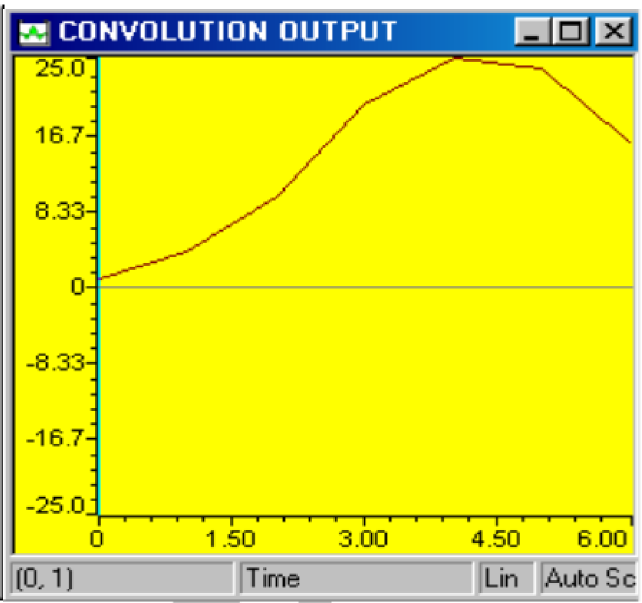
INPUT :

x[n] = {1, 2, 3, 4,0,0,0,0}

h[k] = {1, 2, 3, 4,0,0,0,0}

OUTPUT:





b)PROCEDURE FOR EXECUTING REAL TIME PROGRAMS (EX:IIR FILTERS,FIR FILTERS DESIGNING)

CONNECTING DSP PROCESSOR TO PC

 Connect the dsp processor to the pc using usb cable connector.

 Check the DSK6713 diagnostics (IF you get the “pass”then click on ok).

 Click on ccs studio3.1 desk top icon. Then the window will be opened.

 Go to debug click on connect (then target device will be connected to pc)

TO CREATE PROJECT

 Project new given project name and select the family’TMS320C67XX’Then click ok

 File new source file write down the ‘c’program and save it with.’c’ extension in current project file

 File new dsp/bios.config file select dsk67xx click on dsk6713 and save it in current project.

 Project add files to project add source file

 Project add files to project add library file by following the given path *c/ccs studio3.1/cgtools/c6000/dsk6713/DSK6713.bs/file*.

 Project add files to the project .Add the configuration file.

 Now files are generated and included in generated files . in that open the 3rd file, and copy the header file and paste it in source file. Copy the include files named as ”**dsk6713.h” and “dsk6713_aic23.h”** paste it in current project folder.

 Now compile project.(project compile)

 Project build.

 Project rebuild all.

 File load program project name.pjt debug “project name .out” file click on open debug click on run

 Now apply the input sine wave to line in of dsk6713 kit.

 Observe the output at line out of dsk6713 by using CRO.

INTRODUCTIONTO MATLAB

MATLAB: MATLAB is a software package for high performance numerical computation and visualization provides an interactive environment with hundreds of built in functions for technical computation, graphics and animation .

The MATLAB name stands for Matrix Laboratory.

It also allows you to put a list of your processing requests together in a file and save that combined list with a name so that you can run all of those commands in the same order at some later time. Furthermore, it allows you to run such lists of commands such that you pass in data and/or get data back out (i.e. the list of commands is like a function in most programming languages). Once you save a function, it becomes part of your toolbox (i.e. it now looks to you as if it were part of the basic toolbox that you started with).

For those with computer programming backgrounds: Note that MATLAB runs as an interpretive language (like the old BASIC). That is, it does not need to be compiled. It simply reads through each line of the function, executes it, and then goes on to the next line. (In practice, a form of compilation occurs when you first run a function, so that it ca n run faster the next time you run it.)

MATLAB Windows :

MATLAB works with through three basic windows

Command Window : This is the main window .it is characterized by MATLAB command prompt >> when you launch the application program MATLAB puts you in this window all commands including those for user-written programs ,are typed in this window at the MATLAB prompt

Graphics window: the output of all graphics commands typed in the command window are flushed to the graphics or figure window, a separate gray window with white background color the user can create as many windows as the system memory will allow

Edit window: This is where you write edit, create and save your own programs in files called M files.

Input-output:

MATLAB supports interactive computation taking the input from the screen and flushing, the output to the screen. In addition it can read input files and write output files

Data Type: the fundamental data –type in MATLAB is the array. It encompasses several distinct data objects- integers, real n umbers, matrices, charcter strings, structures and cells.There is no need to declare variables as real or complex,

MATLAB automatically sets the variable to be real.

Dimensioning: Dimensioning is automatic in MATLAB. No dimension statements are required for vectors or arrays .we can find the dimensions of an existing matrix or a vector with the size and length commands.

1. Generation of random signal and plot the same as a waveform showing all the specifications.

Aim: To generate random signal and plot the same as a waveform showing all the specifications.

APPARATUS: PC with MATLAB Software.

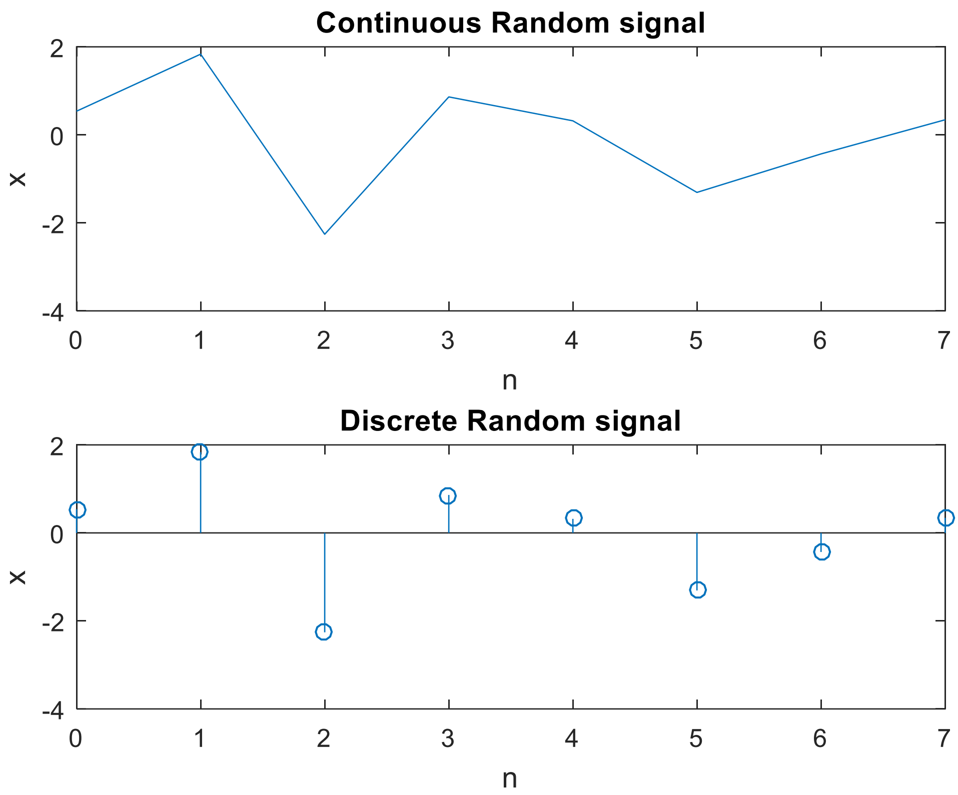
PROCEDURE:

1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

Program:

```
N=input('Enter the value of N:');
n=0:N-1;
x=randn(1,N);
figure;
subplot(2,1,1);
plot(n,x);
xlabel('n');
ylabel('x');
title('Continuous Random signal');
subplot(2,1,2);
stem(n,x);
xlabel('n');
ylabel('x');
title('Discrete Random signal');
```

Output :--



Result : Random signal and plot the same as a waveform showing all the specifications.

VIVA VOCE

1. Random variables give relationship between _____Random event and a real number

2. Which gives the measure of randomness of the random variable?--Variance gives the randomness of the random variable. It is the difference between the mean square value and square of the mean.

3. Random process is a function of ______Random event and time

4. A random process is called as stationary in strict sense if-- Its statistics vary with shift in time origin

5. For a stationary process, auto correlation function depends on--Time difference

2) Finding Power and (or) Energy of a given signal

Aim : To find the Power and (or) Energy of a given signal

APPARATUS: PC with MATLAB Software.

PROCEDURE:

1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

Program:

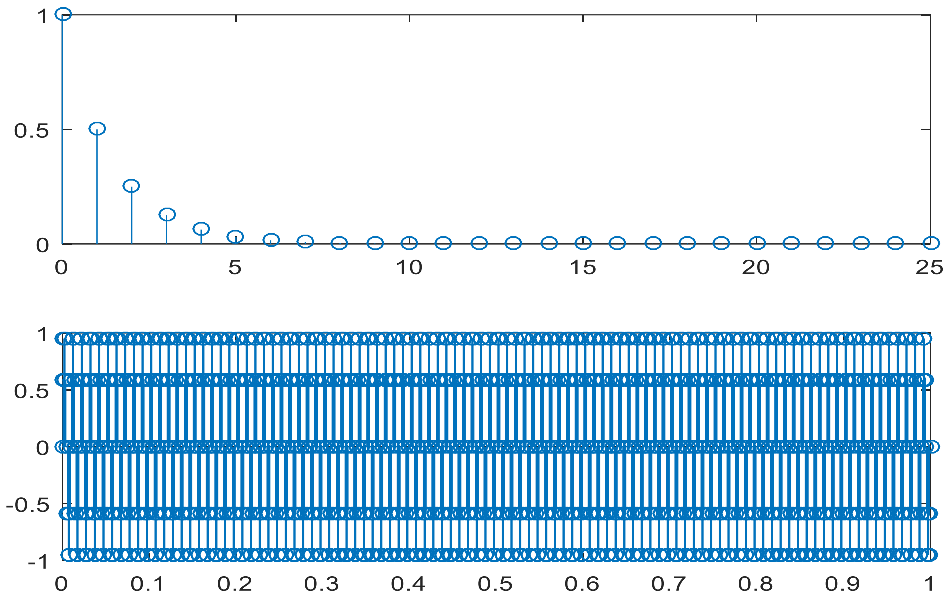
```
%Energy of the Discrete Time Signal

n=0:1:50;

x=(1/2).^n;

figure;

subplot(2,1,1);

stem(n,x);

axis([0 25 0 1]);

disp('The Calculated Energy E of the Signal is ');

E=sum(abs(x).^2);

disp(E);

%Power of the Discrete Time Signal

f=input('enter the frequency = ');

fs=10*f;

n1=0:1/fs:1;

ss=sin(2*pi*f*n1);

disp('The Calculated Power p of the Signal is ');

p=sum(abs(ss).^2)/length(ss);

subplot(2,1,2);
```

```
stem(n1,ss)

disp(p);
```

Output : The Calculated Energy E of the Signal is 1.3333

enter the frequency = 100

The Calculated Power p of the Signal is 0.4995



Result: Power and (or) Energy of a given signal calculated.

VIVA VOCE

1) The signal power of the periodic rectangular pulses of height 1 and width 1, is _

The signal power in the given signal using Parsevals's relation is
P=1t∫T0x2(t)dt =12∫101.dt = 0.5 W.

2) The signal power of the signal x (t) = 2sin 2t + 4sin 4t + 6cos 4t + 2cos 2t with period 0.5 is __Signal power = 0.5(22 + 42 + 62 + 22)
= 0.5(4 + 16 + 36 + 4) = 0.5(20 + 40) = 30 W.

3) A signal is a power signal if the signal has average power equal to _Finite

A signal is said to be a power signal if and only if the average power of the signal is finite. In other words, we can say that a signal is a power signal if the energy of the signal is infinite, i.e., E = ∞.

4) The energy in the time-domain representation of a signal is the same as in the frequency domain representation normalized by _2π

5) A periodic signal has power P/4 equal to average energy per period then rms value of signal is √P/2

3) Convolution and correlation(auto and cross correlation)of discrete sequence without using built in functions for convolutions and correlation operations.

Aim : To write a matlab program for Convolution and correlation(auto and cross correlation)of discrete sequence without using built in functions for convolutions and correlation operations.

APPARATUS: PC with MATLAB Software.

PROCEDURE:

1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

Program:

```
%Linear Convolution

clc;

clear all;

close all;

disp('linear convolution program');

x=input('enter i/p x(n):');

m=length(x);

h=input('enter i/p h(n):');

n=length(h);

x=[x,zeros(1,n)];

l1=0:length(x)-1;

subplot(2,2,1);

stem(l1,x);

title('i/p sequence x(n)is:');

xlabel('---->n');

ylabel('---->x(n)');grid;

h=[h,zeros(1,m)];
```

```
l2=0:length(h)-1;

subplot(2,2,2);

stem(l2,h);

title('i/p sequence h(n)is:');

xlabel('---->n');

ylabel('---->h(n)');grid;

disp('convolution of x(n) & h(n) is y(n):');

y=zeros(1,m+n-1);

for i=1:m+n-1

y(i)=0;

for j=1:m+n-1

if(j<i+1)

y(i)=y(i)+x(j)*h(i-j+1);

end

end

end

disp(y);

l3=m+n-1;

n1=0:l3-1;

subplot(2,2,[3,4]);

stem(n1,y);

title('Linear Convolution of x(n) & h(n) is :');

xlabel('---->n');

ylabel('---->y(n)');

grid;

%Circular Convolution

clc;

clear all;

close all;
```

```
disp('circular convolution program');

x=input('enter i/p x(n):');

m=length(x);

h=input('enter i/p sequence h(n)');

n=length(h);

subplot(2,2,1), stem(x);

title('i/p sequencce x(n)is:');

xlabel('---->n');

ylabel('---->x(n)');grid;

subplot(2,2,2), stem(h);

title('i/p sequencce h(n)is:');

xlabel('---->n');

ylabel('---->h(n)');grid;

disp('circular convolution of x(n) & h(n) is y(n):');

if(m-n~=0)

if(m>n)

h=[h,zeros(1,m-n)];

n=m;

end

x=[x,zeros(1,n-m)];

m=n;

end

y=zeros(1,n);

y(1)=0;

a(1)=h(1);

for j=2:n

a(j)=h(n-j+2);

end

%ciruclar conv
```

```
for i=1:n

y(1)=y(1)+x(i)*a(i);

end

for k=2:n

y(k)=0;

% circular shift

for j=2:n

x2(j)=a(j-1);

end

x2(1)=a(n);

for i=1:n

if(i<n+1)

a(i)=x2(i);

y(k)=y(k)+x(i)*a(i);

end

end

end

disp(y);

subplot(2,2,[3,4]),stem(y);

title('Circular convolution of x(n) & h(n) is:');

xlabel('---->n');

ylabel('---->y(n)');

grid;

%Cross Correlation

clc;

clear all;

close all;

z=input('Enter first sequence x(n):');

n1=length(z);
```

```
x=fliplr(z);

m=length(x);

h=input('Enter Second sequence h(n):');

n=length(h);

l1=0:n1-1;

figure;

subplot(2,2,1);

stem(l1,z);

title('First sequence x(n)is:');

xlabel('---->n');

ylabel('---->x(n)');

grid;

l2=0:n-1;

subplot(2,2,2);

stem(l2,h);

title('Second sequence h(n)is:');

xlabel('---->n');

ylabel('---->h(n)');

grid;

x=[x,zeros(1,m)];

h=[h,zeros(1,m)];

disp('Cross Correlation of x(n) & h(n) is y(n):');

y=zeros(1,m+n-1);

for i=1:m+n-1

y(i)=0;

for j=1:m+n-1

if(j<i+1)

y(i)=y(i)+x(j)*h(i-j+1);

end
```

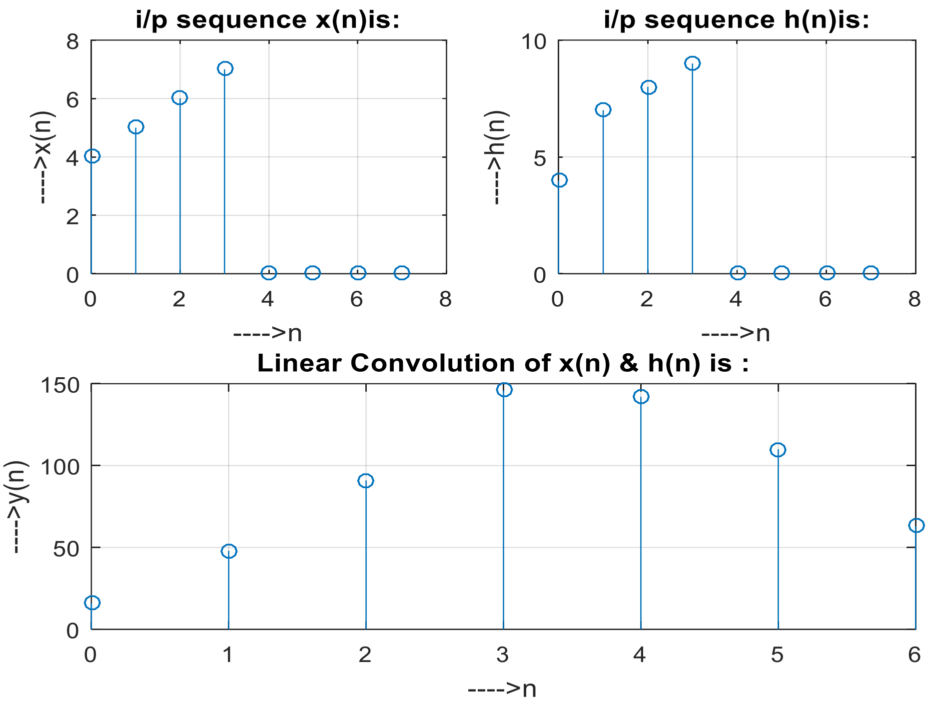
```
end

end

disp(y);

l3=-(m-1):(n-1);

subplot(2,2,[3,4]);

stem(l3,y);

title('Cross Correlation of x(n) & h(n) is :');

xlabel('---->n');

ylabel('---->y(n)');

grid;

%Auto Correlation

clc;

clear all;

close all;

x=input('Enter the sequence x(n):');

n=length(x);

z=fliplr(x);

m=length(z);

l1=0:n-1;

figure;

subplot(2,1,1);

stem(l1,x);

title('Input Sequence x(n)is:');

xlabel('---->n');

ylabel('---->x(n)');

grid;

x=[x,zeros(1,m)];

z=[z,zeros(1,n)];

disp('Auto Correlation of x(n) & x(n) is y(n):');
```

```
y=zeros(1,m+n-1);

for i=1:m+n-1

y(i)=0;

for j=1:m+n-1

if(j<i+1)

y(i)=y(i)+z(j)*x(i-j+1);

end

end

end

disp(y);

l2=-(n-1):(n-1);

subplot(2,1,2);

stem(l2,y);

title('Auto Correlation of x(n) & x(n) is :');

xlabel('---->n');  ylabel('---->y(n)');

grid;
```

Output :

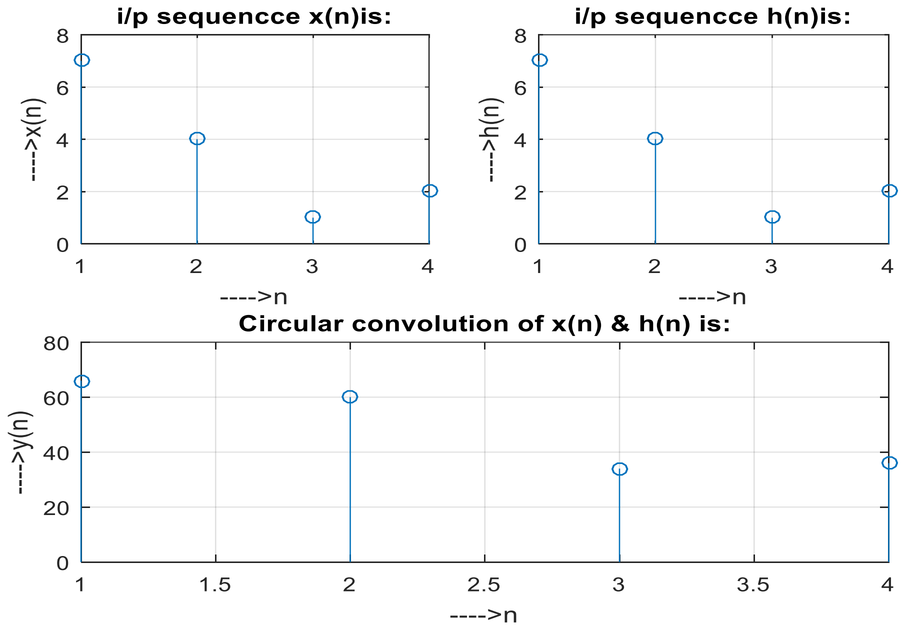
Linear convolution program

```
enter i/p x(n):[4 5 6 7]
enter i/p h(n):[4 7 8 9]
convolution of x(n) & h(n) is y(n):
  16    48    91   146   142   110    63
```

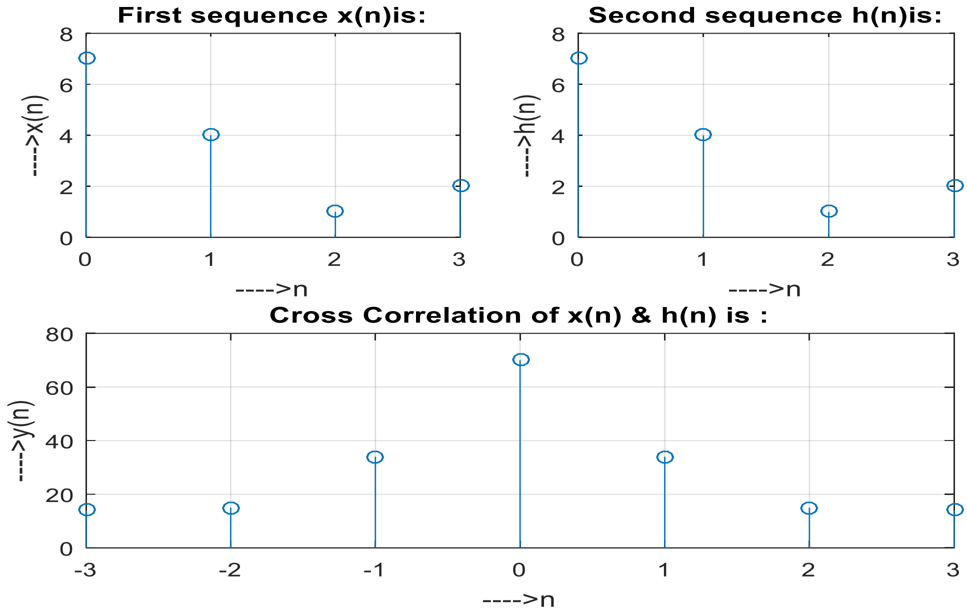


Circular convolution program

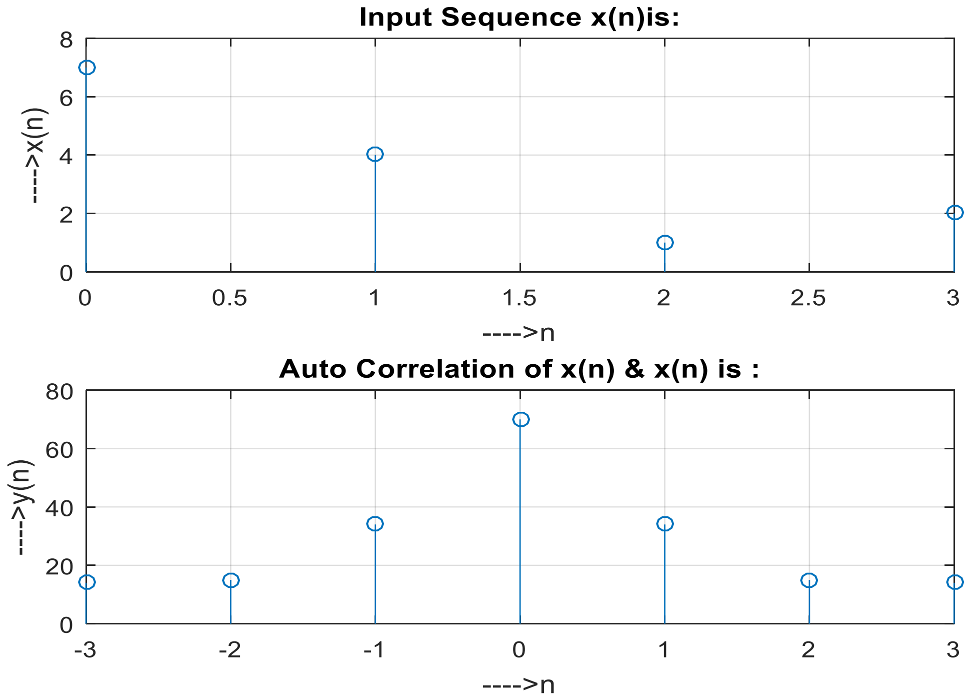
enter i/p x(n):[7 4 1 2]
enter i/p sequence h(n)[7 4 1 2]
circular convolution of x(n) & h(n) is y(n):
66 60 34 36



Enter first sequence x(n):[7 4 1 2]
Enter Second sequence h(n):[7 4 1 2]
Cross Correlation of x(n) & h(n) is y(n):
14 15 34 70 34 15 14



Enter the sequence x(n):[7 4 1 2]
Auto Correlation of x(n) & x(n) is y(n):
14 15 34 70 34 15 14



4. VERIFICATION OF DTFT PROPERTIES

AIM: To Verify DTDT Properties.

APPARATUS: PC with MATLAB Software.

PROCEDURE:

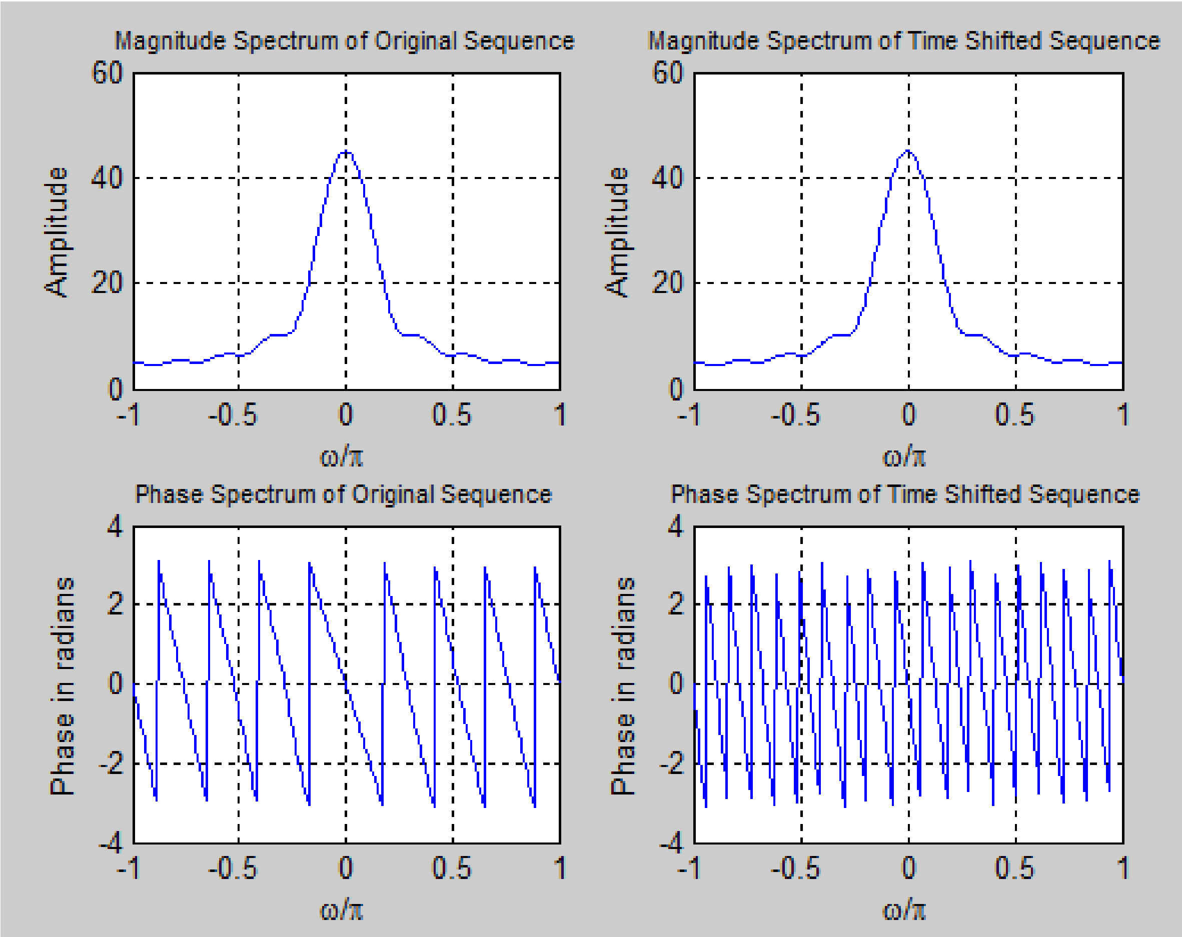
1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

PROGRAM CODE:

A) TIME SHIFTING PROPERTIES OF DTFT

```
clf;
w=-pi:2*pi/255:pi;
D=10;
num=[1 2 3 4 5 6 7 8 9];
h1=freqz(num,1,w);
h2=freqz([zeros(1,D) num],1,w);
subplot(2,2,1);
plot(w/pi,abs(h1));
grid;
title('Magnitude Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2);
plot(w/pi,abs(h2));
grid;
title('Magnitude Spectrum of Time Shifted Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,angle(h1));
grid;
title('Phase Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
subplot(2,2,4);
plot(w/pi,angle(h2));
grid;
title('Phase Spectrum of Time Shifted Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

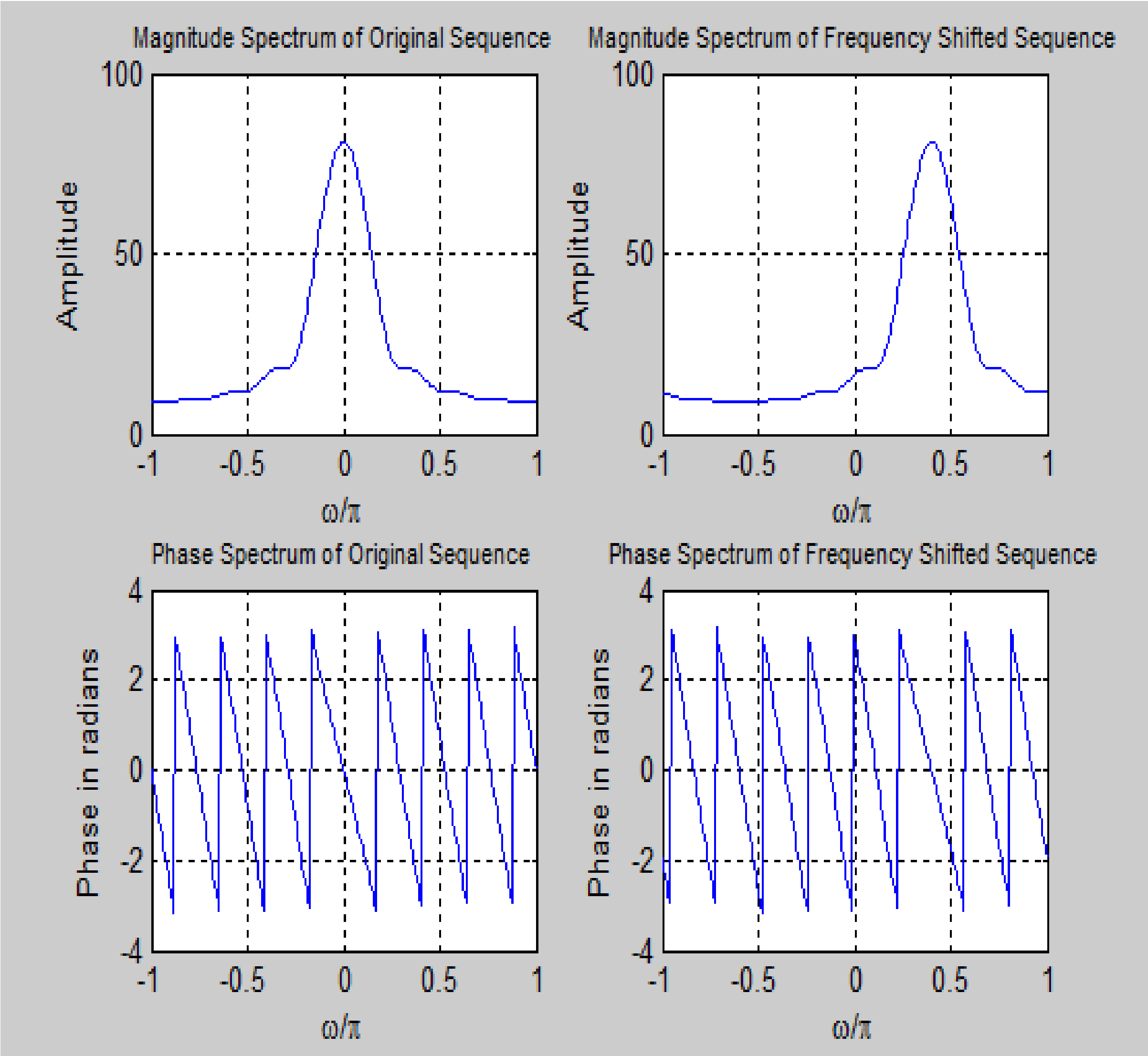
WAVEFORMS:


Magnitude Spectrum of Original Sequence
Magnitude Spectrum of Time Shifted Sequence
Amplitude
ω/π
Phase Spectrum of Original Sequence
Phase Spectrum of Time Shifted Sequence
Phase in radians
ω/π

B) FREQUENCY SHIFTING PROPERTIES OF DTFT

```
clf;
w=-pi:2*pi/255:pi;
wo=0.4*pi;
num1=[1 3 5 7 9 11 13 15 17];
L=length(num1);
h1=freqz(num1,1,w);
n=0:L-1;
num2=exp(wo*i*n).*num1;
h2=freqz(num2,1,w);
subplot(2,2,1);
plot(w/pi,abs(h1));
grid;
title('Magnitude Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2);
plot(w/pi,abs(h2));
grid;
title('Magnitude Spectrum of Frequency Shifted Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,angle(h1));
grid;
title('Phase Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
subplot(2,2,4);
plot(w/pi,angle(h2));
grid;
title('Phase Spectrum of Frequency Shifted Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

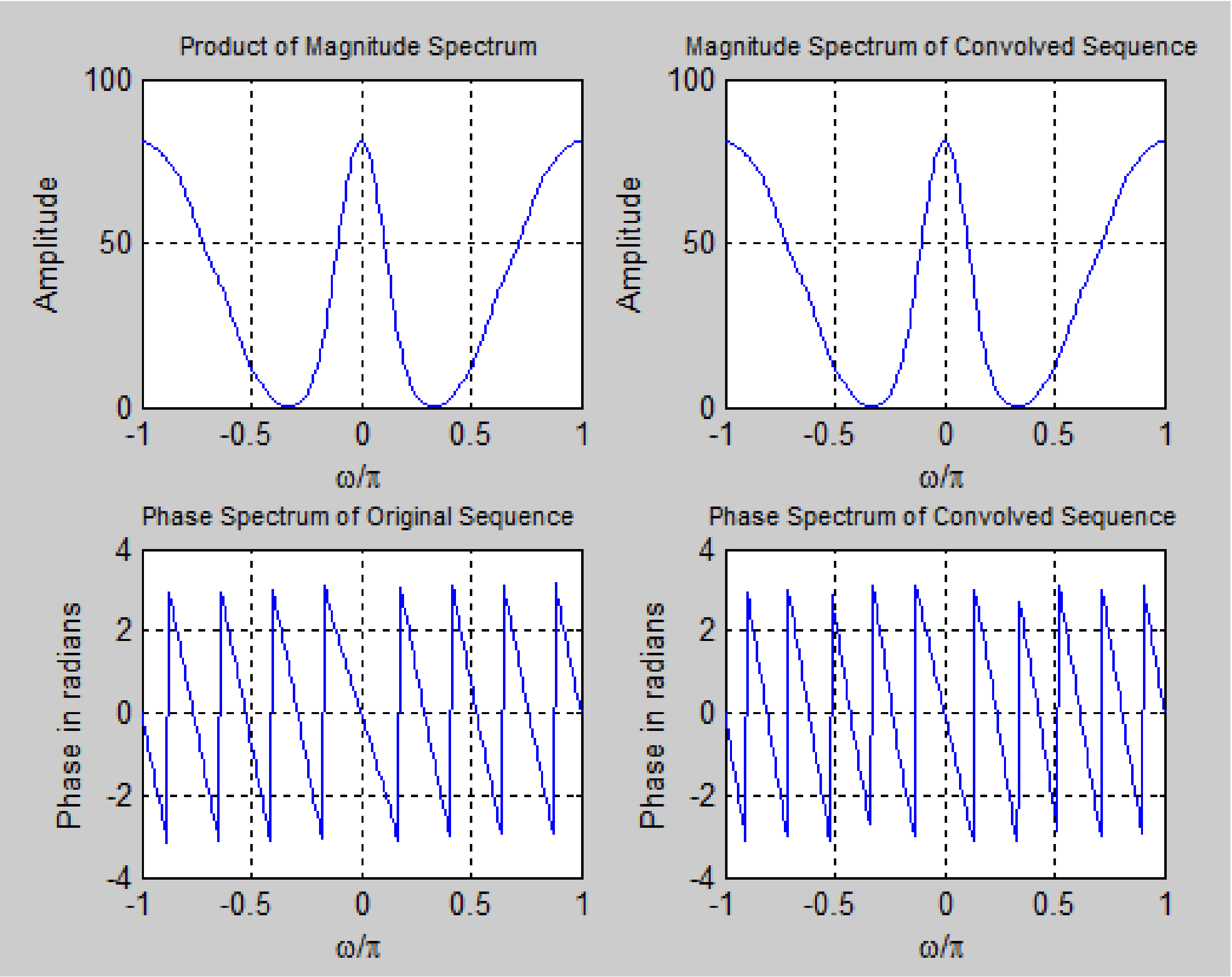
WAVEFORMS:


Magnitude Spectrum of Original Sequence
Magnitude Spectrum of Frequency Shifted Sequence
Amplitude
ω/π
Phase Spectrum of Original Sequence
Phase Spectrum of Frequency Shifted Sequence
Phase in radians

C) CONVOLUTION PROPERTY OF DTFT

```
clf;
w=-pi:2*pi/255:pi;
x1=[1 3 5 7 9 11 13 15 17];
x2=[1 -2 3 -2 1];
y=conv(x1,x2);
h1=freqz(x1,1,w);
h2=freqz(x2,1,w);
hp=h1.*h2;
h3=freqz(y,1,w);
subplot(2,2,1);
plot(w/pi,abs(hp));
grid;
title('Product of Magnitude Spectrum','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2);
plot(w/pi,abs(h3));
grid;
title('Magnitude Spectrum of Convolved Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,abs(hp));
grid;
title('Sum of Phase Spectrum','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,angle(h1));
grid;
title('Phase Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
subplot(2,2,4);
plot(w/pi,angle(h3));
grid;
title('Phase Spectrum of Convolved Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

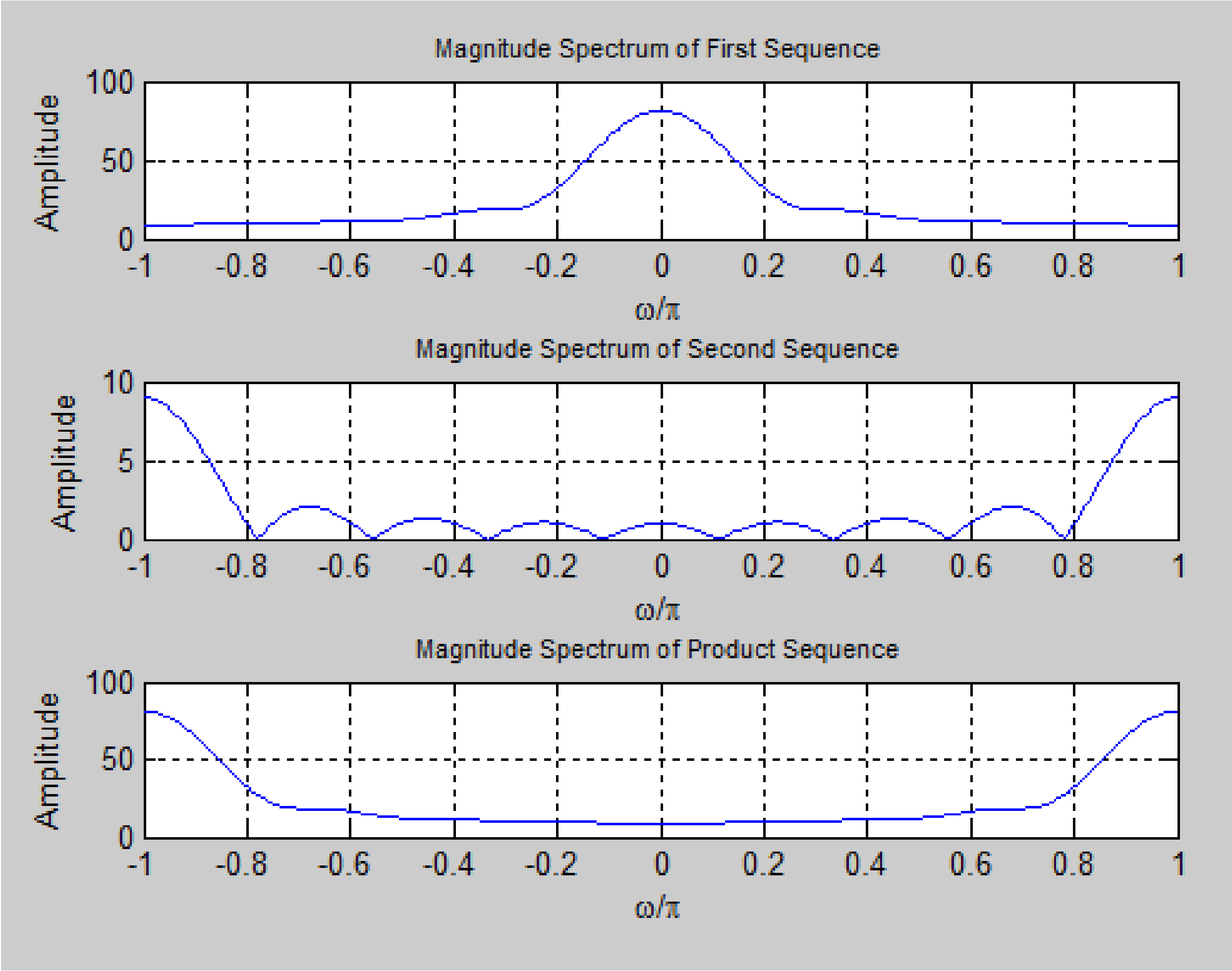
WAVEFORMS:


Product of Magnitude Spectrum
Amplitude
ω/π
Magnitude Spectrum of Convolved Sequence
Amplitude
ω/π
Phase Spectrum of Original Sequence
Phase in radians
ω/π
Phase Spectrum of Convolved Sequence
Phase in radians
ω/π

D) MODULATION PROPERTY OF DTFT

```
clf;
w=-pi:2*pi/255:pi;
x1=[1 3 5 7 9 11 13 15 17];
x2=[1 -1 1 -1 1 -1 1 -1 1];
y=x1.*x2;
h1=freqz(x1,1,w);
h2=freqz(x2,1,w);
h3=freqz(y,1,w);
subplot(3,1,1);
plot(w/pi,abs(h1));
grid;
title('Magnitude Spectrum of First Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(3,1,2);
plot(w/pi,abs(h2));
grid;
title('Magnitude Spectrum of Second Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(3,1,3);
plot(w/pi,abs(h3));
grid;
title('Magnitude Spectrum of Product Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,angle(h1));
grid;
title('Phase Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
subplot(2,2,4);
plot(w/pi,angle(h3));
grid;
title('Phase Spectrum of Convolved Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

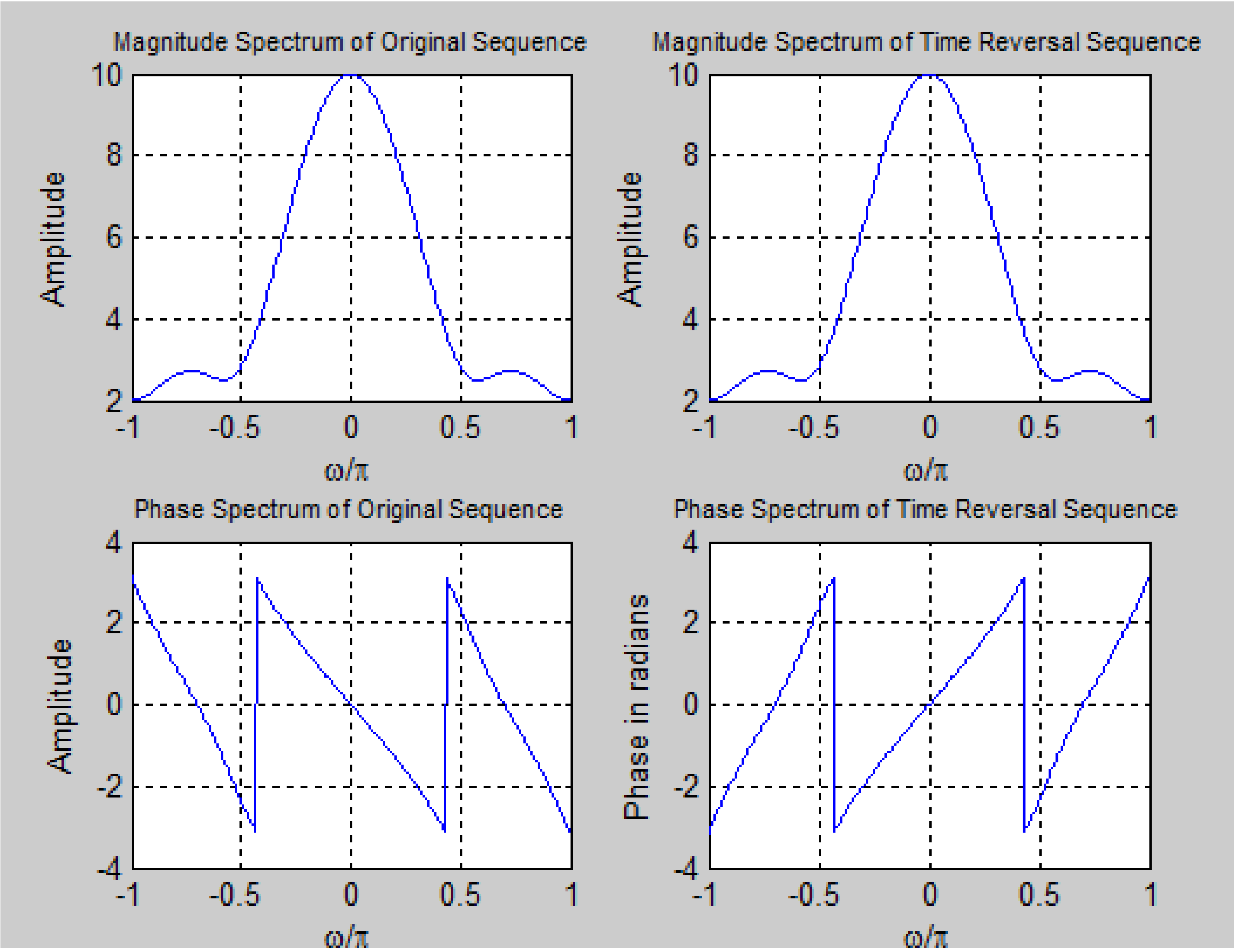
WAVEFORMS:


Magnitude Spectrum of First Sequence
Amplitude
100
50
0
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
ω/π
Magnitude Spectrum of Second Sequence
Amplitude
10
5
0
ω/π
Magnitude Spectrum of Product Sequence
Amplitude
100
50
0
ω/π

E) TIME REVERSAL PROPERTY OF DTFT

```
clf;
w=-pi:2*pi/255:pi;
num=[1 2 3 4];
l=length(num)-1;
h1=freqz(num,1,w);
h2=freqz(fliplr(num),1,w);
h3=exp(w*l*pi).*h2;
subplot(2,2,1);
plot(w/pi,abs(h1));
grid;
title('Magnitude Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2);
plot(w/pi,abs(h3));
grid;
title('Magnitude Spectrum of Time Reversal Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,angle(h1));
grid;
title('Phase Spectrum of Original Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,4);
plot(w/pi,angle(h3));
grid;
title('Phase Spectrum of Time Reversal Sequence','FontSize',8);
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

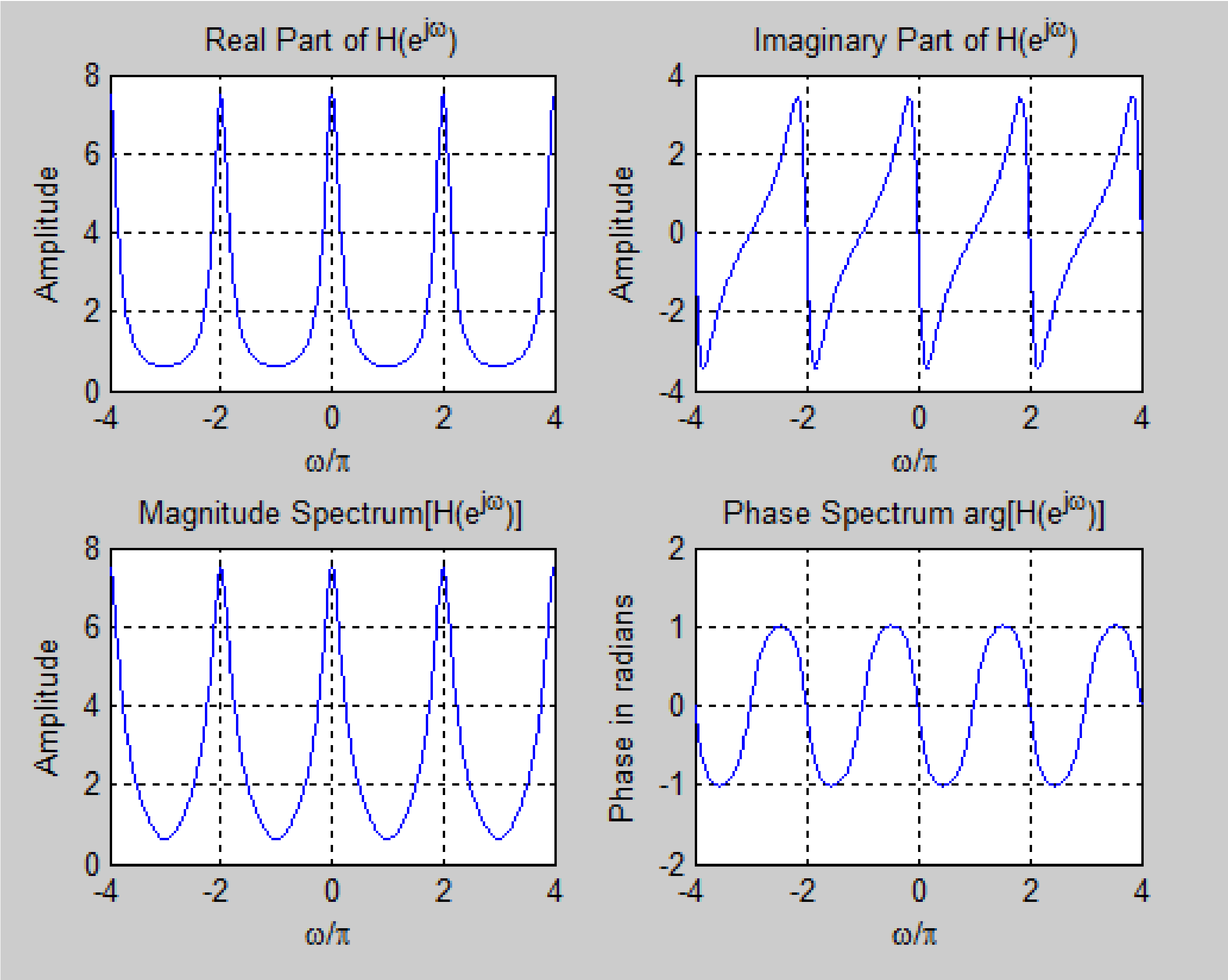
WAVEFORMS:


Magnitude Spectrum of Original Sequence
Magnitude Spectrum of Time Reversal Sequence
Phase Spectrum of Original Sequence
Phase Spectrum of Time Reversal Sequence
Amplitude
Phase in radians
ω/π
10
8
6
4
2
0
-2
-4
-1
-0.5
0
0.5
1

F) EVALUATION OF DTFT 1

```
clf;
w=-4*pi:8*pi/511:4*pi;
num=[2 1];
den=[1 -0.6];
h=freqz(num,den,w);
subplot(2,2,1);
plot(w/pi,real(h));
grid;
title('Real Part of H(e^{j\omega})');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2)
plot(w/pi,imag(h));
grid;
title('Imaginary Part of H(e^{j\omega})');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,abs(h));
grid;
title('Magnitude Spectrum[H(e^{j\omega})]')
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,4);
plot(w/pi,angle(h));
grid;
title('Phase Spectrum arg[H(e^{j\omega})]');
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

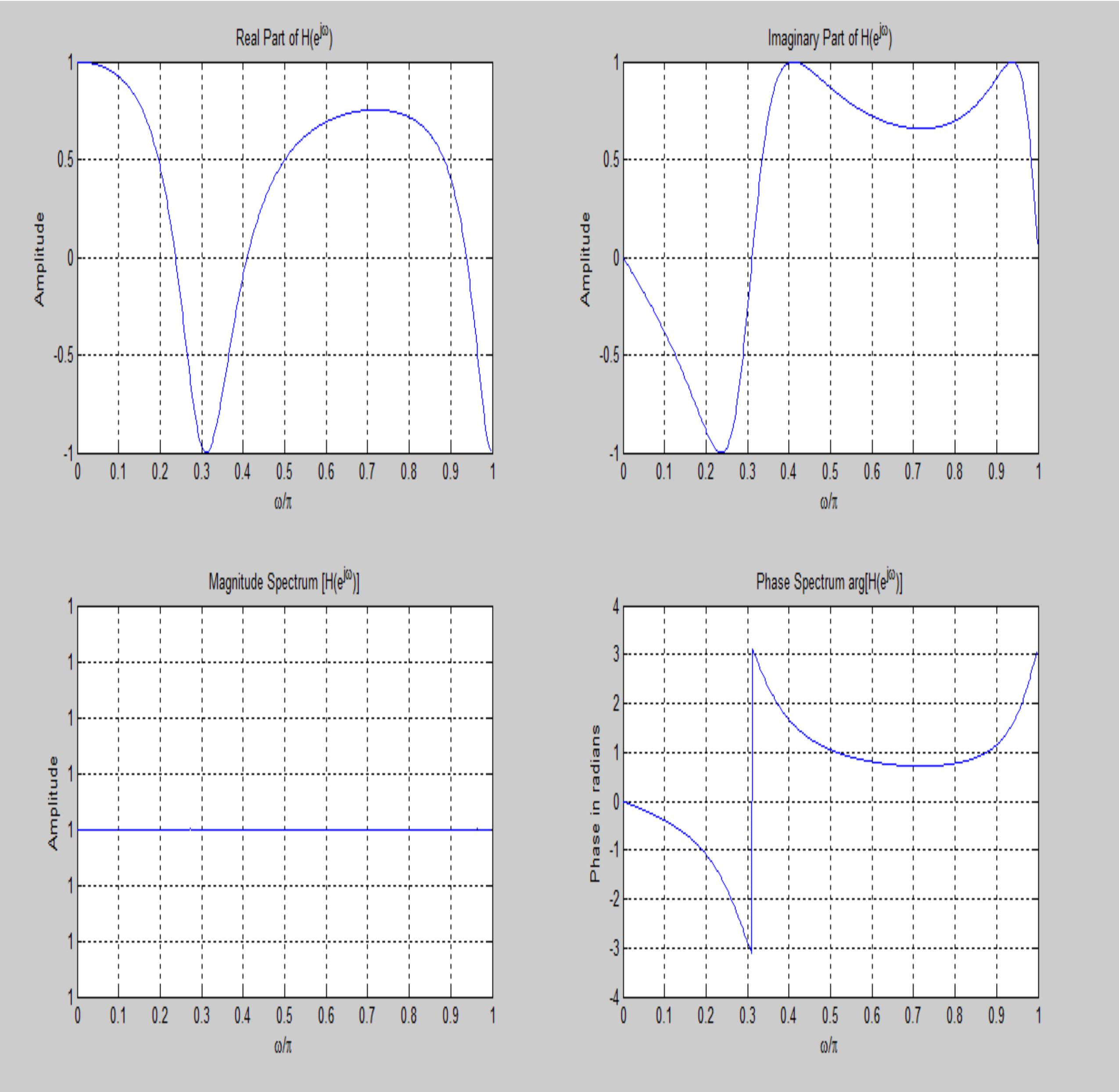
WAVEFORMS:


Real Part of H(e^jω)
Amplitude
ω/π
Imaginary Part of H(e^jω)
Amplitude
ω/π
Magnitude Spectrum[H(e^jω)]
Amplitude
ω/π
Phase Spectrum arg[H(e^jω)]
Phase in radians
ω/π

G) EVALUATION OF DTFT 2

```
clf;
N=512;
num=[0.7 -0.5 0.3 1];
den=[1 0.3 -0.5 0.7];
[h,w]=freqz(num,den,N);
subplot(2,2,1);
plot(w/pi,real(h));
grid;
title('Real Part of H(e^{j\omega})');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2);
plot(w/pi,imag(h));
grid;
title('Imaginary Part of H(e^{j\omega})');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,abs(h));
grid;
title('Magnitude Spectrum [H(e^{j\omega})]');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,4);
plot(w/pi,angle(h));
grid;
title('Phase Spectrum arg[H(e^{j\omega})]');
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

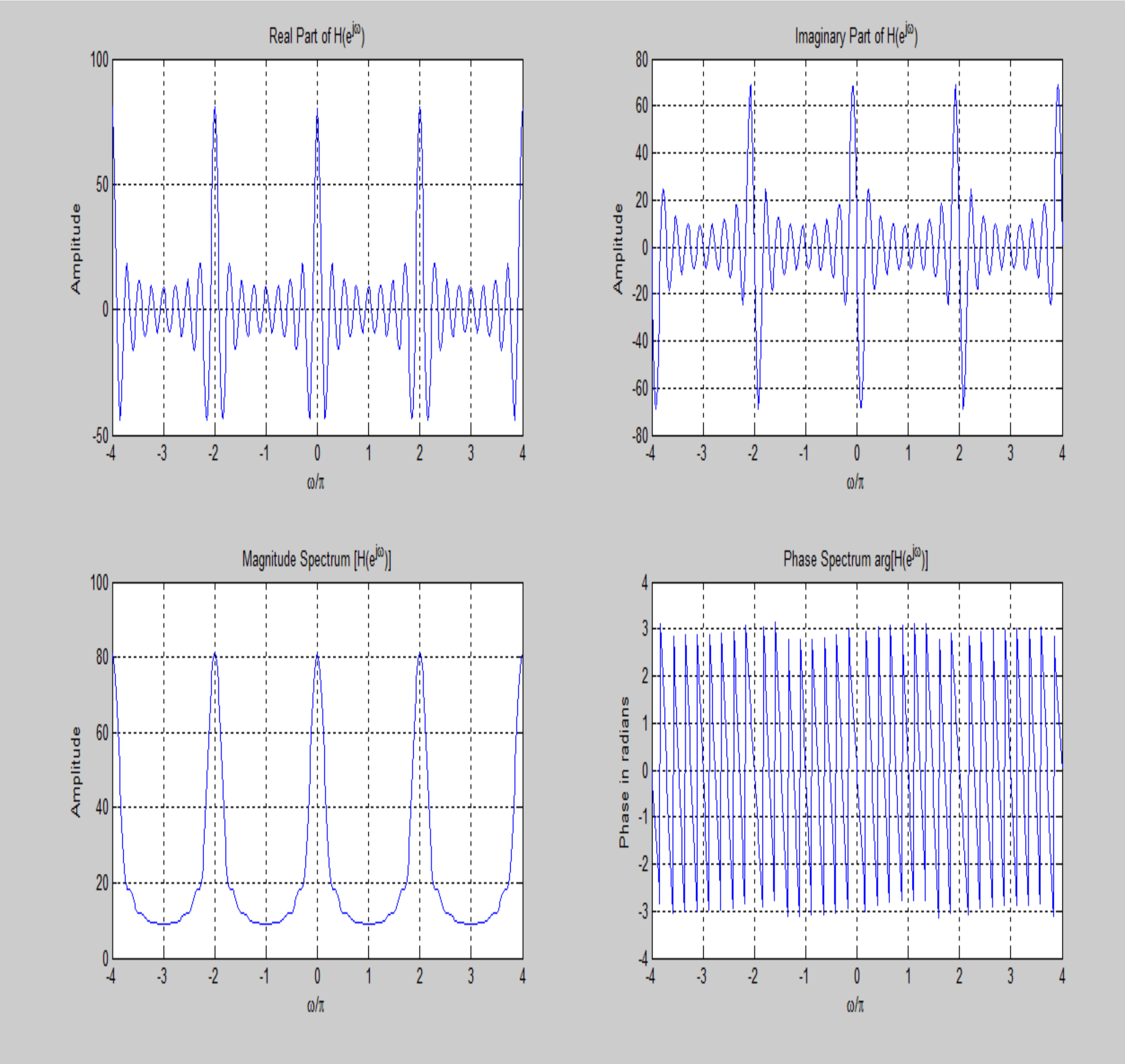
WAVEFORMS:


Real Part of H(e^jω)
Amplitude
ω/π
Imaginary Part of H(e^jω)
Amplitude
ω/π
Magnitude Spectrum [H(e^jω)]
Amplitude
ω/π
Phase Spectrum arg[H(e^jω)]
Phase in radians
ω/π

H) EVALUATION OF DTFT 3

```
clf;
w=-4*pi:8*pi/511:4*pi;
num=[1 3 5 7 9 11 13 15 17];
den=1;
h=freqz(num,den,w);
subplot(2,2,1);
plot(w/pi,real(h));
grid;
title('Real Part of H(e^{j\omega})');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,2);
plot(w/pi,imag(h));
grid;
title('Imaginary Part of H(e^{j\omega})');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,3);
plot(w/pi,abs(h));
grid;
title('Magnitude Spectrum [H(e^{j\omega})]');
xlabel('\omega/\pi');
ylabel('Amplitude');
subplot(2,2,4);
plot(w/pi,angle(h));
grid;
title('Phase Spectrum arg[H(e^{j\omega})]');
xlabel('\omega/\pi');
ylabel('Phase in radians');
```

WAVEFORMS:


Real Part of H(e^jω)
Imaginary Part of H(e^jω)
Magnitude Spectrum |H(e^jω)|
Phase Spectrum arg[H(e^jω)]
Amplitude
Phase in radians
ω/π

5 . Design of N point FFT algorithm

Aim: To write a program to Design of N point FFT algorithm

APPARATUS: PC with MATLAB Software.

PROCEDURE:

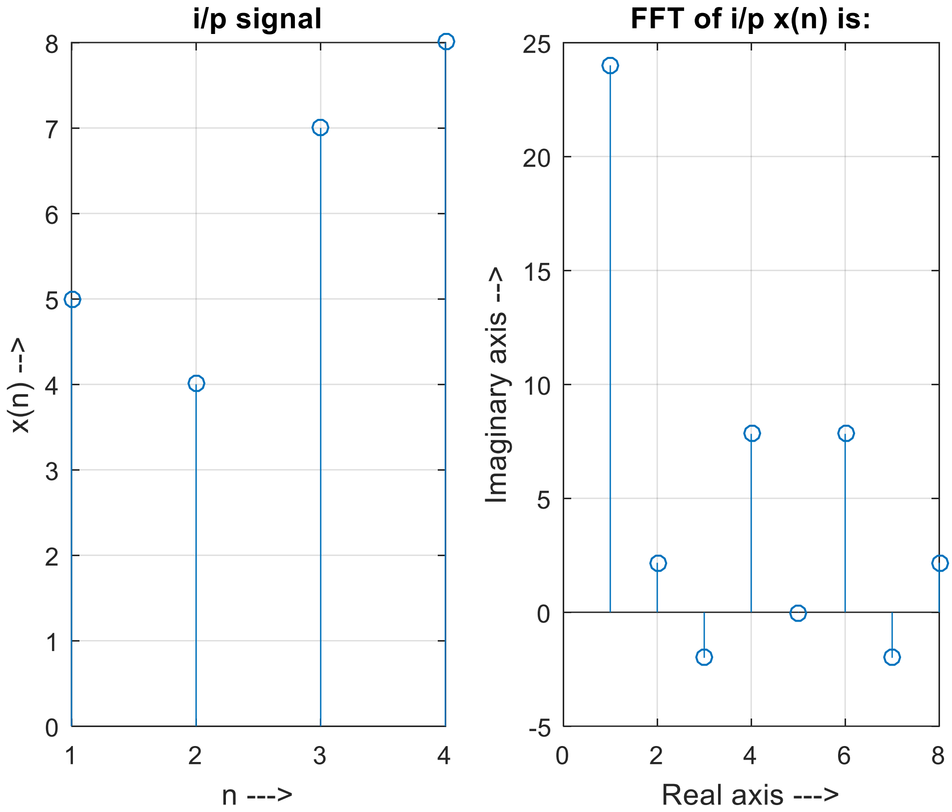
1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

Program:

```
clc;
clear all;
close all;
x=input('Enter the sequence:');
n=input('Enter the length of fft: ');
%compute fft
disp('Fourier transformed signal');
X=fft(x,n);
subplot(1,2,1);
stem(x);
title('i/p signal');
xlabel('n --->');
ylabel('x(n) -->');grid;
subplot(1,2,2);stem(X);
title('FFT of i/p x(n) is:');
xlabel('Real axis --->');
ylabel('Imaginary axis -->');
grid;
```

Output :

Enter the sequence:[5 4 7 8]
Enter the length of fft: 8
Fourier transformed signal



Result :-- N point FFT algorithm using Matlab is designed.

VIVA VOCE

FFT algorithm is designed to perform complex operations.-The FFT algorithm is designed to perform complex multiplications and additions, even though the input data may be real valued. The basic reason for this is that the phase factors are complex and hence, after the first stage of the algorithm, all variables are basically complex valued.

Decimation-in frequency FFT algorithm is used to compute H(k)-The N-point DFT of h(n), which is padded by L-1 zeros, is denoted as H(k). This computation is performed once via the FFT and resulting N complex numbers are stored. To be specific we assume that the decimation-in frequency FFT algorithm is used to compute H(k). This yields H(k) in the bit-reversed order, which is the way it is stored in the memory

How many complex multiplications are need to be performed for each FFT algorithm?-The decimation of the data sequence should be repeated again and again until the resulting sequences are reduced to one point sequences. For N=2v, this decimation can be performed v=log2N times. Thus the total number of complex multiplications is reduced to (N/2)log2N.

6) Design of FIR filter using windowing technique and verify the frequency response of the filter

Aim: To write a matlab program for Design of FIR filter using windowing technique and verify the frequency response of the filter.

APPARATUS: PC with MATLAB Software.

PROCEDURE:

1. OPEN MATLAB

2. File >>New >>Script - Type the program in untitled window

3. File >>Save >> type filename.m in matlab workspace path

4. Debug >>Run. Wave will display at Figure dialog box.

Program:

```
function FIR_Filter_WindowingMethod
% Design and Implementation of FIR Filter Package
% Date: 04/16/2011
% Author: Gang LIU
% Contacts: liug "at" yahoo dot cn
%
% This code is based on the some code from MATLAB Singal Processing
% Toolbox.
%
%. Example MATLAB M-file illustrating FIR filter design and evaluation.
% Finite Impulse Response filter design example
% found in the MATLAB Signal Processing Toolbox
% using the MATLAB FIR1 function (M-file)
close all;
Fs=8000; %Specify Sampling Frequency
Ts=1/Fs; %Sampling period.
Ns=512; %Nr of time samples to be plotted.
t=[0:Ts:Ts*(Ns-1)]; %Make time array that contains Ns elements
%t = [0, Ts, 2Ts, 3Ts,..., (Ns-1)Ts]
f1=500;
f2=1500;
f3=2000;
f4=3000;
x1=sin(2*pi*f1*t); %create sampled sinusoids at different frequencies
x2=sin(2*pi*f2*t);
x3=sin(2*pi*f3*t);
x4=sin(2*pi*f4*t);
x=x1+x2+x3+x4; %Calculate samples for a 4-tone input signal
```

```
%N=16; %FIR1 requires filter order (N) to be EVEN
%when gain = 1 at Fs/2.
%W=[0.4 0.6]; %Specify Bandstop filter with stop band between
%0.4*(Fs/2) and 0.6*(Fs/2)
%B=FIR1(N,W,'DC-1'); %Design FIR Filter using default (Hamming window.
disp('Please specify the filter type');
FilterType={'low','high','bandpass','stop'};
disp('Here is the filter option:');
disp('1: Low Pass Filter ');
disp('2: High Pass Filter');
disp('3: Band Pass Filter');
disp('4: Band Stop Filter');
disp('You only need enter number. EX. 4 means Band Stop filter')
FilterOption=input('FilterOption=');
while FilterOption ~=1 && FilterOption ~=2 && FilterOption ~=3 && FilterOption ~=4
disp('You only need enter number : 1,2,3 or 4. EX. 4 means Band Stop filter');
FilterOption=input('FilterOption=');
end
userFilterType=FilterType{FilterOption};
disp('Enter the order of the filter(Must be a positive even number, should be less than 100');
FilterOrder=input('FilterOrder=');
str=sprintf('Here is all the frequency infromation in this experiment:Fs=%d; f1=%d; f2=%d; f3=%d;
f4=%d;',Fs,f1,f2,f3,f4);
disp(str);
if FilterOption==1
disp('Enter the Edge Freq. in Hz. Note: the value should be within 0~Fs');W=input('omega=');
elseif FilterOption==2
disp('Enter the Edge Freq. in Hz. Note: the value should be within 0~Fs');W=input('omega=');
elseif FilterOption==3
disp('Enter the Edge Freq. in Hz. Note: the value should be within 0~Fs');W=input('omega
cutoff1=');W2=input('omega cutoff2='); W=[W W2];
elseif FilterOption==4
disp('Enter the Edge Freq. in Hz. Note: the value should be within 0~Fs');W=input('omega
cutoff1=');W2=input('omega cutoff2='); W=[W W2];
end
%Fs, W,
W=W./(Fs/2); % [1600 2400]
N=FilterOrder;
Window_Option={'hamming','kaiser','rectwin'};
disp('Here is the Windowing option:');
disp('1: hamming');
disp('2: kaiser');
disp('3: rectwin');
WindowOption=input('WindowOption=');
while WindowOption ~=1 && WindowOption ~=2 && WindowOption ~=3
disp('You only need enter number : 1,2,or 3. EX. 3 means rectwin');
```

```
WindowOption=input('WindowOption=');
end
%WindowOption='rectwin';
switch Window_Option{WindowOption}
case 'hamming'
userWindow=hamming(N+1);
case 'kaiser'
userWindow=kaiser(N+1);
case 'rectwin'
userWindow=rectwin(N+1);
otherwise
disp('Wrong windowing option. EXIT');
eixt
end
%B=FIR1(N,W,'bandpass',userWindow); %Design FIR Filter using default (Hamming window.
B=fir1(N,W,userFilterType,userWindow); %Design FIR Filter using default (Hamming window.
%B %Leaving off semi-colon causes contents of
%B (the FIR coefficients) to be displayed.
A=1; %FIR filters have no poles, only zeros.
figure;
zerophase(B,A); % plot zero phase figure
disp('Checking the Zero-Phase Response of the designed filter, enter any key to continue');
pause;
figure; %Create a new figure window, so previous one isn't lost.
freqz(B,A); %Plot frequency response - both amp and phase response.
disp('Checking the Freq. Response of the designed filter, enter any key to continue');
pause;
figure;
subplot(2,1,1); %Two subplots will go on this figure window.
Npts=200;
plot(t(1:Npts),x(1:Npts)) %Plot first Npts of this 4-tone input signal
title('Time Plots of Input and Output');
xlabel('time (s)');
ylabel('Input Sig');
%Now apply this filter to our 4-tone test sequence
y = filter(B,A,x);
subplot(2,1,2); %Now go to bottom subplot.
plot(t(1:Npts),y(1:Npts)); %Plot first Npts of filtered signal.
xlabel('time (s)');
ylabel('Filtered Sig');
disp('Checking the Filtering Effect of the designed filter in time domain, enter any key to continue');
pause;
figure; %Create a new figure window, so previous one isn't lost.
subplot(2,1,1);
xfftmag=(abs(fft(x,Ns))); %Compute spectrum of input signal.
xfftmagh=xfftmag(1:length(xfftmag)/2);
```

```
%Plot only the first half of FFT, since second half is mirror imag
%the first half represents the useful range of frequencies from
%0 to Fs/2, the Nyquist sampling limit.
f=[1:1:length(xfftmagh)]*Fs/Ns; %Make freq array that varies from
%0 Hz to Fs/2 Hz.
plot(f,xfftmagh); %Plot frequency spectrum of input signal
title('Input and Output Spectra');
xlabel('freq (Hz)');
ylabel('Input Spectrum');
subplot(2,1,2);
yfftmag=(abs(fft(y,Ns)));
yfftmagh=yfftmag(1:length(yfftmag)/2);
%Plot only the first half of FFT, since second half is mirror image
%the first half represents the useful range of frequencies from
%0 to Fs/2, the Nyquist sampling limit.
plot(f,yfftmagh); %Plot frequency spectrum of input signal
xlabel('freq (Hz)');
ylabel('Filtered Signal Spectrum');
```

OUTPUT :

Please specify the filter type

Here is the filter option:

1: Low Pass Filter

2: High Pass Filter

3: Band Pass Filter

4: Band Stop Filter

You only need enter number. EX. 4 means Band Stop filter

FilterOption=1

Enter the order of the filter(Must be a positive even number, should be less than 100

FilterOrder=6

Here is all the frequency infromation in this experiment:Fs=8000; f1=500; f2=1500; f3=2000; f4=3000;

Enter the Edge Freq. in Hz. Note: the value should be within 0~Fs

omega=1000

Here is the Windowing option:

1: hamming

2: kaiser

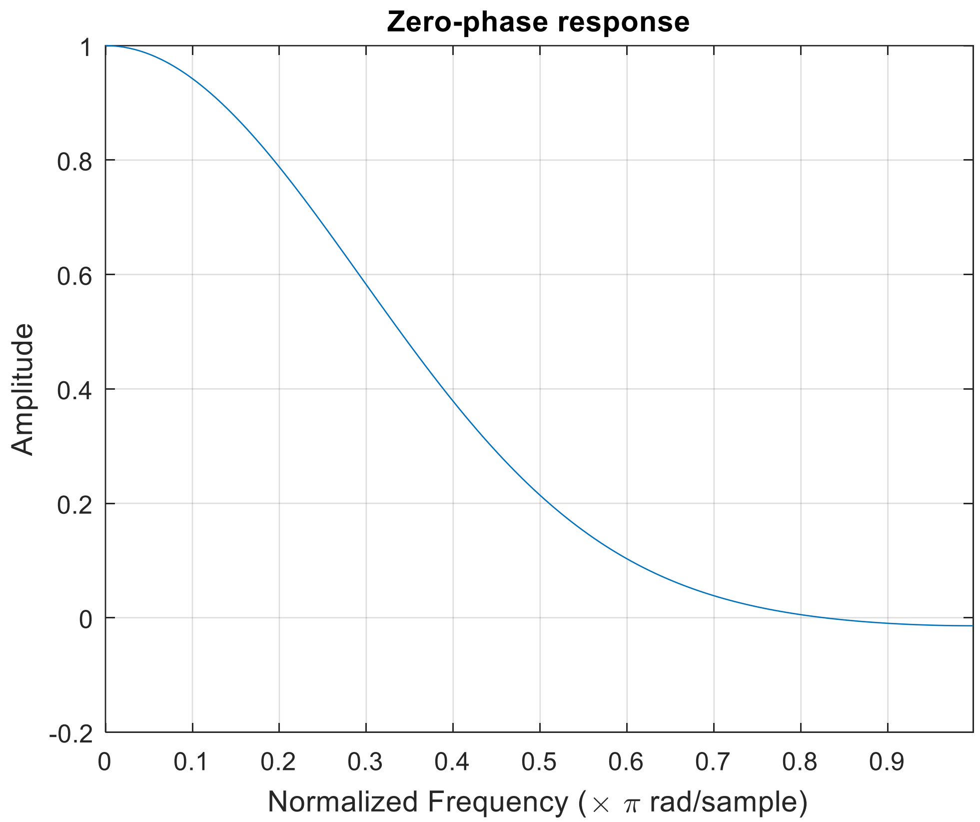
3: rectwin

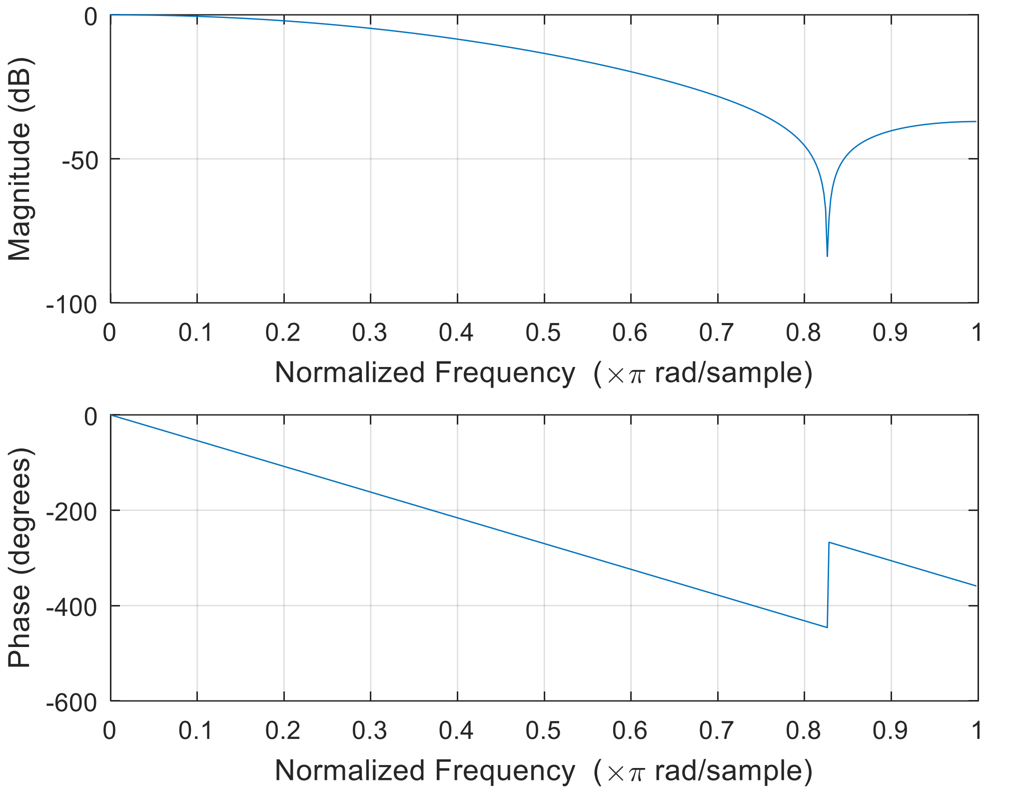
WindowOption=1

Checking the Zero-Phase Response of the designed filter, enter any key to continue

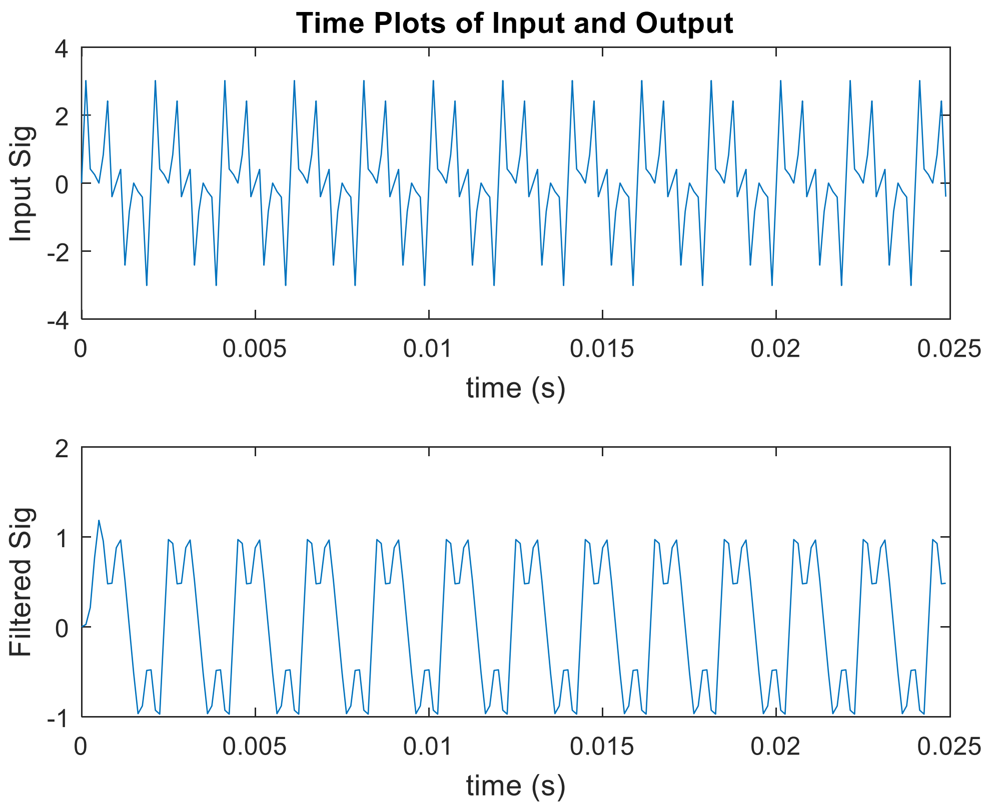
Checking the Freq. Response of the designed filter, enter any key to continue

Checking the Filtering Effect of the designed filter in time domain, enter any key to continue

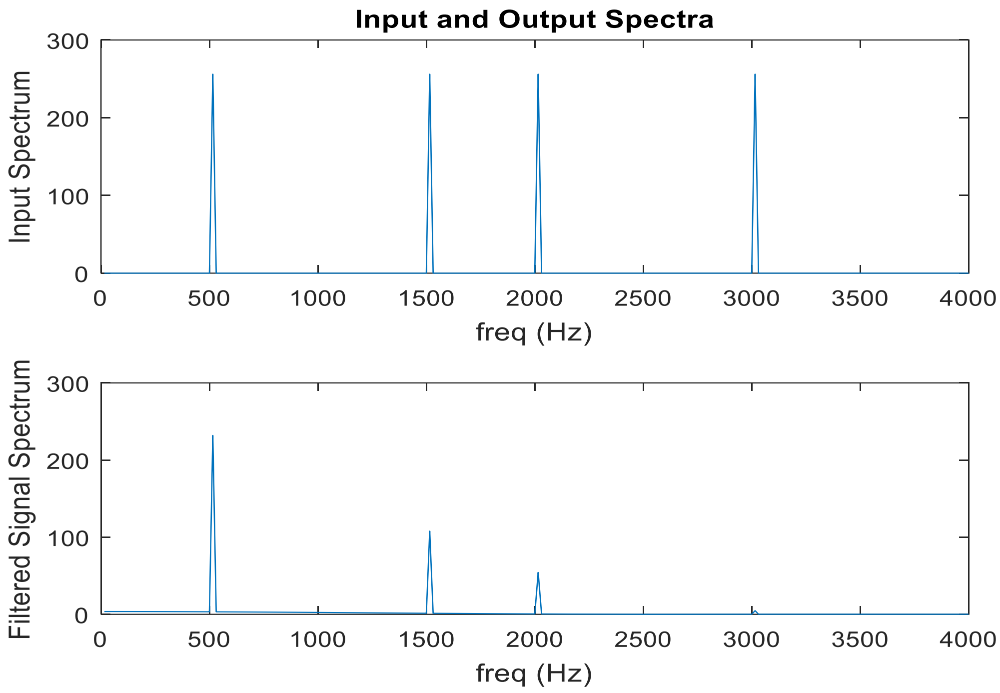




Magnitude (dB)
0
-50
-100
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency ($\times\pi$ rad/sample)
Phase (degrees)
0
-200
-400
-600
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalized Frequency ($\times\pi$ rad/sample)



Time Plots of Input and Output
Input Sig
4
2
0
-2
-4
0 0.005 0.01 0.015 0.02 0.025
time (s)
Filtered Sig
2
1
0
-1
0 0.005 0.01 0.015 0.02 0.025
time (s)



Result:-- FIR Filter using Matlab is designed.

VIVA VOCE

What are the desirable characteristics of the frequency response of window function?

Advantages:

a) FIR filters have exact linear phase.

b) FIR filters are always stable.

c) FIR filters can be realized in both recursive and non recursive structure.

d) Filters with any arbitrary magnitude response can be tackled using FIR sequency.

Disadvantages:

a) For the same filter specifications the order of FIR filter design can be as high as 5 to n10 times that of IIR design.

b) Large storage requirements needed.

c) Powerful computational facilities required for the implementation**.**

The Optimum Equi ripple design Criterion is used for designing FIR Filters with Equal level filteration throughout the Design.

FIR Filter is always stable.

FIR Filter with exactly linear phase can easily be designed.

7) Design of IIR filter using anyof the available methods and verify the frequency response of the filter

Aim: To write a matlab program for design of IIR filter using anyof the available methods and verify the frequency response of the filter.

APPARATUS: PC with MATLAB Software.

PROCEDURE:

1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

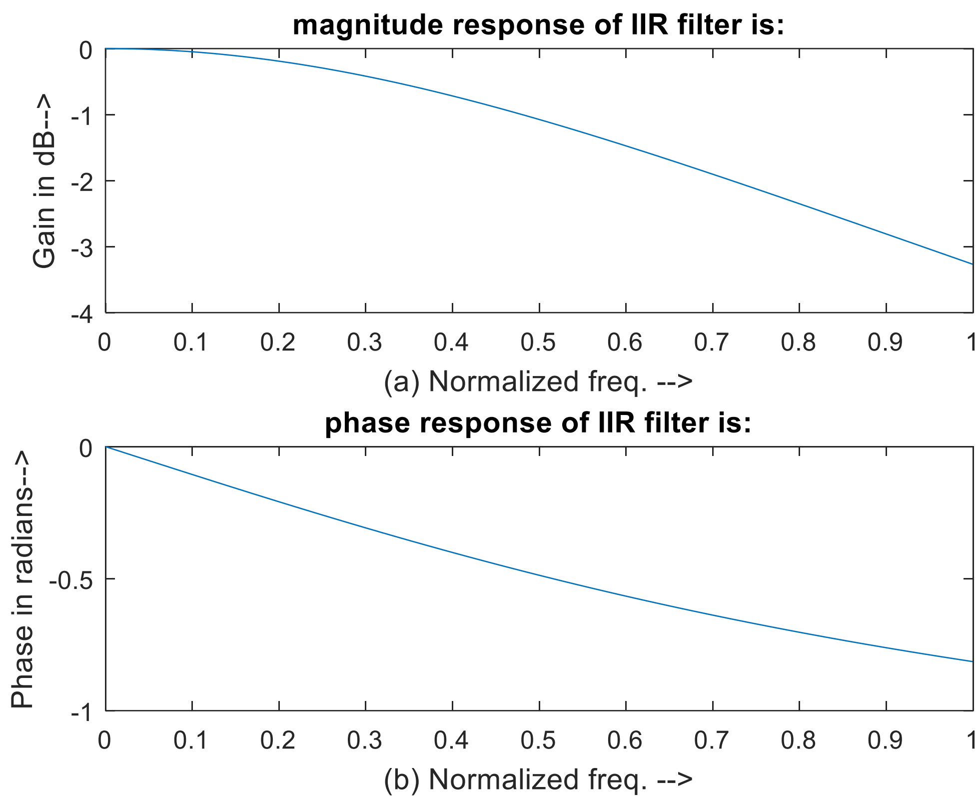
Program:

```
% IIR FILTERS LPF & HPF USING MATLAB

clc;

clear all;

close all;

disp('enter the IIR filter design specifications');

rp=input('enter the passband ripple');

rs=input('enter the stopband ripple');

wp=input('enter the passband freq');

ws=input('enter the stopband freq');

fs=input('enter the sampling freq');

w1=2*wp/fs;w2=2*ws/fs;

[n,wn]=buttord(w1,w2,rp,rs,'s');

% IIR FILTERS LPF & HPF USING MATLAB

clc;

clear all;

close all;

disp('enter the IIR filter design specifications');
```

```
rp=input('enter the passband ripple');

rs=input('enter the stopband ripple');

wp=input('enter the passband freq');

ws=input('enter the stopband freq');

fs=input('enter the sampling freq');

w1=2*wp/fs;w2=2*ws/fs;

[n,wn]=buttord(w1,w2,rp,rs,'s');

c=input('enter choice of filter 1. LPF 2. HPF \n ');

if(c==1)

disp('Frequency response of IIR LPF is:');

[b,a]=butter(n,wn,'low','s');

end

if(c==2)

disp('Frequency response of IIR HPF is:');

[b,a]=butter(n,wn,'high','s');

end

w=0:.01:pi;

[h,om]=freqs(b,a,w);

m=20*log10(abs(h));

an=angle(h);

figure,subplot(2,1,1);plot(om/pi,m);

title('magnitude response of IIR filter is:');

xlabel('(a) Normalized freq. -->');

ylabel('Gain in dB-->');

subplot(2,1,2);plot(om/pi,an);

title('phase response of IIR filter is:');

xlabel('(b) Normalized freq. -->');

ylabel('Phase in radians-->');
```

Output :--

```
enter the IIR filter design specifications

enter the passband ripple.5

enter the stopband ripple.8

enter the passband freq1000

enter the stopband freq2000

enter the sampling freq3000

enter choice of filter 1. LPF 2. HPF

 1

Frequency response of IIR LPF is:
```



Result:-- IIR filters using matlab is designed.

8) Design of analog filters

Aim:To writa a matlab program for design of analog filters

APPARATUS: PC with MATLAB Software.

PROCEDURE:

1. OPEN MATLAB
2. File >>New >>Script - Type the program in untitled window
3. File >>Save >> type filename.m in matlab workspace path
4. Debug >>Run. Wave will display at Figure dialog box.

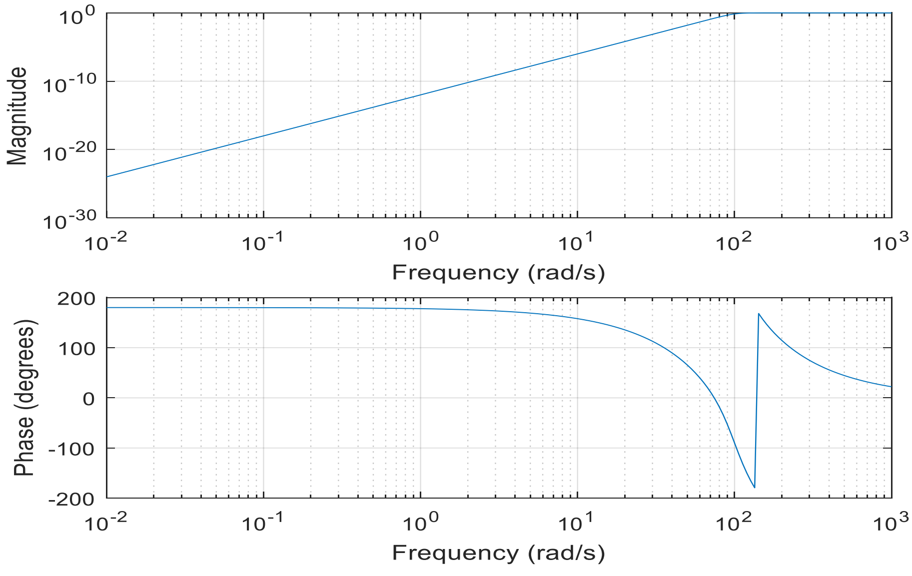
Program:

```
clc;
clear all;
close all;
disp('Enter the Analog filter design specifications');
N=input('Enter the order of the filter');
c=input('Enter the choice of filter 1. LPF 2. HPF 3.BPF 4.BSF \n ');
if(c==1)
disp('Frequency response of Analog LPF is:');
Cf=100;
[b,a]=butter(N,Cf,'S');
freqs(b,a);
end
if(c==2)
disp('Frequency response of Analog HPF is:');
Cf=100;
[b,a]=butter(N,Cf,'HIGH','S');
freqs(b,a);
end
if(c==3)
```

```
disp('Frequency response of Analog BPF is:');

Cf1=[10 100];

[b,a]=butter(N,Cf1,'S');

freqs(b,a);

end

if(c==4)

disp('Frequency response of Analog BPF is:');

Cf1=[10 100];

[b,a]=butter(N,Cf1,'STOP','S');

freqs(b,a);

end
```

OUTPUT :

```
Enter the Analog filter design specifications

Enter the order of the filter6

Enter the choice of filter 1. LPF 2. HPF 3.BPF 4.BSF

 2

Frequency response of Analog HPF is:
```



Result :-- Analog filters using Matlab is designed.

CC STUDIO PROGRAMS

TMS320C6x DIGITAL SIGNAL PROCESSOR

INTRODUCTION

The TMS320C6713 (C6713) is based on the VLIW architecture, which is very well suited for numerically intensive algorithms. The internal program memory is structured so that a total of eight instructions can be fetched every cycle. For example, with a clock rate of 225MHz, the C6713 is capable of fetching eight 32-

bit instructions every 1/(225 MHz) or 4.44 ns. Features of the C6713 include 264 kB of internal memory (8kB as L1P and L1D Cache and 256kB as L2 memory shared between program and data space), eight functional or execution units composed of six arithmetic-logic units (ALUs) and two multiplier units, a 32-bit address bus to address 4 GB (gigabytes), and two sets of 32-bit general-purpose registers.

The C67xx (such as the C6701, C6711, and C6713) belong to the family of the C6x floating-point processors, whereas the C62xx and C64xx belong to the family of the C6x fixed-point processors. The C6713 is capable of both fixed- and floatingpoint processing.

An application-specific integrated circuit (ASIC) has a DSP core with customized circuitry for a specific application. A C6x processor can be used as a standard general-purpose DSp programmed for a specific application. Specific purpose digital signal processors are the modem, echo canceler, and others. A

fixed-point processor is better for devices that use batteries, such as cellular phones, since it uses less power than does an equivalent floating-point processor. The fixed-point processors, C1x, C2x, and C5x, are 16-bit processors with limited dynamic range and precision. The C6x fixed-point processor is a 32- bit processor with improved dynamic range and precision. In a fixed-point processor, it is necessary to scale the data. Overflow, which occurs when an operation such as the addition of two numbers produces a result with more bits than can fit within a processor’s register, becomes a concern.

A floating-point processor is generally more expensive since it has more “real estate” or is a larger chip because of additional circuitry necessary to handle integer as well as floating-point arithmetic. Several factors, such as cost, power consumption, and speed, come into play when choosing a specific DSp.

The C6x processors are particularly useful for applications requiring intensive computations. Family members of the C6x include both fixed-point (e.g., C62x, C64x) and floating-point (e.g., C67x) processors. Other DSp’s are also available from companies such as Motorola and Analog Devices.

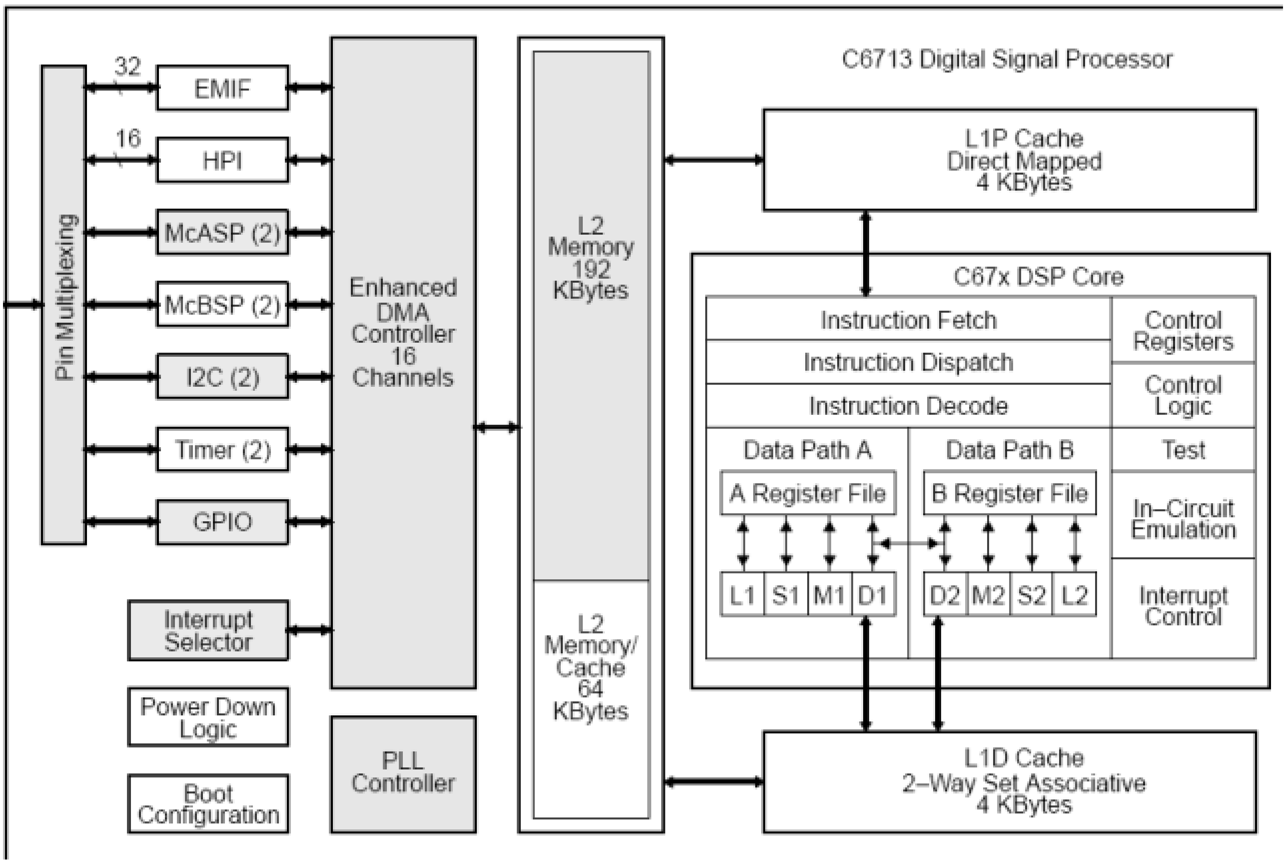


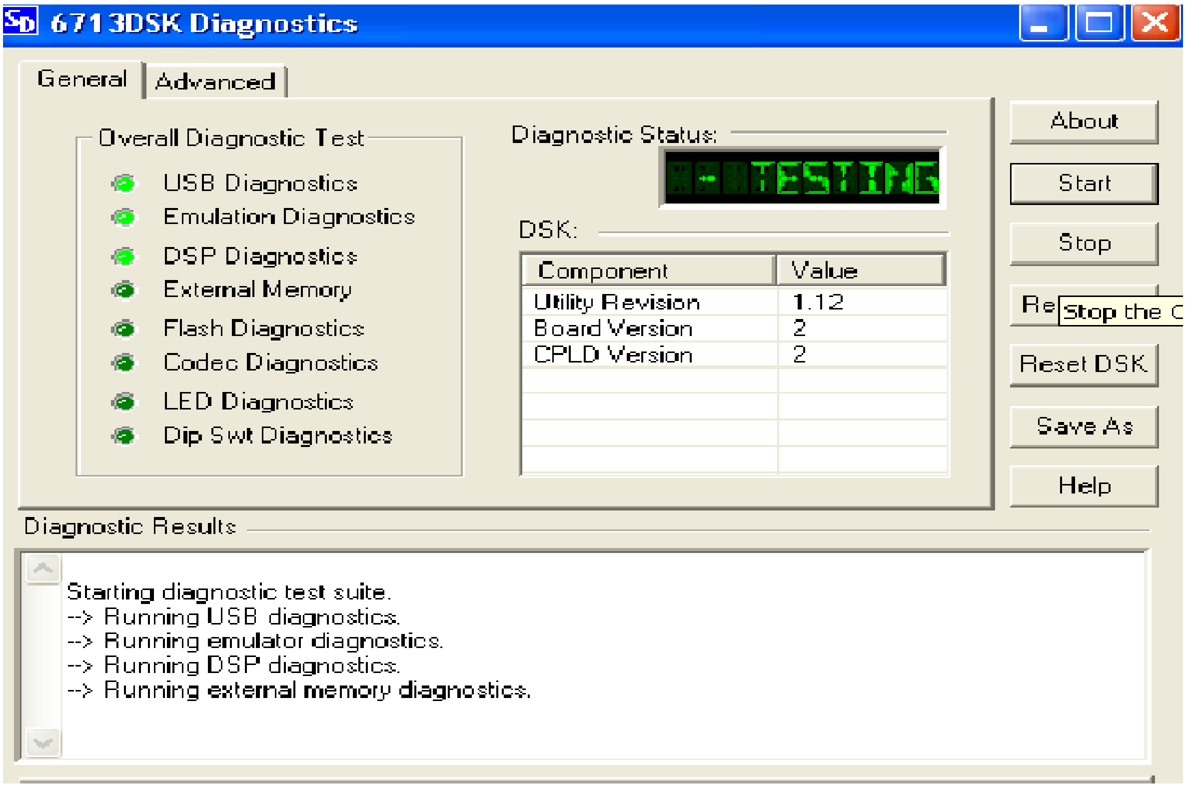
Figure : Functional block diagram of TMS320C6713

The TMS320C6713 onboard the DSK is a floating-point processor based on the VLIW architecture [6–10]. Internal memory includes a two-level cache architecture with 4 kB of level 1 program cache (L1P), 4 kB of level 1 data cache (L1D), and 256 kB of level 2 memory shared between program and data space.

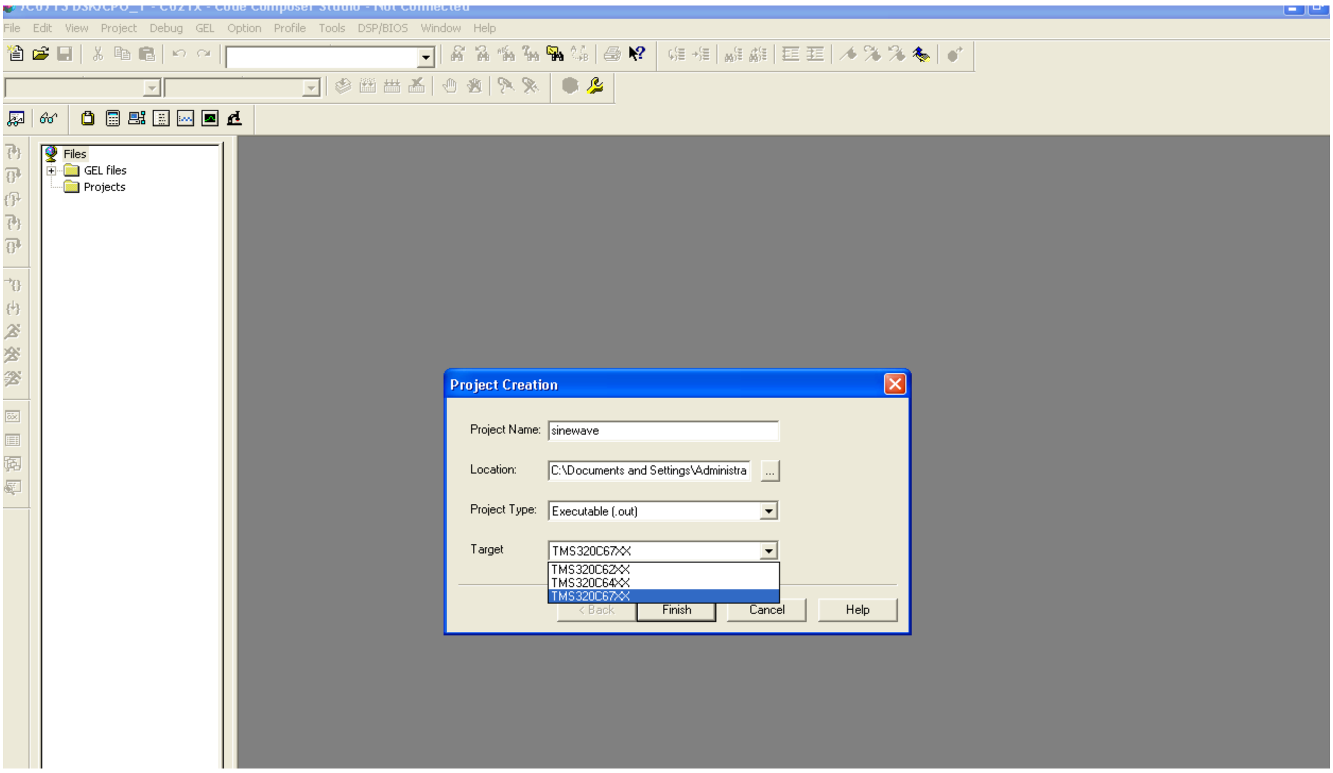
It has a glueless (direct) interface to both synchronous memories (SDRAM and SBSRAM) and asynchronous memories (SRAM and EPROM). Synchronous memory requires clocking but provides a compromise between static SRAM and dynamic DRAM, with SRAM being faster but more expensive than DRAM. On-chip peripherals include two McBSPs, two timers, a host port interface (HPI), and a 32-bit EMIF. It requires 3.3 V for I/O and 1.26 V for the core (internal). Internal buses include a 32-bit program address bus, a 256-bit program data bus to accommodate eight 32-bit instructions, two 32-bit data address buses, two 64- bit data buses, and two 64-bit store data buses.With a 32-bit address bus, the total memory space is 232 = 4GB, including four external memory spaces: CE0, CE1, CE2, and CE3. Figure shows a functional block diagram of the C6713 processor included with CCS.

TMS320C6X- PROCEDURE FOR USING C60 DEBUGGER

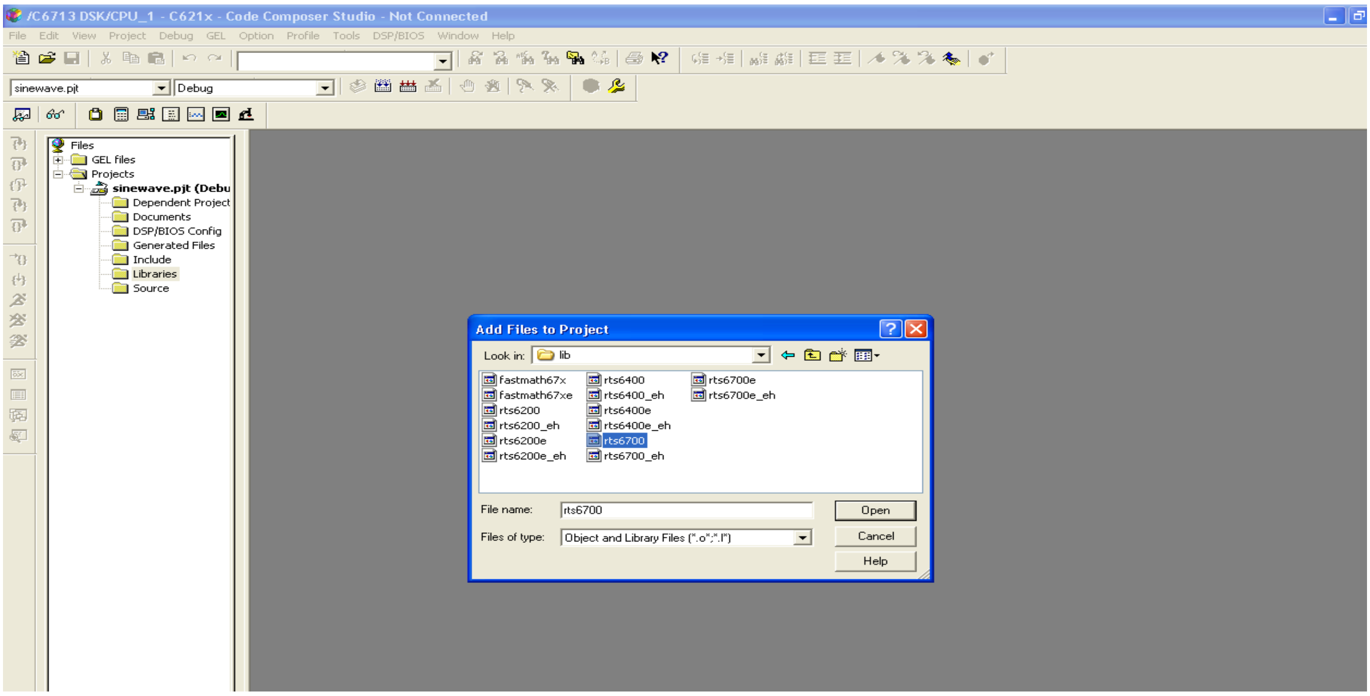
Step1: Open → 6177 Diagnostic → start → Stop → Close



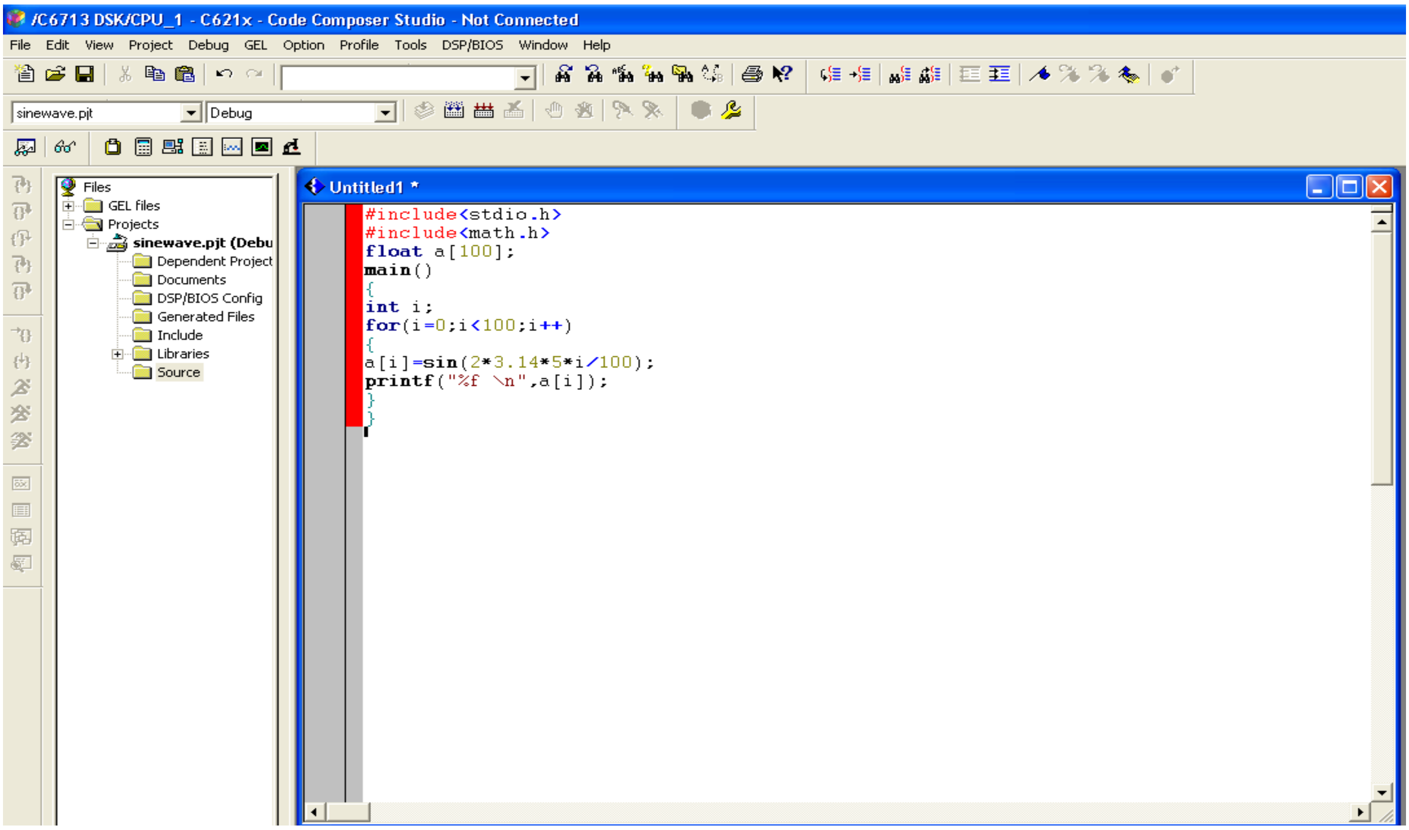
Step 2: Project → New → Project Name → Target → TMS320C67XX



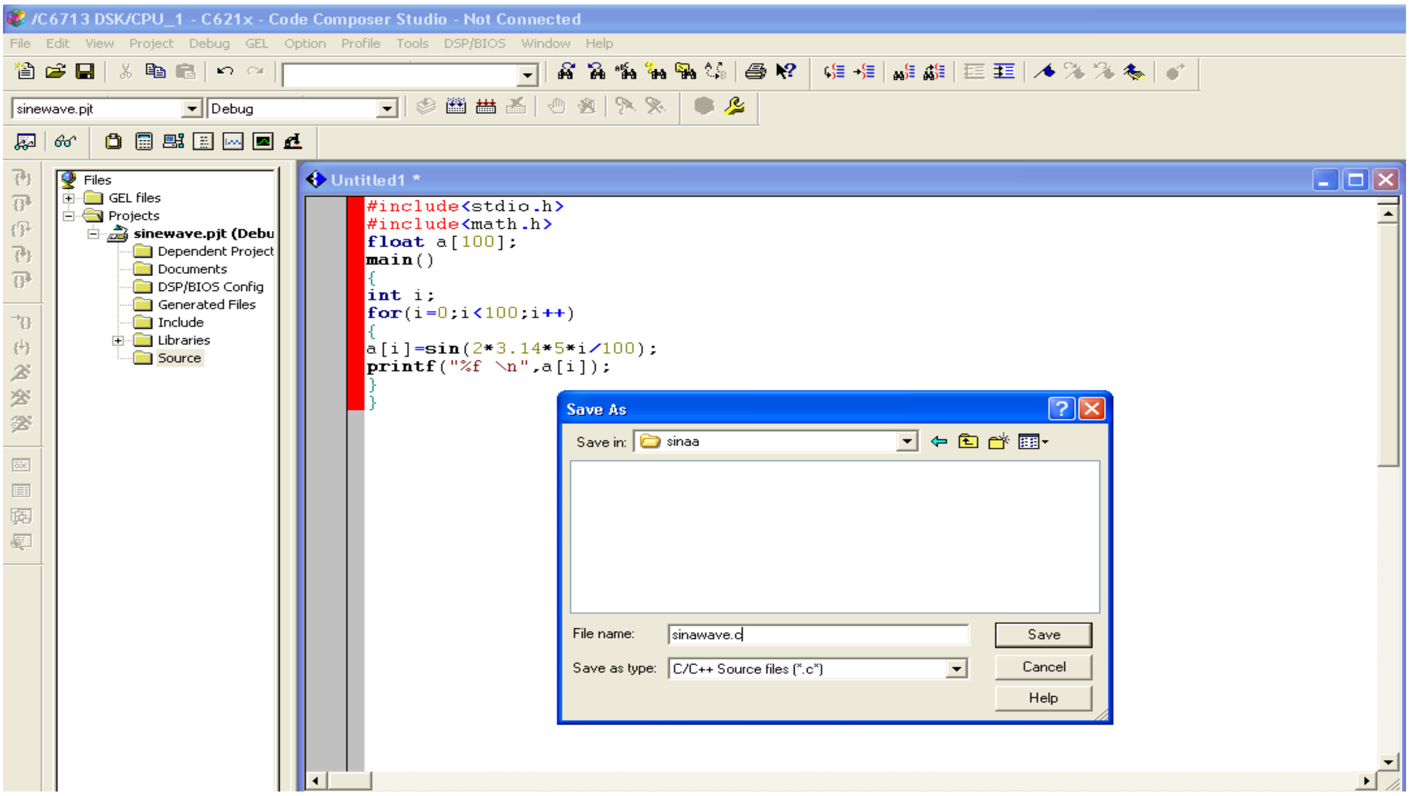
Step 3: Libraries → Add Files to Project → My Project → C600 → CG Tools → Library → rts6700.lib



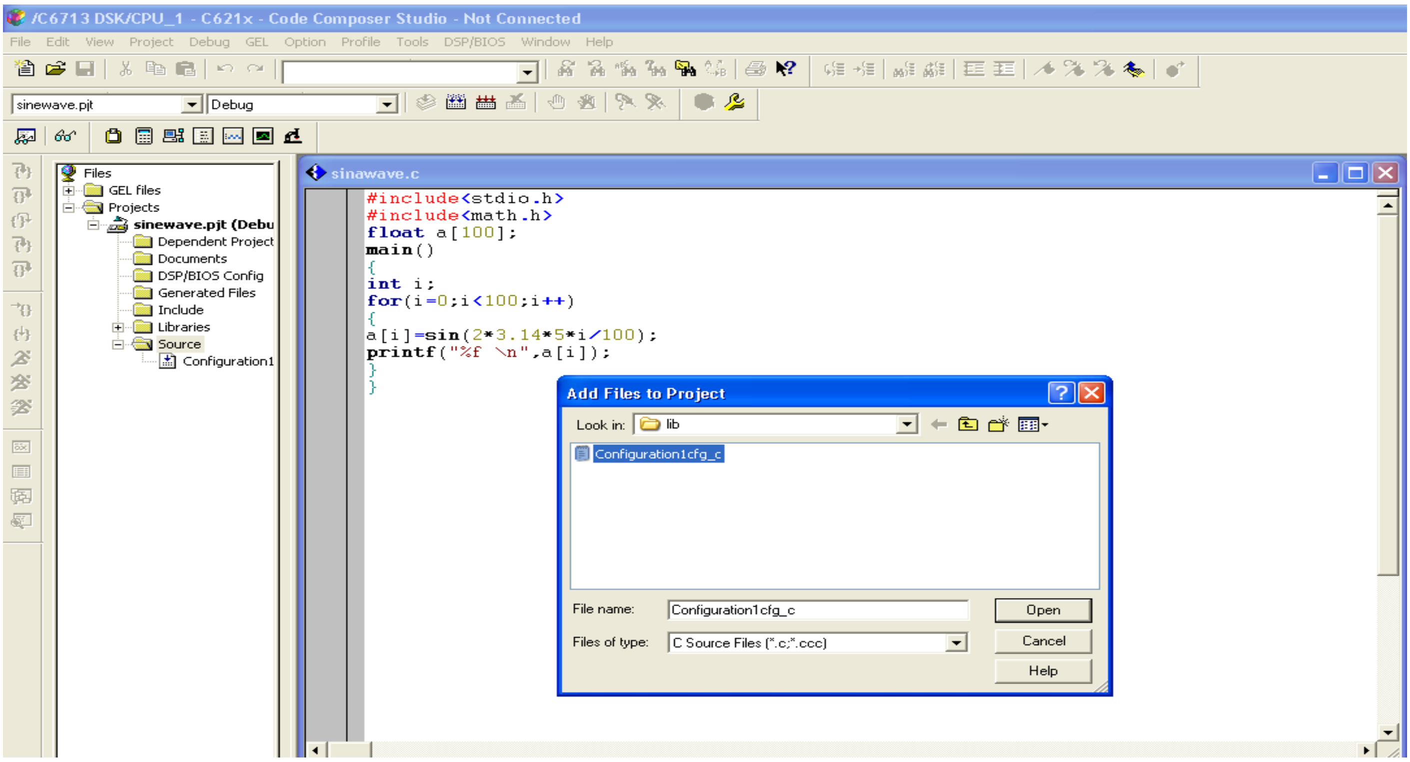
Step 4: File → New → Source File → Write Program in window



Step 5: File → Save as → CStudio → my project → sss.c

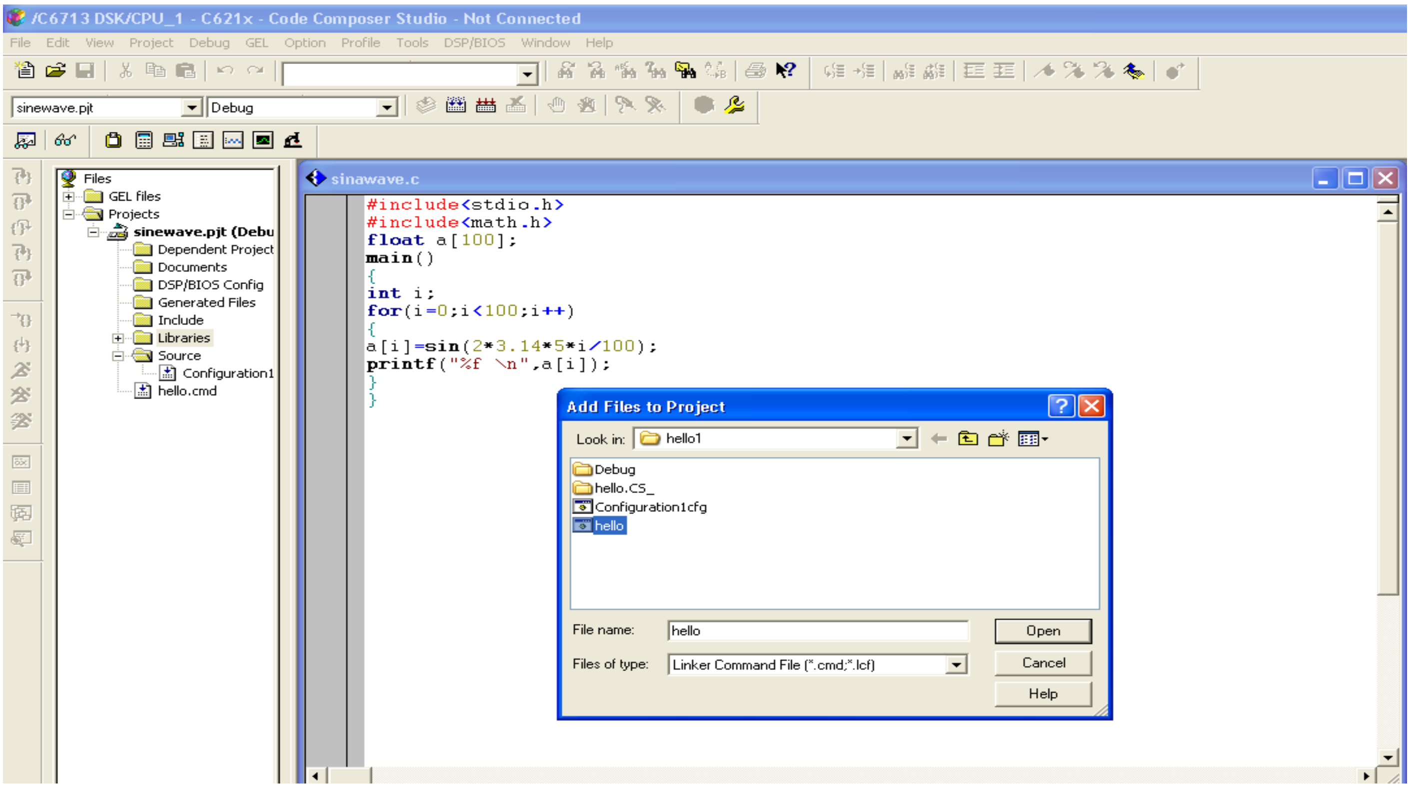


Step 6: Source → Add Files to project → CStudio → lib → confi.c

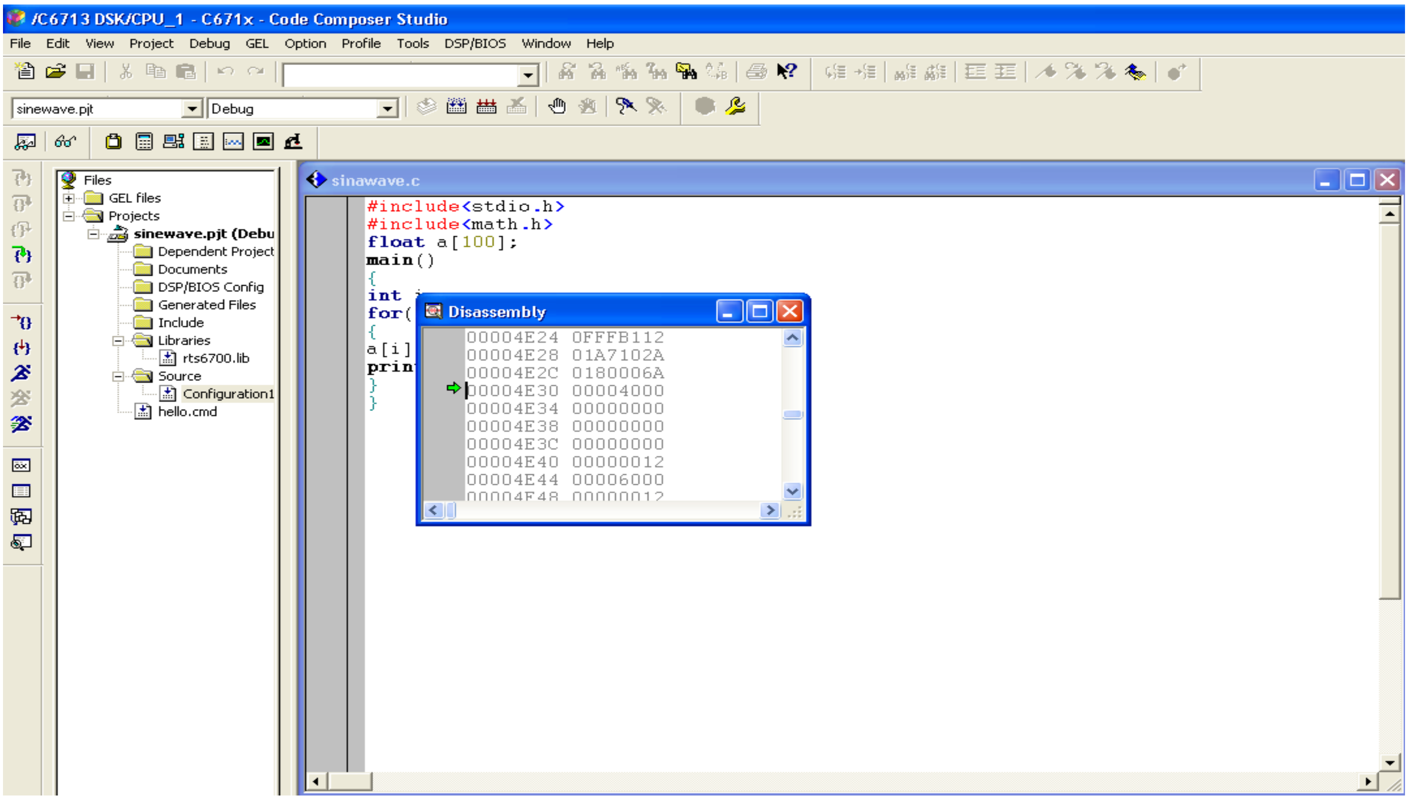


Step 7: Source → Add Files to Project → CStudio → C600 → Tutorial →

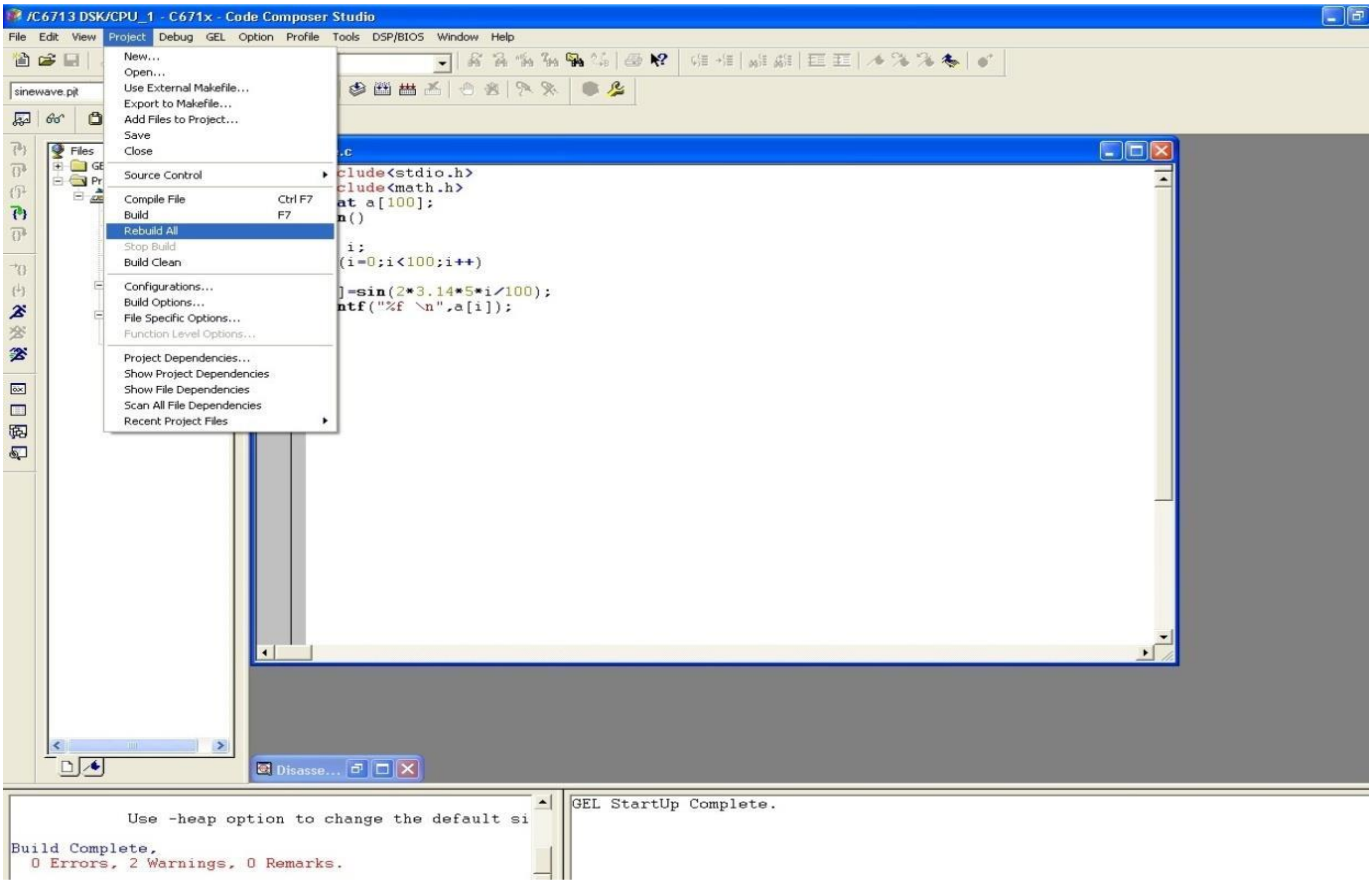
Dsk6713 → hello1 → link cmd → hello.cmd



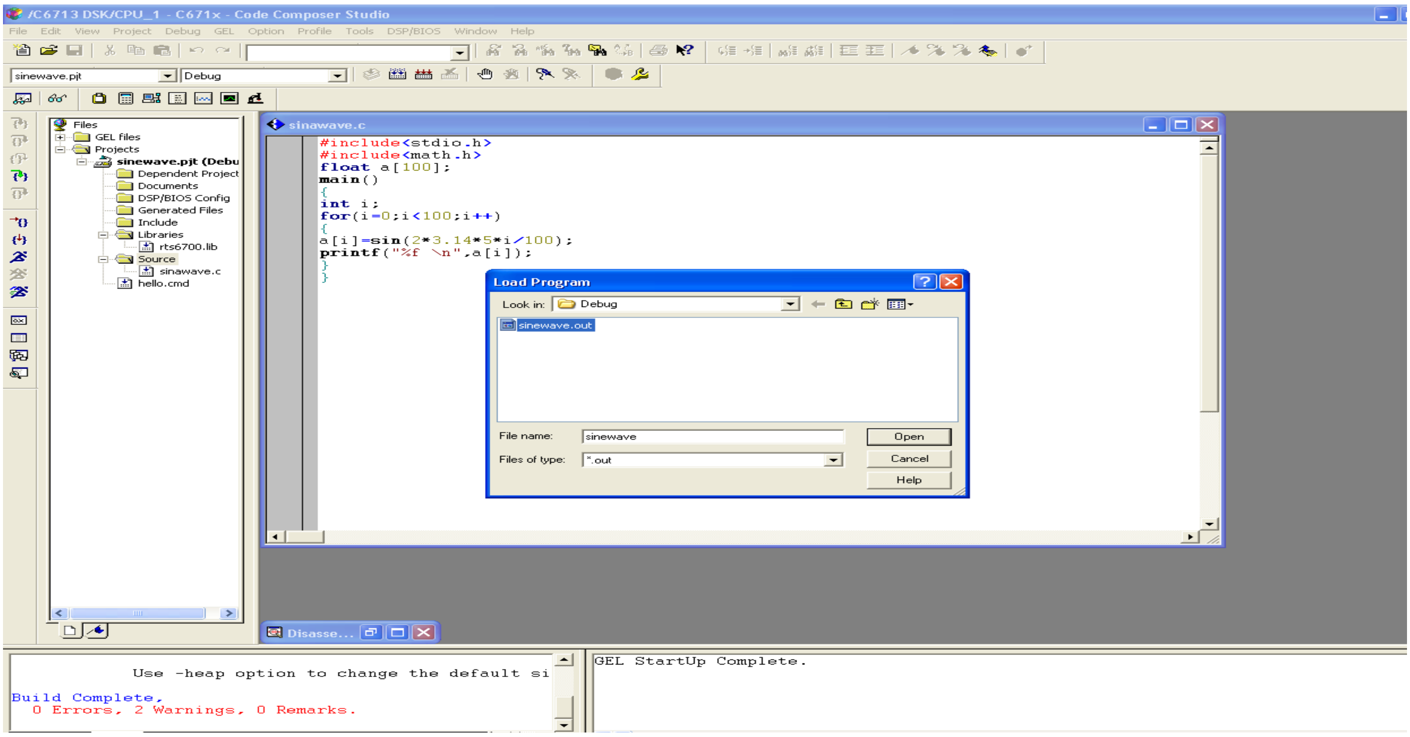
Step 8: Project → Compile File



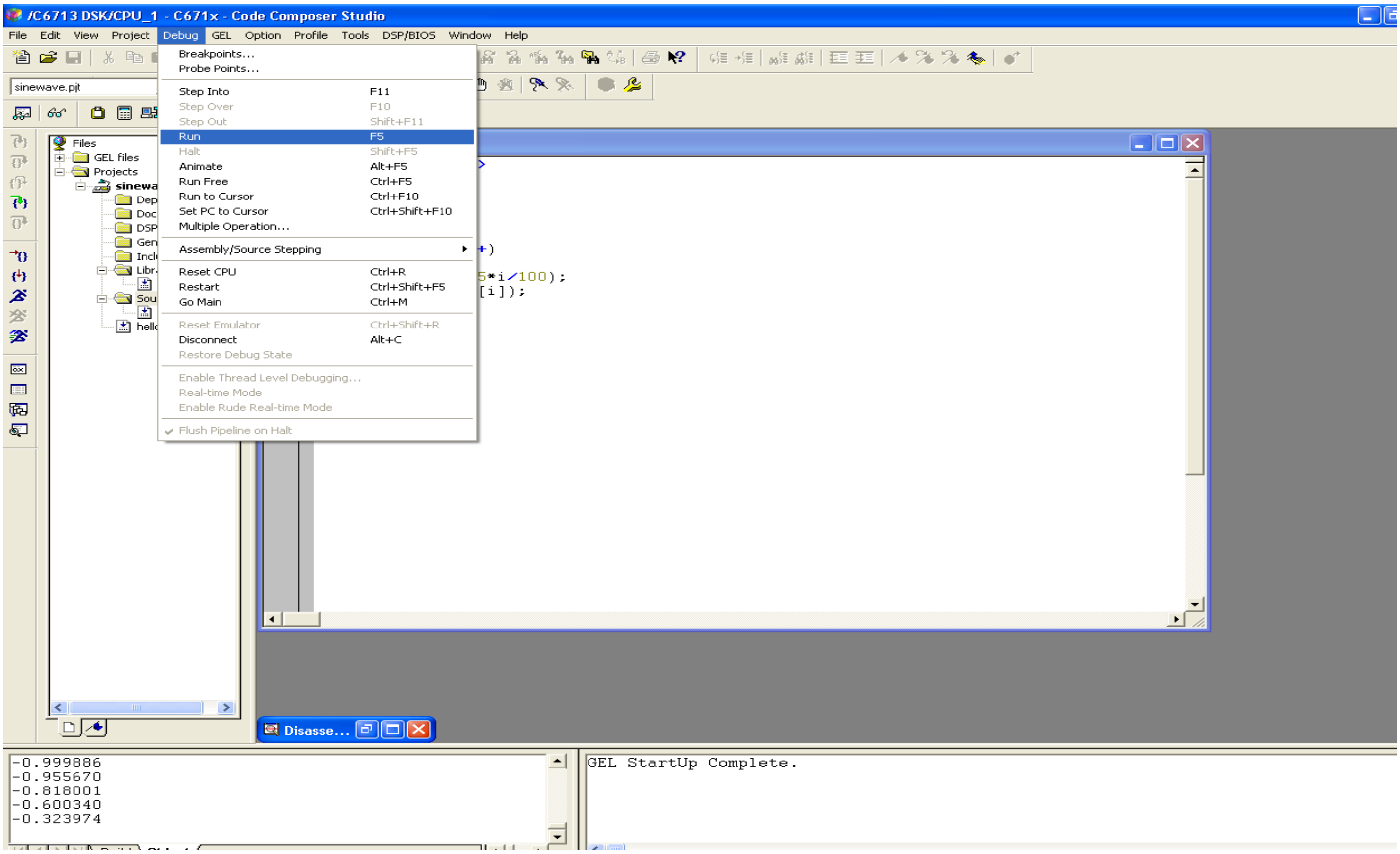
Step 9: Project → Build all



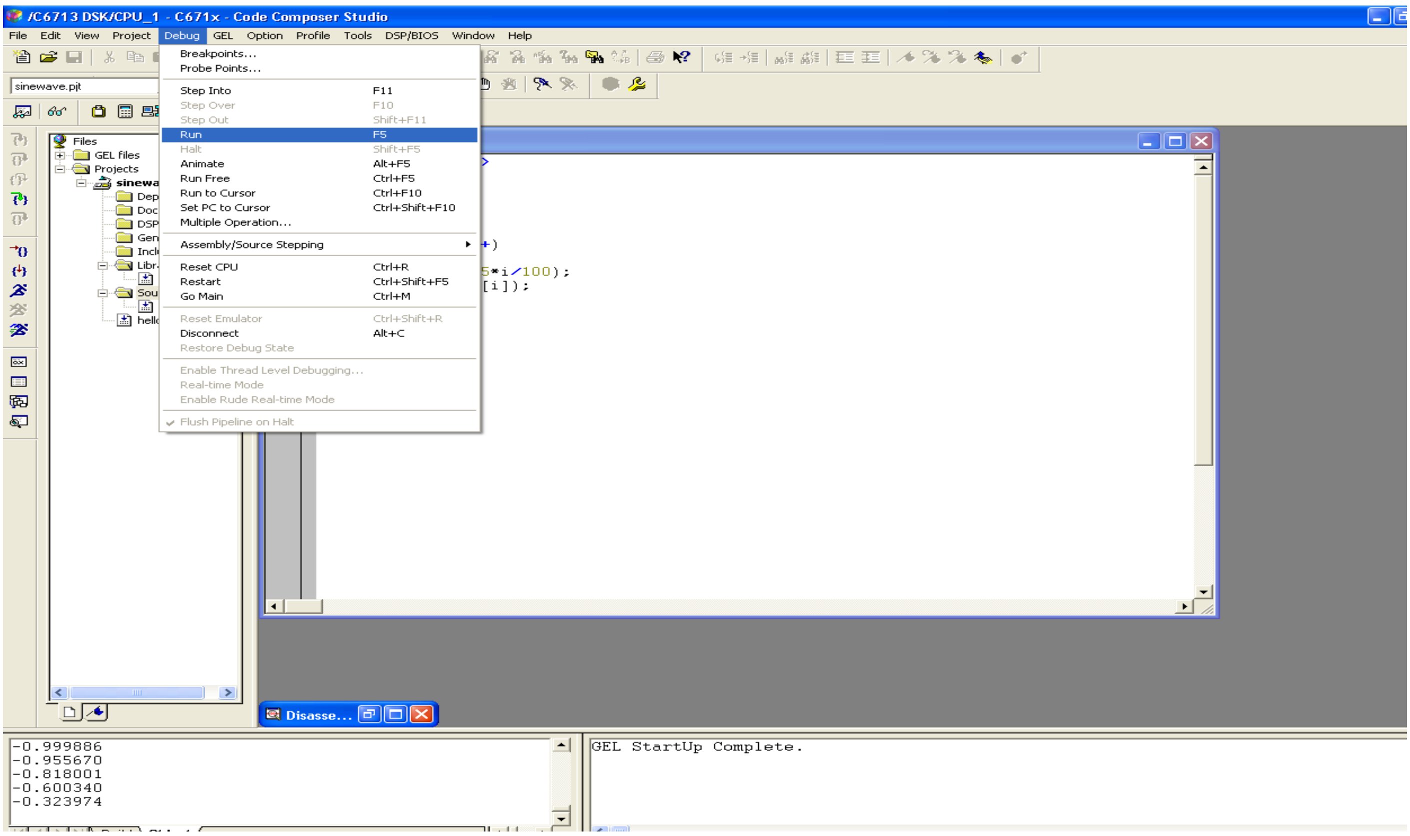
Step 10: File → Load Program → Debug → Output File



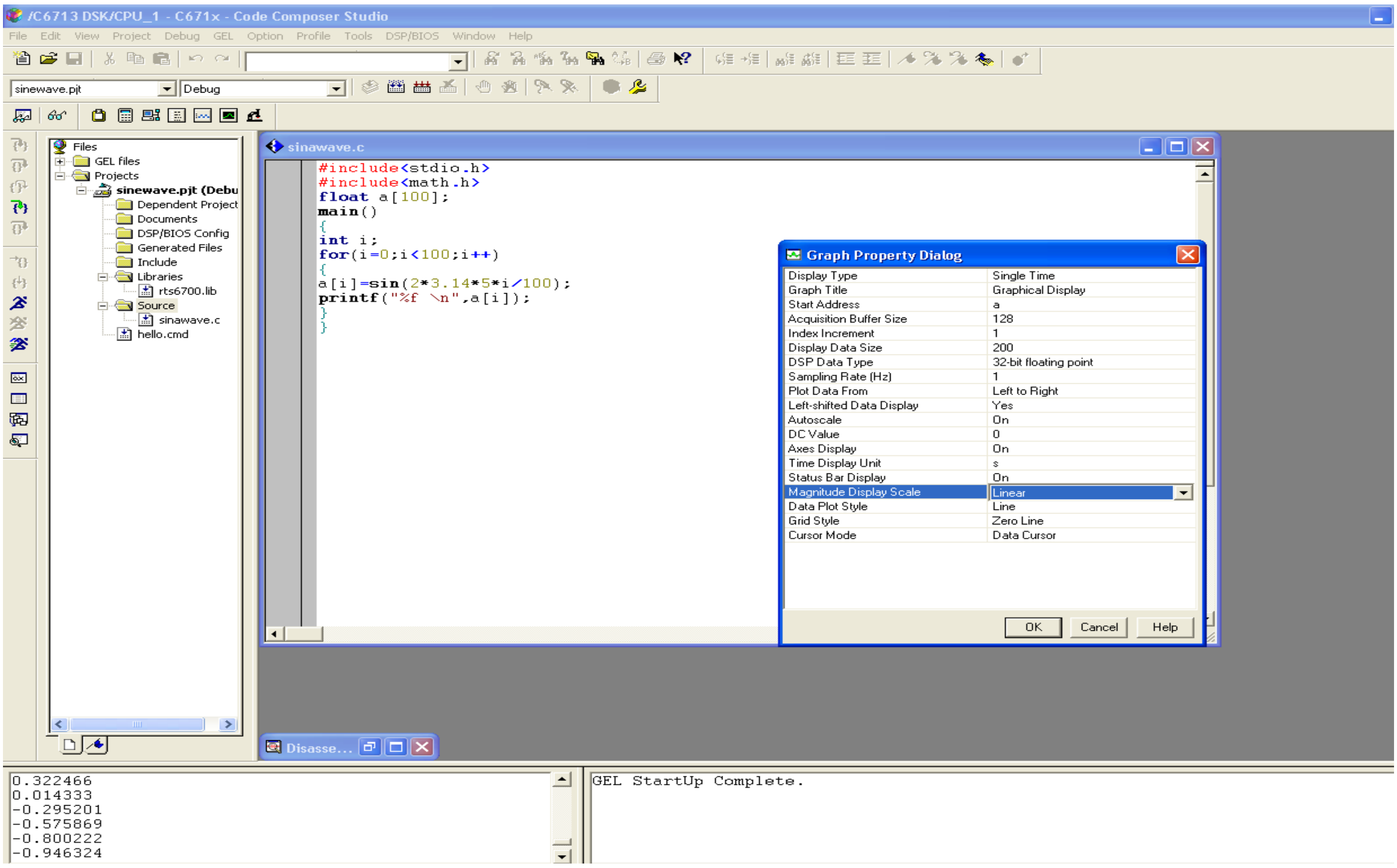
Step 11: Debug → Run



Step 12: Veiw → Graph → Time/Frequency



Step 13: Start Address 'a' and DSP data type- 32 bit floating point



Step 14: We will get th required output ………..!!!!!

TMS320C6748 LCDK

Kit Contents:

- C6748 LCDK Hardware Board
- Power Card
- XDS 100v2 JTAG Emulator
- USB Cable
- SD Card
- User Manual

Description:

The TMS320C6748 DSP development kit (LCDK) is a scalable platform that breaks down development barriers for applications that require embedded analytics and real-time signal processing, including biometric analytics, communications and audio. The low-cost LCDK will also speed and ease your hardware development of real-time DSP applications. This new board reduces design work with freely downloadable and duplicable board schematics and design files. A wide variety of standard interfaces for connectivity and storage enable you to easily bring audio, video and other signals onto the board.

The LCDK does not have an onboard emulator. An external emulator from TI (such as the XDS100, XDS200,XDS510, XDS560) or a third-party will be required to start development.

The TMS320C6748 C6000 DSP processor is a low-power applications processor based on C674x DSP core. This processor provides significantly lower power than other members of the TMS320C6000™ platform of DSPs

The architecture of the device provides benefits of both DSP and reduced instruction set computer (RISC) technologies, incorporating a high-performance TMS320C674x DSP core.

The device DSP core uses a 2-level cache-based architecture. The level 1 program cache (L1P) is a 32- KB direct mapped cache, and the level 1 data cache (L1D) is a 32-KB 2-way, set-associative cache. The level 2 program cache (L2P) consists of a 256-KB memory space that is shared between program and data space. L2 memory can be configured as mapped memory, cache, or combinations of the two..

Processor

 TI TMS320C6748 DSP Application Processor

 456-MHz C674x Fixed/Floating Point DSP

 On-Chip RTC

Memory

 128 MByte DDR2 SDRAM running at 150MHz

 128 MByte 16-bit wide NAND FLASH

 1 Micro SD/MMC Slot

Interfaces

 One mini-USB Serial Port (on-board serial to USB)

 One Fast Ethernet Port (10/100 Mbps) with status LEDs

 One USB Host port (USB 1.1)

 One SATA Port (3Gbps)

 One LCD Port (Beagleboard XM connectors)

 One Leopard Imaging Camera Sensor Input (36-pin ZIP connector)

 Three AUDIO Ports (1 LINE IN-J55 & 1 LINE OUT-J56 & 1 MIC IN-J57)

 14-pin JTAG header (No onboard emulator; external emulator is required)

TMS320C6748 DSP Features

 Highest-Performance Floating-Point Digital Signal Processor (DSP):

- Eight 32-Bit Instructions/Cycle
- 32/64-Bit Data Word
- 375/456-MHz C674x Fixed/Floating-Point
- Up to 3648/2746 C674x MIPS/MFLOPS
- Rich Peripheral Set, Optimized for Audio
- Highly Optimized C/C++ Compiler
- Extended Temperature Devices Available

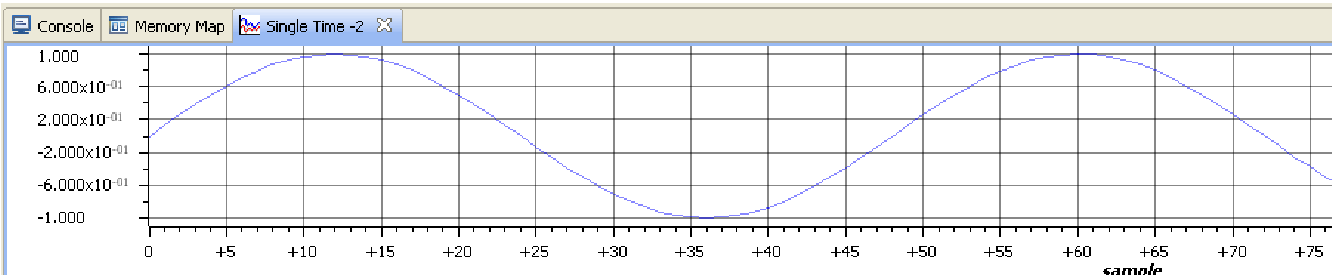
1) GENARATION OF RANDOM AND SQUARE SIGNAL

1.Program --

```
#include<stdio.h>
#include<math.h>
#define PI 3.14
#define PTS 128
float x[PTS];
float y[PTS];
float z[PTS];
float n[PTS];
void main()
{
	int i,j;
```

```
        for (i = 0 ; i < PTS ; i++)

        {

                x[i] = sin(2*PI*i*20/128.0);

                printf("%f\n",x[i]);

                y[i]=0.0;

                n[i]=x[i] + rand() * 10 ;

        }

}
```

RESULT:

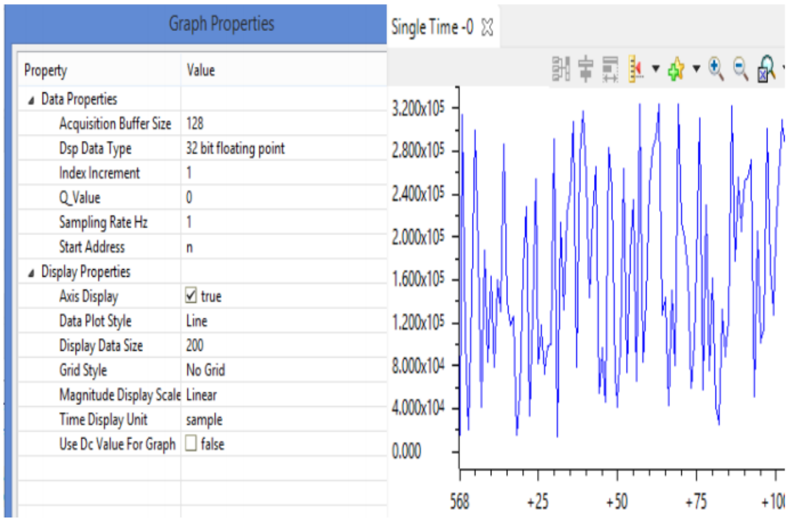


2) Program--

```
        #include <stdio.h>

        #define amp 1  // defined amplitude

        int y[50];    // no. of points to be stored in output variable 'y'

        int i;

        void main()

        {

        for(i=0;i<25;i++)

        {

                        y[i]=amp;

                printf("%d\n",y[i]);

        }

        for(i=25;i<50;i++)
```

```
        {
                y[i]=-amp;
                printf("%d\n",y[i]);
        }
        }
```

RESULT:



2 Finding energy and power of a signal.

1. Program-- Energy of a signal

```
#include<stdio.h>
int main()
{
  int num,i,j,x[32];
  long int sum=0;
  printf("\nEnter the number of samples: ");
  scanf("%d",&num);
  printf("\nEnter samples: ");
  for(j=0;j<num;j++)
     scanf("%d",&x[j]);
  for(i=0;i<=num;i++)
```

```
  {
       sum+=x[i]*x[i];

  }
  printf("\n the energy of above samples is\n %d",sum);
  return 0;
}
```

2. Program--Power of a signal

```
#include<stdio.h>
int main(){
  int num,i,j,x[32];
        float  num1;
  long int sum=0;
  printf("\nEnter the number of samples: ");
  scanf("%d",&num);
  printf("\nEnter samples: ");
  for(j=0;j<num;j++)
     scanf("%d",&x[j]);
  for(i=0;i<=num;i++)
  {
         sum+=x[i]*x[i];

  }
  num=num*2;
  num++;
  num1 = sum / (float) num;
  printf("\n the Average power of above samples is\n %.2f",num1);
  return 0;
}
```

3) Convolution and correlation(auto and cross correlation)of discrete sequence without using built in functions for convolutions and correlation operations.

Program:

PROGRAM: LINEAR.C

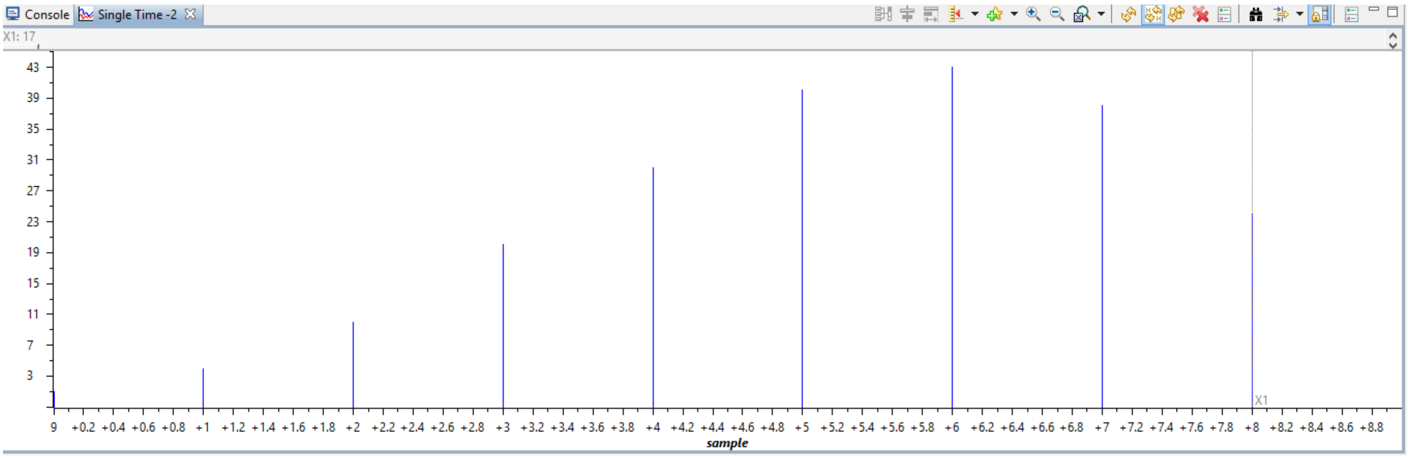
```
#include<stdio.h>
#define LENGHT1 6 /*Lenght of i/p samples sequence*/
#define LENGHT2 4 /*Lenght of impulse response Co-efficients */
int x[2*LENGHT1-1]={1,2,3,4,5,6,0,0,0,0,0}; /*Input Signal Samples*/
int h[2*LENGHT1-1]={1,2,3,4,0,0,0,0,0,0,0}; /*Impulse Response Coefficients*/
int y[LENGHT1+LENGHT2-1];
main()
{
int i=0,j;
for(i=0;i<(LENGHT1+LENGHT2-1);i++)
{
y[i]=0;
for(j=0;j<=i;j++)
y[i]+=x[j]*h[i-j];
}
for(i=0;i<(LENGHT1+LENGHT2-1);i++)
printf("%d\n",y[i]);}
```

INPUT :

x[n] = {1, 2, 3, 4}

h[k] = {1, 2, 3, 4}

OUTPUT : y[r] = {1, 4, 10, 20, 25, 24, 16}



/* program to implement circular convolution */

```
#include<stdio.h>

int m,n,x[30],h[30],y[30],i,j, k,x2[30],a[30];
```

```
void main()

{

printf(" Enter the length of the first sequence\n");

scanf("%d",&m);

printf(" Enter the length of the second sequence\n");

scanf("%d",&n);

printf(" Enter the first sequence\n");

for(i=0;i<m;i++)

scanf("%d",&x[i]);

printf(" Enter the second sequence\n");

for(j=0;j<n;j++)

scanf("%d",&h[j]);

if(m-n!=0) /*If length of both sequences are not equal*/

{

if(m>n) /* Pad the smaller sequence with zero*/

{

for(i=n;i<m;i++)

h[i]=0;

n=m;

}

for(i=m;i<n;i++)

x[i]=0;

m=n;

}

y[0]=0;

a[0]=h[0];

for(j=1;j<n;j++) /*folding h(n) to h(-n)*/

a[j]=h[n-j];

/*Circular convolution*/

for(i=0;i<n;i++)
```

```
y[0]+=x[i]*a[i];

for(k=1;k<n;k++)

{

y[k]=0;

/*circular shift*/

for(j=1;j<n;j++)

x2[j]=a[j-1];

x2[0]=a[n-1];

for(i=0;i<n;i++)

{

a[i]=x2[i];

y[k]+=x[i]*x2[i];

}

}

printf(" The circular convolution is\n");

for(i=0;i<n;i++)

printf("%d \t",y[i]);

}
```

/*cross correlation*/

```
#include<stdio.h>

#include<math.h>

int y[10];

main()

{i

nt i,j;

int x[15]={1,2,3,4,0,0,0,0,0,0,0,0,0,0,0};

int h[15]={4,3,2,1,0,0,0,0,0,0,0,0,0,0,0};

int n=4;

for(i=-(n-1);i<=(n-1);i++)

{
```

```
y[i]=0;

for(j=0;j<=4;j++)

y[i]+= x[j] * h[j-i];

}f

or(i=-(n-1);i<n;i++)

printf("%d\n",y[i]);

}
```

/*auto correlation*/

```
#include<stdio.h>

#include<math.h>

int y[10];

main()

{

int i,j,k;

int x[15]={0,0,0,0,0,1,1,1,1,1,0,0,0,0,0};

int n=15;

k=n-1;

for(i=-(n-1);i<=(n-1);i++)

{

y[i+k]=0;

for(j=0;j<=5;j++)

y[i+k]+=x[j+k] * x[i+j+k];

}

for(i=-(n-1);i<n;i++)

printf("%d\n",y[i+k]);

}
```

4) DTFT of a given signal

Program--

```
#include<stdio.h>

#include<math.h>

int N,k,n,i;

float pi=3.1416,sumre=0, sumim=0,out_real[8]={0.0}, out_imag[8]={0.0};

int x[32];

void main(void)

{

printf("  enter the length of the sequence\n");

   scanf("%d",&N);

   printf("  enter the sequence\n");

   for(i=0;i<N;i++)

   scanf("%d",&x[i]);

for(k=0;k<N;k++)

{

sumre=0;

sumim=0;

for(n=0;n<N;n++)

{

sumre=sumre+x[n]* cos(2*pi*k*n/N);

sumim=sumim-x[n]* sin(2*pi*k*n/N);

}

out_real[k]=sumre;

out_imag[k]=sumim;

printf("X([%d])=\t%f\t+\t%fi\n",k,out_real[k],out_imag[k]);

}

}
```

5) N – point FFT algorithm

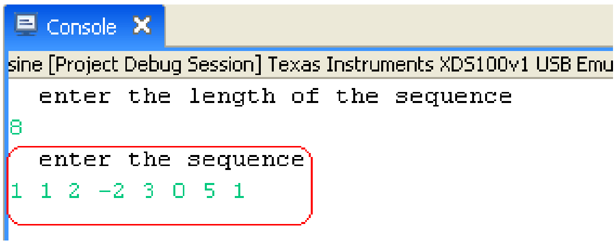
Program--

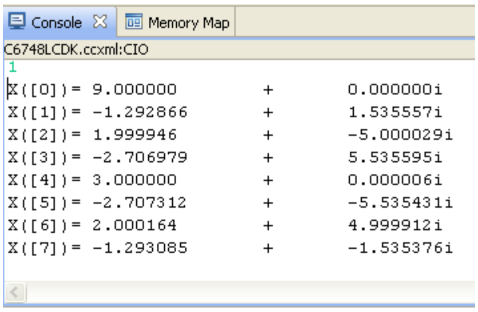
```
#include<stdio.h>

#include<math.h>

#define PI 3.14

typedef struct

        {

                float real,imag;

        }com;

void main()

{

        com xx[8],x[8],temp[8],temp1[8],y[8],a[8],b[8],w[4];

        int i,j=0;//loop counter variables

        printf("\nEnter 8 input samples==");

        for(i=0;i<8;i++)

                scanf("%f",&xx[i].real);

        j=0;

        for(i=0;i<8;i=i+2)

        {

                x[j].real=xx[i].real;

                x[j+1].real=xx[i+4].real;

                if(i==2)

                i=-1;

                j=j+2;

        }

        for(i=0;i<4;i++)

        {

                w[i].real=cos(2*PI*i/8);

                w[i].imag=-sin(2*PI*i/8);

        }

        for(i=0;i<8;i=i+2)
```

```
{
        temp[i].real=x[i].real+x[i+1].real;
        temp[i].imag=x[i].imag+x[i+1].imag;
        temp[i+1].real=x[i].real-x[i+1].real;
        temp[i+1].imag=x[i].imag-x[i+1].imag;
}
for(i=2;i<8;i=3*i)
{
        a[i].real=temp[i].real*w[0].real-temp[i].imag*w[0].imag;
        a[i].imag=temp[i].real*w[0].imag+temp[i].imag*w[0].real;
        a[i+1].real=temp[i+1].real*w[2].real-temp[i+1].imag*w[2].imag;
        a[i+1].imag=temp[i+1].real*w[2].imag+temp[i+1].imag*w[2].real;
        temp[i].real=a[i].real;
        temp[i].imag=a[i].imag;
        temp[i+1].real=a[i+1].real;
        temp[i+1].imag=a[i+1].imag;
}
for(i=0;i<6;i++)
{
        temp1[i].real=temp[i].real+temp[i+2].real;
        temp1[i].imag=temp[i].imag+temp[i+2].imag;
        temp1[i+2].real=temp[i].real-temp[i+2].real;
        temp1[i+2].imag=temp[i].imag-temp[i+2].imag;
        if(i==1)
        i=3;
}
for(i=4;i<8;i++)
{
        b[i].real=temp1[i].real*w[i-4].real-temp1[i].imag*w[i-4].imag;
```

```
            b[i].imag=temp1[i].real*w[i-4].imag+temp1[i].imag*w[i-4].real;

            temp1[i].real=b[i].real;

            temp1[i].imag=b[i].imag;

        }

        for(i=0;i<4;i++)

        {

            y[i].real=temp1[i].real+temp1[i+4].real;

            y[i].imag=temp1[i].imag+temp1[i+4].imag;

            y[i+4].real=temp1[i].real-temp1[i+4].real;

            y[i+4].imag=temp1[i].imag-temp1[i+4].imag;

        }

        printf("\nDFT values==\n");

         for(i=0;i<8;i++)

            printf("\nF(%d)=(%0.1f)+j(%0.1f)\n",i,y[i].real,y[i].imag);

}
```

RESULT :



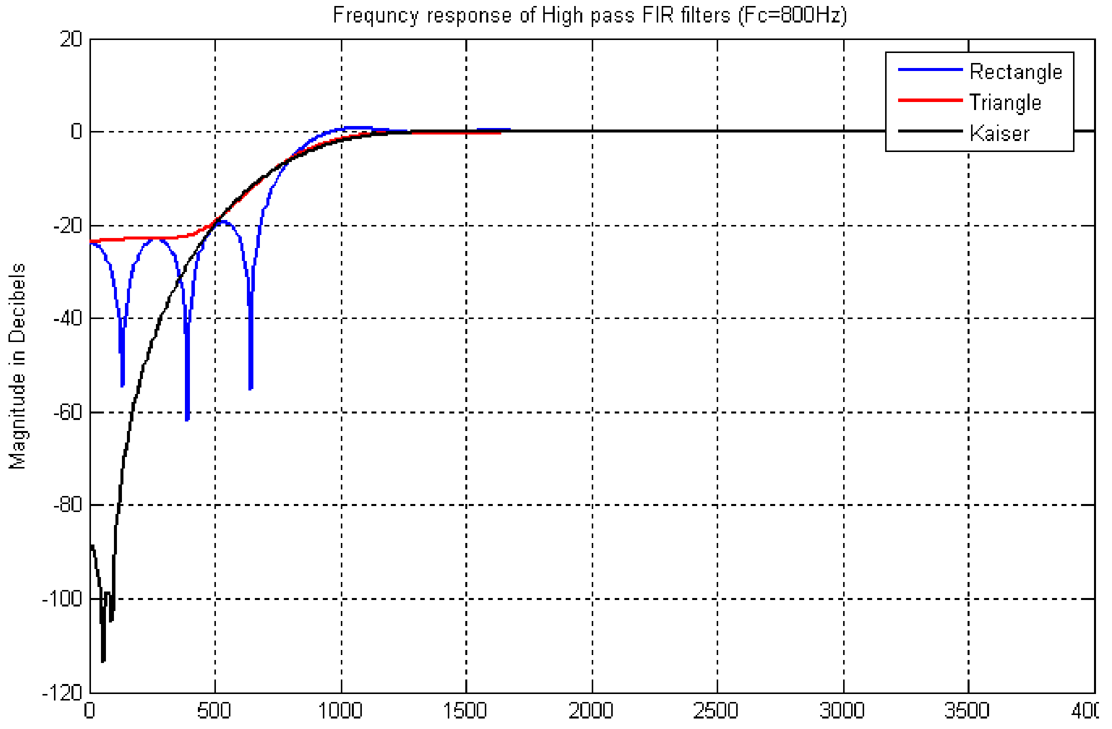


6) Design of FIR filter using windowing technique and verify the frequency response of the filter

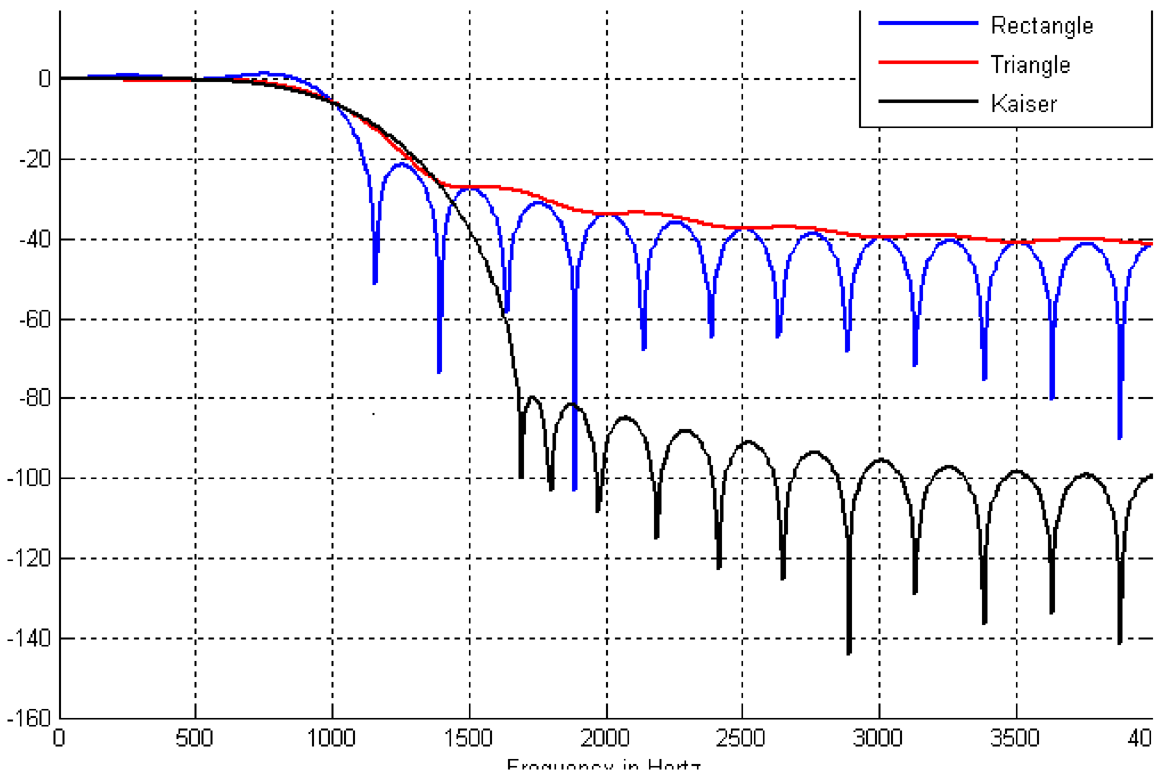
Program:

```
#include<stdio.h>

#include<math.h>

#define pi 3.1415

int n,N,c;

float wr[64],wt[64];

void main()

{ printf("\n enter no. of samples,N= :"); scanf("%d",&N);

printf("\n enter choice of window function\n 1.rect \n 2. triang \n c= :"); scanf("%d",&c);

printf("\n elements of window function are:");

switch(c)

{

case 1:

for(n=0;n<=N-1;n++)

{

wr[n]=1;

printf(" \n wr[%d]=%f",n,wr[n]);

}

break;

case 2:

for(n=0;n<=N-1;n++)

{

wt[n]=1-(2*(float)n/(N-1));

printf("\n wt[%d]=%f",n,wt[n]);

}

break;

}}
```

High Pass FIR filter(Fc= 800Hz).



Low Pass FIR filter (Fc=1000Hz)



7) **Design of IIR filter using any of the available methods and verify the frequency response of the filter**

Program:

```
//IIRFILTERS USING C
#include<stdio.h>
#include<math.h>
int i,w,wc,c,N;
float H[100];
float mul(float, int);
void main()
{
printf("\n enter order of filter ");
scanf("%d",&N);
printf("\n enter the cutoff freq ");
scanf("%d",&wc);
printf("\n enter the choice for IIR filter 1. LPF 2.HPF ");
scanf("%d",&c);
switch(c)
{
case 1:
for(w=0;w<100;w++)
{
H[w]=1/sqrt(1+mul((w/(float)wc),2*N));
printf("H[%d]=%f\n",w,H[w]);
}
break;
case 2:
for(w=0;w<=100;w++)
{
```

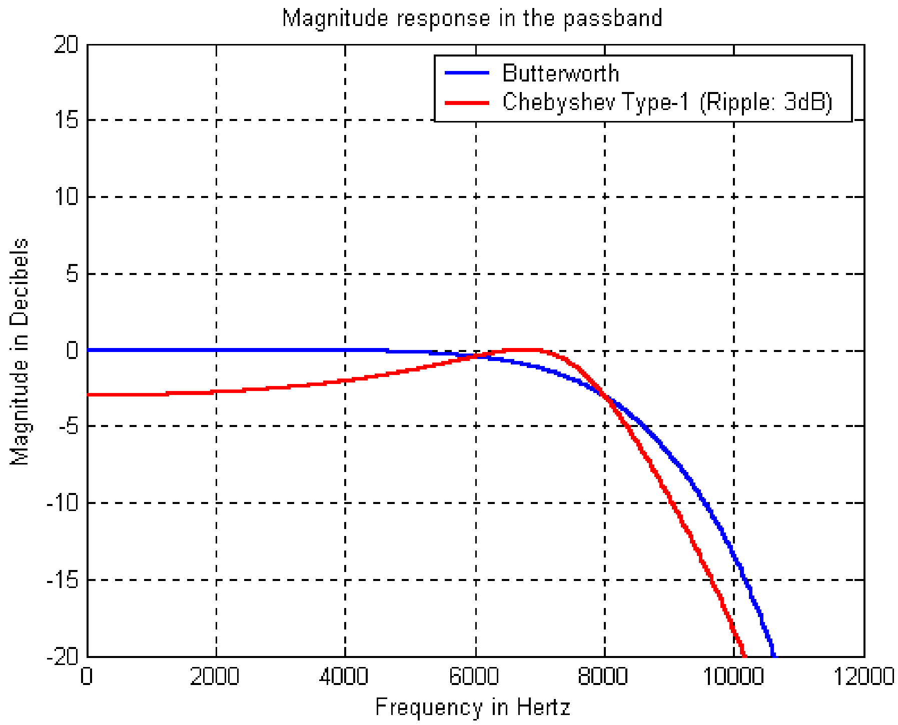
```
H[w]=1/sqrt(1+mul((float)wc/w,2*N));

printf("H[%d]=%f\n",w,H[w]);

}

break;

}}

float mul(float a,int x)

{

for(i=0;i<x-1;i++)

a*=a;

return(a);

}
```

8) Design of Analog filters

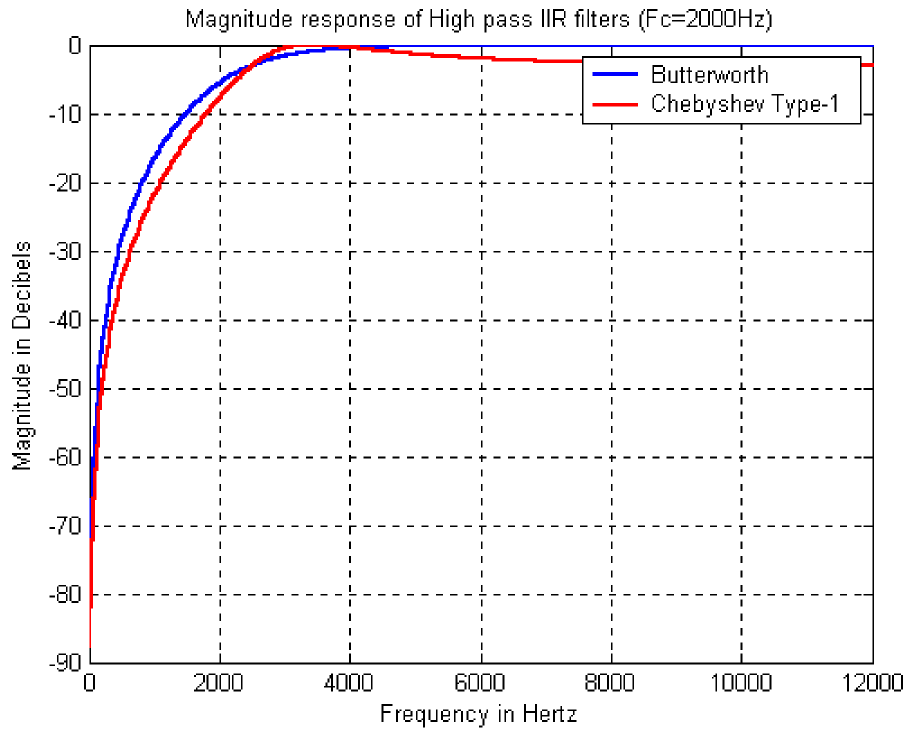
Program:

```
// IIR filter implemented using second order sections
// integer coefficients read from file
#include "L138_LCDK_aic3106_init.h"
#include "elliptic.cof"
int w[NUM_SECTIONS][2] = {0};
interrupt void interrupt4()    //interrupt service routine
{
  int section;    // index for section number
  int input;      // input to each section
  int wn,yn;        // intermediate and output values in each stage
  input = input_left_sample();

//  input = (int)prbs();
  for (section=0 ; section< NUM_SECTIONS ; section++)
  {
//
    wn = input - ((a[section][1]*w[section][0])>>15) -
((a[section][2]*w[section][1])>>15);
//
    yn = ((b[section][0]*wn)>>15) +
((b[section][1]*w[section][0])>>15) +
((b[section][2]*w[section][1])>>15);
    w[section][1] = w[section][0];
    w[section][0] = wn;
    input = yn;              // output of current section will be
input to next
  }
  output_left_sample((int16_t)(yn)); // before writing to codec
  return;                          //return from ISR
}

int main(void)
{

L138_initialise_intr(FS_48000_HZ,ADC_GAIN_0DB,DAC_ATTEN_0DB,LCDK_L
INE_INPUT);
  while(1);
}  // end of main()
```

RESULT :



Magnitude response in the passband
Butterworth
Chebyshev Type-1 (Ripple: 3dB)
Magnitude in Decibels
Frequency in Hertz
20
15
10
5
0
-5
-10
-15
-20
0
2000
4000
6000
8000
10000
12000



Magnitude response of High pass IIR filters (Fc=2000Hz)
Butterworth
Chebyshev Type-1
Magnitude in Decibels
Frequency in Hertz
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
0
2000
4000
6000
8000
10000
12000