INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad - 500 043

Lab Manual:

Digital Signal Processing Laboratory (AECB25)

Prepared by

Dr.S.China Venkateswarlu (IARE10624)

Department Of Electronics and Communication Engineering
Institute of Aeronautical Engineering

April 4, 2022

Contents

INTRODUCTION

1.1 Introduction

This course is intended to enhance the learning experience of the student in topics encountered in AECB23. In this lab, students are expected to gain experience in programming the real-time processing algorithms used in digital signal processing applications. The DSP lab consists of a number of software and hardware experiments illustrating the concepts of digital signal processing using MATLAB software and Texas instruments TMS320C6713 DSP processor. The programming of the DSP chip is done in C language using the code composer studio (CCS) software. How the student performs in the lab depends on his/her preparation, participation, and teamwork. Each team member must participate in all aspects of the lab to insure a thorough understanding of the equipment and concepts. The student, lab assistant, and faculty coordinator all have certain responsibilities toward successful completion of the lab's goals and objectives.

1.1.1 Student Responsibilities

The student is expected tocome prepared for each lab.Lab preparation includes understanding the labexperiment from the lab manual and reading the related textbook material.

Students have to write the allotted experiment for that particular week in the work sheets given and carry them to the Lab. In case of any questions or problems with the preparation, students can contact the Faculty Teaching the Lab course, but in a timely manner.

Students have to be in formal dress code, wear shoes and lab coat for the Laboratory Class.

After the demonstration of experiment by the faculty, student has to perform the experiment individually. They have to note down the observations in the observation Tables drawn in work sheets, do the calculations and analyze the results.

Active participation by each student in lab activities is expected. The student is expected to ask the Faculty any questions they may have related to the experiment.

The student should remain alert and use commonsense while performing the lab experiment.They are also responsible for keeping a professional and accurate record of the labexperiments in the files provided.

1.1.2 Responsibilities of Faculty Teaching the Lab Course

The Faculty shall be completely familiar with each labprior to the laboratory. He/She shall provide the students with details regarding the syllabus and safety review during the first week.Lab experiments should be checked in advance to make sure that everything is in working order.The Faculty should demonstrate and explain the experiment and answer any questions posed by the students.Faculty have to supervise the students while they perform the lab experiments. The Faculty is expected to evaluate the lab worksheets and grade them based on their practical skills and understanding of the experiment by taking Viva Voce. Evaluation of work sheets has to be done in a fair and timely manner to enable the students, for uploading them online through their CMS login within the stipulated time.

1.1.3 Laboratory In-charge Responsibilities

The Laboratory In-charge should ensure that the laboratory is properly equipped, i.e., the Faculty teaching the lab receive any equipment/components necessary to perform the experiments.He/She is responsible for ensuring that all the necessary equipment for the lab is available and in working condition. The Laboratory In-charge is responsible for resolving any problems that are identified by the teaching Faculty or the students.

1.1.4 Course Coordinator Responsibilities

The course coordinator is responsible fo rmaking any necessary corrections in Course Description and lab manual. He/She has to ensure that it is continually updated and available to the students in the CMS learning Portal.

1.2 Lab Policy and Grading

The student should understand the following policy:

ATTENDANCE: Attendance is mandatory as per the academic regulations.

LAB RECORD's: The student must:

1. Write the work sheets for the allotted experiment and keep them ready before the beginning of eachlab.
2. Keep all work in preparation of and obtained during lab.
3. Perform the experiment and record the observations in the worksheets.
4. Analyze the resultsand get the work sheets evaluated by the Faculty.
5. Upload the evaluated reports online from CMS LOGIN within the stipulated time.

Grading Policy:

The final grade of this course is awarded using the criterion detailed in the academic regulations. A large portion of the student's grade is determined in the comprehensive final exam of the Laboratory course (SEE PRACTICALS),resulting in a requirement of understanding the concepts and procedure of each lab experiment for successful completion of the lab course.

Pre-Requistes and Co-Requisties:

The lab course is to be taken during the same semester as AECB23, but receives a separate grade. Students are required to have completed both AECB23 and AECB25 with minimum passing grade or better grade in each.

1.3 Course Goals and Objectives

The Digital signal processing Laboratory is designed to provide the student with the knowledge to use basic concepts of signal processing algorithms and techniques with proficiency. These techniques are designed to complement the concepts introduced in AECB23. In addition, the student should learn how to record experimental results effectively and present these results in a written report.

More explicitly, the class objectives are:

1. To gain proficiency in the use of signal processing algorithms.
2. To enhance understanding of theoretical concepts including:
   - Linear and circular convolution
   - DFT and IDFT
   - Overlap save and overlap add methods
   - DIT-FFT and DIF-FFT
   - IIR Filters
   - FIR Filters
   - DTMF detection
   - Multi-rate signal processing
   - Sine wave generation
3. To develop communication skills through:
   - Verbal interchanges with the Faculty and other students.
   - Preparation of succinct but complete laboratory reports.
   - Maintenance of laboratory worksheets aspermanent, written descriptions of procedures, analysis and results.
4. To compare theoretical predictions with experimental results and to determine the source of any apparent errors.

1.4 Use of Laboratory Instruments

One of the major goals of this lab is to familiarize the student with the proper equipment and techniques for conducting experiments. Some understanding of the lab instruments is necessary to avoid personal or equipment damage.By understanding the device's purpose and following a few simple rules, costly mistakes can be avoided.

The following rules provide a guideline for instrument protection.

1.4.1 Instrument Protection Rules

1. Do not upload, delete or alter any software on the lab PC.
2. Students are not allowed to touch any equipments or other materials in the laboratory area until you are instructed by Teacher or Technician.
3. Before use equipment must be read carefully Labels and instructions. Set up and use the equipment as directed by your teacher.
4. Any failure / break-down of equipment must be reported to the teacher .

1.5 Data Recording and Reports

Students must record their experimental values in the provided tables in this laboratory manual and reproduce them in the lab reports. Reports are integral to recording the methodology and results of an experiment. In engineering practice, the laboratory notebook serves as an invaluable reference to the technique used in the lab and is essential when trying to duplicate a result or write a report. Therefore, it is important to learn to keep accurate data. Make plots of data and sketches when these are appropriate in the recording and analysis of observations. Note that the data collected will be an accurate and permanent record of the data obtained during the experiment and the analysis of the results. You will need this record when you are ready to prepare a lab report.

1.5.1 The Laboratory Worksheets

Students must record their experimental values in the provided tables in this laboratory manual and reproduce them in the lab worksheets. Worksheets are integral to recording the methodology and results of an experiment. Make plots of data and sketches when these are appropriate in the recording and analysis of observations. Note that the data collected will be an accurate and permanent record of the data obtained during the experiment and the analysis of the results.

1.5.2 The Laboratory Files/Reports

Reports are the primary means of communicating your experience and conclusions to other professionals. In this course you will use the lab report to inform your teaching faculty of Lab about what you did and what you have learned from the experience. Engineering results are meaningless unless they can be communicated to others. You will be directed by your teaching faculty of Lab to prepare a lab report on a few selected lab experiments during the semester.

1.5.3 Order of the Lab worksheet

1. COVER PAGE - Cover page must include lab name and number, your name, and the date the lab was performed.
2. Objective: Clearly state the experiment objective in your own words.
3. Prelab: Indicate the required knowledge for performing the concerned experiment
4. Equipment used: Indicate which equipment was used in performing the experiment.
5. Background: Give the brief information about corresponding lab experiment
6. Procedure: Describe the experimental set up and procedure to perform the experiment
7. Result: After completion of experiment, provide a summary of what was learned from this part of the laboratory experiment.

LAB-1 ORIENTATION

2.1 Introduction

In the first lab period, the students should become familiar with the location of equipment and components in the lab, the course requirements, and the teaching instructor. Students should also make sure that they have all of the co-requisites and pre-requisites for the course at this time.

2.2 Objective

To familiarize the students with the lab facilities, equipment, standard operating procedures, lab safety, and the course requirements.

2.3 Prelab Preparation:

Read the Introduction and Appendix of this manual. Download and install the "MATLAB" software on your personal computer, available here.

2.4 Equipment needed

AECB25 lab manual.

2.5 Procedure

1. During the first laboratory period, the instructor will provide the students with a general idea of what is expected from them in this course. Each student will receive a copy of the syllabus, stating the instructor's contact information. In addition, the instructor will review the safety concepts of the course.

2. During this period, the instructor will briefly review the equipment which will be used throughout the semester. The location of instruments, equipment, and components will be indicated. The guidelines for instrument use will be reviewed.

2.6 Further Probing Experiments

Additional experiments pertaining to this lab can be done by the student to enhance their learning experience.

LAB-2 LINEAR CONVOLUTION VS. CIRCULAR CONVOLUTION

3.1 Introduction

Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal Processing. Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response. Linear convolution is the basic operation to calculate the output for any linear time invariant system given its input and its impulse response. Circular convolution is the same thing but considering that the support of the signal is periodic. Circular convolution gives an output of length same as the length of input signal. But linear convolution gives output of larger length.

3.2 Objective

3.2.1 Educational

1. Determine the response of a linear time invariant system by using linear and circular convolution.
2. Understand the difference between linear convolution and a circular convolution.

3.2.2 Experimental

1. Compute linear and circular convolution of two signals using inbuilt function in MATLAB.
2. Compute linear and circular convolution of two signals using mathematical computation in MATLAB.
3. Determine the input signal to a system with given impulse response h(n) and output signal y(n).

3.3 Prelab Preparation:

Read Appendix B and Appendix C of this manual, paying particular attention to the methods of using computer. Prior to coming to lab class, understand the concepts of convolution methods.

3.4 Equipment needed

1. Personal computer
2. MATLAB software

3.5 Background

3.5.1 Linear convolution

Linear convolution is a mathematical operation done to calculate the output of any Linear-Time Invariant (LTI) system given its input and impulse response. The linear convolution of the two sequences is given by

$$y(n) = x(n) * h(n) = \textstyle\sum_{k=-\infty}^{\infty} \; x[k]h[n-k]$$

If x(n) is a sequence of L number of samples and h(n) with M number of samples, after convolution y(n) will have N=L+M-1 samples. It can be used to find the response of a linear filter.Zero padding is not necessary to find the response of a linear filter.

3.5.2 Circular convolution

Circular convolution, also known as cyclic convolution, is a special case of periodic convolution, which is the convolution of two periodic functions that have the same period. Periodic convolution arises, for example, in the context of the discrete-time Fourier transform (DTFT). The circular convolution of the two sequences is given by

$$y(n) = x(n) \circledast h(n) = \textstyle\sum_{k=-\infty}^{\infty} \; x[k]h[n-k] moduloN$$

If x(n) is a sequence of L number of samples and h(n) with M samples, after convolution y(n) will have N=max(L,M) samples .It cannot be used to find the response of a filter.Zero padding is necessary to find the response of a filter.

3.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

3.7 Program

1. a)Generation of linear convolution without using built in function and the function convolution in MATLAB

```
clc;
close all;
clear all;
x=input('Enter x: ');
h=input('Enter h: ');
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
```

```
H=[h,zeros(1,m)];
for i=1:n+m-1
Y(i)=0;
for j=1:m
if(i-j+1¿0)
Y(i)=Y(i)+X(j)*H(i-j+1);
else
end
end
end
Y
stem(Y);
ylabel('Y[n]');
xlabel('—–¿n');
title(Linear 'Convolution of Two Signals without conv function');
```

(b) Generation of circular convolution without using built in function in MATLAB

```
clc;
close all;
x=input('Enter x(n):');
h=input('Enter h(n):');
m=length(x);
n=length(h);
N=max(m,n);
x=[x,zeros(1,N-m)];
h=[h,zeros(1,N-n)];
for n=1:N
Y(n)=0;
for i=1:N
j=n-i+1;
if(j¡=0)
j=N+j;
end
Y(n)=[Y(n)+x(i)*h(j)];
end
end
n=0:N-1;
subplot(311);
disp('First Sequence x(n) is:');
disp(x);
stem(n,x);
xlabel('n');
ylabel('x(n)');
title('First Sequence');
grid on;
subplot(312);
disp('Second Sequence h(n) is:');
disp(h);
stem(n,h);
xlabel('n');
ylabel('h(n)');
```

```
title('Second Sequence');
grid on;
subplot(313);
disp('Convoluted Sequence Y(n) is:')'
disp(Y);
stem(n,Y);
xlabel('n');
ylabel('Y(n)');
title('Circular Convoluted Sequence');
grid on;
```

3.8 Output and waveforms

1(a)linear convolution

```
Enter x: [1 -3 5 8]
    x =
    1 -3 5 8
    Enter h: [6 9 0 2]
    h =
    6 9 0 2
    Y =
    6 -9 3 95 66 10 16
    Using built in function
```

```
>>x=[1 -3 5 8]
    x =
    1 -3 5 8
    >>h=[6 9 0 2]
    h =
    6 9 0 2
    >>conv(x,h)
    ans =
    6 -9 3 95 66 10 16
```

1(b)Circular convolution

```
    Enter x(n): [1 -3 5 8]
Enter h(n): [6 9 0 2]
First Sequence x(n) is: 1 -3 5 8
    Second Sequence h(n) is: 6 9 0 2
    Convoluted Sequence Y(n) is: 72 1 19 95
```

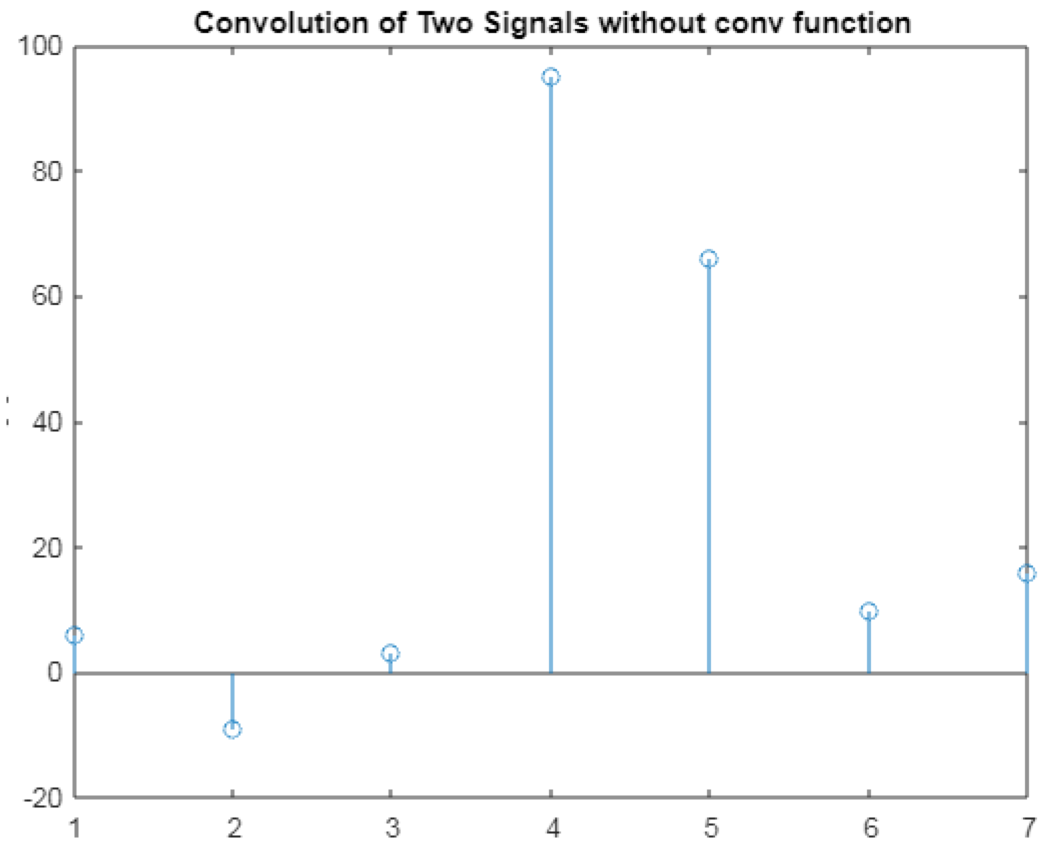


Figure 3.1: Linear convolutoin plot

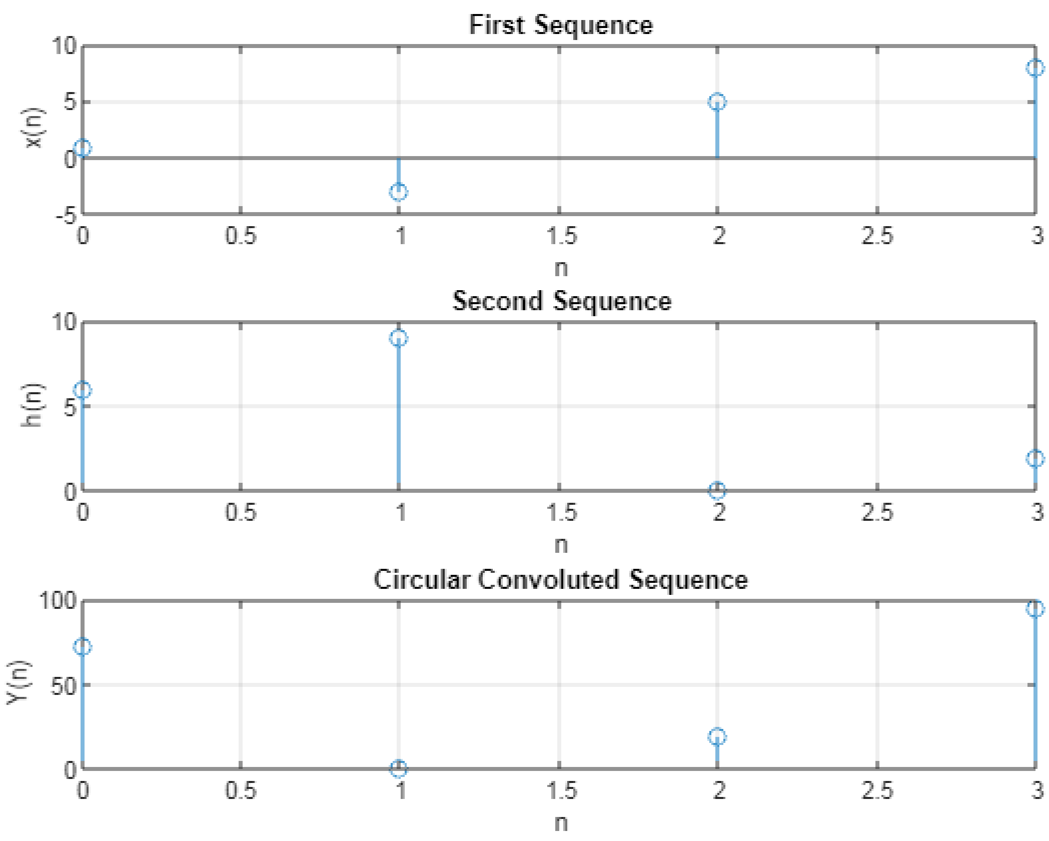


Figure 3.2: Circular convolutoin plot

3.9 Further Probing Experiments

Q1. Determine the unit step response of the linear time invariant system with impulse response h(n)=a^nu(n).

Q2. The impulse response of LTI system is h(n)=[1 2 1-1]Determine the response of the system if input is x[n]=[1 2 3 1].

LAB-3 DFT AND IDFT

4.1 Introduction

The discrete Fourier transform (DFT) and its inverse (IDFT) are the primary numerical transforms relating time and frequency in digital signal processing. The DFT is also used to efficiently solve partial differential equations, and to perform other operations such as convolutions or multiplying large integers. Since it deals with a finite amount of data, it can be implemented in computers by numerical algorithms or even dedicated hardware. The DFT is one of the most powerful tools in digital signal processing which enables us to find the spectrum of a finite-duration signal. There are many circumstances in which we need to determine the frequency content of a time-domain signal.

4.2 Objective

4.2.1 Educational

1. Learn to convert a discrete signal from time domain to frequency domain.
2. Learn to convert a discrete signal from frequency domain to time domain.

4.2.2 Experimental

1. Confirm how the discrete signal has changed from one domain to another domain.
2. Determine the N-DFT of a discrete time signal using the formula.
3. Draw magnitude and phase spectrum for a discrete signal x[n].
4. Determine the IDFT of a discrete signal X[k] using the formula.

4.3 Prelab Preparation:

1. Read and study the Background section of this Laboratory.
2. Read Appendix B, especially the built in function FFT.

4.4 Equipment needed

1. Personal computer
2. MATLAB software

4.5 Background

When a signal is discrete and periodic, we don't need the continuous Fourier transform. Instead we use the discrete Fourier transform, or DFT. The DFT is one of the most powerful tools in digital signal processing which enables us to find the spectrum of a finite-duration signal. There are many circumstances in which we need to determine the frequency content of a time-domain signal. The Discrete Fourier Transform (DFT) of a sequence x[n] is calculated by using the formula,

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}$$

N represents the sequence length and it is calculated by using the command 'length'. The sequence x(n) can be calculated from X(k) using the Inverse Discrete Fourier Transform (IDFT):

$$x[n] = 1/N \sum_{k=0}^{N-1} X[k]e^{j2\pi nk/N}$$

4.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

4.7 Program

```
clc;
close all;
clear all;
xn=input('Enter the sequence x(n)'); %Get the sequence from user
ln=length(xn); %find the length of the sequence
xk=zeros(1,ln); %initilise an array of same size as that of input sequence
ixk=zeros(1,ln); %initilise an array of same size as that of input sequence

   %code block to find the DFT of the sequence
   for k=0:ln-1
for n=0:ln-1
xk(k+1)=xk(k+1)+(xn(n+1)*exp((-i)*2*pi*k*n/ln));
end
end
disp('DFT of x(n) is');
disp(xk);
%code block to plot the input sequence
t=0:ln-1;
subplot(311);
stem(t,xn);
```

```
ylabel ('Amplitude');
xlabel ('Time Index');
title('Input Sequence');
magnitude=abs(xk); % Find the magnitudes of individual DFT points

    %code block to plot the magnitude response
t=0:ln-1;
subplot(312);
stem(t,magnitude);
ylabel ('Amplitude');
xlabel ('K');
title('Magnitude Response');

    % Code block to find the IDFT of the sequence
for n=0:ln-1
for k=0:ln-1
ixk(n+1)=ixk(n+1)+(xk(k+1)*exp(i*2*pi*k*n/ln));
end
end
ixk=ixk/ln;
disp('IDFT of X(k) is');
disp(ixk);
%code block to plot the input sequence
t=0:ln-1;
subplot(313);
stem(t,ixk);ylabel ('Amplitude');xlabel ('Time Index');title('IDFT sequence');
```

4.8 Output and waveforms

```
Enter the sequence x(n)
    [1 1 0 1]
    DFT of x(n) is
    3.0000 + 0.0000i 1.0000 + 0.0000i -1.0000 - 0.0000i 1.0000 + 0.0000i
    IDFT of X[k] is
    1.0000 - 0.0000i 1.0000 + 0.0000i -0.0000 + 0.0000i 1.0000 + 0.0000i
```

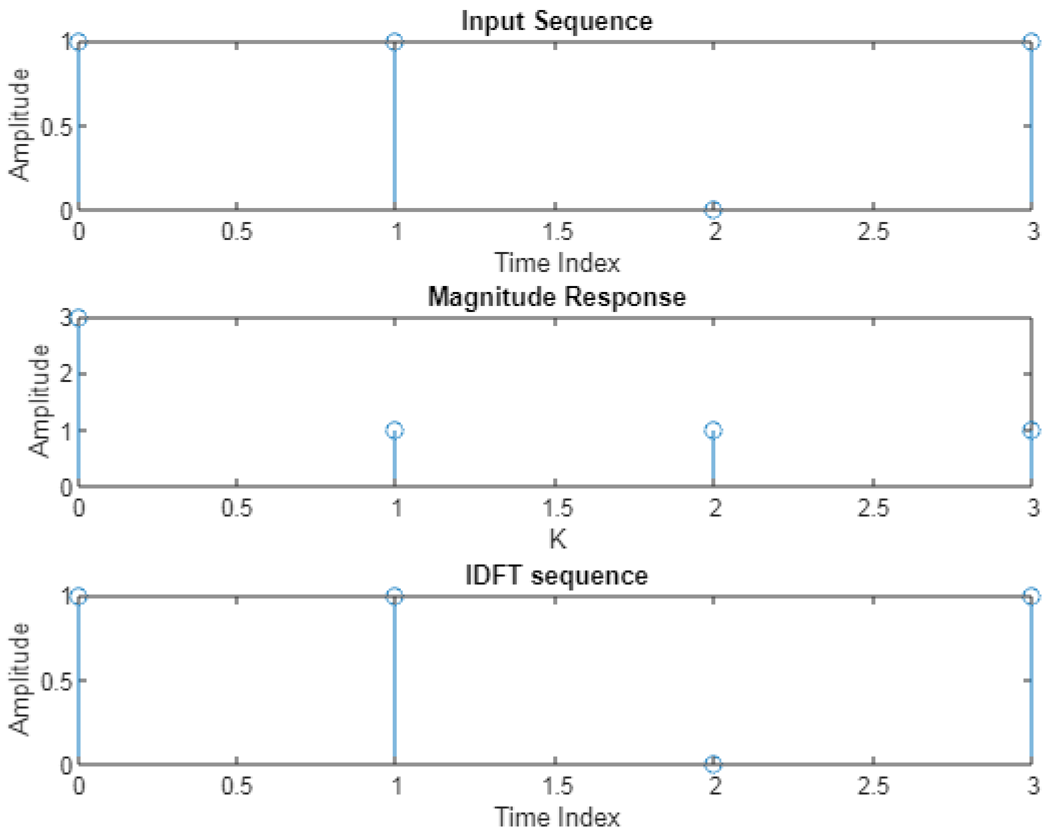


Figure 4.1: DFT and IDFT plot

4.9 Further Probing Experiments

Q1. Let X(K) be the 8 point DFT of the sequence x(n)=A,2,3,4,5,6,7,B. If X(0)=20 and X(4)=0, find A and B .

Q2. Find DFT of following sequence x(n) for N=4 and N=8 and plot magnitude of DFT X(k) and comments on results obtained.

$$x(n) = \begin{cases} 1 & 0 \le n \le 2 \\ 0 & otherwise \end{cases}$$

LAB-4 OVERLAP-ADD AND OVERLAP-SAVE METHODS

5.1 Introduction

Generally in discrete-time processing we talk about linearly convolving a sequence with a FIR filter. In contrast, the FFT performs circular convolution with a filter of equal or lesser length. In computing the DFT or FFT we often have to make linear convolution behave like circular convolution, or vice versa. Two methods that make linear convolution look like circular convolution are overlap-save and overlap-add. The overlap-save procedure cuts the signal up into equal length segments with some overlap. Then it takes the DFT of the segments and saves the parts of the convolution that correspond to the circular convolution. Because there are overlapping sections, it is like the input is copied therefore there is not lost information in throwing away parts of the linear convolution. The overlap-add procedure cuts the signal up into equal length segments with no overlap. Then it zero-pads the segments and takes the DFT of the segments. Part of the convolution result corresponds to the circular convolution. The tails that do not correspond to the circular convolution are added to the adjoining tail of the previous and subsequent sequence. This addition results in the aliasing that occurs in circular convolution.

5.2 Objective

5.2.1 Educational

1. Learn to apply overlap-save method for computing linear convolution.
2. Learn to apply overlap-add method for computing linear convolution.
3. Learn to draw the graphs of linear convolution using overlap-save and overlap-add .
4. Gain experience in the computation and use of overlap-save and overlap-add methods.

5.2.2 Experimental

1. Determine the linear convolution using overlap-save method.
2. Draw and compare the linear convolution using built in function and overlap-save method .
3. Determine the linear convolution using overlap-add method.
4. Draw and compare the linear convolution using built in function and overlap-add method.
5. Explain why the Overlap save is a bit easier to handle and more efficient.

5.3 Prelab

1. Read and study the Background section of this Laboratory.
2. Read Appendix B, built in function conv, and fft.

5.4 Equipment needed

1. Personal computer
2. MATLAB software

5.5 Background

The input signal x(n) is often very long especially in real-time signal monitoring applications. For linear filtering via the DFT, for example, the signal must be limited size due to memory requirements. All N-input samples are required simultaneously by the FFT operator. 1. Complexity of N-FFT is N log(N). 2. If N is too large as for long data sequences, then there is a significant delay in processing that precludes real-time processing. The Strategy for filtering long duration of sequences is:
1. Segment the input signal into fixed-size blocks prior to processing.
2. Compute DFT-based linear filtering of each block separately via the FFT.
3. Fit the output blocks together in such a way that the overall output is equivalent to the linear filtering of x(n) directly. Main advantage: samples of the output y(n) = h(n) * x(n) will be available real-time on a block-by-block basis.
Two approaches to real-time linear filtering of long inputs:
1. Overlap-Add Method
2. Overlap-Save Method

5.5.1 Overlap-Add Algorithm

The overlap-add algorithm is based on splitting the signal xL[k] into non-overlapping segments xp[k] of length P.xL[k] might have to be zero-padded so that its total length is a multiple of the segment length P. Introducing the segmentation of xL[k] into the convolution yields where yp[k]=xp[k] * hN[k]. This result states that the convolution of xL[k]*hN[k] can be split into a series of convolutions yp[k] operating on the samples of one segment(block) only. The length of yp[k] is N+P-1. The result of the overall convolution is given by summing up the results from the segments shifted by multiples of the segment length P.
Algorithm for Overlap-Add Method :

1. Break the input signal x(n) into non-overlapping blocks xm(n) of length L.
2. Zero pad h(n) to be of length N = L + M - 1.
3. Take N-DFT of h(n) to give H(k), k = 0, 1, . . . , N - 1.
4. For each block m:
   (a) Zero pad xm(n) to be of length N = L + M - 1.
   (b) Take N-DFT of xm(n) to give Xm(k), k = 0, 1, . . . , N - 1.
   (c) Multiply Ym(k) = Xm(k)* H(k), k = 0, 1, . . . , N - 1.
   (d) Take N-IDFT of Ym(k) to give ym(n), n = 0, 1, . . . , N - 1.
5. Form y(n) by overlapping the last M - 1 samples of ym(n) with the first M - 1 samples of ym+1(n) and adding the result.

5.5.2 Overlap-Save Algorithm

The overlap-save algorithm, also known as overlap-discard algorithm, follows a different strategy as the overlap-add technique introduced above. It is based on an overlapping segmentation of the input xL[k] and application of the periodic convolution for the individual segments.For the first segment x0[k], N-1 zeros have to be appended to the beginning of the input signal xL[k] for the overlapped segmentation. From the result of the periodic convolution xp[k]*hN[k] the first N-1 samples are discarded, the remaining P-N+1 are copied to the output y[k]. This is indicated by the alternative notation overlap-discard used for the technique
Algorithm for Overlap-Save Method :

1. Insert M - 1 zeros at the beginning of the input sequence x(n).
2. Break the padded input signal into overlapping blocks xm(n) of length N = L + M - 1 where the overlap length is M - 1.
3. Zero pad h(n) to be of length N = L + M - 1.
4. Take N-DFT of h(n) to give H(k), k = 0, 1, . . . , N - 1.
5. For each block m:
   (a) Take N-DFT of xm(n) to give Xm(k), k = 0, 1, . . . , N - 1.
   (b) Multiply Ym(k) = Xm(k)* H(k), k = 0, 1, . . . , N - 1.
   (c) Take N-IDFT of Ym(k) to give ym(n), n = 0, 1, . . . , N - 1.
   (d) Discard the first M - 1 points of each output block ym(n).
6. Form y(n) by appending the remaining (i.e., last) L samples of each block ym(n).

5.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

5.7 Program

```
% Overlap Add method for Linear Convolution
close All
clear All
clc
x=input('x=');
h=input('h=');
L=input('L=');
N1=length(x);
```

```
M=length(h);
lc=conv(x,h)
x=[x zeros(1,mod(-N1,L))];
N2=length(x);
h=[h zeros(1,L-1)];
H=fft(h,L+M-1);
S=N2/L;
index=1:L;
X=[zeros(M-1)];
for stage=1:S
xm=[x(index) zeros(1,M-1)]; % Selecting sequence to process
X1=fft(xm,L+M-1);
Y=X1.*H;
Y=ifft(Y);
Z=X((length(X)-M+2):length(X))+Y(1:M-1); %Samples Added in every stage
X=[X(1:(stage-1)*L) Z Y(M:M+L-1)];
index=stage*L+1:(stage+1)*L;
end
i=1:N1+M-1;
X=X(i)
similarity=corrcoef(X,lc) % Similarity b/w Inbuilt and Calculated one
figure()
subplot(2,1,1)
stem(lc);
title('Convolution Using conv() function')
xlabel('n');
ylabel('y(n)');
subplot(2,1,2)
stem(X);
title('Convolution Using Overlap Add Method')
xlabel('n');
ylabel('y(n)');
% Overlap Save method for Linear Convolution
close All
clear All
clc
x=input('x=');
h=input('h=');
L=input('L=');
N1=length(x);
M=length(h);
lc=conv(x,h)
x=[x zeros(1,mod(-N1,L)) zeros(1,L)];
N2=length(x);
h=[h zeros(1,L-1)];
H=fft(h,L+M-1);
S=N2/L;
index=1:L;
xm=x(index); % For first stage Special Case
x1=[zeros(1,M-1) xm]; %zeros appeded at Start point
X=[];
```

```
for stage=1:S
X1=fft(x1,L+M-1);
Y=X1.*H;
Y=ifft(Y);
index2=M:M+L-1;
Y=Y(index2); %Discarding Samples
X=[X Y];
index3=(((stage)*L)-M+2):((stage+1)*L); % Selecting Sequence to process
if(index3(L+M-1)¡=N2)
x1=x(index3);
end
end;
i=1:N1+M-1;
X=X(i)
similarity=corrcoef(X,lc); % Similarity between Inbuilt and Calculated conv
figure()
subplot(2,1,1)
stem(lc);
title('Convolution Using conv() function')
xlabel('n');
ylabel('y(n)');
subplot(2,1,2)
stem(X);
title('Convolution Using Overlap Save Method')
xlabel('n');
ylabel('y(n)');
```

5.8 Output and waveforms

```
x=[1 2 3 4 5 6 7 8 9]
h=[1 2 2 1]
L=5

    lc = 1 4 9 15 21 27 33 39 45 41 26 9

    X = 1.0000 4.0000 9.0000 15.0000 21.0000 27.0000 33.0000 39.0000 45.0000 41.0000 26.0000
9.0000
    similarity = 1 1 1 1
```

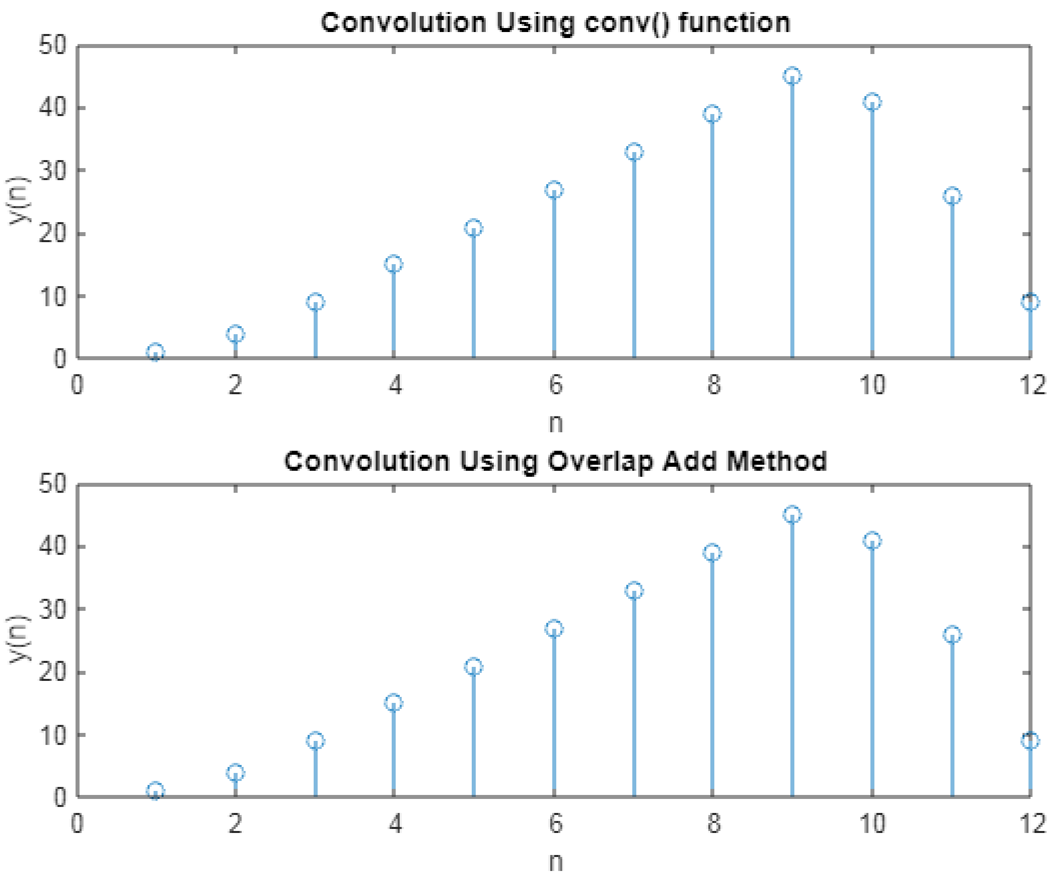


Figure 5.1: Linear convolutoin using overlap add method plot

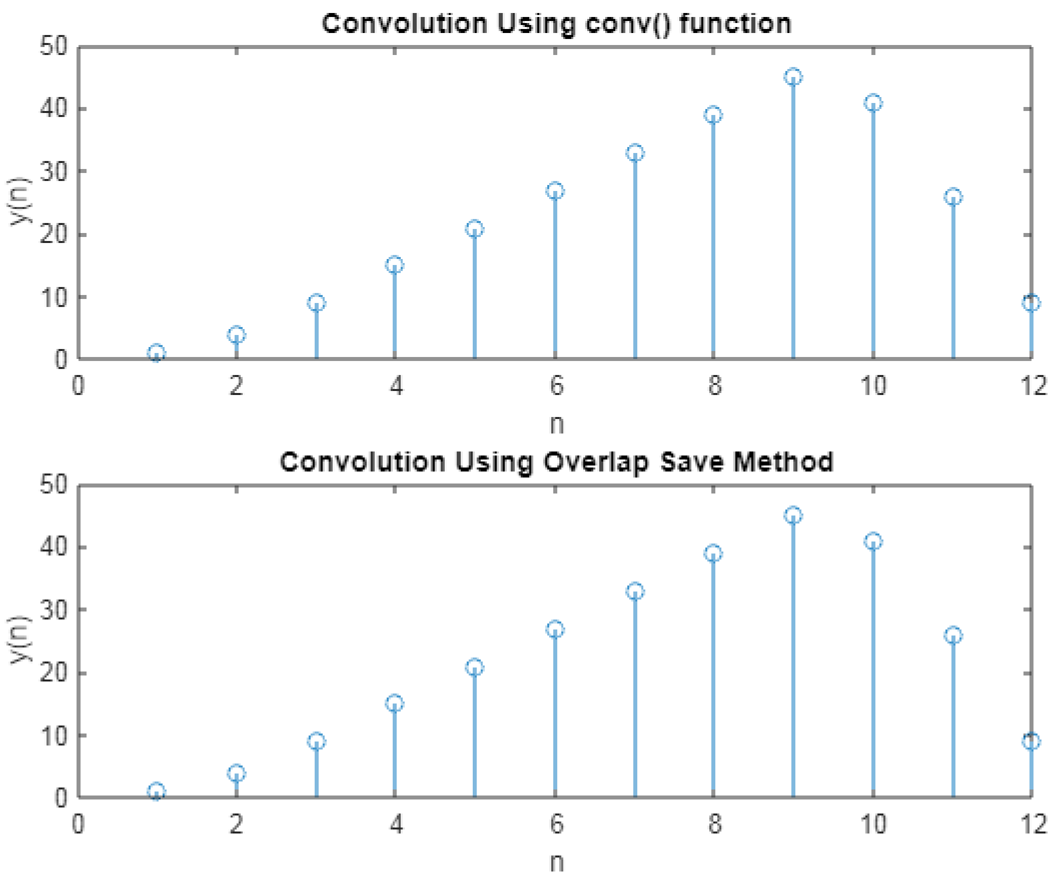


Figure 5.2: Linear convolutoin using overlap save method plot

5.9 Further Probing Experiments

Q1. Simulate the output y(n) of a filter whose impulse response is h(n)=(1,2) and input signal x(n)=(1,2,-1,2,3,-2,-3,-1,1,1,2,-1) using (i) overlap add method (ii) overlap save method

Q2. Simulate the output in MATLAB using built in function conv. Does the simulation support your prediction in question 1.

LAB-5 DIT-FFT ALGORITHM

6.1 Introduction

Fast Fourier Transform is an efficient algorithm developed by Cooley and Turkey in 1965, used to compute the DFT with reduced computations. Due to the efficiency of FFT, it is used for spectrum analysis, convolutions, correlations and linear filtering. To calculate N-point DFT, we require aboutN^2 complex multiplications and N(N-1) complex additions using direct formula. To calculate N-point DFT, we require about N/2 log2N complex multiplications and N log2N complex additions using FFT. The Fast Fourier Transform is useful to map the time-domain sequence into a continuous function of a frequency variable. The most practically implemented choice for r = 2 leads to radix 2 FFT algorithms. So, with N = 2^m, the efficient computation is achieved by breaking the N point DFT into two equal length DFTs and continuing this process until 2 point DFTs are obtained.

6.2 Objective

6.2.1 Educational

1. Learn to apply decimation in time FFT algorithm to compute N-point DFT of a sequence.
2. Learn to draw magnitude and phase spectrum for N-point DFT using DIT-FFT.
3. Gain experience in the construction and use of DIT-FFT butterfly diagrams.

6.2.2 Experimental

1. Confirm decimation in time FFT algorithm in calculating N-point DFT.
2. Draw frequency spectrum for a discrete signal x[n].
3. Determine the number of computations required for calculating N-point DFT using the direct formula and DIT-FFT .

6.3 Prelab Preparation:

1. Read and study the Background section of this Laboratory.
2. Read Appendix B, built in function fft.

6.4 Equipment needed

1. Personal computer
2. MATLAB software

6.5 Background

In this algorithm, the time - domain sequence x[n] is decimated into two equal half sequences, one composed of even – indexed values of x[n], and other composed of odd - indexed values of x[n].i.e.,

g[n]=x[2n] — even indexed sequence

h[n]=x[2n+1] — odd indexed sequence

The N - point DFT of x[n] is given by

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}$$

N represents the sequence length and it is calculated by using the command 'length'.
This can be rewritten in terms of twiddle factor as
$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk}$
$X[k] = \sum_{n=even}^{N-1} x[n]W_N^{nk} + \sum_{n=odd}^{N-1} x[n]W_N^{nk}$
$X[k] = \sum_{n=0}^{N/2-1} x[2n]W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1]W_N^{(2n+1)k}$
$X[k] = \sum_{n=0}^{N/2-1} x[2n]W_N^{2nk} + W_N^k \sum_{n=0}^{N/2-1} x[2n+1]W_N^{2nk}$
$X[k] = \sum_{n=0}^{N/2-1} x[2n]W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x[2n+1]W_{N/2}^{nk}$
$X[k] = \sum_{n=0}^{N/2-1} g[n]W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} h[n]W_{N/2}^{nk}$
$X[k] = G[k] + W_N^k H[k]$
Where G(k) and H(k) are the N/2 – point DFTs of g[n] and h[n] respectively. So, G(k) and H(k) are periodic with period N/2 .i.e.,
G(k+N/2)=G(k)
H(k+N/2)=H(k)
And using the symmetry property of the twiddle factor, WN,we can write
$X[k + N/2] = G[k] - W_N^k H[k]$
From equations X[k] and X[k+N/2] result in the following butterfly diagram For example, for

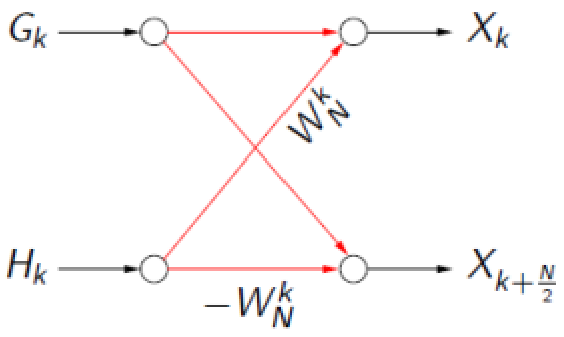


Figure 6.1: Butterfly diagram for DIT-FFT

N = 8, the DFT points in terms of G and H are
$X[0] = G[0] + W_N^0 H[0]$
$X[1] = G[1] + W_N^1 H[1]$
$X[2] = G[2] + W_N^2 H[2]$
$X[3] = G[3] + W_N^3 H[3]$
For the remaining 4 points X(4) to X(7), we use equations X[k+N/2] to get
$X[0] = G[0] - W_N^0 H[0]$
$X[1] = G[1] - W_N^1 H[1]$
$X[2] = G[2] - W_N^2 H[2]$

$X[3] = G[3] - W_N^3 H[3]$
The above process is repeated for calculating the N/2 point DFTs of g[n] and h[n], and this is continued till we get two point DFTs.The overall butterfly diagram for DIT FFT algorithm for N =8 is

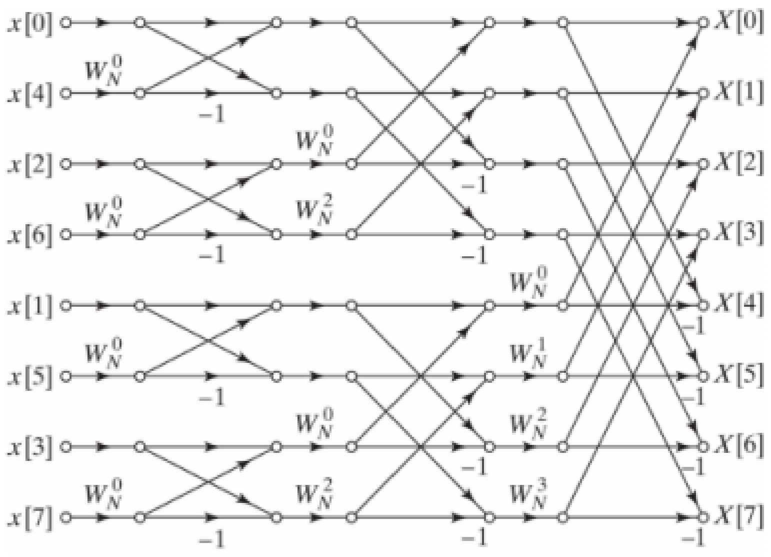


Figure 6.2: Butterfly diagram for 8-point DIT-FFT

Due to repeated decimations, the input sequence is scrambled, and the order of the final input sequence is obtained as follows

Input sequence	Index	Binary form of index	Bit reversed form of index	Decimal representation	Final input sequence order
x[0]	0	000	000	0	x[0]
x[1]	1	001	100	4	x[4]
x[2]	2	010	010	2	x[2]
x[3]	3	011	110	6	x[6]
x[4]	4	100	001	1	x[1]
x[5]	5	101	101	5	x[5]
x[6]	6	110	011	3	x[3]
x[7]	7	111	111	7	x[7]

6.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the ‘FILE’ Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

6.7 Program

```
% Direct computation of FFT
x=[1 2 3 4 4 3 2 1];
N=length(x);
y=fft(x,N);
disp(’FFT using built in command’);
disp(y);
stem(abs(y));
ylabel (’Amplitude’);
xlabel (’N’);
title(’Magnitude Response’);
```

```
    %Matlab Program for FFT using DIT algorithm
clc;
clear all;
close all;
x=input(’enter x[n]:’);
N=length(x);
levels=nextpow2(N);
xn=[x,zeros(1,(2^levels-N)];
x=bitrevorder(xn)
N=length(xn);
tw=cos(2*pi*(1/N)*(0:N/2-1))-j*sin(2*pi*(1/N)*(0:N/2-1));
for level=1:levels;
L=2^level;
twlvl=tw(1:N/L:N/2);
for k=0:L:N-L;
for n=0:L/2-1;
A=x(n+k+1);
B=x(n+k+(L/2)+1)*twlvl(n+1);
x(n+k+1)=A+B;
x(n+k+(L/2)+1)=A-B;
end
end
x
end
XK=x
n=0:N-1;
subplot(2,2,1);stem(n,xn);title(’x(n)’);xlabel(’n’);ylabel(’Amplitude’);
subplot(2,2,2);stem(n,real(XK));title(’Real part of X(K)’);xlabel(’n’);ylabel(’Amplitude’);
subplot(2,2,3);stem(n,imag(XK));title(’Imag part of X(K)’);xlabel(’n’);ylabel(’Amplitude’);
```

6.8 Output and waveforms

```
FFT using built in command
20.0000 + 0.0000i -5.8284 - 2.4142i 0.0000 + 0.0000i -0.1716 - 0.4142i 0.0000 + 0.0000i -0.1716
+ 0.4142i 0.0000 + 0.0000i -5.8284 + 2.4142i
enter x[n]:[1 2 3 4 4 3 2 1]
    XK =
    20.0000 -5.8284 - 2.4142i 0 -0.1716 - 0.4142i 0 -0.1716 + 0.4142i
```

```
0 -5.8284 + 2.4142i
```

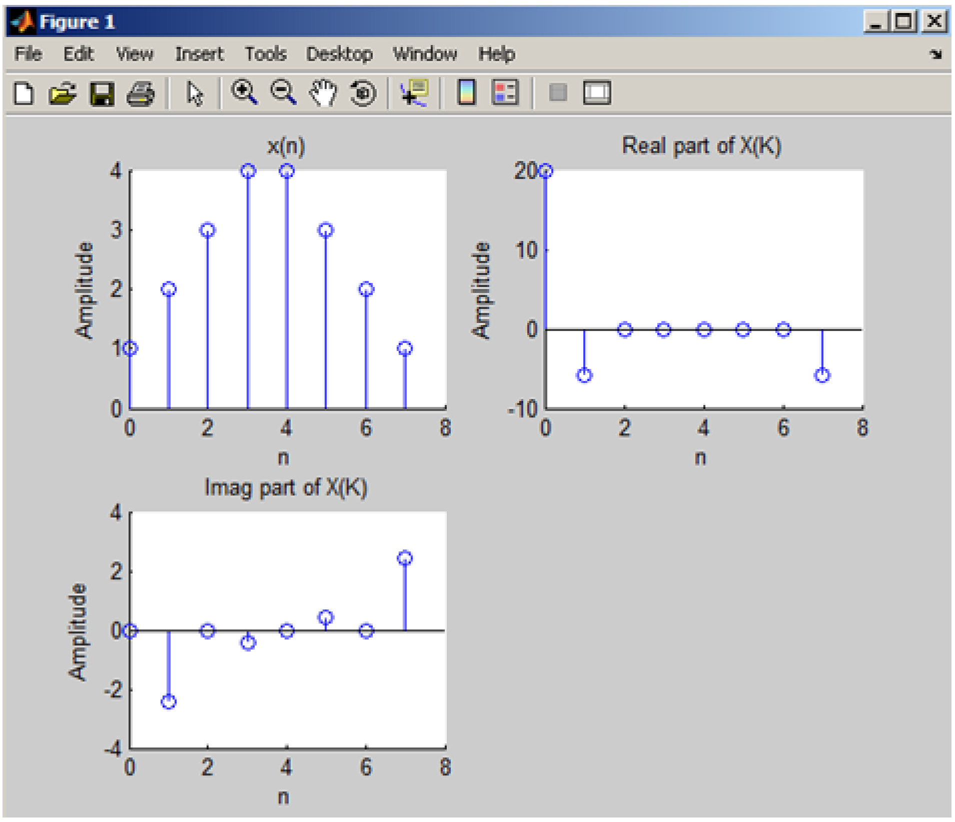


Figure 6.3: DIT-FFT

6.9 Further Probing Experiments

Q1. Compute the IDFT using DIF FFT algorithm given that X (k) = { 4, 1-j2.414, 0, 1-j0.410, 0, 1+jo.414,0, 1+j2.414}.
Q2. Simulate an 8-point DIT FFT with one sinusoid as input, calculate and change parameters on the butterfly structure of the FFT implementation.

LAB-6 DIF-FFT ALGORITHM

7.1 Introduction

Fast Fourier Transform is an efficient algorithm developed by Cooley and Turkey in 1965, used to compute the DFT with reduced computations. Due to the efficiency of FFT, it is used for spectrum analysis, convolutions, correlations and linear filtering. To calculate N-point DFT, we require aboutN^2 complex multiplications and N(N-1) complex additions using direct formula. To calculate N-point DFT, we require about N/2 log2N complex multiplications and N log2N complex additions using FFT. The Fast Fourier Transform is useful to map the time-domain sequence into a continuous function of a frequency variable. The most practically implemented choice for r = 2 leads to radix 2 FFT algorithms. So, with N = 2^m, the efficient computation is achieved by breaking the N point DFT into two equal length sequences in frequency domain for DIF-FFTalgorithm and continuing this process until 2 point DFTs are obtained.

7.2 Objective

7.2.1 Educational

1. Learn to apply decimation in frequency FFT algorithm to compute N-point DFT of a sequence.
2. Learn to draw magnitude and phase spectrum for N-point DFT using DIF-FFT.
3. Gain experience in the construction and use of DIF-FFT butterfly diagrams.

7.2.2 Experimental

1. Confirm decimation in frequency FFT algorithm in calculating N-point DFT.
2. Draw frequency spectrum for a discrete signal x[n].
3. Determine the number of computations required for calculating N-point DFT using the direct formula and DIF-FFT .

7.3 Prelab Preparation:

1. Read and study the Background section of this Laboratory.
2. Read Appendix B, built in function fft.

7.4 Equipment needed

1. Personal computer
2. MATLAB software

7.5 Background

In this algorithm, the frequency - domain sequence X[k] is decimated into two equal half sequences, one composed of even – indexed values of X[k], and other composed of odd - indexed values of X[k].i.e.,

G[k]=X[2k] — even indexed sequence

H[k]=X[2k+1] — odd indexed sequence

The N - point DFT of x[n] is given by

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}$$

N represents the sequence length and it is calculated by using the command 'length'.
This can be rewritten in terms of twiddle factor as

$$\begin{aligned} X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk} &= \sum_{n=0}^{N/2-1} x[n]W_N^{nk} + \sum_{n=N/2}^{N-1} x[n]W_N^{nk} \\ &= \sum_{n=0}^{N/2-1} x[n]W_N^{nk} + \sum_{n=0}^{N/2-1} x[n+N/2]W_N^{nk}W_N^{N/2k} \\ &= \sum_{n=0}^{N/2-1} \{x[n] + (-1)^k\ x[n+N/2]\}W_N^{nk}, k = 0,1,2,....,N-1 \end{aligned}$$

Now, decimating X(k) into even and odd indexed samples,

G[k]=X[2k]=$\sum_{n=0}^{N/2-1} \{x[n] + \ x[n+N/2]\}W_{N/2}^{nk}$,k=0,1,2,....,N/2-1
H[k]=X[2k+1]=$\sum_{n=0}^{N/2-1} \{x[n] - \ x[n+N/2]\}W_{N/2}^{nk}$,k=0,1,2,....,N/2-1

Let g[n]=x[n] + x[n+$\frac{N}{2}$]
Let h[n]=$\{x[n] - \ x[n+\frac{N}{2}]\}W_N^n$

From equations g[n] and h[n] result in the following butterfly diagram

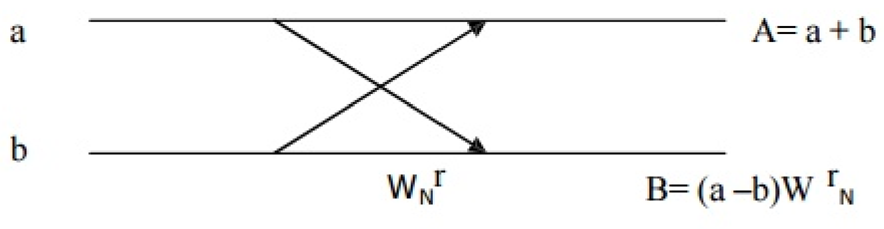


Figure 7.1: Butterfly diagram for DIF-FFT

For example, for N=8, using equations g[n] and h[n], we get

g[0]=x[0]+x[4]
g[1]=x[1]+x[5]
g[2]=x[2]+x[6]

g[3]=x[3]+x[7]
h[0]=$\{x[0]-x[4]\}W_8^0$
h[1]=$\{x[1]-x[5]\}W_8^1$
h[2]=$\{x[2]-x[6]\}W_8^2$
h[0]=$\{x[3]-x[7]\}W_8^3$

The above process is repeated for calculating the N/2 point DFTs, and this is continued till we get two point DFTs.The input sequence is in same order and output sequence is in bit reversal order for DIF-FFT.The overall butterfly diagram for DIF FFT algorithm for N =8 is

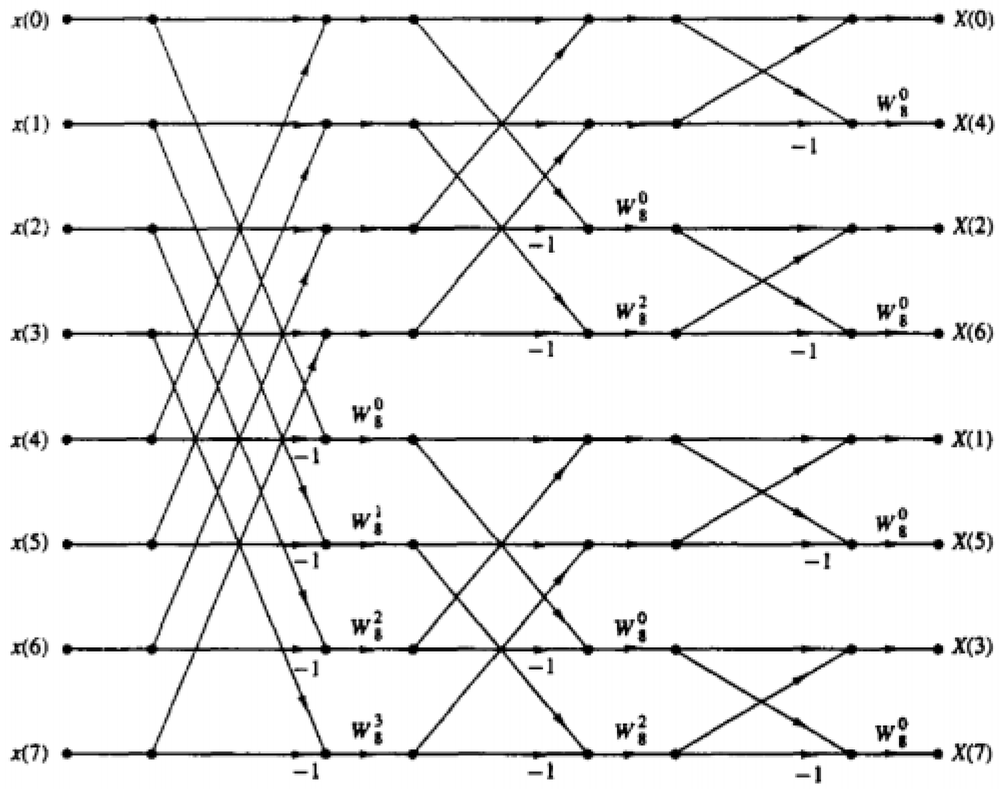


Figure 7.2: Butterfly diagram for 8-point DIF-FFT

7.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

7.7 Program

```
% Direct computation of FFT
x=[1 1 0 0];
N=4;
y=fft(x,N);
stem(abs(y));
ylabel ('Amplitude');
xlabel ('N');
title('Magnitude Response');

    %Matlab Program for FFT using DIF algorithm
clc;
clear all;
close all;
xn=input('enter a sequence:');
p=length(xn);
N=input('enter length of the sequence : '); if N>p xn=[xn,zeros(1,N-p)];
    else
    xn=xn;
    end
    x=xn;
    lev=log2(N);
    tw=exp(-1i*2*pi*(0:N/2-1)/N);
    for level=lev:-1:1; L=2^level; twlvl=tw(1:N/L:N/2);
    for k=0:L:N-L;
    for n=0:L/2-1;
    A=x(n+k+1);
    B=x(n+k+(L/2)+1);
    x(n+k+1)=A+B;
    x(n+k+(L/2)+1)=(A-B)*twlvl(n+1);
    end
    end
    end
    XK=bitrevorder(x);
    l=0:N-1;
    subplot(3,2,1),stem(l,abs(xn));
    title('input sequence');
    xlabel('n');
    ylabel('x[n]');
    subplot(3,2,2),stem(l,abs(XK));
    title('FFT using DIF algorithm');
    xlabel('k');ylabel('X[k]');
    ansft=fft(xn)
    subplot(3,2,3),stem(l,abs(fft(xn)));
    title('FFT using function');
    xlabel('k');ylabel('X[k]');
    %ifft
    x1=conj(XK);
    levels=log2(N);
    tw=exp(-1i*2*pi*(0:N/2-1)/N);
    for level=levels:-1:1; L=2^level;
```

```
twlvl=tw(1:N/L:N/2);
for k=0:L:N-L;
for n=0:L/2-1;
A=x1(n+k+1);
B=x1(n+k+(L/2)+1);
x1(n+k+1)=A+B;
x1(n+k+(L/2)+1)=(A-B)*twlvl(n+1);
end
end
end
x=bitrevorder(x1);
xk=conj(x)/N
ansift=ifft(XK)
subplot(3,2,4),stem(l,abs(xk));
title('IFFT using DIF algorithm-');
xlabel('n');
ylabel('x[n]');
subplot(3,2,5),stem(l,abs(ansift));
title('IFFT using function');
xlabel('n');ylabel('x[n]');
```

7.8 Output and waveforms

```
FFT using built in command
20.0000 + 0.0000i -5.8284 - 2.4142i 0.0000 + 0.0000i -0.1716 - 0.4142i 0.0000 + 0.0000i -0.1716
+ 0.4142i 0.0000 + 0.0000i -5.8284 + 2.4142i
enter x[n]:[1 2 3 4 4 3 2 1]
    XK =
    20.0000 -5.8284 - 2.4142i 0 -0.1716 - 0.4142i 0 -0.1716 + 0.4142i
    0 -5.8284 + 2.4142i
```

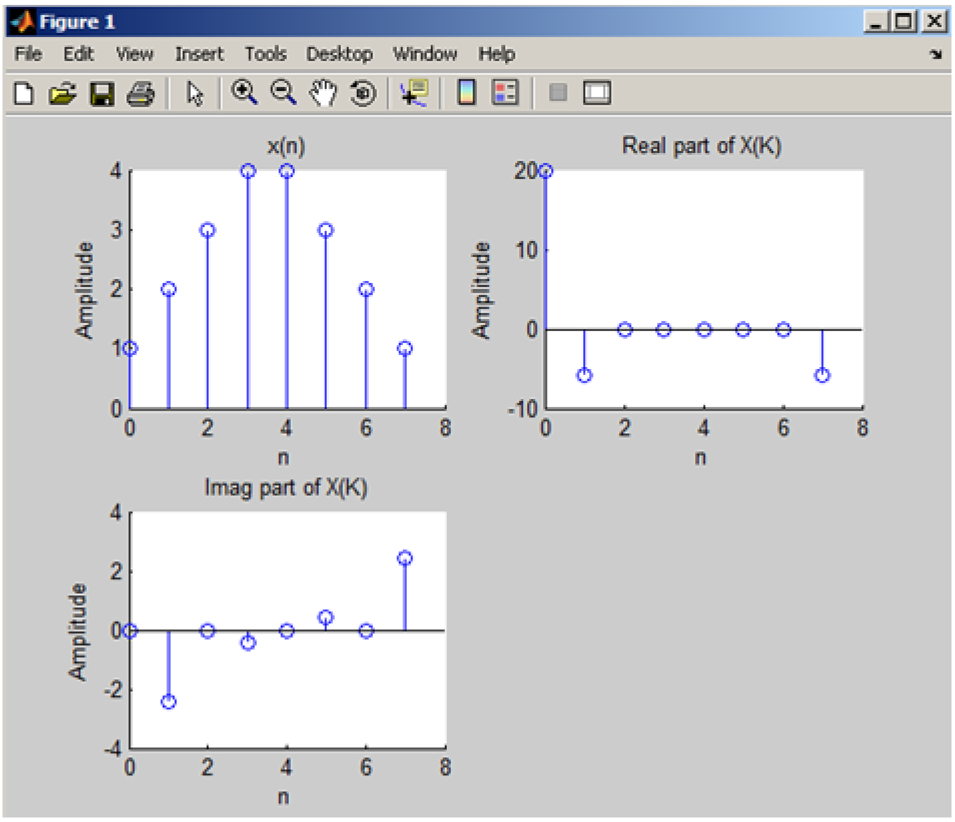


Figure 7.3: DIF-FFT

7.9 Further Probing Experiments

Q1. Simulate DIF- FFTs of different lengths for two sinusoids, observing how the change in resolution affects your output;
Q2. Simulate an 8-point DIF FFT with one sinusoid as input, calculate and change parameters on the butterfly structure of the FFT implementation.

LAB-7 IIR DIGITAL FILTER USING BUTTERWORTH METHOD AND BILINEAR TRANSFORMATION

8.1 Introduction

This laboratory studies the use of Butterworth method and bilinear transformation to design IIR filters. Filters are an essential tool in our complex world of mixed signals — both electronic and otherwise. IIR (infinite impulse response) filters are generally chosen for applications where linear phase is not too important and memory is limited. They have been widely deployed in audio equalization, biomedical sensor signal processing, IOT smart sensors and high-speed telecommunication/RF applications.

8.2 Objective

8.2.1 Educational

1. Learn the four general filter types: High-pass, Low-pass, Band pass, and Band stop.
2. Learn to alter filter type by changing the specifications.
3. Learn magnitude and phase spectrum of IIR filters.
4. Design simple filter.

8.2.2 Experimental

1. Calculate and measure order and cutoff frequency for IIR filters.
2. Design simple IIR low-pass filter.
3. Generate and interpret spectrum plots for filters.

8.3 Prelab Preparation:

1. Study the Background section of this Laboratory exercise.
2. Study textbook regarding origin and meaning of IIR filters.

8.4 Equipment needed

1. Personal computer
2. MATLAB software

8.5 Background

The IIR filter can realize both the poles and zeroes of a system because it has a rational transfer function, described by polynomials in z in both the numerator and the denominato.The transfer functoin of IIR filter of order N is given by

$$H[z] = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=1}^{N} a_k z^{-k}}$$

The difference equation for such a system is described by the following:

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k)$$

M and N are order of the two polynomials bk and ak are the filter coefficients. These filter coefficients are generated using MATLAB. IIR filters can be expanded as infinite impulse response filters. In designing IIR filters, cutoff frequencies of the filters should be mentioned. The order of the filter can be estimated using butter worth polynomial. That's why the filters are named as butter worth filters. Filter coefficients can be found and the response can be plotted.

8.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

8.7 Program

```
% Design a digital lowpass Butterworth filter with pass cut-off frequencies 200 hz and 500 hz respectively.The pass band and stop band attenuations are -5db and -12db repectively. The sampling frequency is 5000hz.
clc;
clear all;
close all;
% pass band cut off frequency
fp = 200;
% stop band cut off frequency
fs = 500;
% sampling frequency
Fs=5000;
% corresponding digital frequencies
Wp=2*fp/Fs;
Ws=2*fs/Fs;
% pass band ripple
Rp = 5;
% stop band ripple
```

Rs = 12;
% Butterworth filter order and cutoff frequency
[n,Wn] = buttord(Wp,Ws,Rp,Rs);
% butter(n,Wn) returns the transfer function coefficients of an nth-order lowpass digital Butterworth filter with normalized cutoff frequency Wn.
[b,a] = butter(n,Wn);
% freqz - Frequency response of digital filter
freqz(b,a)
% sprintf is used to format data into string or character vector
title(sprintf('n = %d Butterworth lowpass Filter',n))

% Design a digital highpass Butterworth filter with pass cut-off frequencies 200 hz and 500 hz respectively.The pass band and stop band attenuations are -5db and -12db repectively. The sampling frequency is 5000hz.
clc;
clear all;
close all;
% pass band cut off frequency
fp = 200;
% stop band cut off frequency
fs = 500;
% sampling frequency
Fs=5000;
% corresponding digital frequencies
Wp=2*fp/Fs;
Ws=2*fs/Fs;
% pass band ripple
Rp = 5;
% stop band ripple
Rs = 12;
% Butterworth filter order and cutoff frequency
[n,Wn] = buttord(Wp,Ws,Rp,Rs);
% butter(n,Wn) returns the transfer function coefficients of an nth-order highpass digital Butterworth filter with normalized cutoff frequency Wn.
[b,a] = butter(n,Wn,'high');
% freqz - Frequency response of digital filter
freqz(b,a)
% sprintf is used to format data into string or character vector
title(sprintf('n = %d Butterworth highpass Filter',n))

%Design a Butterworth digital band pass filter with pass band and stop band attenuation -3db and -10db respectively. The pass band cut-off frequencies are 600hz and 1000hz, and stop band frequencies are 300 hz and 1600 hz. The sampling frequency is 5000 hz.
clc;
clear all;
close all;
% pass band cut off frequency
fp = [600 1000];
% stop band cut off frequency

```
fs = [300 1600];
% sampling frequency
Fs=5000;
% corresponding digital frequencies
Wp=2*fp/Fs;
Ws=2*fs/Fs;
% pass band ripple
Rp = 3;
% stop band ripple
Rs = 10;
% Butterworth filter order and cutoff frequency
    [n,Wn] = buttord(Wp,Ws,Rp,Rs);
% butter(n,Wn) returns the transfer function coefficients of an nth-order bandpass digital But-
terworth filter with normalized cutoff frequency Wn.
    [b,a] = butter(n,Wn,’bandpass’);
% freqz - Frequency response of digital filter
freqz(b,a)
% sprintf is used to format data into string or character vector
title(sprintf(’n = %d Butterworth bandpass Filter’,n))
```

%Design a Butterworth digital band stop filter with pass band and stop band attenuation -3db and -10db respectively. The pass band cut-off frequencies are 600hz and 1000hz, and stop band frequencies are 300 hz and 1600 hz. The sampling frequency is 5000 hz.

```
    clc;
clear all;
close all;
% pass band cut off frequency
fp = [600 1000];
% stop band cut off frequency
fs = [300 1600];
% sampling frequency
Fs=5000;
% corresponding digital frequencies
Wp=2*fp/Fs;
Ws=2*fs/Fs;
% pass band ripple
Rp = 3;
% stop band ripple
Rs = 10;
% Butterworth filter order and cutoff frequency
    [n,Wn] = buttord(Wp,Ws,Rp,Rs);
% butter(n,Wn) returns the transfer function coefficients of an nth-order bandstop digital But-
terworth filter with normalized cutoff frequency Wn.
    [b,a] = butter(n,Wn,’stop’);
% freqz - Frequency response of digital filter
freqz(b,a)
% sprintf is used to format data into string or character vector
title(sprintf(’n = %d Butterworth bandstop Filter’,n))
```

8.8 Output and waveforms

```
%Low pass and high pass filter
n =
    2
    Wn =
    0.1044
% band pass filter n =
    1
    Wn =
    0.2200 0.4292
```

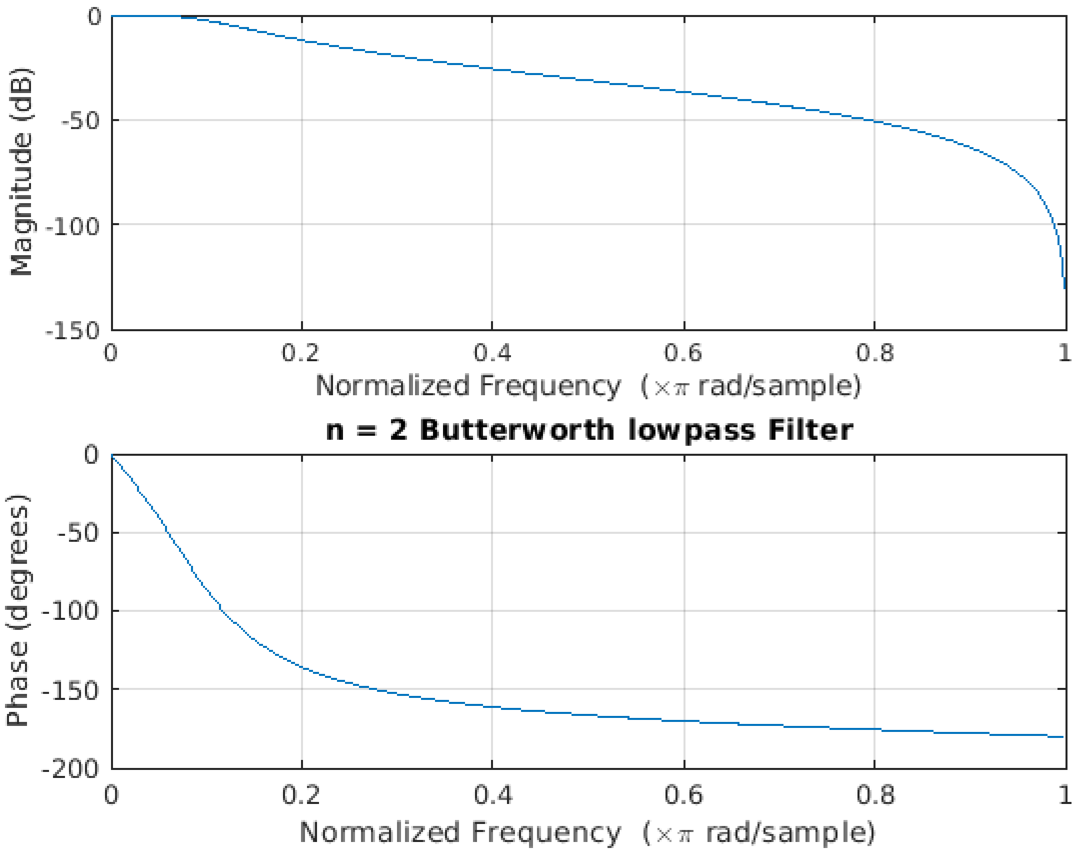


Figure 8.1: Butter worth lowpass IIR filter plot

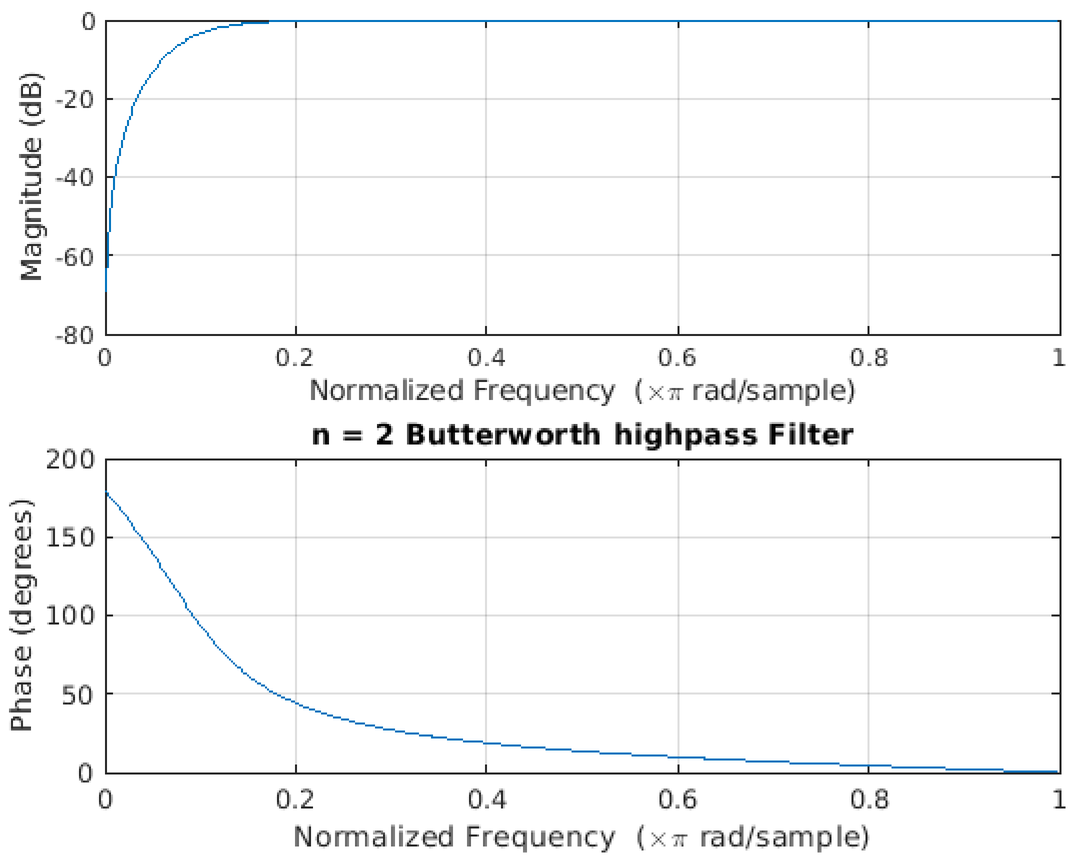


Figure 8.2: Butter worth highpass IIR filter plot

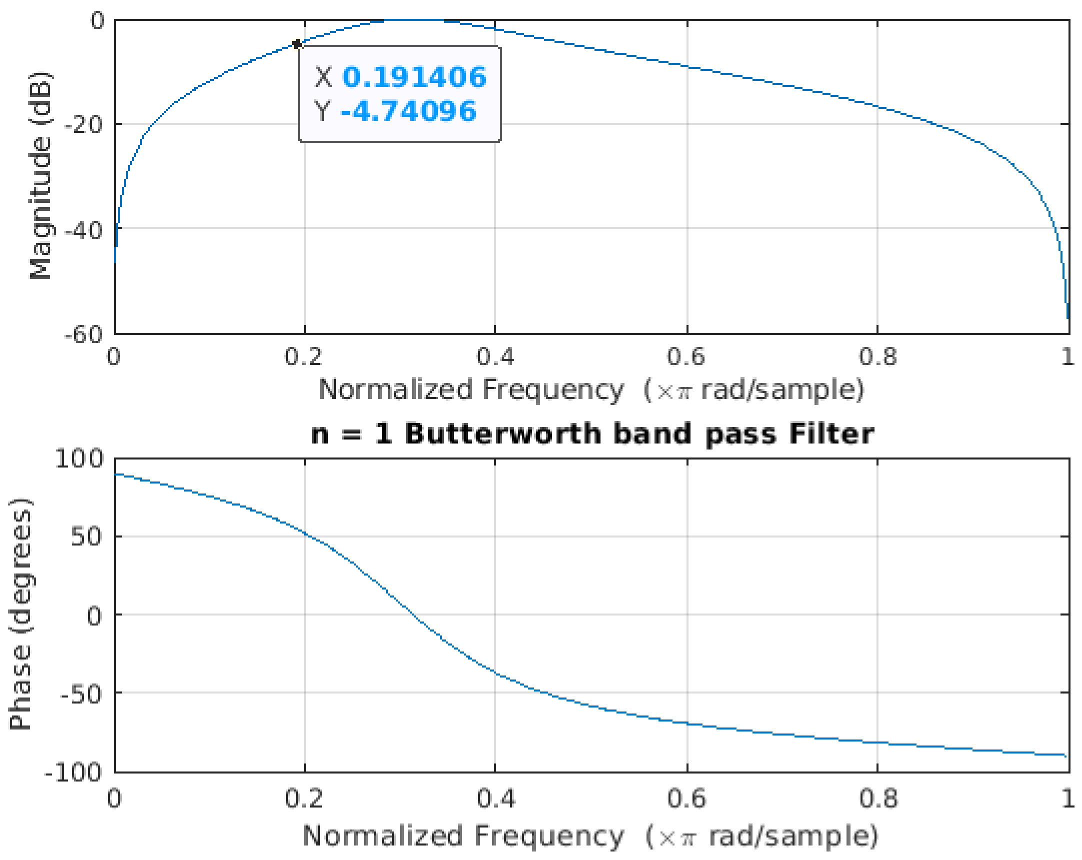


Figure 8.3: Butter worth bandpass IIR filter plot

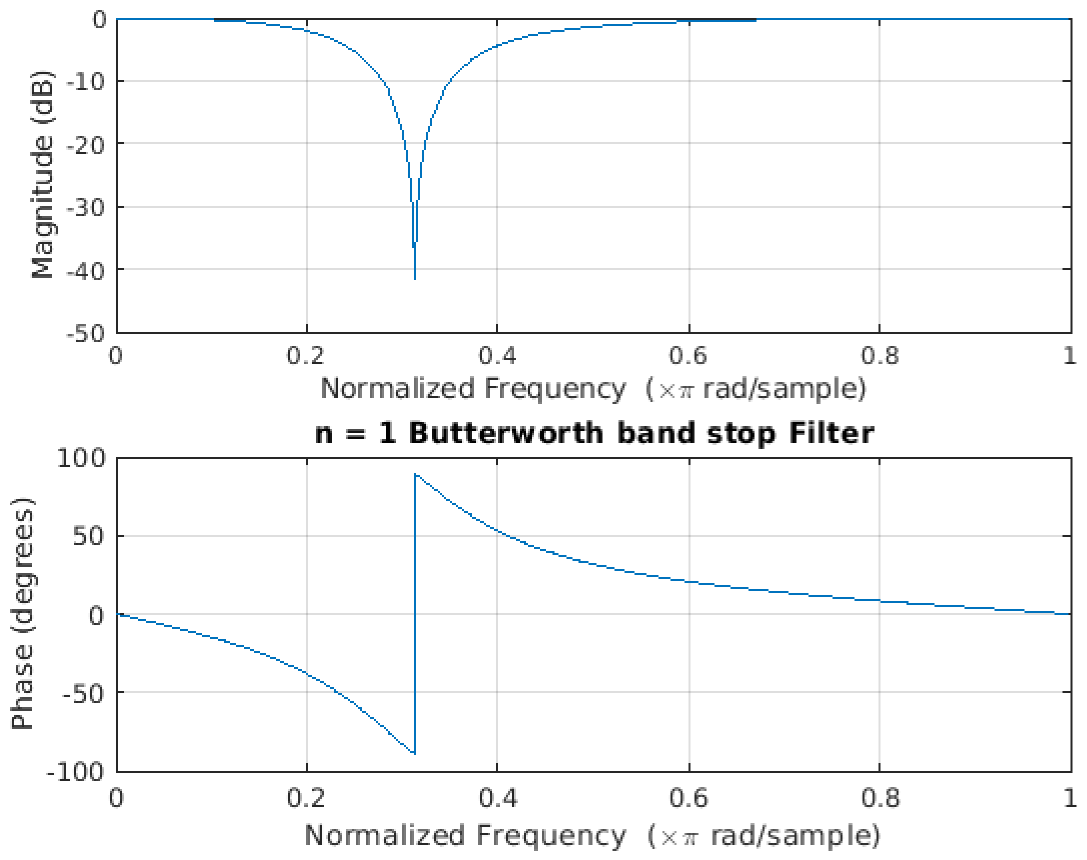


Figure 8.4: Butter worth bandstop IIR filter plot

8.9 Further Probing Experiments

Q1. Consider the system function H(s) =1/s+a .
(a)Determine the discrete-time transfer function H(z) obtained by mapping H(s) to H(z) using the bilinear transformation with T = 2.
(b) Find the range of the constant a for which H(s) is stable and causal.
(c) Verify that if H(s) is stable and causal, then H(z) is also stable and causal

Q2. Design a Butterworth digital low pass filter with pass band and stop band attenuation -3db and -20db respectively. The pass band cut-off frequencys is 0.15π rad, and stop band frequency is 0.35π rad.

LAB-8 IIR DIGITAL FILTER USING CHEBYSHEV (TYPE I AND II) METHOD

9.1 Introduction

The Chebyshev filter is well known and described in all the texts. Its transition band can be made narrower by adding ripple to the pass band. This also makes the group delay variation and subsequent distortion worse. Chebyshev filters are used for distinct frequencies of one band from another. They cannot match the windows-sink filter's performance and they are suitable for many applications. The main feature of Chebyshev filter is their speed, normally faster than the windowed-sinc.

9.2 Objective

9.2.1 Educational

1. To teach the properties of IIR filters using Chebyshev methods.

9.2.2 Experimental

1. Characterize the behavior of IIR digital filters using Chebyshev method, including concepts such as order of filter, transfer function, and spectrum.

9.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding filter designing.
2. Study the Background section below.

9.4 Equipment needed

1. Personal computer
2. MATLAB software

9.5 Background

It is clear from the Butterworth filter, the attenuation of Butterworth filter increases monotonically with frequency , It is also observed from the magnitude spectrum that the attenuation is gradual in pass band and quick in the stop band. In order to increase-the attenuation in the stop band, the order of the filter has to be increased. The increased order leads to complexity in circuit implementation. The Chebyshev filter gives better solution to the same order of Butterworth filter. The major set back of Chebyshev filter is its ripples behaviors either in pass band or in stop band.There are 2 types of Chebyshev Filters; Type 1 and Type 2. Type 1 Chebyshev Filter is all-pole with an equiripple passband and a monotonically decreasing stopband. The magnitude of Frequency Response of type 1 Chebyshev Filter is:

$$H(j\Omega)^2 = \frac{1}{1+\xi^2 C_N^2(\frac{\Omega}{\Omega_p})}$$

N = Filter Order Ω_p = passband cut-off frequency and this parameter that controls pass band ripple amplitude. The discrimination and selectivity factor formula is same as Butterworth IIR filter. The Filter Order, N is determined by

$$N \geq \frac{\cosh^{-1}\sqrt{\frac{10^{0.1\alpha_p - 1}}{10^{0.1\alpha_s - 1}}}}{\cosh^{-1}\frac{\Omega_s}{\Omega_p}}$$

9.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

9.7 Program

```
% Design a Chebyshev type-1 digital low pass filter with pass band and stop band attenuation
-10db and -60db respectively. The pass band cut-off frequency is 80 Hz, and stop band frequency
is 150 Hz. The sampling frequency is 500 Hz.
clc;
clear all;
close all;
% pass band cut off frequency
fp = 80;
% stop band cut off frequency
fs = 150;
% sampling frequency
Fs=500;
% corresponding digital frequencies
Wp=fp/Fs;
Ws=fs/Fs;
% pass band ripple
Rp = 10;
% stop band ripple
Rs = 60;
% Chebyshev type 1 filter order and cutoff frequency
```

```
    [n,Wn] = cheb1ord(Wp,Ws,Rp,Rs);
disp('Chebyshev type 1 order and cutoff frequency');
disp(n);
disp(Wn);
% butter(n,Wn) returns the transfer function coefficients of an nth-order lowpass digital Cheby-
shev type-1 filter with normalized cutoff frequency Wn.
    [b,a] = cheby1(n,Rp,Wp);
% freqz - Frequency response of digital filter
    freqz(b,a);
% sprintf is used to format data into string or character vector
    title(sprintf('n = %d chebyshev type-1 low pass Filter',n))
    % For data sampled at 1000 Hz, design a highpass Chebyshev type 2 filter with less than 3
dB of ripple in the passband defined from 0 to 40 Hz, and at least 60 dB of attenuation in the
stopband defined from 150 Hz to the Nyquist frequency.
    clc;
clear all;
close all;
Wp = 40/500;
Ws = 150/500;
Rp = 3;
Rs = 60;
    [n,Ws] = cheb2ord(Wp,Ws,Rp,Rs);
    disp('The order and cutoff frequency of Chebyshev type 2 filter');
disp(n);
disp(Ws);
    [b,a] = cheby2(n,Rs,Ws,'high');
    freqz(b,a,512,1000)
title(sprintf('n = %d Chebyshev Type II highpass Filter',n));
    % For data sampled at 1000 Hz,design a Chebyshev type 2 bandpass filter with a passband
of 60 Hz to 200 Hz, with less than 3 dB of ripple in the passband, and 40 dB attenuation in the
stopbands that are 50 Hz wide on both sides of the passband to the Nyquist frequency.
clc;
clear all;
close all;
Wp = [60 200]/500;
Ws = [50 250]/500;
Rp = 3;
Rs = 40;
    [n,Ws] = cheb2ord(Wp,Ws,Rp,Rs) [b,a] = cheby2(n,Rs,Ws);
freqz(b,a,512,1000);
title(sprintf('n = %d Chebyshev Type II Bandpass Filter',n));
```

9.8 Output and waveforms

```
Chebyshev type 1 order and cutoff frequency n =
    5
    Wn =
    0.1600
The order and cutoff frequency of Chebyshev type-2 filter 4
    0.3000 n =
    7
```

```
Ws =
0.1000 0.5000
```

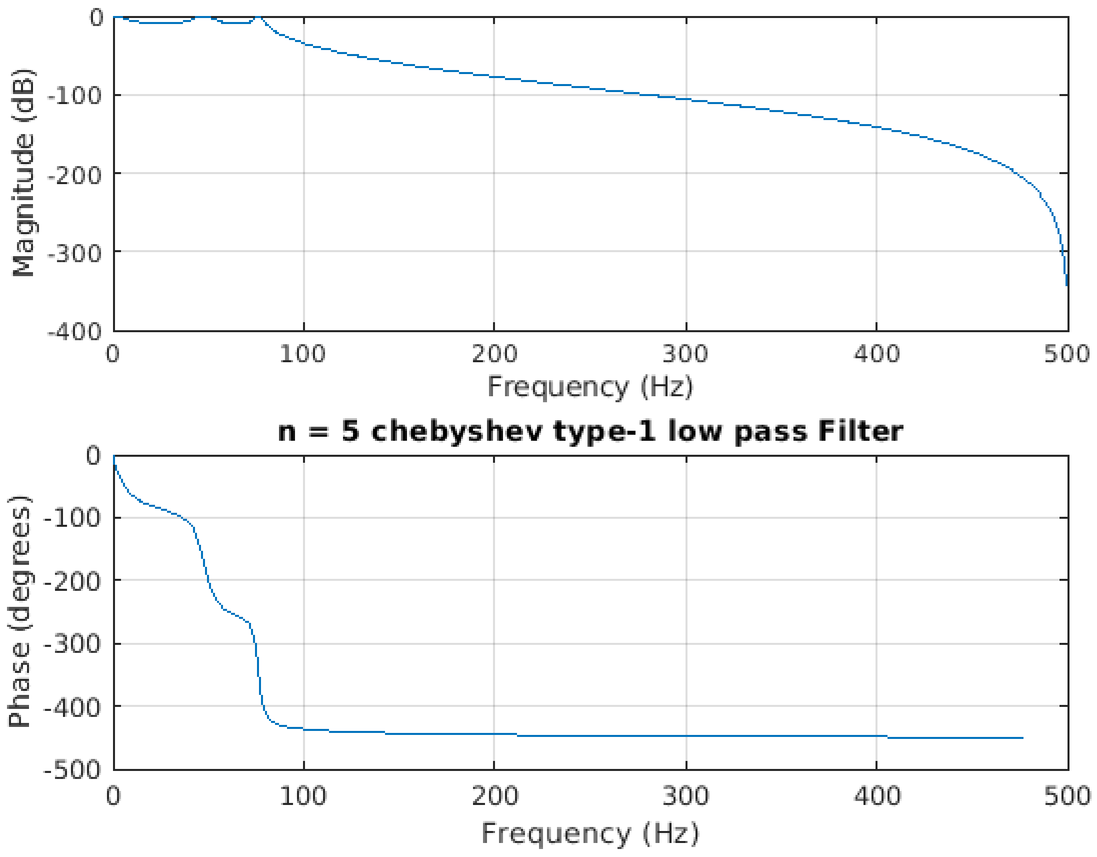


Figure 9.1: Chebyshev type 1 lowpass filter plot

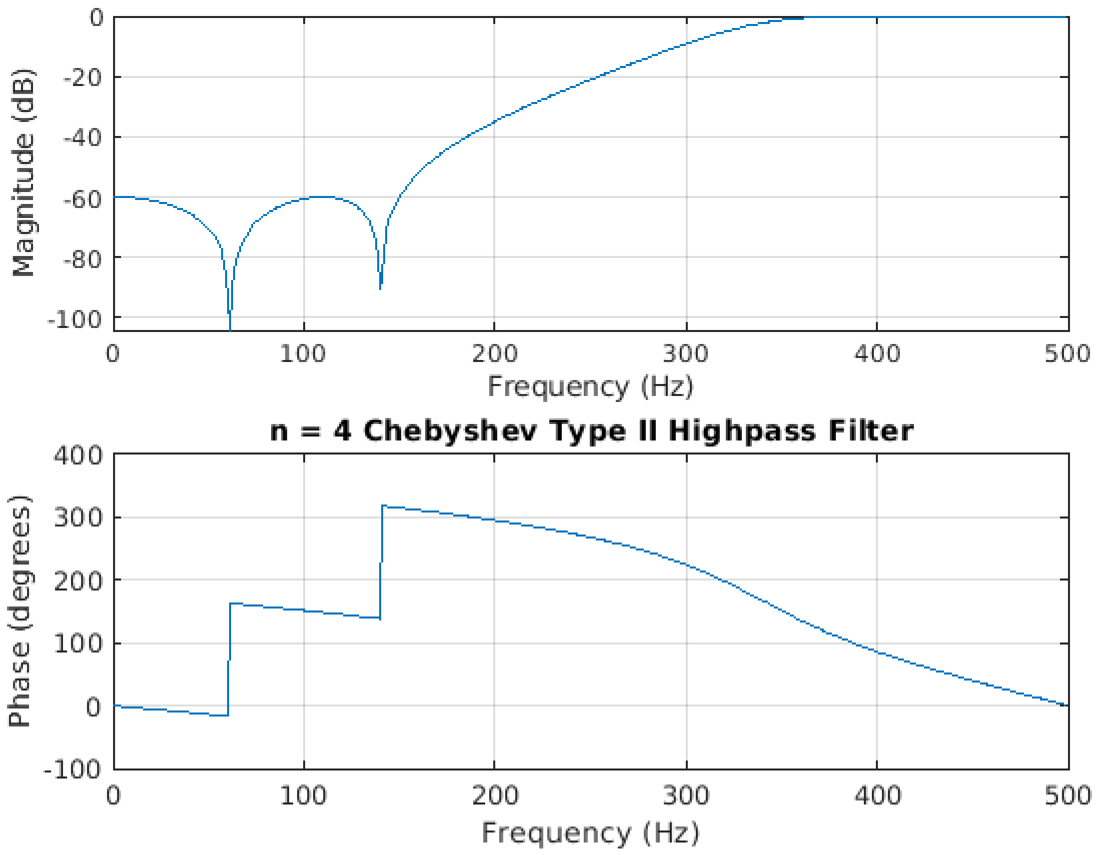


Figure 9.2: Chebyshev type 2 highpass filter plot

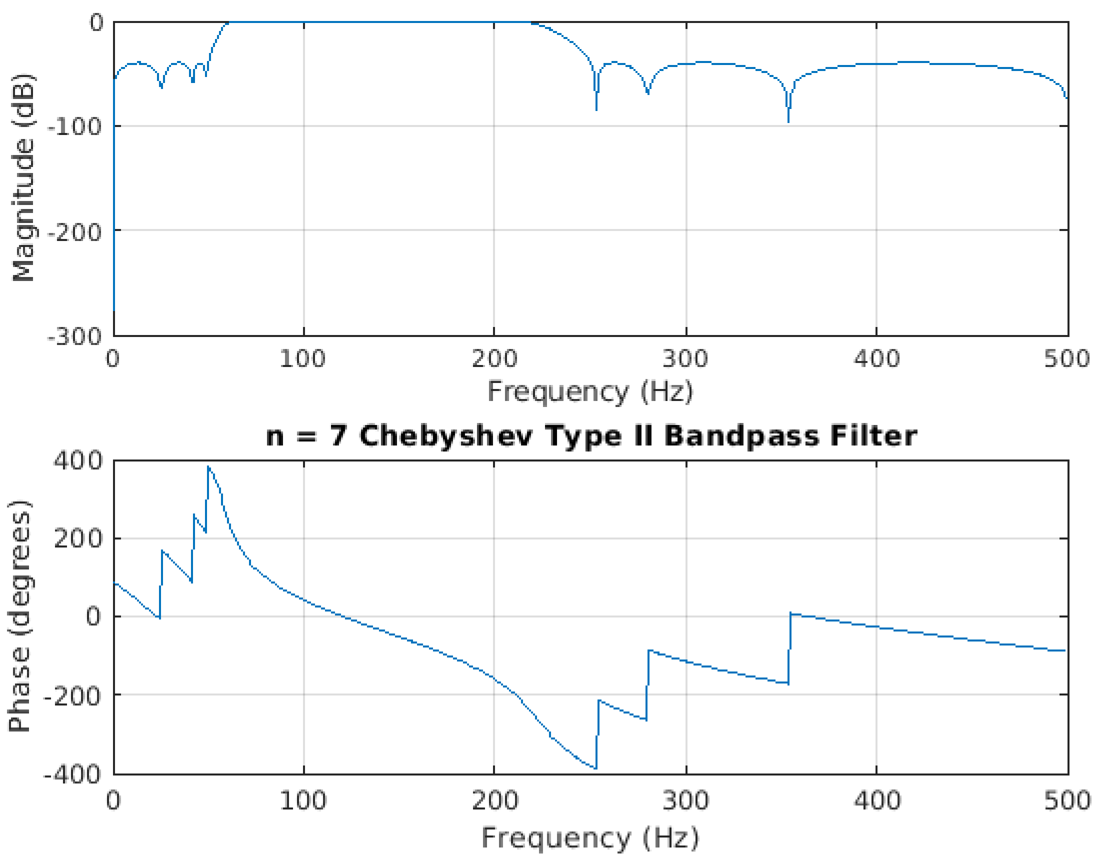


Figure 9.3: Chebyshev type 2 bandpass filter plot

9.9 Further Probing Experiments

Q1. Write a MATLAB program to design a digital high pass Chebyshev type-1 filter to satisfy the following: Pass band cutoff: $\Omega_p = 0.2\pi$ Pass band attenuation: AP=7dB Stop band cutoff: $\Omega_s = 0.3\pi$ Stop band attenuation: AP=7dB Use the bilinear transformation method. Assume T=1s.

Q2. Design a 5th-order analog low pass filter with a cutoff frequency of 2 GHz, 3 dB of pass band ripple and 30 dB of stop band attenuation using Butterworth, Chebyshev type 1 and type 2 methods.. Multiply by 2π to convert the frequency to radians per second. Compute the frequency response of the filter at 4096 points. Also Plot the attenuation in decibels. Express the frequency in gigahertz. Compare the filters.

LAB-9 FIR DIGITAL FILTER USING WINDOWS

10.1 Introduction

Filters are used in a wide variety of applications. Most of the time, the final goal of using a filter is to achieve a kind of frequency selectivity on the spectrum of the input signal. The window method for digital filter design is fast, convenient, and robust, but generally suboptimal. It is easily understood in terms of the convolution theorem for Fourier transforms, making it instructive to study after the Fourier theorems and windows for spectrum analysis.

10.2 Objective

10.2.1 Educational

To gain familiarity with designing of FI filtes using window techniques.

10.2.2 Experimental

To determine the order, cut-off frequency and transfer function of an FIR filter using window techniques.

10.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding filter designing.
2. Study the Background section below.

10.4 Equipment needed

1. Personal computer
2. MATLAB software

10.5 Background

Unlike discrete-time IIR filters which are generally obtained by transforming continuous time IIR systems, FIR filters are almost always implemented in discrete time: a desired magnitude response is approximated directly in the DT domain. In addition, a linear phase constraint is often imposed during FIR filter design. The window method consists of simply “windowing” a theoretically ideal filter impulse response h(n) by some suitably chosen window function w(n) , yielding

h(n)=hd(n).w(n)

Where hd(n) is desired impulse response of infinite duration and w(n) is a finite length window. There are many types of windows tat may be used in window filter design. They are rectangular window, triangular window, hamming window, hanning window, Blackman window, Kaiser window. However the desired frequency response of a filter designed with window method has to satisfy the following factors.
1.The width of the main lobe of window.
2.The peak side lobe amplitude level of window.
Ideally the main-lobe width should be narrow and peak side lobe amplitude should be small. However for a fixed length window, these can not be minimized independently. Some general properties of windows are as follows:
1. As the length of window sequence N increases, the width of main lobe decreases, which results in a decrease in transition width between pass band and stop band.
2. The peak side lobe amplitude is determined by the shape of the window, and it is essentially independent of window length.
3.If the window shape is changed to decrease the side lobe amplitude , the width of main lobe will generally increases.

10.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

10.7 Program

```
%FIR filters using windows
%rectangular
clc;
close all;
clear all;
format long;
rp=input('enter the passband ripple:(default:0.04)');
rs=input('enter the stopband ripple:(default: 0.05)');
fp=input('enter the passband frequency:(default:1500)');
fs=input('enter the stopband frequency:(default:2000)');
f=input('enter the sampling frequency:(default:9000)');
wp=2*(fp/f);
ws=2*(fs/f);
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2) =0)
```

```
n1=n;
n=n-1;
end;
y=boxcar(n1);
%Lowpass filter
b=fir1(n,wp,y);
    [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,1);
plot(o/pi,m);
ylabel('gain in db—-¿');
xlabel('Normalised frequency—-¿');
title('FIR filter using Rectangular window of LPF —-');
grid on;
%triangular or bartlett
clc;
close all;
clear all;
format long;
rp=input('enter the passband ripple:(default:0.04)');
rs=input('enter the stopband ripple:(default:0.02)');
fp=input('enter the passband frequency:(default:1500)');
fs=input('enter the stopband frequency:(default:2000)');
f=input('enter the sampling frequency:(default:8000)');
wp=2*(fp/f);
ws=2*(fs/f);
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2) =0)
n1=n;
n=n-1;
end;
y=triang(n1);
%Lowpass filter
b=fir1(n,wp,y);
    [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,1);
plot(o/pi,m);
ylabel('gain in db—-¿');
xlabel('Normalised frequency—-¿');
title('FIR filter using Triangular window of LPF —-');
grid on;
%Highpass filter
b=fir1(n,wp,'high',y); [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,2);
plot(o/pi,m);
ylabel('gain in db—-¿');
```

```
xlabel('Normalised frequency—-¿');
title('FIR filter using Triangular window of HPF —-');
grid on ;
%hamming
clc;
close all;
clear all;
format long;
rp=input('enter the passband ripple:(default:0.04)');
rs=input('enter the stopband ripple:(default: 0.05)');
fp=input('enter the passband frequency:(default:1500)');
fs=input('enter the stopband frequency:(default:2000)');
f=input('enter the sampling frequency:(default:9000)');
wp=2*(fp/f);
ws=2*(fs/f);
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2) =0)
n1=n;
n=n-1;
end;
y=hamming(n1);
%Lowpass filter
b=fir1(n,wp,y);
   [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,1);
plot(o/pi,m);
ylabel('gain in db—-¿');
xlabel('Normalised frequency—-¿');
title('FIR filter using Rectangular window of LPF —-');
grid on;
%Highpass filter
b=fir1(n,wp,'high',y); [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,2);
plot(o/pi,m);
ylabel('gain in db—-¿');
xlabel('Normalised frequency—-¿');
title('FIR filter using Rectangular window of HPF —-');
title('Magnitude response of high pass filter');
grid on;
%hanning
clc;
close all;
clear all;
format long;
rp=input('enter the passband ripple:(default:0.04)');
rs=input('enter the stopband ripple:(default: 0.05)');
```

```
fp=input('enter the passband frequency:(default:1500)');
fs=input('enter the stopband frequency:(default:2000)');
f=input('enter the sampling frequency:(default:9000)');
wp=2*(fp/f);
ws=2*(fs/f);
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2) =0)
n1=n;
n=n-1;
end;
y=hanning(n1);
%Lowpass filter
b=fir1(n,wp,y); [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,1);
plot(o/pi,m);
ylabel('gain in db—-¿');
xlabel('Normalised frequency—-¿');
title('FIR filter using Rectangular window of LPF —-');
grid on;
%Highpass filter
b=fir1(n,wp,'high',y);
    [h,o]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,1,2);
plot(o/pi,m);
ylabel('gain in db—-¿');
xlabel('Normalised frequency—-¿');
title('FIR filter using Rectangular window of HPF —-');
title('Magnitude response of high pass filter');
grid on;
```

10.8 Output and waveforms

```
enter the passband ripple:(default:0.04)
0.04
enter the stopband ripple:(default:0.02)
0.02
enter the passband frequency:(default:1500)
1500
enter the stopband frequency:(default:2000)
2000
enter the sampling frequency:(default:8000)
8000
```

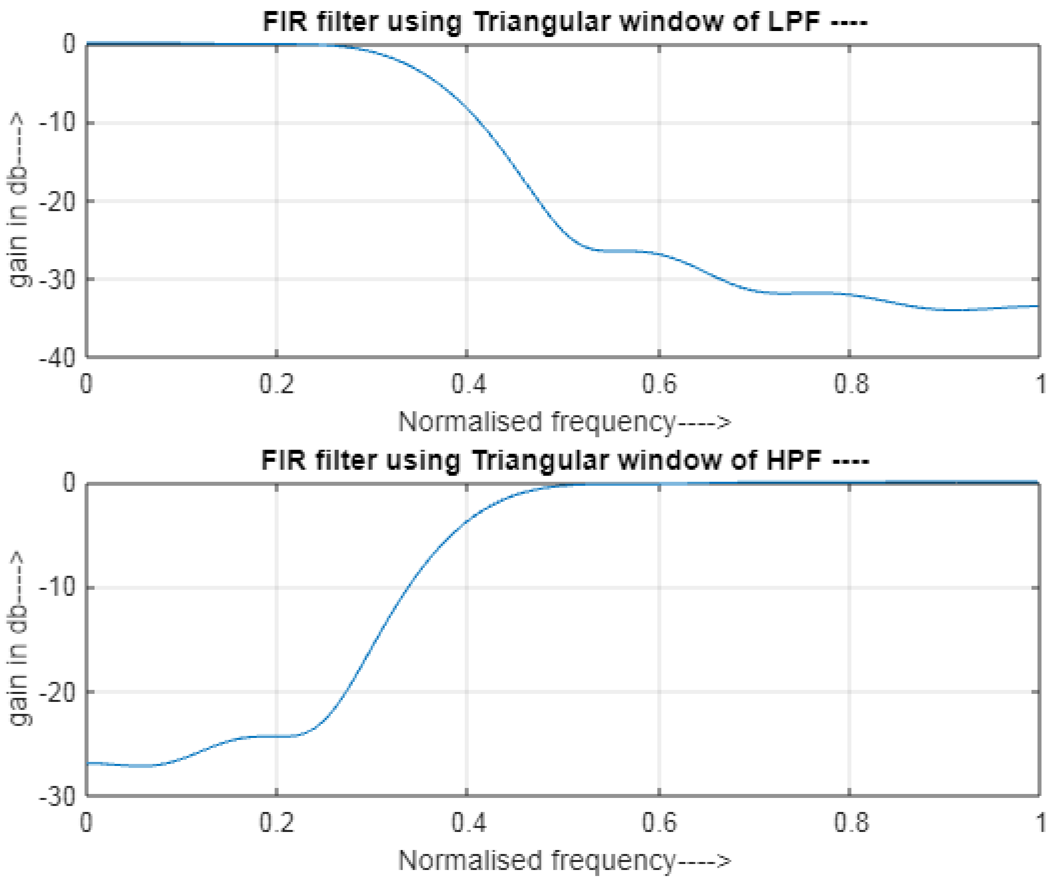


Figure 10.1: FIR low pass and high pass filter plot using windows

10.9 Further Probing Experiments

Q1. Design a HPF of length 7 with cut off frequency of 2 rad/sec using Hamming window. Plot the magnitude and phase response.
Q2. Design a high-pass filter with fs=200Hz and fp=300Hz which exhibits attenuation greater than 40dB in the stop-band. We need the pass-band ripple to be less than 0.2dB. Assume that the sampling frequency, fs, is 1200Hz.

LAB-10 FIR DIGITAL FILTER USING FREQUENCY SAMPLING METHOD

11.1 Introduction

The finite impulse response (FIR) filter is one of the most basic elements in a digital signal processing system, and it can guarantee a strict linear phase frequency characteristic with any kind of amplitude frequency characteristic. Besides, the unit impulse response is finite; therefore, FIR filters are stable system The frequency sampling method is use to design recursive and non-recursive FIR filters for both standard frequency selective filters and with arbitrary frequency response. The main idea of the frequency sampling design method is that a desired frequency response can be approximated by sampling it at N evenly spaced points and then obtaining N-point filter response.

11.2 Objective

11.2.1 Educational

The objective of this laboratory exercise are to gain familiarity with designing of FIR filters using frequency sampling technique.

11.2.2 Experimental

To determine the frequency response and transfer function of an FIR filter using frequency sampling technique.

11.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding filter designing.
2. Study the Background section below.

11.4 Equipment needed

1. Personal computer
2. MATLAB software

11.5 Background

The frequency sampling method is use to design recursive and non-recursive FIR filters for both standard frequency selective filters and with arbitrary frequency response. The main idea of the frequency sampling design method is that a desired frequency response can be approximated by sampling it at N evenly spaced points and then obtaining N-point filter response.

A continuous frequency response is then calculated as an interpolation of the sampled frequency response. The approximation error would then be exactly zero at the sampling frequencies and would be finite in frequencies between them. The smoother the frequency response being approximated, the smaller will be the error of interpolation between the sample points.

There are two distinct types of Non-Recursive Frequency Sampling method of FIR filter design, depending on where the initial frequency sample occurred. The type 1 designs have the initial point at $\Omega = 0$, whereas the type 2 designs have the initial point at $f = 1/2N or \Omega = \pi/N$.
Procedure for Type-1 Design:
1) Choose the desired frequency response Hd(Ω)
2) Sample Hd(w) at N -points and generate the sequence H(k).
3) The N-point inverse DFT of the sequence H(k) gives the impulse response of the filter h(n). For practical realization of the filter, samples of impulse response should be real. This can happen if all the complex terms appear in conjugate pairs.Unlike the window method, this technique can be used for any given magnitude response.
This method is useful for the design of non-prototype filters where the desired magnitude response can take any irregular shape. Major advantage of Frequency sampling method lies in the efficient frequency sampling structure, which is obtained when most of the frequency samples are zero. One disadvantage with this method is that the frequency response obtained by interpolation is equal to the desired frequency response only at the sampled points. At the other points, there will be a finite error present.

11.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

11.7 Program

```
% response of low-pass filter using frequency sampling method clc;
clear all;
close all;
N=20;
alpha=(N-1)/2;
k=0:N-1;
wk=2*pi*k/N;
% desired ideal frequency response of the filter
Hr=[1,1,1,zeros(1,15),1,1];
n1=0:floor((N-1)/2);
n2=floor((N-1)/2)+1:N-1;
angH=[-alpha*2*pi*n1/N,alpha*2*pi*(N-n2)/N];
```

```
H=Hr.*exp(i*angH);
% calculating desired impulse response
h=real(ifft(H,N));
% calculating the frequency response with b=h and a=1 for FIR filters
    [h1,w]=freqz(h,1);
subplot(3,1,1);
stem(Hr);
title('frequency samples');
subplot(3,1,2);
stem(h);
title('impulse samples');
subplot(3,1,3);
plot(20*log10(abs(h1)));
title('magnitude response of LPF');
```

11.8 Output and waveforms

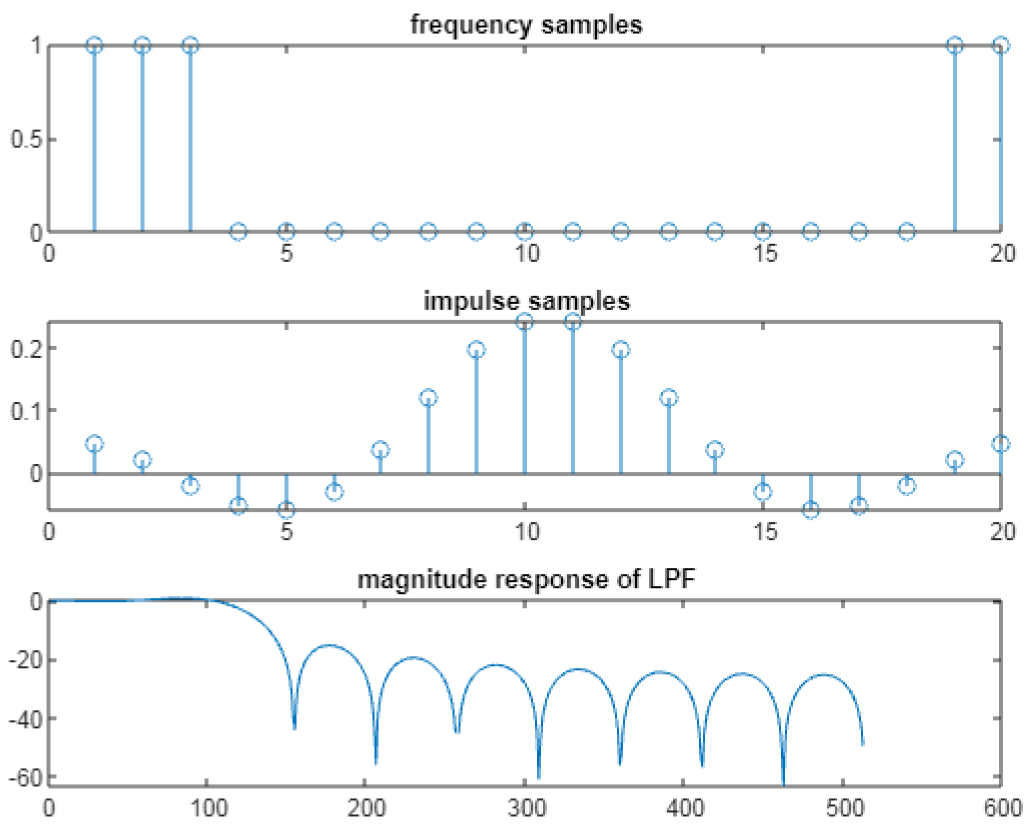


Figure 11.1: FIR low pass and high pass filter plot usingfrequency sampling method

11.9 Further Probing Experiments

Q1. Use the frequency sampling method to design a 9-tap highpass FIR filter with a cutoff frequency of 0.25 pi radians per sample.

Q2. Use the frequency sampling method to design a 25-tap bandpass FIR filter with a cutoff frequency of 0.3 pi radians per sample.

LAB-11 OPTIMUM EQUIRIPPLE FIR DIGITAL FILTER

12.1 Introduction

Design of FIR filters by windowing is straightforward. The window method although simple does not provide for individual control over approximation errors in different bands. Frequency selective filters designed by windowing often have the property that the error is greatest on either side of a discontinuity of the ideal frequency response, and the error becomes smaller for frequencies away from the discontinuity. The optimal method is hence based on the concept of equiripple pass band and stop band.

12.2 Objective

12.2.1 Educational

The objective of this laboratory exercise is to gain familiarity with designing of optimum equiripple FIR filters using Parks-McClellan algorithm.

12.2.2 Experimental

To determine the frequency response of an optimum FIR filter using Parks-McClellan algorithm.

12.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding filter designing.
2. Study the Background section below.

12.4 Equipment needed

1. Personal computer
2. MATLAB software

12.5 Background

Design of FIR filters by windowing is straightforward. However, we wish to design a filter that is the “best” that can be achieved for a given value of ‘M’. We know that the rectangular window provides the best mean-square approximation to the desired-frequency for a given value of M. However this approximation shows adverse behavior at discontinuities of Hd(w).The window method although simple does not provide for individual control over approximation errors in different bands. Frequency selective filters designed by windowing often have the property that the error is greatest on either side of a discontinuity of the ideal frequency response, and the error becomes smaller for frequencies away from the discontinuity. But, if the ripples are distributed more evenly over the pass band and stop band, a better approximation of the desired frequency response can be achieved. The optimal method is hence based on the concept of equiripple pass band and stop band. In the optimal method, the objective is to determine the filter coefficients, h(n), such that the value of the maximum weighted error, is minimized in the pass band and stop band, i.e. Algorithmic techniques in designing filters to meet the above mentioned specifications must systematically vary the impulse response h[n] to obtain the best results. Design algorithms have been developed in which some parameters (M, ws, wp, ds and dp) are fixed and an iterative procedure is used to obtain the optimum adjustments of others. The most dominant of the optimum solution generation algorithms is the Parks-McClellan algorithm. The Parks-McClellan algorithm is based on reformulating the filter design problem as a problem in polynomial approximation.The M-file function remez in the Signal Processing Toolbox of Matlab implements the Parks-McClellan Algorithm, which uses the Remez exchange algorithm and Chebyshev approximation to arrive at the desired filter.More about the Remez exchange algorithm could be found in "Digital Filter Design" by T.W. Parks and C.S Burrus. The filters designed are optimal in the sense that they minimize the maximum error between the desired and actual frequency responses. They are sometimes called minmax filters.

12.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the 'FILE' Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

12.7 Program

```
%LPF
% Use the Parks-McClellan algorithm to design an FIR lowpass filter of order 17. Specify normalized stopband frequency of 0.4 pi rad/sample and normalized passband frequency of 0.3 pi rad/sample. Plot the ideal and actual magnitude responses.
clc;
clear all;
close all;
f = [0, 0.3 ,0.4, 0.6, 0.7 ,1];
a = [1, 1, 0, 0, 0, 0];
b = firpm(17,f,a);
    [h,w] = freqz(b,1,512);
plot(f,a,w/pi,abs(h))
legend('Ideal','firpm Design')
xlabel ('Radian Frequency'), ylabel( 'Magnitude');
    %HPF
clc;
clear all;
```

```
close all;
f = [0 0.3 0.4 0.6 0.7 1];
a = [ 0 0 0 0 1 1];
b = firpm(17,f,a);
    [h,w] = freqz(b,1,512);
plot(f,a,w/pi,abs(h))
legend('Ideal','firpm Design')
xlabel 'Radian Frequency'), ylabel ('Magnitude');
%BPF % Use the Parks-McClellan algorithm to design an FIR bandpass filter of order 17. Spec-
ify normalized stopband frequencies of 0.3 pi and 0.7 pi rad/sample and normalized passband
frequencies of 0.4 pi and 0.6 pi rad/sample. Plot the ideal and actual magnitude responses.
clc;
clear all;
close all;
f = [0 0.3 0.4 0.6 0.7 1];
a = [0 0 1 1 0 0];
b = firpm(17,f,a);
%Parks-McClellan algorithm to design an FIR bandpass filter [h,w] = freqz(b,1,512);
plot(f,a,w/pi,abs(h))
legend('Ideal','firpm Design')
xlabel ('Radian Frequency'), ylabel ('Magnitude');
    %BSF
clc;
clear all;
close all;
f = [0 0.3 0.4 0.6 0.7 1];
a = [ 1 1 0 0 1 1];
b = firpm(17,f,a);
    [h,w] = freqz(b,1,512);
plot(f,a,w/pi,abs(h));
legend('Ideal','firpm Design');
xlabel ('Radian Frequency'), ylabel( 'Magnitude');
title(['magnitude response of band stop FIR optimum equiripple filter using ' Parks-McClellan
algorithm ']);
```

12.8 Output and waveforms

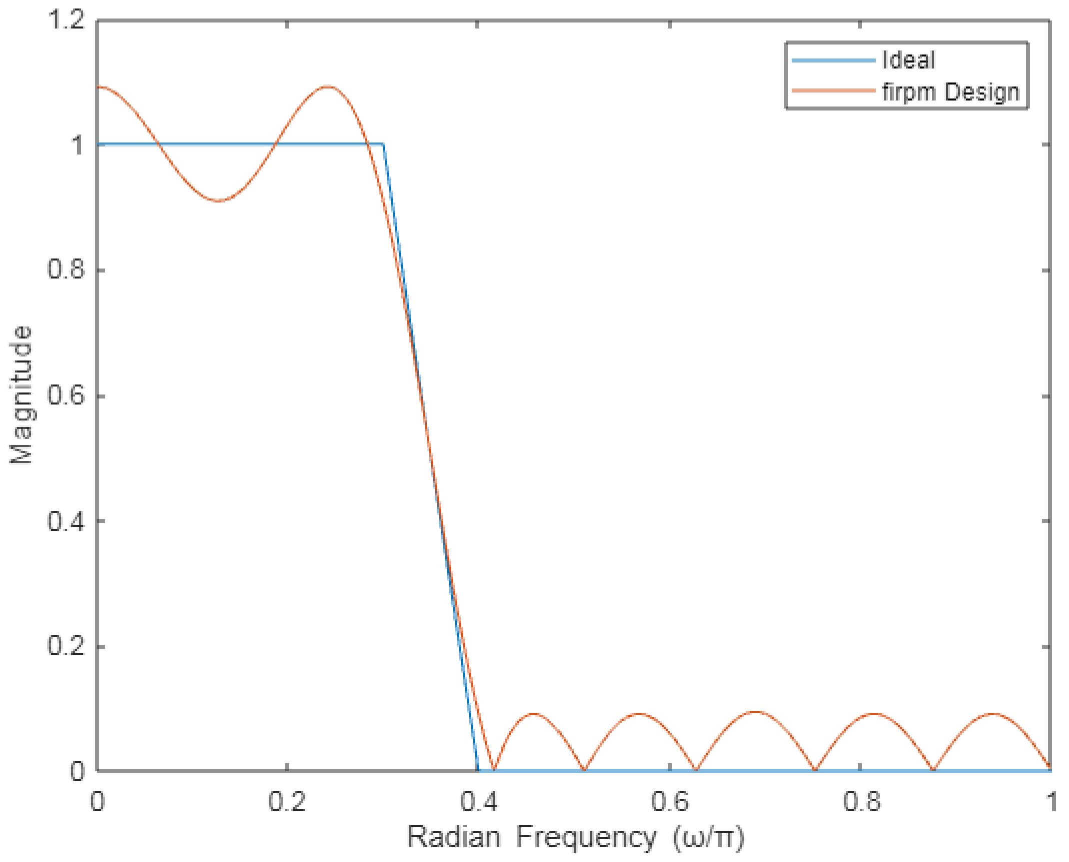


Figure 12.1: Optimum FIR low pass filter plot using Parks-McClellan algorithm

12.9 Further Probing Experiments

Q1. Use the Parks-McClellan algorithm to design an FIR band stop filter of order 21. Specify normalized stop band frequencies of 0.25πi and 0.65π rad/sample and normalized pass band frequencies of 0.45π and 0.55π rad/sample. Plot the ideal and actual magnitude responses.

Q2. Use the Parks-McClellan algorithm to design an FIR low pass filter of order 35. Specify normalized stopband frequency of 0.5 πrad/sample and normalized passband frequency of 0.4π rad/sample. Plot the ideal and actual magnitude responses.

LAB-12 DTMF TONE GENERATION AND DETECTION

13.1 Introduction

Dual Tone Multi Frequency (DTMF) is a system of signal tones used in the field of communication whose application ranges from voice mail and help desks to telephone banking and controlling robotics designs [1]. The DTMF signal is generated by the sum of two sinusoidal tones. One of them is selected from a group of 697 Hz, 770 Hz, 852 Hz, 941 Hz named as low frequency group and the second one is selected from a set of 1209 Hz, 1336 Hz, 1477 Hz, 1633 Hz called high frequency group. By addition of two sinusoidal tones, four frequencies from each group gives a total of 16 combinations, which represented ten decimal digits, four alphabet characters and two special characters.

13.2 Objective

13.2.1 Educational

The objective of this laboratory exercise is to gain familiarity with generation and detection of DTMF tone using Goertzel Algorithm in MATLAB software.

13.2.2 Experimental

To find frequency deviation, twist, energy and time duration tests on the DTMF signals. The algorithm recognizes the DTMF tones if they satisfy the recommendations, otherwise they are rejected.

13.3 Prelab Preparation:

1. Read the chapter in your textbook on DTMF detection.
2. Study the Background section below.

13.4 Equipment needed

1. Personal computer
2. MATLAB software

13.5 Background

DTMF is a method of representing digits with tones for communications . DTMF tones are used by all touch tone phones to represent the digits on a touch tone keypad. DTMF signaling used for many applications such as telephone banking, routing customer support calls ,system control, voicemail, and similar applications . A DTMF signal represents one of sixteen touchtone symbols

(0-9, A-D, #, *) as shown in figure 1. Each symbol is represented by one of four frequencies in a low frequency band and one of four frequencies in a higher frequency band. The symbols are shown in a matrix format. The columns are represented by frequencies in a band between 1 kHz (kilohertz) and 2 kHz, and the rows are represented by frequencies in a band between 500 Hz and 1 kHz. Whenever a key of a touchtone keypad is depressed, the DTMF is generated by adding two signals and transmitted to device. The device that receives this dual tone signal must detect which one of the four low frequencies and which one of the four high frequencies has been received to determine which symbol is to be determined .In communications, Discrete Fourier Transform (DFT) is used for frequency detection. DFT converts the signal from time domain to frequency domain. Goertzel Algorithm Goertzel algorithm is used widely and it is most popular method in the worldwide. This algorithm uses IIR filters that are tuned to eight DTMF frequencies.

13.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the ‘FILE’ Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

13.7 Program

```
% DTMF tone generator
clc;
close all;
clear all
fs=8000;
t=[0:1:204]/fs;
x=zeros(1,length(t)); x(1)=1;
y852=filter([0 sin(2*pi*852/fs) ],[1 -2*cos(2*pi*852/fs) 1],x);
y1209=filter([0 sin(2*pi*1209/fs) ],[1 -2*cos(2*pi*1209/fs) 1],x);
y7=y852+y1209;
subplot(2,1,1);plot(t,y7);grid
ylabel('y(n) DTMF: number 7');
xlabel('time (second)');
title('signal tone number 7')
Ak=2*abs(fft(y7))/length(y7);Ak(1)=Ak(1)/2;
f=[0:1:(length(y7)-1)/2]*fs/length(y7);
subplot(2,1,2);plot(f,Ak(1:(length(y7)+1)/2));grid
ylabel('Spectrum for y7(n)');
xlabel('frequency (Hz)');
title('absolute value of signal tone number 7')
```

13.8 Output and waveforms

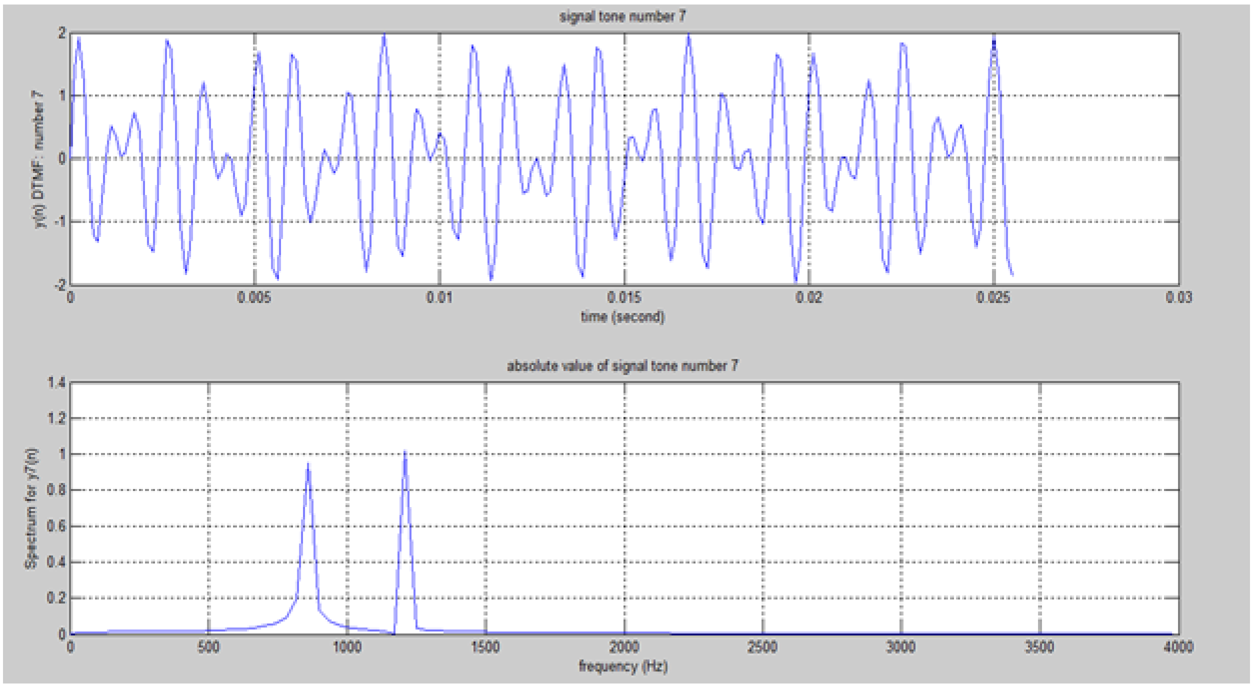


Figure 13.1: DTMF generation

13.9 Further Probing Experiments

Q1. Generate and detect DTMF signal tone '5' using the Goertzel algorithm.

Q2. Design a touch-tone telephone using dual tone multi frequency (DTMF) scheme to encode in MATLAB.

LAB-13 SAMPLING RATE CONVERSION

14.1 Introduction

"Multirate" means "multiple sampling rates". A multirate DSP system uses multiple sampling rates within the system. Whenever a signal at one rate has to be used by a system that expects a different rate, the rate has to be increased or decreased, and some processing is required to do so. Therefore "Multirate DSP" refers to the art or science of changing sampling rates. "Resampling" means combining interpolation and decimation to change the sampling rate by a rational factor. Resampling is done to interface two systems with different sampling rates.

14.2 Objective

14.2.1 Educational

The objective of this laboratory exercise is to gain familiarity with designing of multirate signal processing using decimation and interpolation.

14.2.2 Experimental

To determine the sampling rate of a multirate signal processing system.

14.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding multirate signal processing.
2. Study the Background section below.

14.4 Equipment needed

1. Personal computer
2. MATLAB software

14.5 Background

"Decimation" is the process of reducing the sampling rate. "Downsampling" is a more specific term which refers to just the process of throwing away samples, without the lowpass filtering operation. The most immediate reason to decimate is simply to reduce the sampling rate at the output of one system so a system operating at a lower sampling rate can input the signal. But a much more common motivation for decimation is to reduce the cost of processing: the calculation and/or memory required to implement a DSP system generally is proportional to the sampling rate, so the use of a lower sampling rate usually results in a cheaper implementation.

“Upsampling” is the process of inserting zero-valued samples between original samples to increase the sampling rate. (This is called “zero-stuffing”.) “Interpolation”, is the process of upsampling followed by filtering. The filtering removes the undesired spectral images. The primary reason to interpolate is simply to increase the sampling rate at the output of one system so that another system operating at a higher sampling rate can input the signal.

14.6 Procedure

1. Click on the MATLAB Icon on the desktop.
2. MATLAB window opens.
3. Click on the ‘FILE’ Menu on menu bar.
4. Click on NEW M-File from the file Menu.
5. An editor window open, start typing commands.
6. Now SAVE the file in directory.
7. Then Click on DEBUG from Menu bar and Click Run.

14.7 Program

```
clc;
close all;
clear all;
L = input('Enter Up-sampling factor :');
M = input('Enter Down-sampling factor :');
N = input('Enter number of samples :');
n = 0:N-1;
x = sin(2*pi*0.43*n) + sin(2*pi*0.31*n);
y = resample(x,L,M);
subplot(2,1,1); stem(n,x(1:N));
axis([0 29 -2.2 2.2]);
title('Input Sequence');
xlabel('Time index n'); ylabel('Amplitude');
subplot(2,1,2);
m = 0:(N*L/M)-1;
stem(m,y(1:N*L/M));
axis([0 (N*L/M)-1 -2.2 2.2]);
title('Output Sequence');
xlabel('Time index n'); ylabel('Amplitude');
```

14.8 Output and waveforms

```
Enter Up-sampling factor :
6 Enter Down-sampling factor :
3
Enter number of samples :
200
```

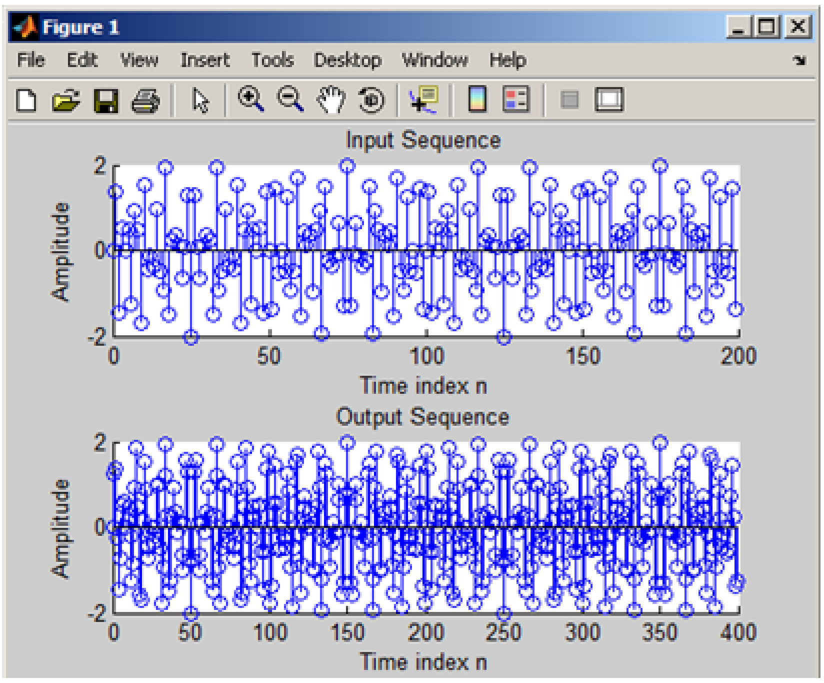


Figure 14.1: Multirate sampling by a factor of I/D

14.9 Further Probing Experiments

Q1. Perform I/D sampling rate conversion on the input signal x(n)=cos(2*pi*30*n)for different sampling rates.

Q2. For the given data sequence x(n)=1,4,5,6,7,8,9, find the the output sequence which is up sampled by 2.

LAB-14 SINEWAVE GENERATION

15.1 Introduction

The sine wave has found its usage in various applications, including factory testing for connectivity. One method of sine-wave generation is based on a positive feedback system that employs the principle of oscillation. Oscillation can only be achieved if the system satisfies the Bakhausen Criterion. Following the principles of an analog oscillator, a digital oscillator is implemented using a special two-pole bandpass filter.

15.2 Objective

15.2.1 Educational

The objective of this laboratory exercise is to gain familiarity with generating of sine wave using DSP processor.

15.2.2 Experimental

To determine the sinusoidal coefficients using MATLAB and implement sine wave generation on DSP processor.

15.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding filter designing.
2. Study the Background section below.

15.4 Equipment needed

1. TMS 320C6713 Kit
2. MATLAB software
3. RS232 Serial Cable
4. Software – CCStudio v3.1
5. Personal computer

15.5 Background

The basic idea is to keep track of the starting and present phases of the waveform so that a continuous-phase sine wave can be obtained. By knowing or remembering the previous phase that has been output, the next amplitude to be output is then calculated. The next phase (that corresponds to the next amplitude) is incremented according to the desired output frequency. With this calculated new phase, the amplitude can be read off the look-up table and output to the CODEC. Assume that the starting phase of the output signal to be stored in data memory is InitPhase. The phase step of the desired frequency is stored in Phase Step. The magnitude of the phase step is calculated . Phase Step Desired frequency Sampling frequency 2 2f Therefore, if the desired frequency is 10 kHz with a sampling frequency of 44.1 kHz, the phase step is: PhaseStep 10000 44100 2 0.4535 (radians) 81.6 degrees. The amplitude to be output at every sample interval is sin(0.4535 pi n), where n = 0, 1, 2,. . . . Therefore, the phase of the next output sample is: Present Phase to be output = Previous phase + Phase Step. If the starting phase of the signal is 0 degree, then the next phase is 81.6 degrees, 163.2 degrees, and so on. These phase angles are used for calculating which amplitude to read from the cosine look-up table.

15.6 Procedure

1. Connect CRO to the LINE OUT socket
2. Now switch ON the DSK and bring up Code Composer Studio on PC
3. Create a new project with name sinewave.pjt
4. From File menu select New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as “ sinewave.cdb”
5. Add sinewave.cdb to the current project
6. Create a new source file and save it as sinewave.c
7. Add the source file sinewave.c to the project
8. Add the library file “dsk6713bsl.lib” to the project (Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.li
9. Copy files “dsk6713.h” and “dsk6713_aic23.h” to the Project folder (Path: C:\CCStudio v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.out file))
11. Run the program (F5)
12. Observe the waveform that appears on the CRO screen and ccstudio simulator.

15.7 Program

```
% matlab code to generate the sine values of the look table
n=1:48;
x=sin(2*pi*n*1000/48000);
x1=round(x*2î5);
/ c program for generation of sine wave using c6713 DSK
#include ”sinewavecfg.h”
#include ”dsk6713.h”
#include ”dsk6713_aic23.h”
```

```
short loop=0;
short gain=1;
Int16 outbuffer[256];
const short BUFFERLENGTH=256;
int i=0;
DSK6713_AIC23_Config config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
Int16 sine_table[48]={4277,8481,12540,16384,19948,23170,25997,28378,30274,31651,32488, 32766,32488,31651,3
4277,-8481, -12540,-16384,-19948,-23170,-25997,-28378,-30274,-31651,-32488,-32766,-32488,-31651,
-30274,-28378,-25997,-23170,-19948,-16384,-12540,-8481,-4277,0};
Uint32 fs=DSK6713_AIC23_FREQ_48KHZ;
void main()
{ DSK6713_AIC23_CodecHandle hCodec;
DSK6713_init();
hCodec=DSK6713_AIC23_openCodec(0,& config);
DSK6713_AIC23_setFreq(hCodec, fs);
while(1)
{
outbuffer[i]=sine_table[loop];
while(!DSK6713_AIC23_write(hCodec, sine_table[loop])*gain);
i++;
if(i==BUFFERLENGTH) i=0;
if(++loop¿47) loop=0;
}
}
```

15.8 Output and waveforms

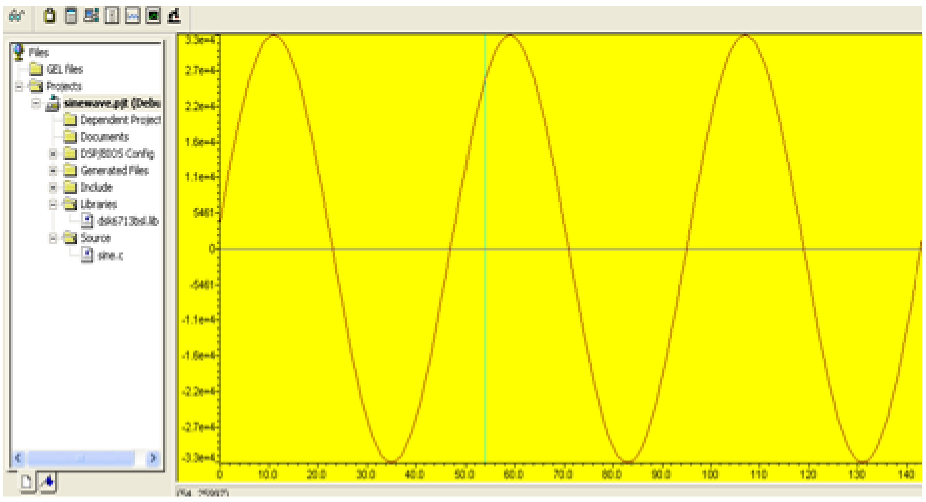


Figure 15.1: Sinewave generation plot in CCS studio simulator

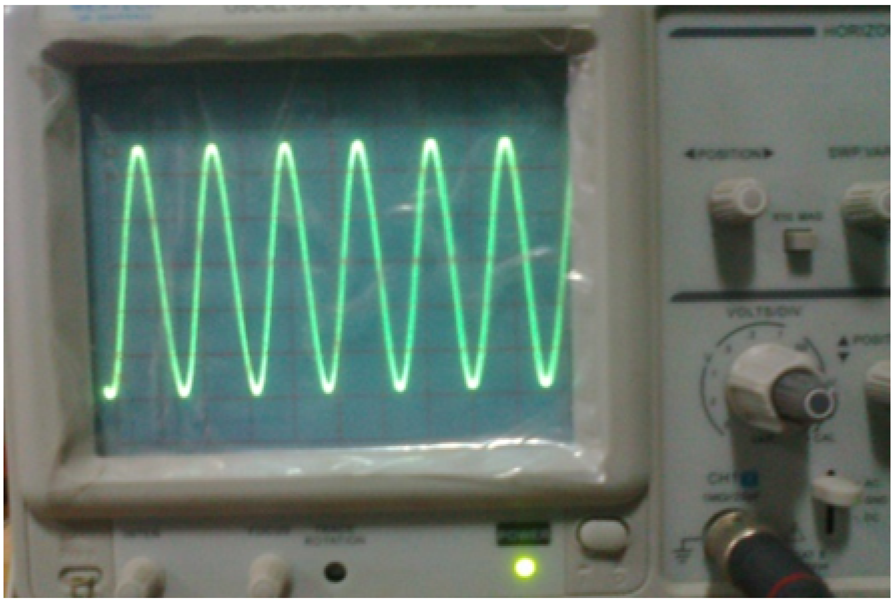


Figure 15.2: Sinewave generation plot in CRO

15.9 Further Probing Experiments

Q1. Generate cosine wave using TMS320C6713 DSP processor.

Q2. Generate the signal x(n)=sin(3π)+2cos(3π)using DSP processor.

LAB-15 IIR AND FIR FILTERS USING DSP KITS

16.1 Introduction

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying). IIR can be unstable, whereas FIR is always stable. IIR, when compared to FIR, can have limited cycles, but FIR has no limited cycles. IIR is derived from analog, whereas FIR has no analog history. IIR filters consist of zeros and poles, and require less memory than FIR filters, whereas FIR only consists of zeros.

16.2 Objective

16.2.1 Educational

The objective of this laboratory exercise is to gain familiarity with designing of IIR and FIR filters using DSP kits.

16.2.2 Experimental

To design and observe the characteristics of IIR and FIR filters using DSKC6713 processor.

16.3 Prelab Preparation:

1. Review the sections of your digital signal processing(AECB23) textbook regarding filter designing.
2. Study the Background section below.

16.4 Equipment needed

1. TMS 320C6713 Kit
2. Power cord
3. RS232 Serial Cable
4. Software – CCStudio v3.1
5. Personal computer

16.5 Background

In signal processing, a finite impulse response (FIR) filter is a filter whose impulse response (or response to any finite length input) is of finite duration, because it settles to zero in finite time. This is in contrast to infinite impulse response (IIR) filters, which may have internal feedback and may continue to respond indefinitely (usually decaying). IIR can be unstable, whereas FIR is always stable. IIR, when compared to FIR, can have limited cycles, but FIR has no limited cycles. IIR is derived from analog, whereas FIR has no analog history. IIR filters consist of zeros and poles, and require less memory than FIR filters, whereas FIR only consists of zeros. Historically, digital IIR filters have been derived from their analog counterparts. There are several common types of analog filters: Butterworth which have maximally flat passbands in filters of the same order, Chebyshev type I which are equiripple in the passband, Chebyshev type II which are equiripple in the stopband, and Elliptic filters which are equiripple in both the passband and the stopband. The digital version of these can be obtained from analog designs through the bilinear transformation and impulse invariance techniques.FIR filters have been designed with windows , Fourier series or frequency sampling methods.

16.6 Procedure

1. Connect CRO to the LINE OUT socket
2. Now switch ON the DSK and bring up Code Composer Studio on PC
3. Create a new project with name sinewave.pjt
4. From File menu select New->DSP/BIOS Configuration->Select dsk6713.cdb and save it as " sinewave.cdb"
5. Add sinewave.cdb to the current project
6. Create a new source file and save it as sinewave.c
7. Add the source file sinewave.c to the project
8. Add the library file "dsk6713bsl.lib" to the project (Path: C:\CCStudio\C6000\dsk6713\lib\dsk6713bsl.li
9. Copy files "dsk6713.h" and "dsk6713_aic23.h" to the Project folder (Path: C:\CCStudio v3.1\C6000\dsk6713\include)
10. Build (F7) and load the program to the DSP Chip (File->Load Program(.out file))
11. Run the program (F5)
12. Observe the waveform that appears on the CRO screen and ccstudio simulator.

16.7 Program

```
 // c program for generation of fir filter using c6713 DSK
#include "firfiltercfg.h"
#include "dsk6713.h"
#include "dsk6713_aic23.h"
#include "stdio.h"
Float filter_coeff[ ]={-0.020203,-0.016567,0.009656,0.027335,0.011411,-0.023194,- 0.033672,0.000000,0.043293,0.0
0.025105,-0.082004,-0.041842,0.115971,0.303048, 0.386435,0.303048,0.115971,-0.041842,-0.082004,-
0.025105,0.038657,0.043293,0.000000,-0.033672,-0.023194,0.011411,0.027335,0.009656,-0.016567,-
0.020203};
```

```
/FIR Low pass Rectangular Filter pass band range 0-1500Hz
DSK6713_AIC23_Config config={0x0017,0x0017,0x00d8,0x00d8,0x0011,0x0000,0x0000,0x0043,0x0081,0x0001};
void main()
{
DSK6713_AIC23_CodecHandle hCodec;
Uint32 l_input, r_input,l_output, r_output;
DSK6713_init();
hCodec = DSK6713_AIC23_openCodec(0, &config);
DSK6713_AIC23_setFreq(hCodec, 1);
while(1)
{
while(DSK6713_AIC23_read(hCodec, &l_input));
while(DSK6713_AIC23_read(hCodec, &r_input));
l_output=(Int16)FIR_FILTER(&filter_coeff ,l_input);
r_output=l_output;
while(DSK6713_AIC23_write(hCodec, l_output));
while(DSK6713_AIC23_write(hCodec, r_output));
} DSK6713_AIC23_closeCodec(hCodec);
}
signed int FIR_FILTER(float * h, signed int x)
{
int i=0;
signed long output=0;
static short int in_buffer[100];
in_buffer[0] = x;
for(i=30;i¿0;i–)
in_buffer[i] = in_buffer[i-1];
for(i=0;i¡32;i++)
output = output + h[i] * in_buffer[i];
return(output);
}
```

16.8 Output and waveforms

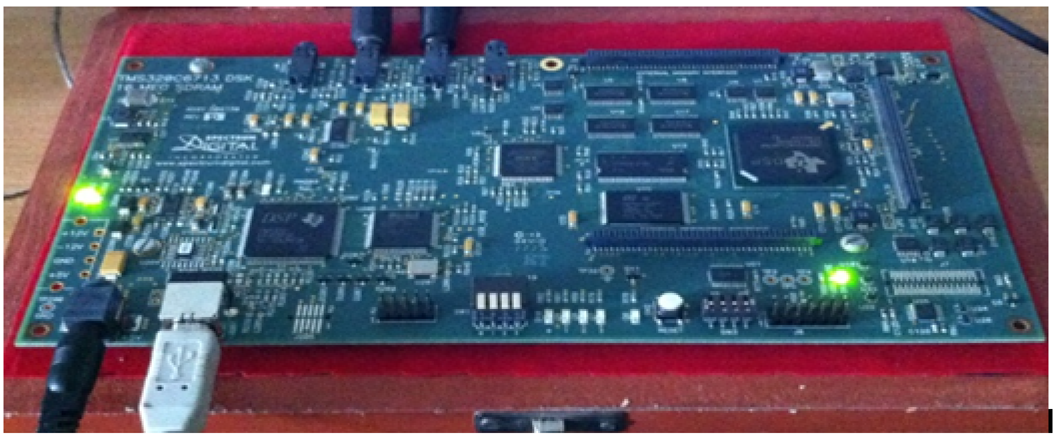


Figure 16.1: TMS320C6713 DSP processor

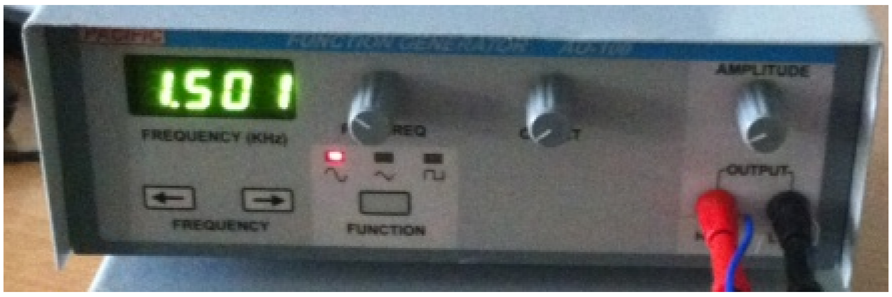


Figure 16.2: Input signal frequency

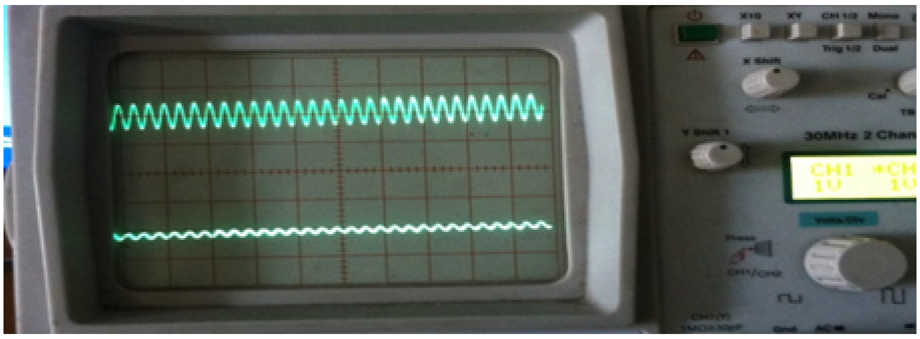


Figure 16.3: Low pass filter plot using C6713 DSK processor

16.9 Further Probing Experiments

Q1. Design 3th order High pass IIR filter using Chebyshev filter.

Q2. Design 7th order low pass IIR filter using Butterworth method.

Appendix A - Introduction to MATLAB

MATLAB is a software package for high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. At its core ,MATLAB is essentially a set (a “toolbox”) of routines (called “m files” or “mix files”) that sit on your computer and a window that allows you to create new variables with names (e.g. voltage and time) and process those variables with any of those routines (e.g. plot voltage against time, find the largest voltage, etc). 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 can run faster the next time you run it.)

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow learning and applying specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include Image processing, signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

The main features of MATLAB

1. Advance algorithm for high performance numerical computation, especially in the Field matrix algebra.

2. A large collection of predefined mathematical functions and the ability to define one’s own functions.

3. Two-and three dimensional graphics for plotting and displaying data.

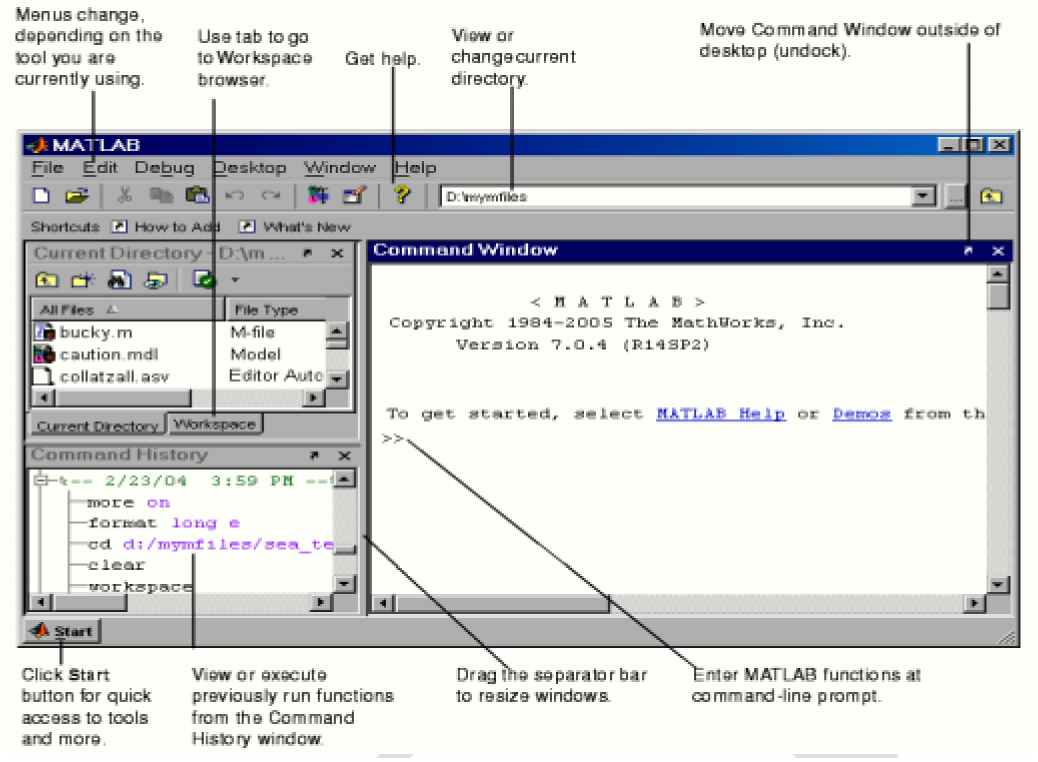
4. Powerful, matrix or vector oriented high level programming language for individual applications.

5. Toolboxes available for solving advanced problems in several application areas.

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB shortcut icon (MATLAB 7.0.4) on your Windows desktop. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

1. The Command Window
2. The Command History
3. The Workspace
4. The Current Directory
5. The Help Browser

: The graphical interface to the MATLAB workspace When MATLAB is started for the first time, the screen looks like the one that shown in the Figure 1. This illustration also shows the default configuration of the MATLAB desktop. You can customize the arrangement of tools and documents to suit your needs. Dimensioning is automatic in MATLAB. No dimension


Menus change, depending on the tool you are currently using.
Use tab to go to Workspace browser.
Get help.
View or change current directory.
Move Command Window outside of desktop (undock).
MATLAB
File Edit Debug Desktop Window Help
D:\mymfiles
Shortcuts How to Add What's New
Current Directory - D:\m...
All Files
File Type
bucky.m
M-file
caution.mdl
Model
collatzall.asv
Editor Auto
Current Directory
Workspace
Command History
Command Window
Start
Click Start button for quick access to tools and more.
View or execute previously run functions from the Command History window.
Drag the separator bar to resize windows.
Enter MATLAB functions at command-line prompt.

```
            < M A T L A B >
 Copyright 1984-2005 The MathWorks, Inc.
        Version 7.0.4 (R14SP2)


 To get started, select MATLAB Help or Demos from th
>>
```

```
⊟-%-- 2/23/04   3:59 PM --%
   ├more on
   ├format long e
   ├cd d:/mymfiles/sea_te
   ├clear
   └workspace
```

Figure A.1: Graphical interface of the MATLAB

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.

The functional unit of data in any MATLAB program is the array. An array is a collection of data values organized into rows and columns, and known by a single name. MATLAB variable is a region of memory containing an array, which is known by a user specified name. MATLAB variable names must begin with a letter, followed by any combination of letters, numbers, and the underscore character. Only the first 31characters are significant; if more than 31 are used, the remaining characters will be ignored. If two variables are declared with names that only differ in the 32nd character, MATLAB will treat them as same variable.

Spaces cannot be used in MATLAB variable names, underscore letters can be substituted to create meaningful names. It is important to include a data dictionary in the header of any program that you write. A data dictionary lists the definition of each variable used in a program. The definition should include both a description of the contents of the item and the units in which it is measured.

MATLAB language is case-sensitive. It is customary to use lower-case letters for ordinary variable names. The most common types of MATLAB variables are double and char. MATLAB is weakly typed language. Variables are not declared in a program before it is used.

MATLAB variables are created automatically when they are initialized. There are three common ways to initialize variables in MATLAB:

1. Assign data to the variable in an assignment system.

2. Input data into the variable from the keyboard.

3. Read data from a file.

The semicolon at the end of each assignment statement suppresses the automatic echoing of values that normally occurs whenever an expression is evaluated in an assignment statement.

Appendix B - Introduction to TMS 320 C6713 DSK

The high–performance board features the TMS320C6713 floating-point DSP. Capable of performing 1350 million floating point operations per second, the C6713 DSK the most powerful DSK development board. The DSK is USB port interfaced platform that allows to efficiently develop and test applications for the C6713. With extensive host PC and target DSP software support, the DSK provides ease of use and capabilities that are attractive to DSP engineers. The 6713 DSP Starter Kit (DSK) is a low-cost platform which lets customers evaluate and develop applications for the Texas Instruments C67X DSP family. The primary features of the DSK are:

1. 225 MHz TMS320C6713 Floating Point DSP
2. AIC23 Stereo Codec
3. Four Position User DIP Switch and Four User LEDs
4. On-board Flash and SDRAM

TIs Code Composer Studio development tools are bundled with the 6713DSK providing the user with an industrial-strength integrated development environment for C and assembly programming. Code Composer Studio communicates with the DSP using an on-board JTAG emulator through a USB interface. The TMS320C6713 DSP is the heart of the system. It is a core member of Texas Instruments C64X line of fixed point DSPs whose distinguishing features are an extremely high performance 225MHz VLIW DSP core and 256Kbytes of internal memory. On-chip peripherals include a 32-bit external memory interface (EMIF) with integrated SDRAM controller, 2 multi-channel buffered serial ports (McBSPs), two on-board timers and an enhanced DMA controller (EDMA). The 6713 represents the high end of TIs C6700 floating point DSP line both in terms of computational performance and on-chip resources.

The 6713 has a significant amount of internal memory so many applications will have all code and data on-chip. External accesses are done through the EMIF which can connect to both synchronous and asynchronous memories. The EMIF signals are also brought out to standard TI expansion bus connectors so additional functionality can be added on daughter card modules. DSPs are frequently used in audio processing applications so the DSK includes an on-board codec called the AIC23. Codec stands for coder/decoder, the job of the AIC23 is to code analog input samples into a digital format for the DSP to process, then decode data coming out of the DSP to generate the processed analog output. Digital data is sent to and from the codec on McBSP1. The DSK has 4 light emitting diodes (LEDs) and 4 DIP switches that allow users to interact with programs through simple LED displays and user input on the switches. Many of the included examples make use of these user interfaces Options. The DSK implements the logic necessary to tie board components together in a programmable logic device called a CPLD. In addition to random glue logic, the CPLD implements a set of 4 software programmable registers that can be used to access the on-board LEDs and DIP switches as well as control the daughter card interface. The DSK uses a Texas Instruments AIC23 (part #TLV320AIC23) stereo codec for input and output of audio signals. The codec samples analog signals on the microphone or line inputs and converts them into digital data so it can be processed by the DSP. When the DSP is finished with the data it uses the codec to convert the samples back into analog signals

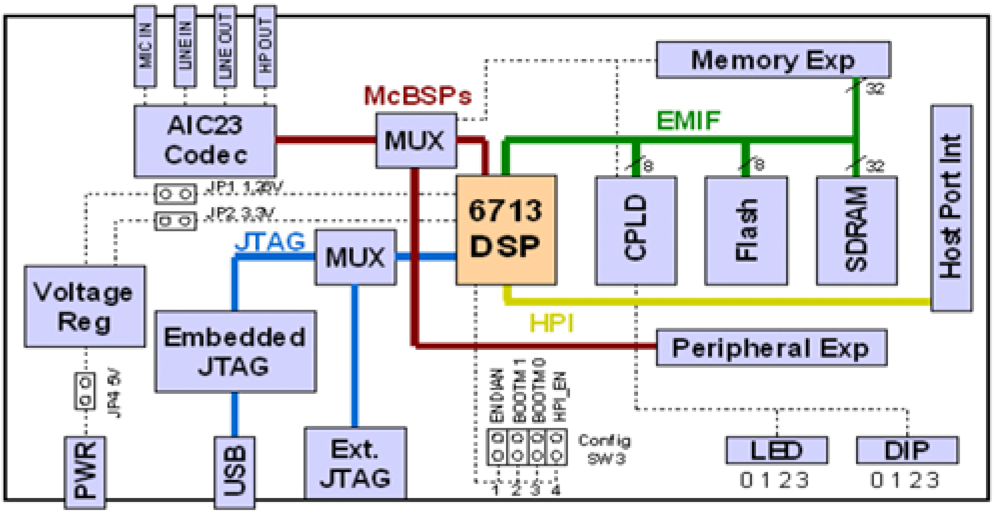


Figure B.1: TMS320C6713 DSK OVERVIEW BLOCK DIAGRAM

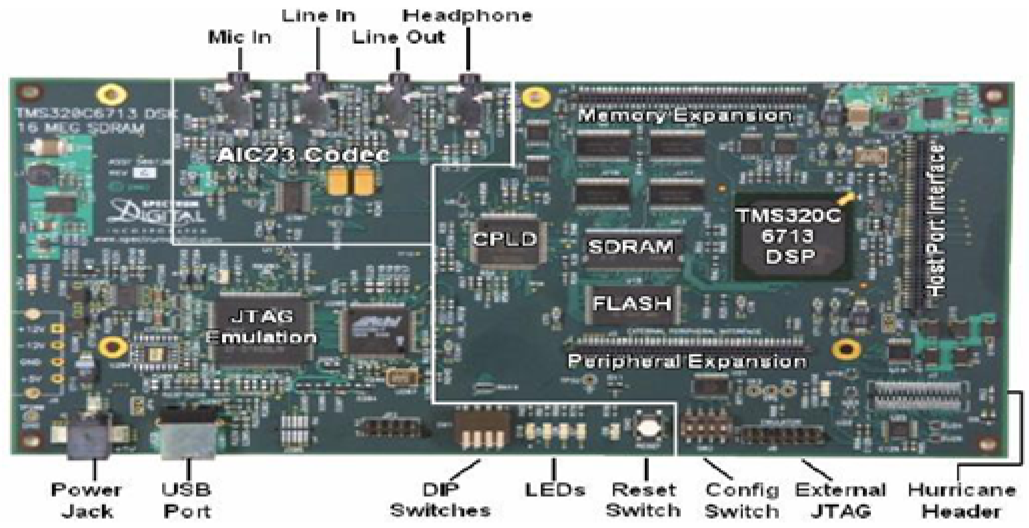


Figure B.2: TMS320C6713 DSK Processor

on the line and headphone outputs so the user can hear the output. The codec communicates using two serial channels, one to control the codec's internal configuration registers and one to send and receive digital audio samples. McBSP0 is used as the unidirectional control channel. It should be programmed to send a 16-bit control word to the AIC23 in SPI format. The top 7 bits of the control word should specify the register to be modified and the lower 9 should contain the register value. The control channel is only used when configuring the codec, it is generally idle when audio data is being transmitted, McBSP1 is used as the bi-directional data channel. All audio data flows through the data channel. Many data formats are supported based on the three variables of sample width, clock signal source and serial data format. The DSK examples generally use a 16-bit sample width with the codec in master mode so it generates the frame sync and bit clocks at the correct sample rate without effort on the DSP side. The preferred serial format is DSP mode which is designed specifically to operate with the McBSP ports on TI DSPs.

Appendix C - Introduction to code composer studio

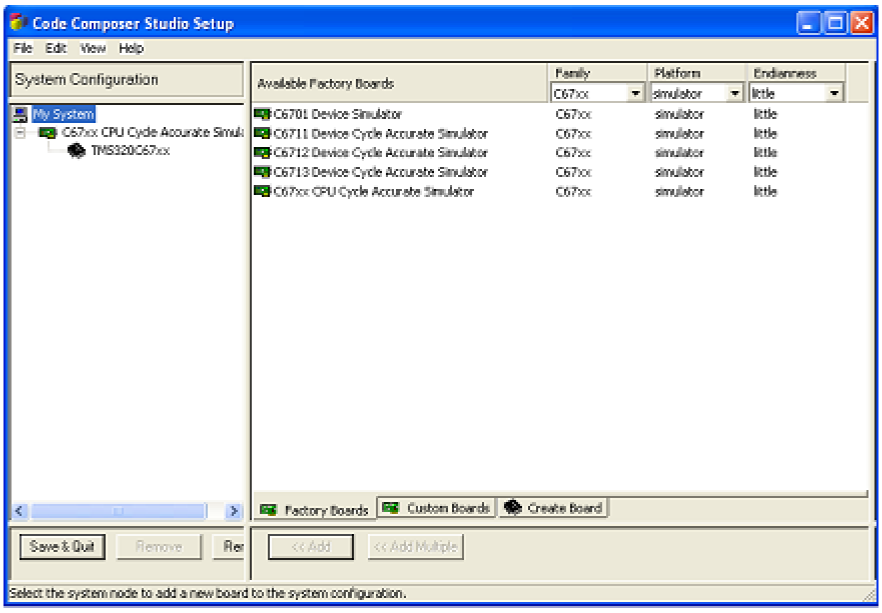
Code Composer is the DSP industry's first fully integrated development environment (IDE) with DSP-specific functionality. With a familiar environment liked MS-based C+, Code Composer lets you edit, build, debug, profile and manage projects from a single unified environment. Other unique features include graphical signal analysis, injection/extraction of data signals via file I/O, multi-processor debugging, automated testing and customization via a C-interpretive scripting language and much more.

CODE COMPOSER FEATURES INCLUDE:

1. Debug IDE
2. Advanced watch windows
3. Integrated editor
4. File I/O, Probe Points, and graphical algorithm scope probes
5. Advanced graphical signal analysis
6. Interactive profiling
7. Automated testing and customization via scripting
8. Visual project management system
9. Compile in the background while editing and debugging
10. Multi-processor debugging
11. Help on the target DSP

TO CREATE A SYSTEM CONFIGURATION USING A STANDARD CONFIGURATION FILES:

1. Start CCS Setup by double clicking on the Setup CCS desktop icon.
2. Select Family as c67xx, Platform as simulator,Endean's as little
3. Click the Import button (File goto import) to import our selection (c67xx_sim.ccs) to the System configuration currently being created in the CCS Setup window.
4. Click the Save and Quit button to save the configuration in the System Registry.
5. Click the Yes button to start the CCS IDE when we exit CCS Setup. The CCS Setup Closes and the CCS IDE automatically opens using the configuration we just created.



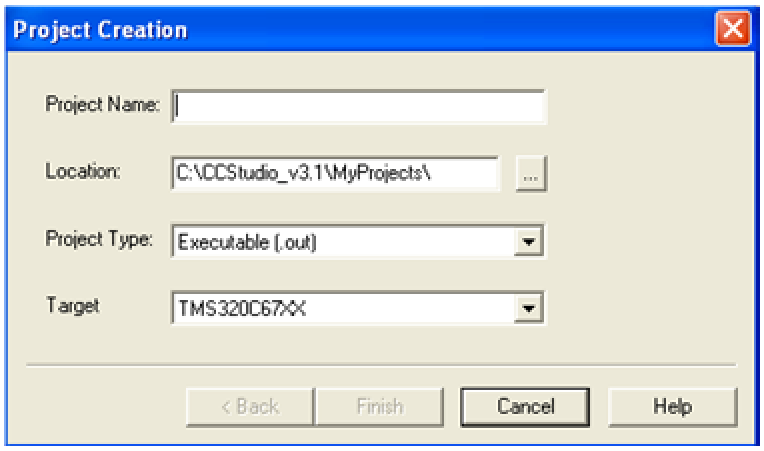
Available Factory Boards	Family	Platform	Endianness
	C67xx	simulator	little
C6701 Device Simulator	C67xx	simulator	little
C6711 Device Cycle Accurate Simulator	C67xx	simulator	little
C6712 Device Cycle Accurate Simulator	C67xx	simulator	little
C6713 Device Cycle Accurate Simulator	C67xx	simulator	little
C67xx CPU Cycle Accurate Simulator	C67xx	simulator	little

Select the system node to add a new board to the system configuration.

Figure C.1: Code composer studio setup

PROCEDURE TO WORK ON CODE COMPOSER STUDIO

1. **Creating a New Project:**From the Project menu, choose New. In the Project Name field, type the name we want for our project. Each project we create must have a unique name, and Click Finish. The CCS IDE creates a project file called projectname.pjt. This file stores our project settings and references the various files used by our project. The Project Creation wizard window displays.



Project Name:

Location: C:\CCStudio_v3.1\MyProjects\

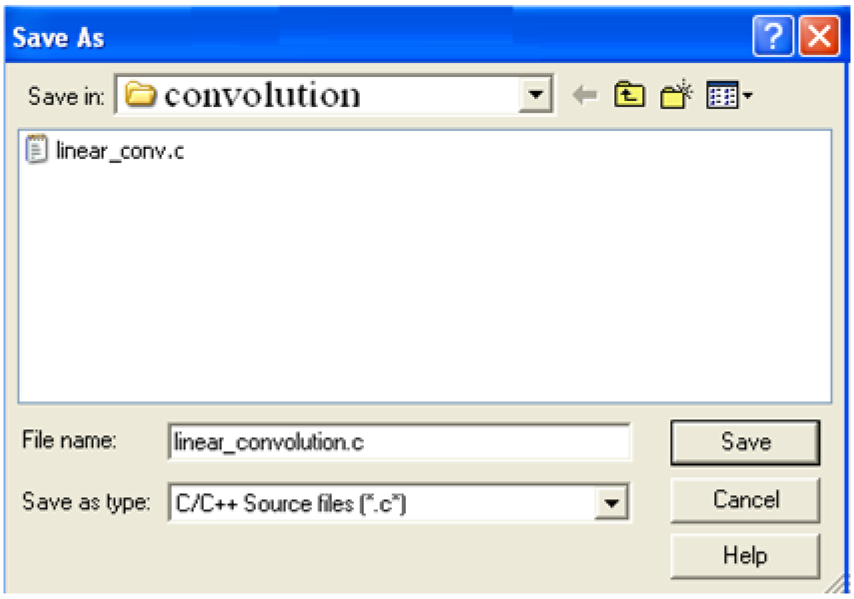
Project Type: Executable (.out)

Target: TMS320C67XX

Figure C.2: Project creation using Code composer studio

2. **Creating a source file**Create a new source file using ‘File then new then source file pull down menu and save the source file with .c extension in the current project name directory.
Path: C:\ CCStudio_v3.1\ MyProjects\Project Name

3. **Add files to our project (source file\ library file\ linker file)**
SOURCE FILE: Add the .C source file in the project using Project then add files to project pull down menu.
Path: C:\CCStudio_v3.1\ My Projects\Project Name\filename



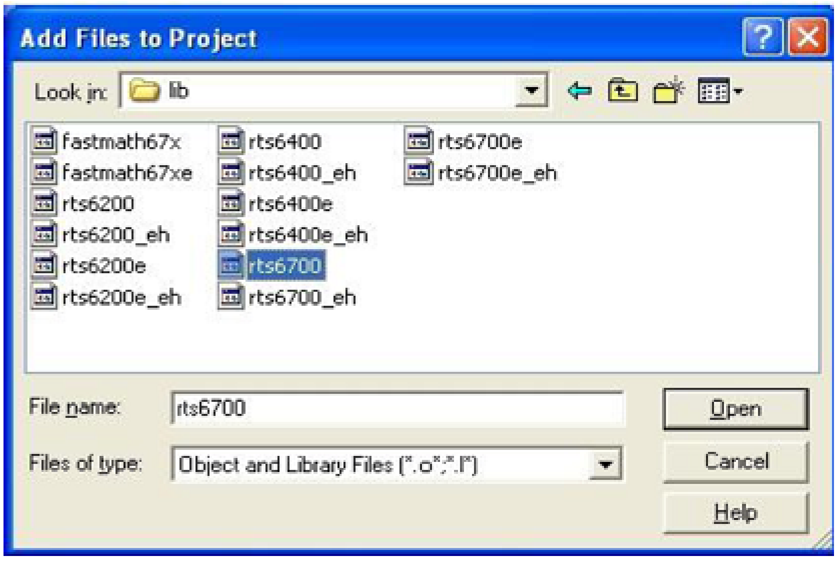
linear_conv.c

File name: linear_convolution.c

Save as type: C/C++ Source files (*.c*)

Figure C.3: Saving file using Code composer studio

LIBRARY FILE: Add the library file in the project using Project then add files to project pull down menu. Files of type: Object and Library Files. Path: C:\CCStudio_v3.1\ C6000\ cgtools\ lib \ rts6700.lib



fastmath67x, fastmath67xe, rts6200, rts6200_eh, rts6200e, rts6200e_eh, rts6400, rts6400_eh, rts6400e, rts6400e_eh, rts6700, rts6700_eh, rts6700e, rts6700e_eh

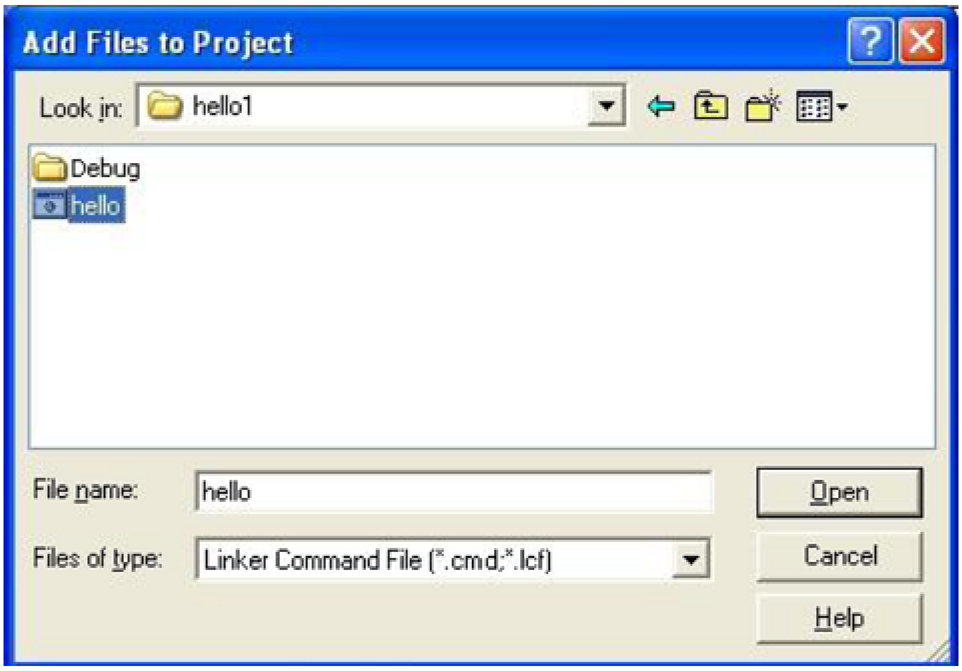
File name: rts6700

Files of type: Object and Library Files (*.o*;*.l*)

Figure C.4: Adding library file using Code composer studio

4. **LINKER FILE:** Add the linker file in the project using 'Project-¿ add files to project' pull down menu. Files of type: Linker command Files. Path: C:\CCStudio_v3.1\ tutorial\ dsk6713\ hello1 \ hello.cmd

5. **Building and Running the Program (compile\ Build\ Load Program\ Run)**
   COMPILE: Compile the program using the 'Project-compile' pull down menu or by clicking the shortcut icon on the left side of program window.
   BUILD: Build the program using the 'Project-Build' pull down menu or by clicking the shortcut icon on the left side of program window.



File name: hello

Files of type: Linker Command File (*.cmd;*.lcf)

Figure C.5: Addinglinker file using Code composer studio

LOAD PROGRAM: Load the program in program memory of DSP chip using the 'File-load program' pull down menu. Files of type:(.out) Path: C:\CCStudio_v3.1\ MyProjects\Project Name\ Debug\ Project ame.out
RUN: Run the program using the Debug- then run pull down menu or by clicking the shortcut icon on the left side of program window.

6. **observe output using graph**Choose View then Graph then Time or Frequency. In The Graph Property Dialog, Change The Graph Title, Start Address, And Acquisition Buffer Size, Display Data Size, Dsp Data Type, Auto Scale, And Maximum Y Value Properties To The Values.