FIR FILTER DESIGN AND ITS APPLICATION IN RADIO ASTRONOMY

PROJECT REPORT

2000 - 2001

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION

BANGALORE UNIVERSITY

by

ANUPAMA K **ASHWINI U**

SOUMYA M

Under the guidance of

External Guide

Prof. C.R. SUBRAHMANYA
Dept. of Astronomy & Astrophysics
Raman Research Institute
Bangalore - 560 080.

Internal Guide

Prof. D. KUMARASWAMY
Dept. of E & C
M.S.R.I.T.
Bangalore - 560 054.

Department of Electronics & Communication

M.S. RAMAIAH INSTITUTE OF TECHNOLOGY

BANGALORE - 560 054

RAMAN RESEARCH INSTITUTE
C.V. Raman Avenue, Sadashivanagar, Bangalore-560 080 India

Prof. C. R. Subrahmanya
Department of Astronomy &Astrophysics

CERTIFICATE

This is to certify that the project work entitled

"FIR FILTER DESIGN AND ITS APPLICATION IN RADIO ASTRONOMY"

carried out by

ANUPAMA K **ASHWINI U**

SOUMYA M

submitted in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF ENGINEERING** in **ELECTRONICS AND COMMUNICATION** by the **BANGALORE UNIVERSITY**, has been carried out under my guidance during the period of November 2000 to August 2001. This report has not been submitted elsewhere for a similar degree.

C. R. Subrahmanya
Project Guide

Phone : (80) 334 0122 Fax : (80) 334 0492 Telegrams : RAMANINST e-mail : root@rri.ernet.in

M.S. RAMAIAH INSTITUTE OF TECHNOLOGY

Department of Electronics & Communication

BANGALORE - 560 054

CERTIFICATE

This is to certify that the project entitled **"FIR FILTER DESIGN AND ITS APPLICATION IN RADIO ASTRONOMY"**, done in partial fulfillment for the award of the Degree of Bachelor of Engineering in Electronics and Communication from the Bangalore University during the year 2000-2001 was duly submitted by

ANUPAMA K
97GDEE2006

ASHWINI U
97GDEE2008

SOUMYA M
97GDEE2050

6/8/2001

Prof. D. KUMARASWAMY
Dept. of E & C
M.S.R.I.T.
Bangalore - 560 054.

6/8/01

Prof. M.S. SUDHI
H.O.D, Dept. of E & C
M.S.R.I.T.
Bangalore - 560 054.

ACKNOWLEDGEMENTS

In the execution of the project "**FIR FILTER DESIGN AND ITS APPLICATION IN RADIO ASTRONOMY**" we received constant help and encouragement directly and indirectly from various people to whom we are indebted. In particular, we would like to express our sense of gratitude to the under-mentioned, without whose active help and guidance the successful completion of this project would not have been possible.

Prof. N. Kumar, Director, Raman Research Institute and **Dr. D. K. Ravindra**, Head of Radio Astronomy Laboratory, RRI, for granting us the permission to carry out this project.

Prof. C. R. Subrahmanya (External guide), Department of Astronomy and Astrophysics, RRI, for his technical guidance and support right from the conceptual stage up to the level of implementation of this project

Prof. M. S. Sudhi, Head of Department of Electronics and Communication, M. S. Ramaiah Institute of Technology, for allowing us to undertake the project at Raman Research Institute.

Prof. D. Kumaraswamy (Internal guide), Department of Electronics and Communication, M. S. Ramaiah Institute of Technology, for his constant support and guidance during the course of the project.

All the faculty members at M. S. Ramaiah Institute of Technology for their timely suggestions and finally to our parents for all the support and encouragement they have extended to us.

CONTENTS

SYNOPSIS

1. **PREAMBLE** 1
   - **1.1** General introduction
   - **1.2** Statement of the problem
   - **1.3** Objectives
   - **1.4** Methodology

2. **INTRODUCTION TO SIGNAL PROCESSING** 3
   - **2.1** Signals and systems
   - **2.2** Sampling theorem
   - **2.3** Types of discrete time signals
   - **2.4** Advantages/disadvantages of digital over analog signal processing
   - **2.5** Applications of digital signal processing

3. **FOURIER ANALYSIS** 8
   - **3.1** Introduction to Fourier analysis
   - **3.2** Fourier transform
   - **3.3** Discrete Fourier transform
   - **3.4** Fast Fourier transform

4. **INTRODUCTION TO DIGITAL FILTERS** 16
   - **4.1** Filtering
   - **4.2** Advantages of digital filters
   - **4.3** Applications of digital filters
   - **4.4** Study of digital filters
     - **4.4.1** Digital filter as a system
     - **4.4.2** Characterization of digital filters
     - **4.4.3** Properties of FIR filters
   - **4.5** Desirability of linear phase
   - **4.6** Filtering of long data sequences
   - **4.7** Filter bank
   - **4.8** Advantages/disadvantages of FIR over IIR filters

5. **DESIGN OF FIR FILTERS** 32
   - **5.1** Issues in filter design
   - **5.2** Design of FIR filters by windowing

6. IMPLEMENTATION **46**
- **6.1** Comments on the programming style
- **6.2** LINUX
- **6.3** Gnuplot
- **6.4** FFTW

7. REALIZATION OF FIR FILTERS USING SOFTWARE **50**
- **7.1** Quantitative comparison of various windows
- **7.2** Quantitative comparison of LPF with different windows

8. APPLICATION TO RADIO ASTRONOMY **63**
- **8.1** Introduction to radio astronomy
- **8.2** Methanol maser
- **8.3** Radio telescope
- **8.4** Application of the FIR filter

9. CONCLUSION **71**
- **9.1** Achievements
- **9.2** Scope for future work in this field

APPENDIX
- Flowcharts
- Programs

BIBLIOGRAPHY

SYNOPSIS

Digital filters are major components of Digital Signal Processing (DSP) systems. This project report describes the study of digital filters, development of software for the design of digital FIR filters and its application to a specific problem in radio astronomy.

In the study phase of the project we focused on the digital filtering techniques, in particular the Windowing technique. Various types of filters and their governing mathematical models were studied.

In the design and implementation phase, a C program on LINUX platform was developed for obtaining the magnitude and phase plots of the window functions and the frequency response and impulse response of the filters. The user would be able to choose between Lowpass, Highpass, Bandpass and Bandstop filter design using any of the windows such as Rectangular, Bartlett, Hanning, Hamming, Blackman and Kaiser. A quantitative comparison of different window functions and the resulting responses of filter were made. DFT was used to analyze the frequency response of the filters. During the initial stages, the DFT was directly computed. In the later stages, FFTW, a library from public domain for Fast Fourier Transform (FFT) was used to increase the speed of computation. The C programs developed call "gnuplot" to display the magnitude and phase spectra of the filters. It is a command driven interactive function plotting program, a powerful tool on LINUX platform for graphics.

In the final phase of this project a bandpass filter was designed using this software and implemented for analyzing the data obtained from a Methanol Maser source observed using a 10.4m diameter radio telescope operating in the centimeter wave band.

Preamble

1. PREAMBLE

1.1 GENERAL INTRODUCTION

Today **Digital Signal Processing** (DSP) has pervaded several fields like speech synthesis and analysis, satellite communication, telephony, medical imaging, radar and sonar, to name just a few. The advancement in the IC technology has led to the availability of very high-speed analog-to-digital (A/D) converters. This has revolutionalized the design of receiver systems for satellite communication, radio astronomy and many other branches of engineering. Before the advent of high-speed A/D converters, mostly the conversion was carried out after bringing down the frequency of the received signal to the video band. But today the main emphasis is, how soon in the chain of a receiver system can we digitize. With this, the digital signal processors have become very important.

DSP is the mathematics, the algorithms and the techniques used to manipulate the signals after they have been converted to a digital form. Digital filters are a very important part of DSP. Their extraordinary performance is one of the key reasons for the popularity of DSP. Digital filters are divided into **Finite Impulse Response** (FIR) and **Infinite Impulse Response** (IIR) filters. Digital filters are used for two general purposes namely separation of signals that have been combined and restoration of signals that have been distorted in some way.

1.2 STATEMENT OF THE PROBLEM

"Development of software for the design of Finite Impulse Response (FIR) digital filter and apply the filter in processing the data acquired from the observations of a spectral line using the 10.4m radio telescope at the Raman Research Institute (RRI)".

1.3 OBJECTIVES

- To design a **Finite Impulse Response** (FIR) filter using the windowing technique for a given filter tap length or for the required rolloff rate, sidelobe suppression and peak ripple.
- To design a **filter bank,** which selects the required number of channels of specified bandwidth.

- To demonstrate the **application** of a FIR filter in the case study of a signal received from a radio telescope.

1.4 METHODOLOGY

The course of the project began with the study of digital filters and the techniques of digital filter design. The **window method** of digital filter design was chosen.

A study of the various windows was carried out and a quantitative comparison of the different window functions was done.

Using these windows the filter was designed based on the specifications of operating bandwidth, cutoff frequency, taplength or rolloff transition width from passband to the stopband and the degree of suppression in the stopband.

For obtaining the magnitude and phase spectra of the window functions and the filters a C program that generates the Discrete Fourier Transform (**DFT**) was developed. A Fast Fourier Transform (**FFT**) library called the **FFTW** was used in the later part of the project to increase the speed of computation.

The operating system on which the C programs were developed is **LINUX**. The graphs were plotted using an application program called **gnuplot**.

Finally, a filter specific to observation of a celestial radio source using the RRI telescope was designed and applied for actual observations.

Introduction to signal processing

2. INTRODUCTION TO SIGNAL PROCESSING

2.1 SIGNALS AND SYSTEMS

A signal can be defined as a function that conveys information, generally about the state or behavior of a physical system. Although signals can be represented in many ways, in all cases the information is contained in a pattern of variations of some form. For example, the signal may take the form of a pattern of time variations or spatially varying pattern. Signals are represented mathematically as a function of one or more independent variables. For example, a speech signal would be represented mathematically as a function of time and picture would be represented as a brightness function of two spatial variables.

The independent variable of the mathematical representation of a signal may be either continuous or discrete. **Continuous-time signals** are signals that are defined at a continuum of times and thus are represented by continuous variable functions. **Discrete-time signals** are defined at discrete times and thus the independent variable takes on only discrete values i.e., discrete-time signals are represented as sequence of numbers.

In addition to the fact that the independent variables can be either continuous or discrete, the signal amplitude may be either continuous or discrete. **Digital signals** are those for which both time and amplitude are discrete. Continuous-time, continuous-amplitude signals are sometimes called **analog signals**.

The signal generation is usually associated with a system that responds to a stimulus or force. The stimulus in combination with the system is called the **signal source**. A **system** is defined as a physical device that performs an operation on a signal. For example, a filter used to reduce the noise and interference corrupting a desired information-bearing signal is called a system. When we pass a signal through a system, as in filtering, we say that we have processed the signal. The type of operation that it performs on the signal characterizes the system. For example, if the operation is linear the system is called linear, if the operation on the signal is nonlinear, the system is said to be nonlinear, and so forth. Such operations, which are concerned with representation, transformation and manipulation of signals and the information they contain, are usually referred to as **signal processing**.

Signal processing may be classified along the same lines as signals. That is, continuous-time systems are systems for which both the input and output are continuous-time signals and discrete-time systems are those for which both the input and output are discrete-time signals. Similarly analog systems are systems for

which the input and output are analog signals and digital systems are those for which both the input and output signals are digital signals. **Digital signal processing** deals with the transformation of signals that are discrete in both amplitude and time. The term mixed signal processing is often used in the context of systems, which include anti-aliasing filters and analog-to-digital converters in the first stage of DSP.

Discrete-time signals may arise by the sampling a continuous-time signal or they may be generated directly by some discrete time process. Whatever the origin of discrete time signals, digital signal processing systems have many attractive features. They can be realized with great flexibility using general-purpose digital computers or they can be realized with digital hardware. They can, if necessary, be used to simulate analog systems or more importantly to realize signal transformations impossible to realize with analog hardware. Thus, discrete representations of signal are often desirable when sophisticated signal processing is required.

2.2 SAMPLING THEOREM

A message signal may originate from a digital or analog source. To process an analog signal using digital signal processing techniques, it is necessary to convert the signal into a sequence of numbers. One of the important steps before formatting the analog signal for digital processing is the process of sampling. This is achieved by taking samples of the analog signals at periodic intervals of time.

Shannon introduced the sampling theorem to communication theory in 1949. It is for this reason that the theorem is also referred to as the '**Shannon sampling theorem'.**

For the sampling process to be of practical utility it is necessary to choose the sampling rate properly so that the discrete time signal uniquely defines the original continuous time signal. This is given by the sampling theorem which states that "any band limited signal having no spectral components above **w Hz** can be uniquely determined by values sampled at uniform intervals of T seconds, provided $\mathbf{T \leq 1/2w}$ or $\mathbf{f_s \geq 2w}$ where f_s is called as sampling rate and T_s is called as the sampling interval ". The rate $\mathbf{f_{sNQ} = 2w}$ is called as the **Nyquist rate.**

However, for a bandpass signal x(t) with a spectrum of bandwidth **B** and upper frequency limit $\mathbf{f_u}$, it is not necessary to sample the signal at twice the highest frequency . It is sufficient to sample the signal at twice the bandwidth 2B since the bandwidth is the one that contains the information and not the signal frequency. However, one has to ensure that none of the harmonics of the sampling

frequency falls within the band of interest. Thus x(t) can be uniquely determined by instantaneous values if the sampling rate $\mathbf{f_s \geq 2f_u/m}$, where **m** is the largest integer not exceeding f_u/B. (higher sampling rates are not always usable unless they exceed $2f_u$)i.e.,$\mathbf{m \leq f_u/B}$. The sampling rate for a bandpass signal depends on the ratio f_u/B. If $\mathbf{f_u/B >> 1}$, then the minimum sampling rate approaches 2B.

2.3 TYPES OF DISCRETE TIME SIGNALS

A **continuous-time** signal can be represented by a function x(t) whose domain is a range of numbers (t1,t2), where $-\infty \leq t1,t2 \leq \infty$. Similarly a **discrete-time** signal can be represented by a function x(nT), where T is constant and n is an integer in the range (n1,n2) such that $-\infty \leq n1$ and $n2 \leq \infty$. Alternatively a discrete time signal can be represented by x(n).

Continuous-time signals and discrete-time signals can be non-quantized or quantized. A **non-quantized** signal is one that can assume any value within a specific range, whereas a **quantized** signal is one that can assume only a finite number of discrete values.

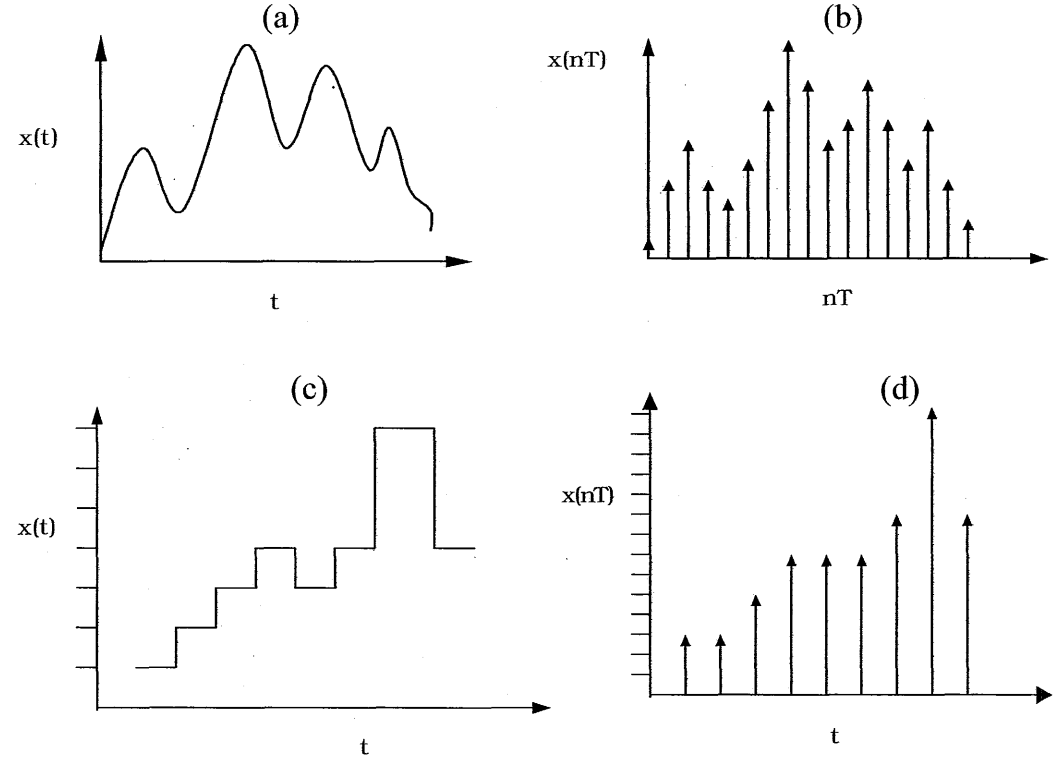


Figure 2.1: (a): Non-quantized continuous time signal
(b): Non-quantized discrete time signal

(c): Quantized continuous time signal
(d): Quantized discrete time signal

2.4.1 ADVANTAGES OF DIGITAL OVER ANALOG SIGNAL PROCESSING

A digital programmable system allows **flexibility** in reconfiguring the digital signal processing operations simply by changing the program. Reconfiguration of an analog system usually implies a redesign of the hardware followed by tuning, testing and verification to see that it operates properly.

Accuracy considerations also play an important role in determining the form of the signal processor. Tolerances in analog circuit components make it extremely difficult for the system designer to control the accuracy of an analog signal processing system. On the other hand, a digital system provides much better control of accuracy requirements.

Digital signals are easily stored on magnetic media (tape or disk) without deterioration or loss of signal fidelity beyond that introduced in analog-to-digital (A/D) conversion. As a consequence, the signals become transportable and can be processed off-line in a remote laboratory. It is possible to incorporate a variety of coding, compression and error checking algorithms into digital processing to enhance the **reliability** and **efficiency** of communication and archival. The digital signal processing methods also allows for the implementation of more sophisticated signal processing algorithms. It is usually very difficult to perform precise mathematical operations on signals in analog form but these same operations can be routinely implemented on a digital computer using software.

In some cases a digital implementation of the signal processing system is more **economical** than its analog counterpart. The lower cost may be due the fact that the digital hardware is **less expensive**, or may be a consequence of the flexibility for modifications provided by the digital implementation.

As a consequence of these advantages, digital signal processing has been applied in practical systems covering a broad range of disciplines.

2.4.2 DISADVANTAGES OF DIGITAL OVER ANALOG SIGNAL PROCESSING

- The implementation of a discrete time system in special purpose hardware can result in performance degradation.

- Another practical limitation is the speed of operation of A/D converters and digital signal processors. Signals having extremely wide bandwidths require fast-sampling-rate A/D converters and fast digital signal processors. Hence there are analog signals with large bandwidths for which a digital signal processing approach is beyond the state of the art of digital hardware.

2.5 APPLICATIONS OF DIGITAL SIGNAL PROCESSING

- **Space**: Data compression, Intelligent sensory analysis by remote space probes
- **Medical**: Diagnostic imaging (CT, MRI, ultrasound), Electrocardiogram analysis, Medical image storage/retrieval
- **Commercial**: Image and sound compression for multimedia presentation, Movie special effects, Video conference calling
- **Telephone**: Voice and data compression, Echo cancellation, Signal multiplexing, Filtering
- **Military**: Radar, sonar, secure communication
- **Industrial**: Oil and mineral prospecting, Process monitoring and control, Nondestructive testing, CAD and design tools
- **Scientific**: Radio astronomy, Earthquake recording and analysis, Data acquisition, Spectral analysis, Simulation and modeling

Fourier Analysis

3. FOURIER ANALYSIS

3.1 INTRODUCTION TO FOURIER ANALYSIS

Fourier analysis is a family of mathematical techniques, all based on decomposing signals into sinusoids. A time domain signal can be either continuous or discrete and it can be either periodic or aperiodic. This leads to the definition of four types of Fourier transforms: the **discrete Fourier transform** (discrete, periodic), the **discrete time Fourier transform** (discrete, aperiodic), the **Fourier series** (continuous, periodic) and the **Fourier transform** (continuous, aperiodic). If a signal is discrete in one domain, it will be periodic in the other. Likewise, if a signal is continuous in one domain it will be aperiodic in the other.

Each of the four Fourier transforms can be subdivided into **real** and **complex** versions. The real version is the simplest, using ordinary numbers and algebra for the synthesis and decomposition. The complex versions are immensely more complicated requiring the use of complex numbers.

3.2 FOURIER TRANSFORM

The Fourier transform technique forms the basic foundation for all digital signal processing applications. It is been named after Jean Baptiste Joseph Fourier (1768-1830), a French mathematician and physicist. Fourier transform provides a link between the time domain and frequency domain descriptions of a signal. The waveform of a signal and its spectrum (i.e., frequency content) are natural vehicles to understand the signal.

By definition, the Fourier transform of the signal g(t) is given by the integral

$$G(f) = \int_{-\infty}^{+\infty} g(t)e^{-j2\pi ft}\,dt \qquad (3.1)$$

where g(t) is a non-periodic, deterministic signal, expressed as some function of time.

Given the Fourier transform G(f), the original signal g(t) is recovered exactly using the formula for the inverse Fourier transform

$$g(t) = \int_{-\infty}^{\infty} G(f)e^{j2\pi ft}\,df \qquad (3.2)$$

For example the Fourier transform of a sinusoid (time domain) consists of two delta functions (frequency domain). Similarly rect (t) forms a Fourier transform pair with sinc (f).

Rect (t) ←→ sinc (f)

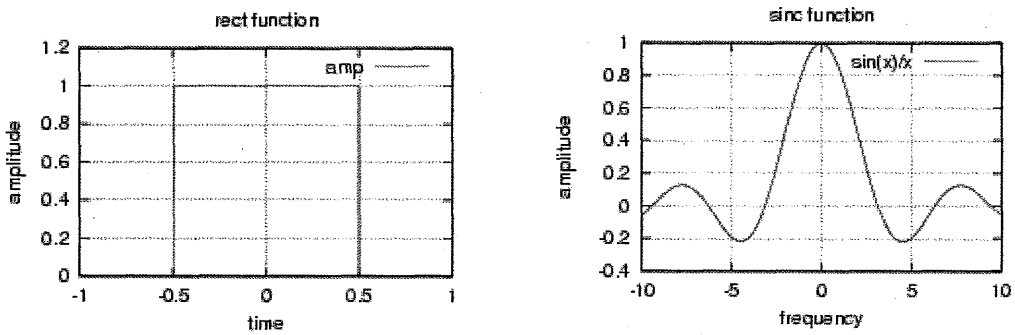


Figure 3.1: Graphical representation of a Fourier transform pair

For finite duration sequences, it is possible to develop an alternative Fourier representation, referred to as **'Discrete Fourier Transform'** (DFT). The DFT in itself is a sequence rather than a function of a continuous variable, and it corresponds to samples, equally spaced in frequency, of the Fourier transform of the signal.

3.3 DISCRETE FOURIER TRANSFORM

The only type of Fourier transform that can be used in DSP is the DFT. In other words, digital computers can only work with information that is discrete and finite in length. DFT is used in three common ways. First, the DFT can calculate a signal's frequency spectrum. This is the direct examination of information encoded in the frequency, phase and amplitude of the component sinusoids. Second, DFT can find a system's frequency response from the system's impulse response, and vice versa. This allows systems to be analyzed in the frequency domain, just as convolution allows systems to be analyzed in the time domain. Third, the DFT can be used as an intermediate step in more elaborate signal processing techniques. The classic example of this is FFT convolution.

The sequence say, x(n) is a sequence of N input data samples then the DFT produces a sequence of N samples X(k) spaced equally in frequency.

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N} \qquad (3.3)$$

where $e^{-j2\pi nk/N} = \cos(2\pi nk/N) - j\sin(2\pi nk/N)$

Hence DFT can be viewed as a tool to fit sum of sinusoids of different frequencies, amplitudes and phases to approximate the original signal in terms of known signals.

The sine and cosine waves used in the DFT are commonly called the DFT **basis functions**. In other words, the output of the DFT is a set of numbers that represent amplitudes. The basis functions are a set of sine and cosine waves with unity amplitude. If we assign each amplitude (the frequency domain) to the proper sine or cosine wave (the basis functions), the result is a set of scaled sine and cosine waves that can be added to form the time domain signal. The DFT basis functions are generated from the equations

$$c_k[n] = \cos(2\pi kn/N)$$
$$s_k[n] = \sin(2\pi kn/N) \qquad (3.4)$$

where $c_k[\ ]$ is the cosine wave for the amplitude held in ReX[k] and $s_k[\ ]$ is the sine wave for the amplitude held in ImX[k].

A graphical illustration of how a DFT pair is obtained from a Fourier transform pair is shown figure 3.2. Consider the example function h(t) and its Fourier transform H(f) illustrated in the fig (a). It is desired to modify this Fourier transform pair in such a manner that the pair is amenable to digital computer computation. This modified pair, termed the discrete Fourier transform is to approximate as closely as possible, the continuous Fourier transform.

To determine the Fourier transform of h(t) by means of digital analysis techniques, it is sampled by multiplying h(t) by the sampling function, shown in the fig (b). The sample interval is T. The sampled function $\hat{h}(t)$ and its Fourier transform are as shown in fig (c). This Fourier transform pair represents the first modification to the original pair, which is necessary in defining a discrete transform pair. Note that to this point the modified transform pair differs from the original transform pair only by the aliasing effect, which results from sampling. If the waveform h(t) is sampled at a frequency of at least twice the largest frequency component of h(t), there is no loss of information as a result of sampling. If the function h(t) is not band limited, then sampling will introduce aliasing as illustrated in fig (c). To reduce this error, we have only one recourse and that is to sample faster by choosing T smaller.

The Fourier transform pair in fig (c) is not suitable for machine computation because an infinity of samples of h(t) is considered. It is necessary to truncate the sampled function h(t) so that only a finite number of points, say N are considered. The rectangular or truncation function and its Fourier transform are illustrated in fig (d). The product of the infinite sequence of impulse functions representing h(t)

and the truncation function yields the finite length time function shown in fig (e). Truncation introduces a second modification of the original Fourier transform pair. This effect is to convolve the aliased frequency transform of fig (c) with the Fourier transform of truncation function [fig (d)]. As shown in fig (e) the frequency transform now has a ripple to it. To reduce this effect, the truncation function is increased in length so that the sinc(f) function will approach an impulse function.

The modified transform pair of fig (e) is still not an acceptable DFT pair because the frequency transform is a continuous function. For machine computation, only sample values of the frequency function can be computed. It is necessary to modify the frequency transform by the frequency sampling function shown in fig (f). The frequency-sampling interval is 1/To.

The DFT pair of fig (g) is acceptable for the purpose of digital machine computation since both the time and frequency domains are represented by discrete values. As illustrated in fig (g), the original time function is approximated by N samples. The original Fourier transform H(f) is also approximated by N samples . These N samples define the DFT pair and approximate the original Fourier transform pair. It is important to note that the sampling in time domain resulted in a periodic function of frequency; sampling in the frequency domain resulted in a periodic function of time. Hence, the DFT requires that both the original time and frequency functions be modified such that they become periodic functions. N time samples and N frequency values represent one period of the time and frequency domain waveforms respectively.

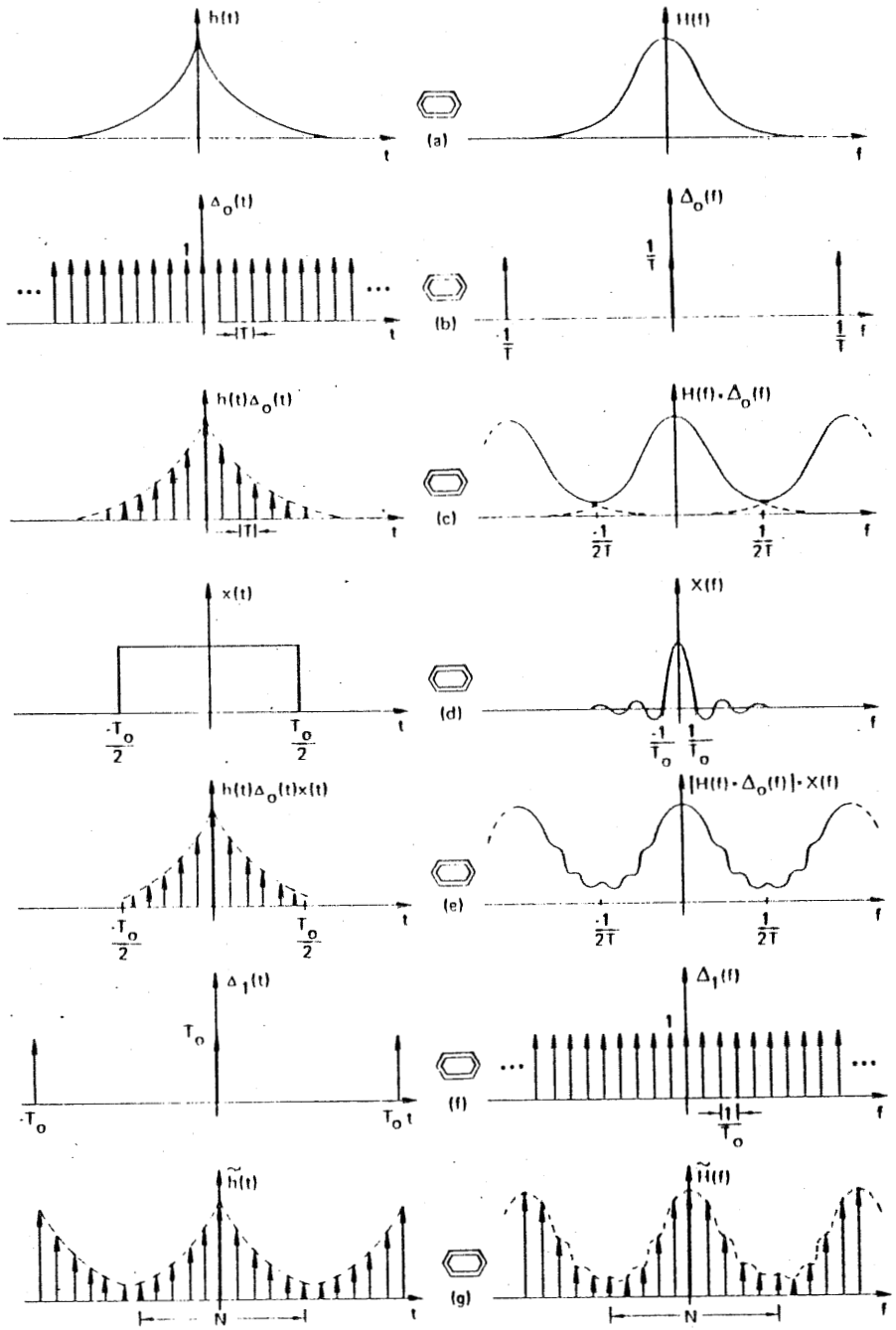


Figure 3.2: Graphical development of the discrete Fourier transform

3.4 FAST FOURIER TRANSFORM

The **multiply-accumulate** is a basic building block in DSP. The speed of DSP computers is often specified by how long it takes to perform a multiply-accumulate operation. The problem of excessive execution time is commonly handled by using a better algorithm for implementing the DFT. Fast Fourier Transform (FFT) is one such very sophisticated algorithm. FFT produces exactly the same result as other conventional Fourier transform algorithms. However, the execution time is dramatically reduced. For signals with thousands of samples FFT can be hundreds of times faster. The disadvantage is program complexity.

In evaluating the **DFT** equation for N points, **NxN complex multiplications** are to be performed. The complex multiplications can be broken down into simpler chains of addition and multiplication while removing a lot of redundancy, which would substantially reduce the computational time. For example, if a 1024-point DFT is performed 1024x1024 multiplications are required. It is possible to break 1024 points into two 512-point DFT's and end up with the same result. This is called decimation. This helps us in finding and removing redundancies in calculations of the two halves, one such algorithm to find the DFT much faster is the Fast Fourier Transform (FFT). The **FFT** reduces the number of complex multiplications to $\mathbf{N \log_2 (N)}$.

There are two basic classes of FFT based on the manner in which this principle is implemented namely

- Decimation in time (**DIT**)
- Decimation in frequency (**DIF**)

Decimation in time derives its name from the fact that in the process of arranging the computations into smaller transformations, the sequence x(n) is decomposed successively into smaller sub-sequences. It implements the two N/2-point transforms using the even and odd elements of the input sequence respectively. These two transforms are then merged by a further N/2 2-point transform to generate the required output elements. The block diagram and flow graph are as shown.

Decimation in frequency performs the same two sets of operations but in the reverse order. The sequence of DFT coefficients is decomposed into smaller sub-sequences. The input samples are first processed in pairs by N/2 2-point transforms followed by the two N/2-point transforms, which generate odd and even components of the output sequence directly. The block diagram and flow graph are as shown.

Depending on whether the decimation was done in groups of 2,4,8 etc the FFT is called the Radix 2,4 or 8 respectively. It is clear that if the inputs are

supplied in a numerically ascending time order the output will be in the bit reversed order. Bit-reversing technique can be implemented in DSP software to return the output in the same order as the input. This basic element in the calculation of the DFT is termed as ***Butterfly***. The FFT is made up of many butterfly calculations.

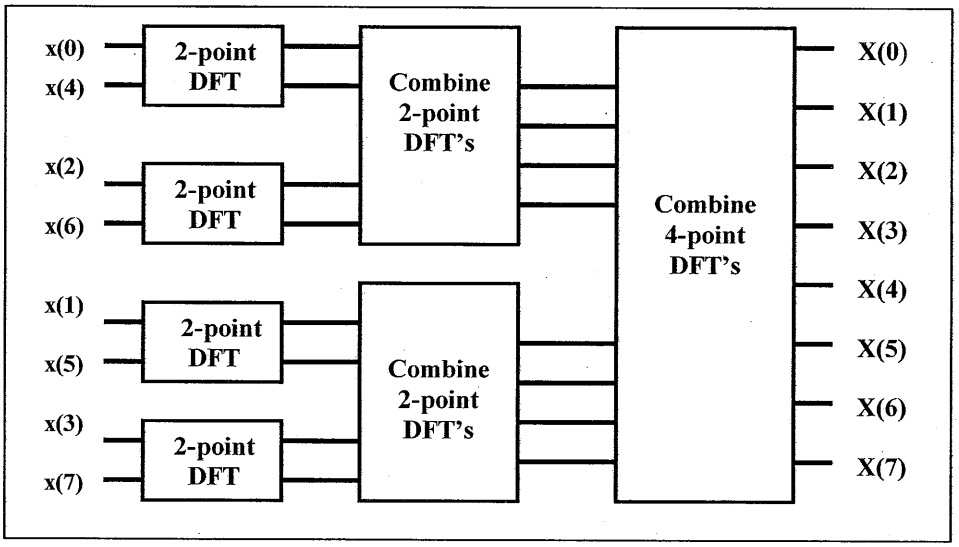


Figure 3.3: Decimation in time

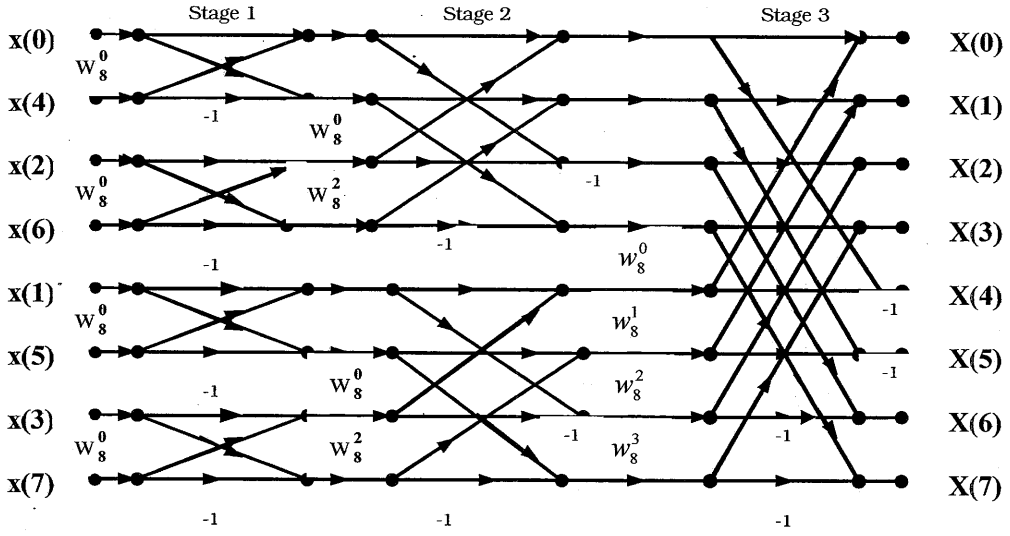


Figure 3.4: Flow graph of decimation in time

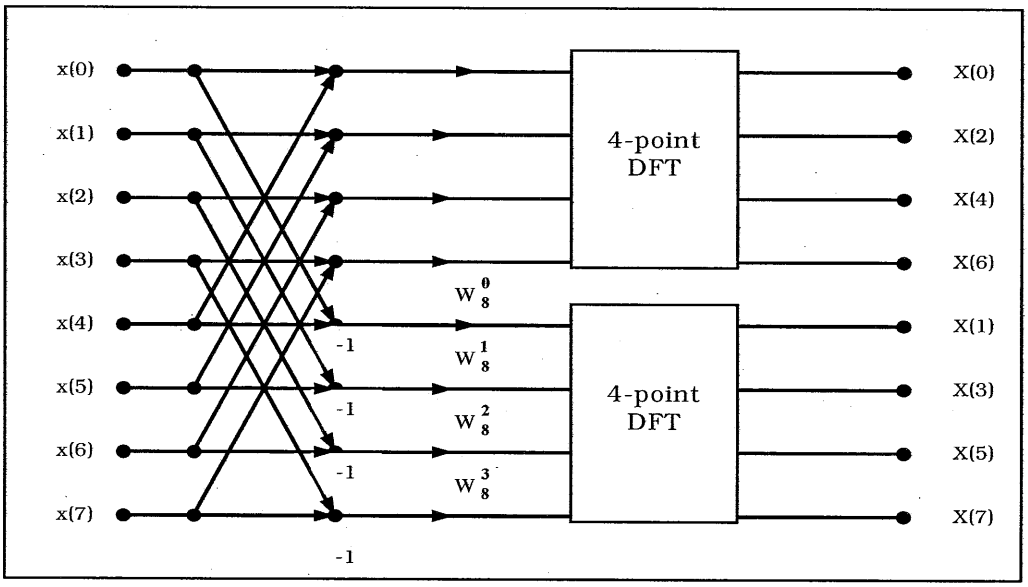


Figure 3.5: Decimation in frequency

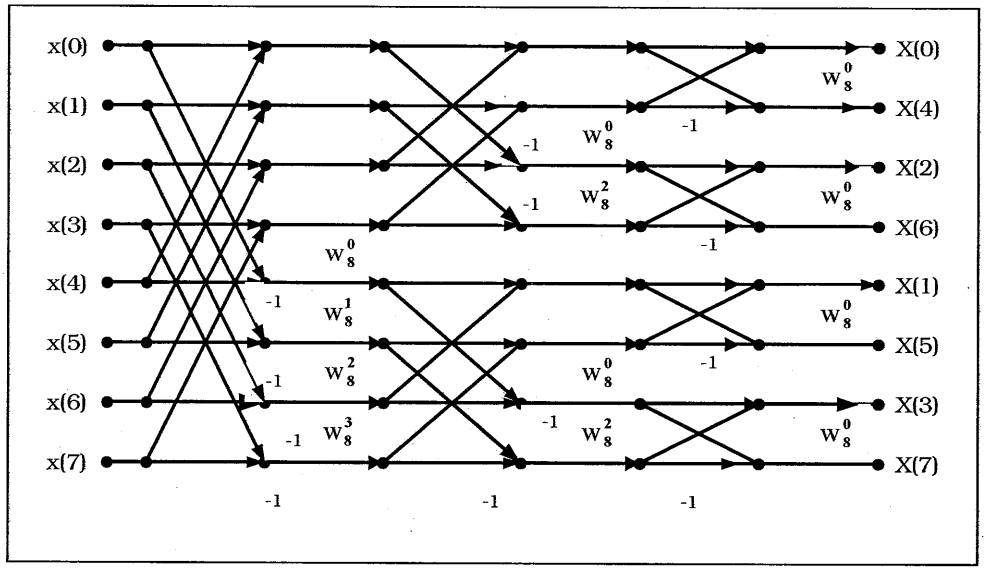


Figure 3.6: Flow graph of decimation in frequency

A pictorial representation of the FFT algorithm showing the radix-2 Butterfly stages for the radix-8 FFT are given in the figures. Similarly, one can construct algorithm for radix-3, radix-5 etc. In general efficient algorithms exist for a combination of powers of prime factors.

Introduction to Digital Filters

4. INTRODUCTION TO DIGITAL FILTERS

4.1 FILTERING

Filtering is a process by which the frequency spectrum of a signal can be modified, reshaped or manipulated according to some desired specifications. Filter is a system that passes certain frequency components and rejects all others to a large extent. The passband of a filter refers to those frequencies, which are passed, while the stopband refers to those frequencies that are blocked. The transition band is in between these. A fast rolloff means that the transition band is very narrow. The division between the passband and stopband is called the cutoff frequency.

Parameters that measure how well the filter performs in frequency domain are:

- The filter must have a **fast rolloff** to separate closely spaced frequency bands.
- For the passband frequencies to move through the filter unaltered there must be **no passband ripple.**
- Lastly, to adequately block the stopband frequencies, it is necessary to have **good stopband attenuation.**

The uses of filtering are manifold, e.g., to eliminate signal contamination such as noise, to remove signal distortion brought about by an imperfect transmission channel or by inaccuracies in measurement, to separate two or more distinct signals which were mixed in order to maximize channel utilization, to resolve signals into their frequency components, to demodulate signals, to convert discrete-time signals into continuous-time signals and to band limit signals.

Digital filter is a digital system that can be used to filter discrete-time signals. It can be implemented by means of software (computer programs), dedicated hardware or a combination of software and hardware. Software digital filters may be implemented using a low level language on a general purpose DSP chip or in terms of a high-level language on a PC or a workstation. At the other extreme, hardware digital filters can be designed using a number of highly specialized interconnected VLSI chips and reprogrammable devices like Field Programmable Gate Array (FPGA). Both software and hardware digital filters can be used to process real-time and non-real-time (recorded) signals, except that the hardware implementation is usually much faster and can process signals whose frequency spectra extend to much higher frequencies.

4.2 ADVANTAGES OF DIGITAL FILTERS

The advantages to be gained by the use of digital filters include the traditional advantage associated with digital system in general.

- **Feasibility**: The characteristics of a digital filter can be changed easily by simply reading in a new set of parameters, which determines the characteristic of the filter. Thus, time varying filters can be easily implemented. More importantly, a single filter structure can be used to realize a multiplicity of filtering functions on time-shared basis and for the design of adaptive filters.
- **Reliability**: The use of digital components offers a more reliable system, since this avoids a number of problems such as component tolerances, influence of spurious environmental signals and large physical size of component for low frequency operation that arise in many analog systems.
- **Modularity**: Digital filters can be highly modularized structure making them suitable for large-scale integration. For e.g., it is technically feasible to put one or two second order digital filter sections with programmable characteristics on a single LSI chip.
- **Dynamic retunability** is possible when it is implemented on FPGAs and DSP chips
- Superior performance (**linear phase** for example)
- Component tolerances are less critical.
- **Accuracy** can be tailored to specific requirements.
- Physical size is small.

In case of analog filters, any desired change in frequency response will be costlier, in terms of required changes in hardware where as in the case of digital filters, the basic overall structure can remain fixed and yet allow changes via programming.

4.3 APPLICATIONS OF DIGITAL FILTERS

- Medical imaging
- Channel equalization
- Enhancement of edges and highlighting smooth features in aerial photographs, satellite weather photos and enhancement of video transmissions from lunar and deep space probes

- Seismic data analysis as required in oil exploration, earthquake measurement and nuclear test monitoring
- Multirate signal processing systems used in Facsimile, teletyping etc.
- Digital filter banks
  - Spectral analysis and signal synthesis
  - Graphic equalizers
  - Brightness and contrast adjustments in image processing
  - Multi-processing systems

4.4 STUDY OF DIGITAL FILTERS

A digital filter, like an analog filter, can be represented by a network, which comprises of a collection of interconnected elements. Analysis of a digital filter is a process of determining the response of the filter network to a given excitation. Design of a digital filter, on the other hand, is a process of synthesizing and implementing a filter network so that a set of prescribed excitation results in a set of desired responses. This chapter is an introduction to the analysis of digital filters. The fundamental concepts of time-invariance, causality etc., as applied to digital filters is discussed.

4.4.1 THE DIGITAL FILTER AS A SYSTEM

The block diagram as shown below can represent a digital filter. Input x(nT) and output y(nT) are the excitation and the response of the filter respectively. The response is related to the excitation by a rule of correspondence called convolution. 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. Systems are described by a signal called the impulse response. Convolution is important because it relates the three signals of interest: the input signal x(n), the output signal y(n) and the impulse response h(n). We can indicate this fact notationally as

$$x(n) * h(n) = y(n)$$

The output signal y(n) can also be obtained by transforming the two signals- x(n) and h(n) into the frequency domain, multiplying them and then transforming the result back into the time domain. This replaces one convolution with two DFTs, a multiplication and an inverse DFT. Even though the intermediate steps are very different, the output is identical to the standard convolution algorithm.

Like other systems, digital filters can be classified as time-invariant or time-variant, causal or non-causal and linear or non-linear.

LINEARITY

A filter is called linear if it has two mathematical properties: **homogeneity** and **additivity**. A third property, shift invariance, is not a strict requirement for linearity, but it is a mandatory property for most DSP techniques. A system is said to be homogeneous if an amplitude change in the input results in an identical amplitude change in the output. That is, if x(n) results in y(n), then kx(n) results in ky(n), for any signal, x(n), and any constant, k.

A system is said to be additive if added signals pass through it without interacting.

A system is said to be shift invariant if a shift in the input signal causes an identical shift in the output signal.

A digital filter is linear if and only if it satisfies the conditions

$$R\alpha x(nT) = \alpha R x(nT)$$

$$R[x_1(nT) + x_2(nT)] = Rx_1(nT) + Rx_2(nT)$$

for all possible values of α and excitations $x_1(nT)$ and $x_2(nT)$

If this condition is violated for any pair of excitations or any constant, then the filter is non-linear.

TIME INVARIANCE

A digital filter is said to time-invariant if its response to an arbitrary excitation does not depend on the time of application of the excitation. As in other types of systems, the response of a digital filter depends on number of internal system parameters. In a time-invariant digital filter, these parameters do not change with time.

Digital filters like other systems have internal storage or memory elements that can store values of signals. Such elements can serve as sources of internal

signals and, consequently, a nonzero filter response maybe produced in a case where the excitation is zero. If all the memory elements of a digital filter are empty or their contents are set to zero, a filter is said to be relaxed. The response of such a filter is zero for all n if the excitation is zero for all n.

Formally, an initially relaxed digital filter with excitation x(nT) and response y(nT), such that x(nT) = y(nT) = 0 for n<0, is said to be time-invariant if and only if

$$Rx(nT - kT) = y(nT - kT)$$

for all possible excitations x(nT) and all integer k. this must be the case, if internal parameters do not change with time.
A filter that does not satisfy the above requirements is said to be time-dependent.

CAUSALITY

A causal digital filter is one whose response at a specific instant is independent of subsequent values of excitation. More precisely, an initially relaxed digital filter in which x(nT) = y(nT) = 0 is said to be causal if and only if

$$Rx_1(nT) = Rx_2(nT) \quad for\ n \le k$$

For all possible distinct excitations $x_1(nT)$ and $x_2(nT)$ such that

$$x_1(nT) = x_2(nT) \quad for\ n \le k$$

Conversely, if

$$Rx_1(nT) \neq Rx_2(nT) \quad for\ n \le k$$

For at least one pair of distinct excitations $x_1(nT)$ and $x_2(nT)$ such that

$$x_1(nT) = x_2(nT) \quad for\ n \le k$$

Then the filter is non-causal.

STABILITY

A filter is said to be stable if its transfer function remains same for any operating period. More precisely, a filter is said to be stable if and only if

$$\sum_{n=0}^{\infty} h(n) \neq \infty \quad \text{and} \quad \sum_{n=0}^{\infty} y(n) \neq \infty$$

where h(n) and y(n) are the impulse response and the output signal respectively.
If the system output tends to infinity as n tends to infinity, then the system is said to be unstable.

4.4.2 CHARACTERIZATION OF DIGITAL FILTERS

Digital filters are characterized in terms of difference equations. On the other hand analog filters are characterized in terms of differential equation. Two types of digital filters can be identified as recursive or **Infinite Impulse Response** (IIR) and nonrecursive **or Finite Impulse Response** (FIR) filters.

IIR FILTERS

The response of an IIR filter is a function of elements in the excitation as well as the response sequence. In case of a linear, time-invariant, causal filter

$$y(nT) = \sum_{i=0}^{N} a_i x(nT - iT) - \sum_{i=0}^{N} b_i y(nT - iT) \qquad (4.1)$$

i.e., if instant nT is taken to be the present, the present response is a function of the present and past N values of the excitation as well as the past N values of the response. If $b_i = 0$, an IIR filter reduces to a FIR filter and essentially the FIR filter is a special case of the IIR one.

FIR FILTERS

FIR filter is characterized by a unit sample response that has finite duration. The response of non-recursive filter at instant nT is of the form

$$y(nT) = f\{\cdots, x(nT - T), x(nT), x(nT + T), \ldots\}$$

If we assume linearity and time-invariance, y(nT) can be expressed as

$$y(nT) = \sum_{i=-\infty}^{\infty} a_i x(nT - iT) \qquad (4.2)$$

where a_i represents constants. Now assuming the filter is causal and noting that x(nT+T), x(nT+2T), ... are subsequent values of the excitation with respect to instant nT,we must have

$a_i = 0$ for $i < -1$

And so $y(nT) = \sum_{i=0}^{\infty} a_i x(nT - iT)$

If, in addition, $x(nT) = 0$ for $n<0$ and $a_i = 0$ for $i>N$

$$y(nT) = \sum_{i=0}^{n} a_i x(nT - iT) + \sum_{i=n+1}^{\infty} a_i x(nT - iT)$$

$$= \sum_{i=0}^{N} a_i x(nT - iT) + \sum_{i=N+1}^{n} a_i x(nT - iT)$$

$$= \sum_{i=0}^{N} a_i x(nT - iT) \qquad (4.3)$$

Therefore a linear, time-invariant, causal, FIR filter can be represented by Nth-order difference equation, which is the order of the filter.

4.4.3 PROPERTIES OF FIR FILTERS

Constant delay filters

A FIR causal filter can be characterized by the transfer function

$$H(z) = \sum_{n=0}^{N-1} h(nT) z^{-n} \qquad (4.4)$$

Its frequency response is given by

$$H(e^{j\omega T}) = M(\omega) e^{j\theta(\omega)} = \sum_{n=0}^{N-1} h(nT) e^{-j\omega nT} \qquad (4.5)$$

where $M(\omega) = \left| H\left(e^{j\omega T}\right) \right|$

and $\theta(\omega) = \arg H\left(e^{j\omega T}\right)$ (4.6)

The phase and group delays of a filter are given by

$$\tau_p = -\left(\frac{\theta(\omega)}{\omega}\right)$$

and $\tau_g = -\frac{d(\theta(\omega))}{d\omega}$ respectively.

For constant phase delay as well as group delay the phase response must be linear, i.e.,

$$\theta(\omega) = -\tau\omega$$

and thus from equations (4.2) and (4.3)

$$\therefore \theta(\omega) = -\tau\omega = \tan^{-1}\left(\frac{\sum_{n=0}^{N-1} h(nT) \sin \omega nT}{\sum_{n=0}^{N-1} h(nT) \cos \omega nT} \right)$$

Consequently, $$\tan \omega\tau = \frac{\sum_{n=0}^{N-1} h(nT)\sin \omega nT}{\sum_{n=0}^{N-1} h(nT)\cos \omega nT}$$

and accordingly,

$$\sum_{n=0}^{N-1} h(nT)\sin(\omega\tau - \omega nT) = 0$$

The solution to the equation can be shown to be

$$\tau = \frac{(N-1)T}{2} \qquad (4.7)$$

$$h(nT) = h[(N-1-n)T] \quad \text{for } 0 \le n \le N-1 \qquad (4.8)$$

Therefore, a FIR filter, unlike a IIR filter, can have a constant phase and group delays over the entire base band. It is only necessary for the impulse response to be symmetrical about the midpoint between samples (N-2)/2 for odd N. The required symmetry is illustrated in figure 4.2 for N=10 and N=11.

In many applications only group delay need be constant, in which case the phase response can have the form

$$\theta(\omega) = \theta_0 - \tau\omega$$

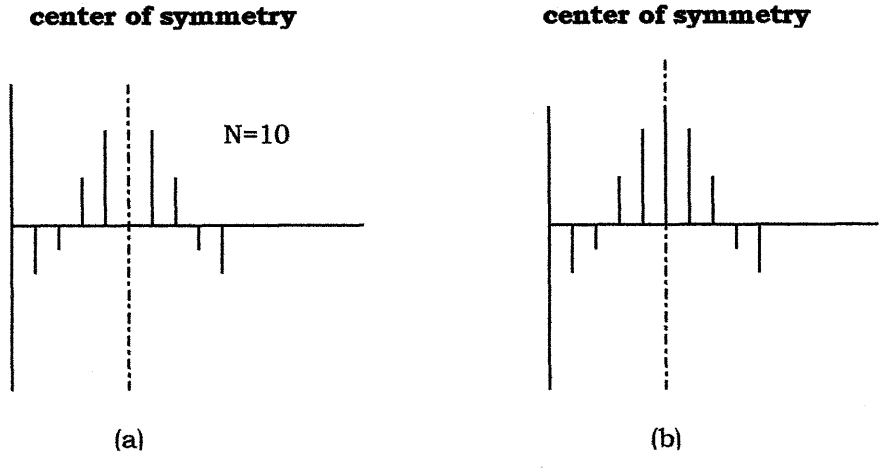


Figure 4.1: Impulse response for constant phase and group delays
(a) N=even (b) N=odd

where θ_0 is a constant. On using the above procedure, a second class of constant-delay, FIR filters can be obtained. When $\theta_0 = \pm 1/2$ the solution is

$$\tau = \frac{(N-1)T}{2}$$

$$h(nT) = -h(N-1-n)T \qquad (4.9)$$

In this case the impulse response is anti-symmetrical about the midpoint between samples (N-2) / 2 and N/ 2 for even N or about sample (N– 1)/ 2 for odd N.

Frequency response

Equations (4.8) and (4.9) lead to some simple expressions for the frequency response. For a symmetrical impulse response with N odd, equation (4.5) can be expressed as

$$H\left(e^{j\omega T}\right) = \sum_{n=0}^{(N-3)/2} h(nT)e^{-j\omega nT} + h\left[\frac{(N-1)T}{2}\right]e^{-j\omega(N-1)T/2} + \sum_{n=(N+1)/2}^{N-1} h(nT)e^{-j\omega nT} \qquad (4.10)$$

By using the equation (4.8) and then letting N-1-n =m, m=n the last summation in the above equation can be expressed as

$$\sum_{n=(N+1)/2}^{N-1} h(nT)e^{-j\omega nT} = \sum_{n=(N+1)/2}^{(N-1)/2} h[(N-1-n)T]e^{-j\omega nT}$$

$$= \sum_{n=0}^{(N-3)/2} h(nT)e^{-j\omega(N-1-n)T} \qquad (4.11)$$

Now from equations (4.8) and (4.10) we get

$$H(e^{j\omega T}) = e^{-j\omega(N-1)T/2}\left\{h\left(\frac{(N-1)T}{2}\right) + \sum_{n=0}^{(N-3)/2} 2h(nT)\cos\left[\omega\left(\left(\frac{N-1}{2}\right)-n\right)T\right]\right\}$$

And hence with (N-1)/2-n =k we have

$$H(e^{j\omega T}) = e^{-j\omega(N-1)T/2}\sum_{k=0}^{(N-1)/2} a_k \cos\omega kT$$

where , $a_0 = h\left[\frac{(N-1)T}{2}\right]$ (4.12)

$$a_k = 2h\left[\left(\frac{N-1}{2}-k\right)T\right] \qquad (4.13)$$

4.5 DESIRABILITY OF LINEAR PHASE

A signal is said to have **zero phase** if it has left-right **symmetry around sample number zero**. A signal is said to have **linear phase** if it has **left-right symmetry**,

but around some other point other than zero. This means that any linear phase signal can be changed into a zero phase signal simply by shifting left or right. Lastly a signal is said to have non-linear phase if it does not have left-right symmetry.

The frequency spectrum of any signal is composed of two parts, the magnitude and the phase. The frequency spectrum of a signal that is symmetrical around zero has a phase that is zero. Likewise, the frequency spectrum of a signal that is symmetrical around some nonzero point has a phase that is a straight line, i.e., a linear phase. Lastly, the frequency spectrum that is not symmetrical has a phase that is not a straight line i.e., it has a non-linear phase.

In application such as speech processing and data transmission, it is desirable to design a filter that has linear phase. For e.g., a linear phase prevents the dispersion of pulses. To see the effect of linear phase, consider a discrete system whose system function is

$$H(e^{j\omega}) = |H(e^{j\omega})| e^{-j\omega\tau} \qquad (4.14)$$

For sinusoidal input $x(n) = A\cos(n\omega_0)$, we have for the output

$$\begin{aligned} y(n) &= \mathrm{Re}\left[H(e^{j\omega_0}) A e^{jn\omega_0}\right] \\ y(n) &= \mathrm{Re}\left[|H(e^{j\omega_0})| e^{-j\tau\omega_0} A e^{jn\omega_0}\right] \\ y(n) &= A|H(e^{j\omega_0})|\cos(n-\tau)\omega_0 \end{aligned} \qquad (4.15)$$

If ω_0 is in the passband of the system then $|H(e^{j\omega_0})|$ ideally, and the filter output is simply the input signal delayed by a certain amount. It is not possible to design a linear phase analog filter from lumped parameter elements, but it is possible to design a linear phase discrete time filter.

4.6 FILTERING OF LONG DATA SEQUENCES

In practical applications involving linear filtering of signals, the input sequence x(n) is often a very long sequence. This is especially true in some real-time signal processing applications concerned with signal monitoring and analysis.

Since linear filtering performed via the DFT involves operations on a block of data, which by necessity must be limited in size due to limited memory of a digital computer, a long input signal sequence must be segmented to fixed size blocks prior to processing. Since the filtering is linear, successive blocks can be processed one at a time via the DFT and the output blocks are fitted together to form the overall output signal sequence.

We now describe two methods for linear FIR filtering a long sequence on a block-by-block basis using the DFT. The input sequence is segmented into blocks and each block is processed via the DFT and IDFT to produce a block of output data. The output blocks are fitted together to form an overall output sequence which is identical to the sequence obtained if the long block had been processed via time domain convolution.

The two methods are called the overlap-save method and the overlap-add method. For both methods we assume that the FIR filter has duration M. The input data sequence is segmented into blocks of L points, where, by assumption, L>>M without loss of generality. The overlap-save method has been discussed here.

In the **overlap-save** method the size of the input data blocks is N = L+M-1 and the size of the DFTs and IDFT are of length N. Each data block consists of the last M-1 data points of the previous data block followed by L new data points to form a data sequence of length N = L+M-1. An N-point DFT is computed for each data block. The impulse response of the FIR filter is increased in length by appending L-1 zeroes and an N-point DFT of the sequence is computed once and stored. The multiplication of the two N-point DFTs {H(k)} and $\{X_m(k)\}$ for the m^{th} block of data yields

$$\hat{Y}_m(k) = H(k)X_m(k) \qquad k = 0,1,2,\ldots,N-1 \tag{4.16}$$

Then the N-point IDFT yields the result

$$\hat{Y}_m(n) = \{\hat{y}_m(0)\hat{y}_m(1)..\hat{y}_m(M-1)\hat{y}_m(M)..\hat{y}_m(N-1)\} \tag{4.17}$$

Since the data record is of length N, the first M-1 points $y_m(n)$ are corrupted by aliasing and must be discarded. The last L points of $y_m(n)$ are exactly the same as the result from linear convolution and, as a consequence,

$$\hat{y}_m(n) = y_m(n), n = M, M+1,\ldots,N-1 \tag{4.18}$$

To avoid loss of data due aliasing, the last M-1 points of each data record are saved and these points become the first M-1 data points of the subsequent record, as indicated above. To begin the processing, the first M-1 points of the first record are set to zero. Thus the blocks of data sequences are

$$\begin{aligned}
x_1(n) &= \{\underbrace{0,0,\ldots,,0,0}_{M-1\ points}, x(0), x(1),\ldots,x(L-1)\} \\
x_2(n) &= \{\underbrace{x(L-M+1),\ldots,x(L-1)}_{M-1\ data\ points}, \underbrace{x(2L),\ldots,x(2L-1)}_{L\ new\ data\ points}\} \\
x_3(n) &= \{\underbrace{x(2L-M+1),\ldots,x(2L-1)}_{M-1\ data\ points\ from\ x_2(n)}, \underbrace{x(2L),\ldots,x(3L-1)}_{L-1\ new\ data\ points}\}
\end{aligned} \tag{4.19}$$

and so forth. The resulting data sequences from the IDFT are given by the equations, where the first M-1 points are discarded due to aliasing and the remaining L points constitute the desired result from linear convolution. This segmentation of the input data and the fitting of the output data blocks together to form the output sequence are graphically illustrated in the figure

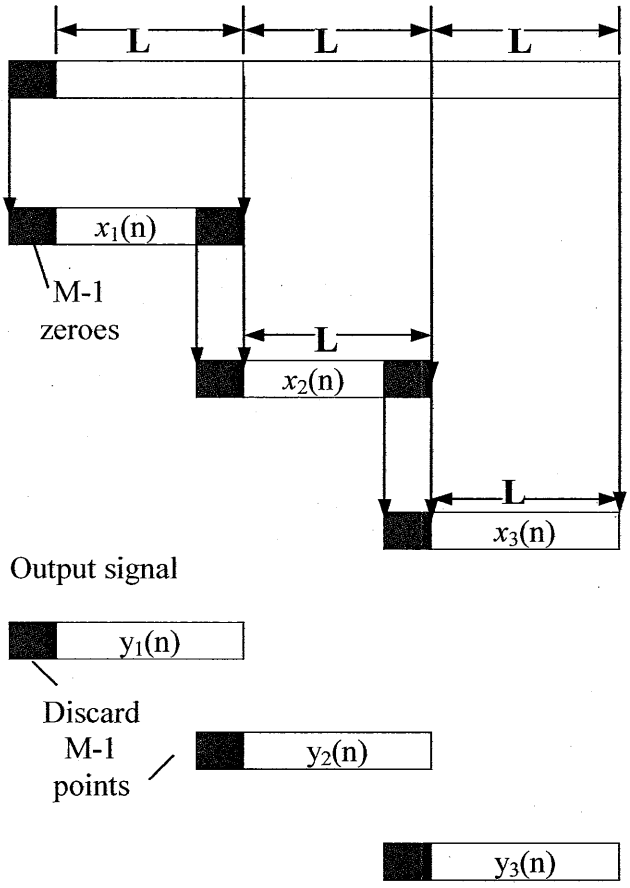


Figure 4.2: Linear FIR filtering by the overlap-save method

4.7 FILTER BANK

In many practical applications of DSP, one is faced with the problem of changing the sampling rate of a signal, either increasing it or decreasing it by some amount. For example, in telecommunication systems that transmit and receive different

types of signals (e.g., teletype, facsimile, speech, video, etc.), there is a requirement to process the various signals at different rates commensurate with the corresponding bandwidths of the signals. The process of converting a signal from a given rate to a different rate is called sampling rate conversion. Systems that employ multiple sampling rates in the processing of digital signals are called *multirate digital signal processing systems.* Multirate signal processing finds several applications in filter banks. Filter banks are systems, which consist of a bank of filters used to process simultaneously different bands of frequencies.

Filter banks are generally categorized as two types, analysis filter banks and synthesis filter banks. An analysis filter bank consists of a set of filters, with system functions $\{H_z(k)\}$, arranged in a parallel bank as shown in the figure(a). The frequency response characteristic of this filter bank splits the signal into a corresponding number of sub-bands. On the other hand, a synthesis filter bank consists of a set of filters with system functions $\{G_k(z)\}$, arranged as shown in the figure (b) with corresponding inputs $\{y_k(n)\}$. The outputs of the filters are summed to form the synthesized signal $\{x(n)\}$.

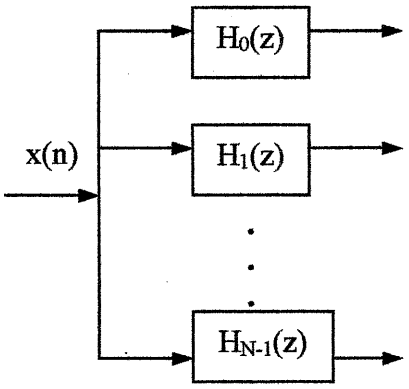


Figure 4.3: Analysis of filter bank

Filter banks are often used for performing spectrum analysis and signal synthesis. When a filter bank is employed in the computation of the DFT of a sequence $\{x(n)\}$, the filter bank is called a DFT filter bank. An analysis filter bank consisting of N filters $\{H_k(z)\}$,k=0,1,…,N-1} is called a uniform DFT filter bank if $H_k(z)$,k=1,2,…,N-1, are derived from a prototype filter $H_0(z)$, where

$$H_k(\omega) = H_0\left(\omega - \frac{2\pi k}{N}\right), k = 1,2,...,N-1$$

Hence the frequency response characteristics of the filter $\{H_k(z)$,k=0,1,2,…,N-1} are simply obtained by uniformly shifting the frequency response of the prototype

filter by multiples of $2\pi/N$. In the time domain the filters are characterized by their impulse responses, which can be expressed as

$$h_k(n) = h_0(n)e^{j2\pi nk/N}, k = 0,1,...,N-1$$

where $\{ h_0(n) \}$ is the impulse response of the prototype filter.

The uniform DFT analysis filter bank can be realized as shown in the following figure where the frequency components in the sequence $\{x(n)\}$ are translated in the frequency to lowpass by multiplying x(n) with the complex exponentials $\exp(-j2\pi nk/N)$, k=1,...,N-1, and the resulting product signals are passed through a lowpass filter with impulse response $\{h_0(n)\}$.Since the output of the lowpass filter is relatively narrow in bandwidth, the signal can be decimated by a factor $D \leq N$. The resulting decimated output signal can be expressed as

$$X_k(m) = \sum_n h_0(mD-n)x(n)e^{-j2\pi nk/N} \begin{cases} k = 0,1,...,N-1 \\ m = 0,1,... \end{cases}$$

where $\{X_k(m)\}$ are samples of the DFT at frequencies $\omega_k=2\pi k/N$.

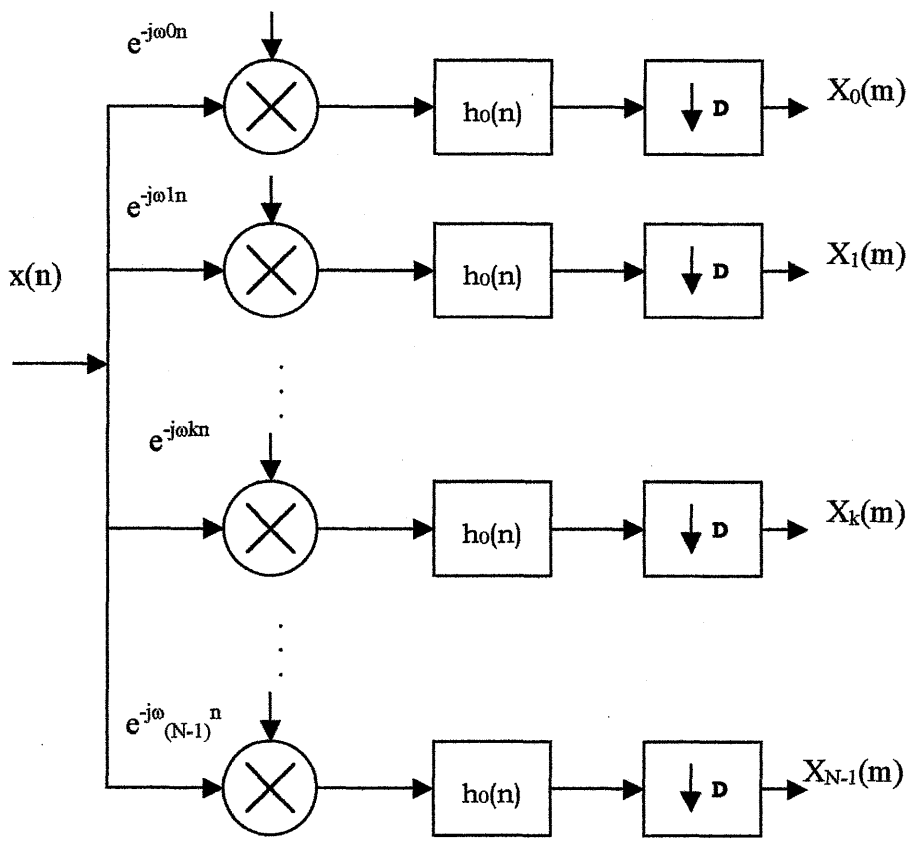


Figure 4.4: Detailed block diagram of analysis filter bank

Design of FIR filters

5. DESIGN OF FIR FILTERS

5.1 ISSUES IN FILTER DESIGN

General process of designing a digital filter involves the following four basic steps

- Approximation problem for filters
- Synthesis of filters
- Implementation of filters
- Application of the filter in the fields of Radar theory, Sonar theory, communications, speech, seismology etc.

Major factors that influence the choice of specific realization are:

- **Computational complexity**: It refers to number of arithmetic operations required to compute an output value for the system. However recent multifold increase in processing power has shifted attention to other factors such as number of times a fetch from the memory is performed and number of times a comparison between two numbers is performed per sample.
- **Memory requirements**: It refers to number of memory locations required to store the system parameters, past inputs, past outputs and any intermediate computed values.

FIR filters are almost entirely restricted to discrete-time implementations. Consequently the design techniques for FIR filters are based on directly approximating the desired frequency response of the discrete-time system. Further more, most techniques for approximating the magnitude response of an FIR system assume a linear phase constraint. The methods used for designing the FIR filters are

- **Windowing method**: This method employs an N-point tapered window sequence to truncate the infinite duration of the ideal desired unit sample response.
- **Frequency sampling method**: This method is fast and simple. It is useful for adaptive filters or for an intermediate stage in a more complicated algorithm where speed is important. Unfortunately, it gives the least control over the total frequency response.
- **Parks-McClellan algorithm**: This algorithm minimizes the Chebyshev error, but the design algorithm can be slow.

In our project the design of FIR filters is based on the windowing method.

5.2 DESIGN OF FIR FILTERS BY WINDOWING

Since $H(e^{j\omega})$, the frequency response of any digital filter, is periodic in frequency, it can be expanded in a Fourier series. The resultant series is of the form

$$H(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h(n)e^{-j\omega n} \qquad (5.1)$$

$$\text{where } h(n) = \frac{1}{2\pi}\int_0^{2\pi} H(e^{j\omega})e^{j\omega n}d\omega \qquad (5.2)$$

The coefficients of the Fourier series h(n) are easily recognized as being identical to the impulse response of a digital filter. There are two difficulties with the representation of these equations for designing FIR filters. First the filter impulse response is infinite in duration since the summation in the equation (5.1) extends to $\pm\infty$. Second the filter is unrealizable because the impulse response begins at $-\infty$; i.e., no finite amount of delay can make the impulse response realizable. Hence the filter resulting from a Fourier series representation of $H(e^{j\omega})$ is an unrealizable IIR filter.

One possible way of obtaining an FIR filter that approximates $H(e^{j\omega})$ would be to truncate the infinite Fourier series at $n=\pm M$. Direct truncation of the series leads to Gibbs phenomenon, however which manifests itself as a fixed percentage overshoot and ripple before and after an approximated discontinuity in the frequency response. Thus for example, in the approximation of such standard filters, as the ideal lowpass or bandpass filter, the largest ripple in the frequency response is about 9% of the size of the discontinuity and its amplitude does not decrease with increasing impulse response duration i.e., including more and more terms in the Fourier series does not decrease the amplitude of the largest ripple. Instead, the overshoot is confined to a smaller and smaller frequency range as N is increased. Since any reasonable design technique must be capable of designing good approximations to ideal lowpass filters, direct truncation of equation (5.1) is not a reasonable way of obtaining an FIR filter.

A successful way of obtaining an FIR filter is to use the weighting sequence called w(n), called a window, to modify the Fourier coefficients h(n) in equation (5.1) to control convergence of the Fourier series .

The technique of windowing is illustrated. The desired periodic frequency response $H(e^{j\omega})$ and its Fourier series coefficients h(n) are as shown in figure 5.1(a). Figure 5.1(b) shows a finite duration weighting sequence w(n) with Fourier transform $W(e^{j\omega})$. $W(e^{j\omega})$, for most reasonable windows, consists of a central lobe, which contains most of the energy of the window, and the sidelobes, which decay rapidly. To produce an FIR approximation to $H(e^{j\omega})$, the sequence

$\hat{h}(n) = h(n).w(n)$ is formed. Outside the interval $-M \le n \le M, \hat{h}(n)$ is zero exactly. Figure 5.1(b) shows $\hat{h}(n)$ and its Fourier transform $\hat{H}(e^{j\omega})$ which is readily seen to be the circular convolution of the $H(e^{j\omega})$ and $W(e^{j\omega})$, since $\hat{h}(n)$ is the product of the sequences $h(n)$ and $w(n)$.Finally, Figure 5.1(e) shows the realizable sequence g(n), which is a shifted version of $\hat{h}(n)$ and may be used as the desired filter impulse response.

There are several noteworthy effects of windowing the Fourier coefficients of the filter on the resulting frequency response. A major effect is that discontinuities in $H(e^{j\omega})$ become transition bands between values on either side of the discontinuity. Since the final frequency response of the filter is the circular convolution of the ideal frequency response with the window's frequency response, it is clear that the width of these transition bands depend on the width of the mainlobe of $W(e^{j\omega})$. A secondary effect of windowing is that the ripples from the sidelobes of $W(e^{j\omega})$ produce approximation errors for all ω. Finally since the filter frequency response is obtained via a convolution relation, it is clear that the resulting filters are never optimal in any sense, even though the windows from which they are obtained may satisfy some reasonable optimality criterion.

The discussion above leads to the questions of what are the desirable window characteristics and how closely are they attained in practice. The desirable window characteristics are

- **Small width of the mainlobe** of the frequency response of the window containing as much of the total energy as possible.
- **Sidelobes** of the frequency response that **decrease in energy** rapidly as ω tends to π.

There have been many window schemes proposed that approximate the desired characteristics. The window functions namely, rectangular window, generalized Hamming window and the Kaiser window have been studied. These are summarized below.

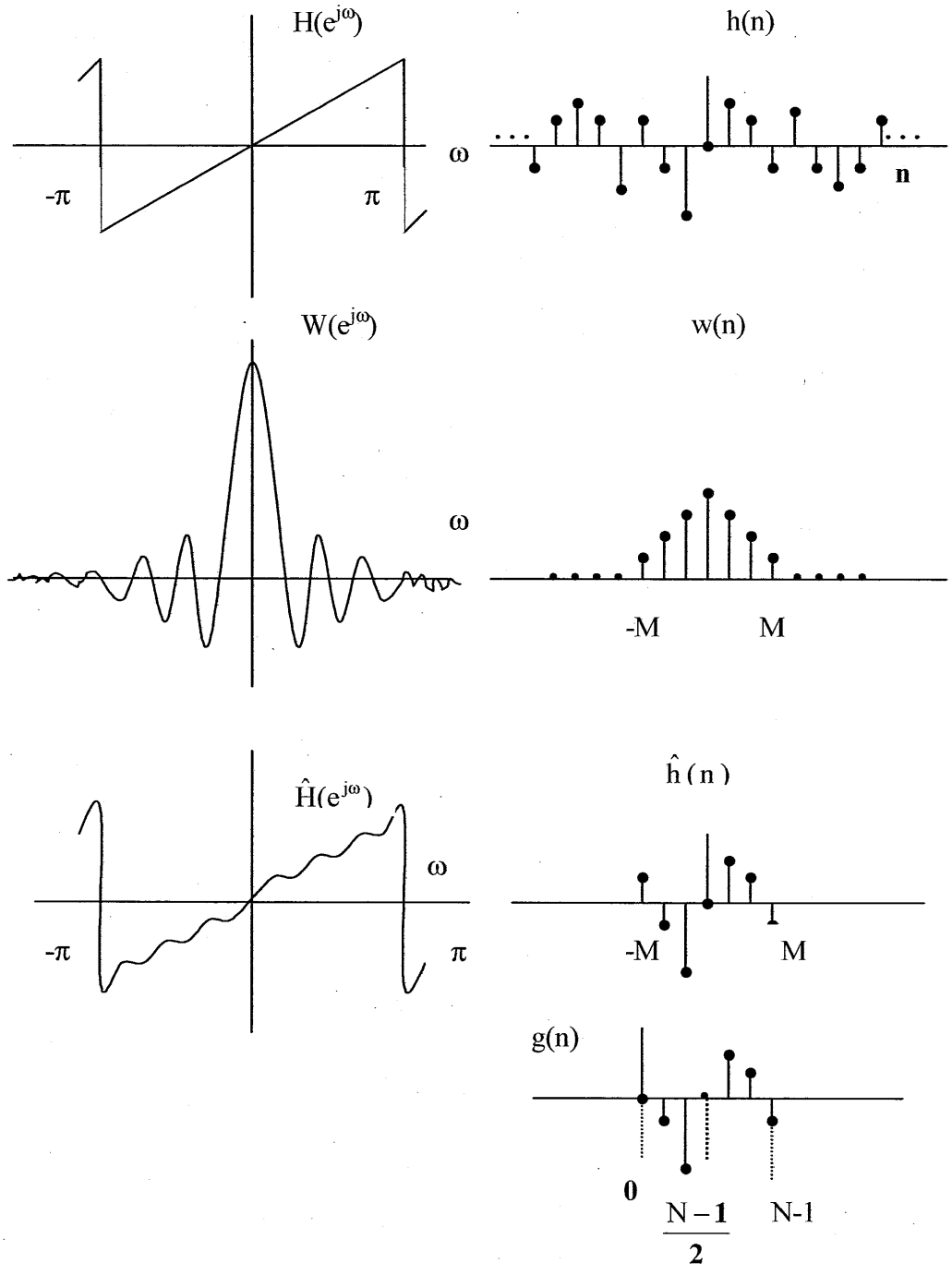


Figure 5.1: Illustration of windowing

RECTANGULAR WINDOW

The M point rectangular window, which corresponds to direct truncation (with no modification) of the Fourier series, has the weighting function

$$w(n) = 1 \quad for\ 0 \le M$$
$$= 0 \quad elsewhere \tag{5.3}$$

The frequency response of this window is

$$W\left(e^{j\omega}\right) = \frac{\sin(\omega M/2)}{\sin(\omega/2)} \tag{5.4}$$

'GENERALIZED' HAMMING WINDOW

The second type of window namely the generalized Hamming window has the form

$$w(n) = \alpha + (1-\alpha)\cos(2\pi n/M) \quad for\ 0 \le n \le M$$
$$= 0 \quad elsewhere \tag{5.5}$$

where α is in the range $0 \le \alpha \le 1.0$. If $\alpha = 0.54$, the window is called Hamming window; if it is 0.5 it is called Hanning window.

The frequency response can be obtained by the observation that the window can be represented as the product of the rectangular window and an infinite window of the form of equation that describes the above window for all n.

The frequency response of the generalized Hamming window is therefore the convolution of the frequency response of the rectangular window with an impulse train, which can be written as

$$w_{ham}(n) = w_{rect}(n)\left[\alpha + (1-\alpha)\cos\left(\frac{2\pi n}{M}\right)\right]$$

$$W_{ham}\left(e^{j\omega}\right) = W_{rect}\left(e^{j\omega}\right)\left[\alpha u_0(\omega) + \frac{1-\alpha}{2}u_0\left(\omega - \frac{2\pi}{M}\right) + \frac{1-\alpha}{2}u_0\left(\omega + \frac{2\pi}{M}\right)\right] \tag{5.6}$$

$$W_{ham}\left(e^{j\omega}\right) = \alpha W_{rect}\left(e^{j\omega}\right) + \frac{(1-\alpha)}{2}W_{rect}\left(e^{j\left(\omega - \frac{2\pi}{M}\right)}\right) + \frac{(1-\alpha)}{2}W_{rect}\left(e^{j\left(\omega + \frac{2\pi}{M}\right)}\right)$$

KAISER WINDOW

The trade-off between the mainlobe width and sidelobe area can be quantified by seeking the window function that is maximally concentrated around $\omega = 0$ in the frequency domain. A near optimal filter, as suggested by Kaiser can be formed

using the zeroth-order modified Bessel function of the first kind, a function that is much easier to compute. The Kaiser window is defined as

$$w(n)=\begin{cases}\dfrac{I_0\left[\beta\left(1-\left[(n-\alpha)/\alpha^2\right]\right)^{1/2}\right]}{I_0(\beta)} & \text{for } 0\le n\le M\\ 0 & \text{elsewhere}\end{cases} \qquad (5.7)$$

where $\alpha =$ M/2, and $I_0(.)$ represents the zeroth-order modified Bessel function of the first kind given by $I_0(x)=1+\sum_{k=1}^{\infty}\left[\frac{1}{k!}\left(\frac{x}{2}\right)^k\right]^2$

In contrast to the other window, the Kaiser window has two parameters like, the length (M+1) and a shape parameter β. β is a constant that specifies a frequency response trade off between the peak height of the sidelobe ripples and the width or energy of the mainlobe. By varying M+1 and β, shape can be adjusted to trade side-lobe amplitudes for main-lobe width. When β = 0, the Kaiser window reduces to a rectangular window. The values of M and β can be predicted in advance to meet a specific frequency–selective filter application. Given that the peak approximation error δ is fixed, β is determined and the passband frequency ω_p is defined such that $|H(e^{j\omega})| \ge 1-\delta$. The stopband cutoff frequency ω_s is defined to be the lowest frequency such that $|H(e^{j\omega})| \le \delta$ .The transition region has a width $\Delta\omega = \omega_s - \omega_p$ for the lowpass approximation.

Defining $A = -20\log_{10}\delta$, the value of β needed to achieve a specified value of A is given by

$$\begin{aligned}\beta &= 0.1102(A-8.7) && A>50\\ &= 0.5942(A-21)^{0.4}+0.007886(A-21) && 21\le A\le 50\\ &= 0.0 && A<21\end{aligned} \qquad (5.8)$$

The value of M is given by

$$M=\frac{A-8}{2.285\Delta\omega} \qquad (5.9)$$

A closed form expression for the frequency response of the digital Kaiser window has not been obtained, although Kaiser has shown that, in the continuous time case, the frequency response is proportional to

$$\frac{\sin\left(\beta\sqrt{(\omega/\omega_\beta)^2-1}\right)}{\sqrt{(\omega/\omega_\beta)^2-1}} \qquad (5.10)$$

Where ω_β is approximately the spectral width of the central lobe of the frequency response because an analytical result for the frequency response of the Kaiser

window is not available, actual plots of it's frequency response will serve to illustrate the properties of this window.

Type of window	Window functions
Rectangular window	$w(n)=1 \quad for\ 0 \le M$ $=0 \quad elsewhere$
Bartlett window	$w(n)=2n/M \quad for\ 0 \le n \le M/2$ $=2-2n/M \quad for\ M/2 \le n \le M$ $=0 \quad elsewhere$
Hanning window	$w(n)=0.5-0.5\cos(2\pi n/M) \quad for\ 0 \le n \le M$ $=0 \quad elsewhere$
Hamming window	$w(n)=0.54-0.46\cos(2\pi n/M) \quad for\ 0 \le n \le M$ $=0 \quad elsewhere$
Blackmann window	$w(n)=0.42-0.56\cos(2\pi n/M)+0.08\cos(4\pi n/M) \quad for 0 \le n \le M$ $=0 \quad elsewhere$
Kaiser window	$w(n)=\begin{cases} \dfrac{I_0\left[\beta\left(1-\left[(n-\alpha)/\alpha^2\right]\right)^{1/2}\right]}{I_0(\beta)} & for 0 \le n \le M \\ 0 & elsewhere \end{cases}$

5.2.1 RELATIONSHIP OF THE KAISER WINDOW TO OTHER WINDOWS

The basic principle of the window design method is to truncate the ideal impulse response with finite-length window. The corresponding effect in the frequency domain is that the ideal frequency response is convolved with the Fourier transform of the window. If the ideal filter is a lowpass filter, the discontinuity in its frequency response is smeared as the mainlobe of the Fourier transform of the window moves across the discontinuity in the convolution process. To a first approximation, the width of the resulting transition band is determined by the width of the mainlobe of the Fourier transform of the window, and the passband and stopband ripples determined by the sidelobes of the Fourier transform of the window. Because the passband and stopband ripples are produced by the integration of the symmetric window sidelobes, the ripples in the passband and the stopband are approximately the same. Furthermore, to a very good approximation, the maximum passband and stopband are not dependent on M and can be changed only by changing the shape of the window used. This is illustrated by Kaiser's

formula for the window shape parameter, which is independent of M. The comparison of various windows can be seen in the following table.

Type of Window	Approximate width of mainlobe	Peak sidelobe amplitude (relative) (dB)	Peak approximation error $20\log_{10}\delta$ (dB)	Equivalent Kaiser window B	Transition width of equivalent Kaiser window
Rectangular	$4\pi/(M+1)$	-13	-21	0	$1.8\pi/M$
Bartlett	$8\pi/M$	-25	-25	1.33	$2.37\pi/M$
Hanning	$8\pi/M$	-31	-44	3.86	$5.01\pi/M$
Hamming	$8\pi/M$	-41	-53	4.86	$6.27\pi/M$
Blackman	$12\pi/M$	-57	-74	7.04	$9.19\pi/M$

The table gives a comparison of various windows. The last two columns of the table compare the Kaiser window with the other windows. The fifth column gives the Kaiser window shape parameter that yields the same peak approximation error (δ) as the window indicated in the first column. The sixth column shows the corresponding transition width for filters designed with for the filters designed with the Kaiser window. This formula would be a much better predictor of the transition width for other windows than would the mainlobe width given in the third column of the table.

5.2.2 ISSUES WITH WINDOWING

Although windowing appears to be the most attractive technique for designing FIR filters, there are several issues that often impede their use in many cases. One problem is that one needs a closed form expression for the Fourier series coefficient h(n) where

$$h(n) = \frac{1}{2\pi}\int_0^{2\pi} H(e^{j\omega})e^{j\omega n}d\omega \qquad (5.11)$$

When $H(e^{j\omega})$ is complicated or cannot easily be put in a closed form mathematical expression (and sometimes even when it can) this equation is often cumbersome or difficult to evaluate. Without an expression for the unwindowed coefficients, it is difficult to contemplate the use of windows.

Another issue with windows is that windowing offers little design flexibility. For example, in the design of a lowpass filter, the passband edge frequency generally cannot be specified exactly since the window "smears" the discontinuity in frequency. Thus the ideal lowpass filter frequency response $H(e^{j\omega})$

with cutoff frequency f_c is smeared by the window to give a frequency response with passband cutoff frequency f_1 and stopband cutoff frequency f_2. Although in many designs such as the ideal lowpass filter, this effect can be compensated, in more complicated filters this is no trivial matter.

However the advantage of windowing method of filter design is that it is reasonably straightforward to obtain the FIR filter with **minimal computational effort**. Major reason for the success of windows is their **simplicity** and the fact that closed form expressions is often available for the window coefficients. They are **stable**, produce few surprises, and can be pushed to **good performance** levels. These frequency domain characteristics are obtained at the expense of poor performance in the time domain, including excessive ripple and overshoot in the impulse response. When carried out by standard convolution, these filters are easy to program, but slow to execute. The FFT can be used to dramatically increase the speed of these filters.

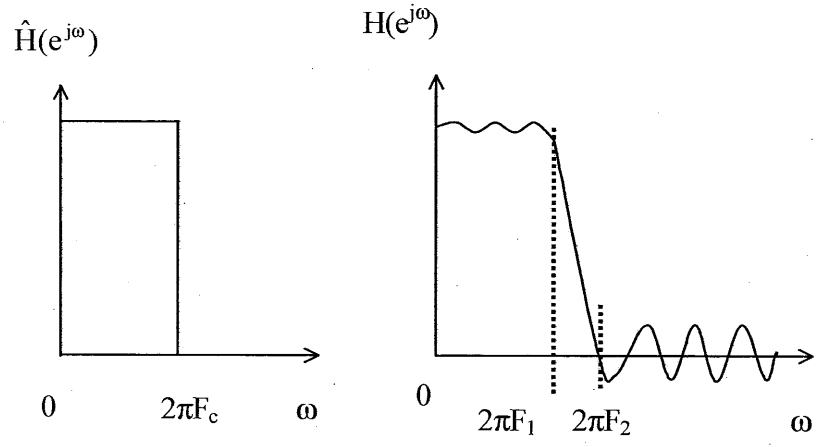


Figure 5.2: Smearing due to windows

5.2.3 DESIGN OF A LOWPASS FILTER

A lowpass filter is designed to block all frequencies above the cutoff frequency (the stopband), while passing all frequencies below (the passband). The ideal lowpass filter with generalized linear phase has the frequency response

$$H_{lp}(e^{j\omega}) = e^{j\omega M/2} \qquad |\omega| \le \omega_c$$
$$= 0 \qquad \omega_c < |\omega| < \pi$$

The corresponding impulse response can be found by evaluating the inverse transform of $H_{lp}(e^{j\omega})$

$$h_{lp}(n)=\frac{1}{2\pi}\int_{-\omega_c}^{\omega_c} e^{-j\omega M/2}e^{j\omega n}=\left(\frac{\sin[\omega_c(n-M/2)]}{\pi(n-M/2)}\right) \qquad for\ n\neq M/2$$

$$=\frac{\omega_c}{\pi} \qquad for\ n=M/2$$

It is easily shown that h_{lp} (M-n) = h_{lp}(n), so if we use a symmetric window in the equation a linear phase system is obtained.
Thus, h(n) is

$$h(n)=h_{lp}(n)w(n)$$

where w(n) can be any of the windows specified above and are used to truncate any ideal impulse response to obtain a causal FIR approximation.

To design a filter, three parameters are to be selected: the cutoff frequency, bandwidth and the filter tap length M. For computational purpose, the cutoff frequency ω_c is a fraction of sampling frequency where the sampling frequency is chosen as twice the bandwidth and thus it is made to lie between 0 and 0.5.

With the use of the design formulas for the Kaiser window, it is straightforward to design lowpass FIR filter to meet prescribed specifications. The procedure is as follows

Step 1: First the specifications must be established, namely ω_s (stopband frequency), ω_p (passband frequency) maximum tolerable approximation error δ. It must be the same both in the passband and the stopband.

Step 2: The cutoff frequency of the underlying ideal lowpass filter must be found. Due to the symmetry of the approximation at the discontinuity of $H_d(e^{j\omega})$, we would set

$$\omega_c=\frac{\omega_s-\omega_p}{2}$$

Step 3:

$$\Delta\omega=\omega_s-\omega_p$$

$$A=-20\log_{10}\delta$$

β And M are calculated.

$\alpha = M/2$

Step 4:

$$w(n)=\begin{cases}\dfrac{I_0\left[\beta\left(1-\left[(n-\alpha)/\alpha^2\right]\right)^{1/2}\right]}{I_0(\beta)} & for\ 0\le n\le M\\ 0 & elsewhere\end{cases}$$

5.2.4 DESIGN OF A HIGHPASS FILTER

Windowing method of design of filters is not restricted to the design of lowpass filters only. It to also be used to design highpass, bandpass and bandstop filters. A common strategy in filter design is to device the lowpass filter first and then transform it to highpass, bandpass and bandstop.

A highpass filter is designed to block all frequencies below the cutoff frequency (the stopband), while passing all frequencies above (the passband).

A delta function impulse response passes the entire signal, while a lowpass impulse response passes only the low frequency components. By superposition, a filter kernel consisting of a delta function minus the lowpass filter kernel will pass the entire signal minus the low frequency components. Thus a highpass filter is obtained.

The ideal highpass filter with generalized linear phase has the frequency response

$$H_{hp}(e^{j\omega}) = 0 \qquad |\omega| \le \omega_c$$
$$= e^{-j\omega M/2} \qquad \omega_c < |\omega| < \pi$$

The corresponding impulse response can be found by evaluating the inverse transform of $H_{hp}(e^{j\omega})$ or we can observe that

$$H_{hp}(e^{j\omega}) = e^{-j\omega M/2} - H_{lp}(e^{j\omega}),$$

where $H_{lp}(e^{j\omega})$ is the impulse response of a lowpass filter as given in the above equation.

Thus, $h_{hp}(n)$ is

$$h_{hp}(n) = \left(\frac{\sin\pi(n-M/2)}{\pi(n-M/2)}\right) - \left(\frac{\sin\omega_c(n-M/2)}{\pi(n-M/2)}\right) \qquad for\ n \neq M/2$$
$$= 1 - \frac{\omega_c}{\pi} \qquad for\ n = M/2$$

It is easily shown that $h_{hp}(M-n) = h_{hp}(n)$, so if we use a symmetric window in the equation a linear phase system is obtained.

Thus, h(n) is

$$h(n) = h_{hp}(n)\,w(n)$$

where w(n) can be any of the windows specified above and are used to truncate any ideal impulse response to obtain a causal FIR approximation. Once h[n] is obtained the further design procedure employed is same as that adopted in case of a LPF.

Kaiser window method for the design of highpass filter can be used to design filters to meet the required specifications. Suppose that we wish to design a filter to meet the highpass specifications

$$|H(e^{j\omega})| \le \delta_2, |\omega| \le \omega_s$$
$$1-\delta_1 \le |H(e^{j\omega})| \le 1+\delta_1, \omega_p \le |\omega| \le \pi$$

where ω_s is the stopband frequency
ω_p is the passband frequency
δ_1 is the peak approximation error allowed in the passband
δ_2 is the peak approximation error allowed in the stopband
Since the ideal response also has discontinuity, we can apply Kaiser's formulae to estimate the values of β and M for the specified pass and stopband frequencies and peak approximation errors.

5.2.5 DESIGN OF A BANDPASS FILTER

Windowing method of design of filters can be extended to the design of bandpass filters when the bandwidth of the required bandpass filter and the center frequency of the band or the lower cutoff frequency and upper cutoff frequency are specified. A bandpass filter is one that passes only a band of frequencies between a lower cutoff frequency and an upper cutoff frequency and suppresses the other frequencies.

Design of a bandpass filter given the upper and lower cutoff frequencies can be achieved using the following relation

$$h_{bp}(n) = \left\{ \frac{\sin\omega_u(n-\alpha) - \sin\omega_l(n-\alpha)}{\pi(n-\alpha)} \right\} \qquad for\ n \neq \alpha$$
$$= \frac{\omega_2 - \omega_1}{\pi} \qquad for\ n = \alpha$$

It may be observed that $h_{bp}(M-n) = h_{bp}(n)$, so if we use a symmetric window in the equation a linear phase system is obtained.
Thus, h(n) is

$$h(n) = h_{bp}(n)w(n)$$

where w(n) can be any of the windows specified above and are used to truncate any ideal impulse response to obtain a causal FIR approximation. Once h(n) is obtained the further design procedure employed is same as that adopted in case of a LPF.

Bandpass filter can also be obtained by convolving the lowpass and highpass filter kernels.

Bandpass filters are the **frequency-translated** version of lowpass filters. Translation of signal in lowpass signal to a frequency band centered on a carrier frequency f_c can be achieved by using one of the frequency shift property of Fourier transformations. This property states that:

$$\text{If } x(n) \Leftrightarrow X(f), \text{ then } x(n)e^{-j2\pi f_c n} \Leftrightarrow X(f - fc)$$

Design of a bandpass filter using this property is done as follows: A lowpass filter of bandwidth half the required bandwidth is designed using the Kaiser's method of design of lowpass filter for the specified peak approximation error, transition bandwidth and other specifications. The filter obtained is translated to the required band using the above relation.

5.2.6 DESIGN OF A BANDSTOP FILTER

Bandstop filter is a filter that attenuates a band of frequencies and passes all other frequencies. Bandstop filter is specified by the lower and upper cutoff frequencies of the band of frequencies to be attenuated. The bandstop filter can be obtained by the addition of two filters one being a highpass filter and the other being a lowpass filter. The lowpass filter should be designed to have an upper cutoff frequency equal to the lower cutoff frequency of the bandstop filter and the highpass filter should be designed to have a lower cutoff frequency equal to the upper cutoff frequency of the bandstop filter. The filter coefficients of the bandstop filter is given by

$$h_{bs}(n) = \left\{ \frac{\sin\omega_l(n-\alpha) + \sin\pi(n-\alpha) - \sin\omega_u(n-\alpha)}{\pi(n-\alpha)} \right\} \qquad \text{for } n \neq \alpha$$

$$= \frac{\omega_1 + \pi - \omega_2}{\pi} \qquad \text{for } n = \alpha$$

Using a symmetric window in the equation a linear phase system is obtained.

$$h(n) = h_{bs}(n)w(n)$$

where w(n) can be any of the windows specified above and are used to truncate any ideal impulse response to obtain a causal FIR approximation. Once h(n) is

obtained the further design procedure employed is same as that adopted in case of a LPF.

5.2.7 DESIGN OF FILTER BANK

Filter bank is a system, which consists of a bank of filters used to process simultaneously, different bands of frequencies. A filter bank with L bandpass filters can be designed using the specifications of upper and lower cutoff frequencies for all the filters. These bandpass filters can be designed as mentioned in the section 5.3.5. The filter co-efficients for each of the bandpass filters are determined and summed up to obtain the filter bank co-efficients.

Design of a bandpass filter given the upper and lower cutoff frequencies can be achieved using the following relation

$$h_{bpk}(n) = \left\{ \frac{\sin \omega_{2k}(n-\alpha) - \sin \omega_{lk}(n-\alpha)}{\pi(n-\alpha)} \right\} \qquad for\ n \neq \alpha$$

$$= \frac{\omega_{2k} - \omega_{1k}}{\pi} \qquad for\ n = \alpha$$

Using a symmetric window in the equation a linear phase system is obtained. Thus, $h_k(n)$ is

$$h_k(n) = h_{bpk}(n) w(n)$$

where $h_k(n)$ is the filter co-efficients of the k^{th} filter and w(n) can be any of the windows specified above. Once $h_k(n)$ is obtained the filter bank is designed as

$$h(n) = \sum_{k=1}^{L} h_k(n)$$

The above design procedures are implemented using software in the implementation phase of our project and applied in the case study of a spectral line observation.

Implementation

6. IMPLEMENTATION

The second phase of the project is the implementation. This section gives a brief introduction of the tools and techniques used in the implementation of the project.

6.1 COMMENTS ON PROGRAMMING STYLE

The four common measures of DSP software are reliability, maintainability, extensibility and efficiency.

A **reliable** program is one that seldom (if ever) fails. This especially important in DSP because tremendous amounts of data are often processed using the same program. If the program fails due to one sequence of data passing through the program, it may be difficult or impossible, to ever determine what caused the problem.

Since most programs of any size occasionally fail, a **maintainable** program is one that is easy to fix. A truly maintainable program is one that can be fixed by someone other than the original programmer. It is also some times important to be able to maintain a program on more than one type of processor, which means that in order for a program to be truly maintainable, it must be portable.

An **extensible** program is one that can be easily modified when the requirements change, new functions need to be added, or new hardware features need to be exploited.

An **efficient** program is often the key to a successful DSP implementation of a desired function. An efficient DSP program will use the processing capabilities of the target computer (whether general purpose or dedicated) to minimize the execution time. In a typical DSP system it often means minimizing the number of operations per unit sample or maximizing the number of operations per second usually means a lower overall system cost as fast computers typically cost more than slow computers. For e.g., it could be said that the FFT algorithm reduced the cost of speech processing (both implementation cost and development cost) such those inexpensive speech recognition and generation processors are now available for use for the general public.

Unfortunately, DSP programs often forsake maintainability and extensibility for efficiency. Such is the case for most currently available programmable signal processing integrated circuits. These devices are usually programmed in assembly language in such a way that it is often impossible for changes to be made by any one but the original programmer and after a few months even the original programmer may have to rewrite the program to add

additional functions. Often a compiler is not available for the processor or the processor's architecture is not well suited to efficient generation of code from a compiled language. The current trend in programmable signal processors appears to be toward high-level languages. Many of the DSP chip manufacturers are supplying C compilers for their more advanced products.

The following five coding situations will lessen the software quality of DSP programs:

- Functions that are too big or have several purposes.
- A main program that does not use functions.
- Functions that are tightly bound to the main program.
- Lack of meaningful variable names and comments.

An oversized function might be defined as one that exceeds two pages of source listing. A function with more than one purpose lacks strength. Other programs and programmers can use a function with one clearly defined purpose. Functions with many purposes will find limited utility and limited acceptance by others.

A main program that does not use function will often result in an extremely long and hard to understand program. Also, because complicated operations often can be independently tested when placed in short functions, the program may be easier to debug. However taking this rule to the extreme can result in functions that are tightly bound to the main program thus weakens the entire program.

A program that does not use meaningful variables and comments is guaranteed to be very difficult to understand and maintain.

6.2 LINUX OPERATING SYSTEM

Linux is one of the most important and powerful operating system that exists to date. The popularity and success of LINUX is because it is **portable**, **machine independent**, it aids **multi-user operations**, **background processing**, **hierarchical file systems**. It offers an excellent variety of tools for software development. With multitasking, multi-user capabilities, TCP/IP support, graphical X windows, LINUX has become a network operating system of choice of PC users and PC based businesses world over. LINUX was developed in early 1990's by Linus Torvald at the University of Helsinki along with other programmers around the world. LINUX is a PC version of Unix OS that has been used for decades on mainframes and minicomputers. LINUX provides the advantages of speed, efficiency and flexibility, which distinguish it from other operating systems.

LINUX can be generally divided into four major components: the kernel, the shell, the file structure and the utilities. Kernel is the core program that runs

programs and manages the hardware. Shell provides an interface for the user. File structure organizes the way files are stored on a storage device such as the disk. LINUX has a great number of utilities, which are classified as editors, filters and communication programs.

6.3 GNUPLOT

Gnuplot is a **powerful interactive plotting program** inbuilt on Linux platform. Gnuplot is a command-driven interactive function-plotting program. If files are given, gnuplot loads each file with the load command, in the order specified. Gnuplot exits after the last file is processed. Gnuplot plots any number of functions, built up of C operators, C library functions, and some of the functions not present in C like **(square), sgn( ), etc. It also supports plotting of data files. Gnuplot can also be used to compare actual data to theoretical curves. Gnuplot supports user-defined X and Y ranges (optional auto-ranging), smart axes scaling, smart tic marks, labeling of X and Y-axes, user-defined constants and functions.

Some of the commonly used commands of gnuplot are:

- **plot**-This command plots the function or the data to be plotted on the screen.

  Syntax-plot <function>
  plot "datafile" using column_m:column_n

- **multiplot-** This command is used to plot many plots on the same page.
- **replot-** This command is used to overwrite on an already existing plot
- **set-** This command is used to set the environment of the plot as required by the user.
  - set xrange[a:b]- sets the x-range of the plot from a to b.
  - set yrange[a:b]- sets the y-range of the plot from a to b.
  - set auto- sets the plot environment to default settings.
  - set xlabel, set ylabel, set title- are some of the other commands using set command.
  - set term- is another very important command used to set the terminal environment. Gnuplot provides the x11 terminal type for use with X servers. This terminal type is set automatically at startup if the DISPLAY environment variable is set, if the TERM environment variable is set to xterm, or if the *-display command* line option is used. The terminal can be set to postscript environment or jpeg environment or any other environment. For terminal type x11,

gnuplot accepts the standard X Toolkit options and resources such as geometry, font and background.

6.4 FFTW

A **library** from public domain for **Fast Fourier Transform (FFT)** called the FFTW was used to improve the computational efficiency. Generally in FFT algorithms speed was the direct consequence of clever algorithms that minimized the number of arithmetic computations. In present day general-purpose microprocessors, however, the performance of a program is mostly determined by complicated interactions of the code with the processor pipeline and by the structure of the memory. Design for performance under these conditions requires an intimate knowledge of the computer architecture. FFTW is an adaptive high performance implementation of **Cooley-Tukey** Fast Fourier Transform algorithm and Radix of the higher prime factors available as a library callable from C or Fortran programs

The user must first create a plan, which can be then used at will. In FFTW the computation of the transform is accomplished by an executor that consists of highly optimized composable blocks of C code called codelets. A codelet is a specialized piece of code that computes part of the transform. A combination of codelets applied by the executor is specified by a special data structure called a plan. The plan is determined at runtime, before the computation begins by a planner, which uses a dynamic programming algorithm to find fast composition of codelets. The planner tries to minimize the actual execution time and not the number of floating point operations. Consequently, the planner measures the run-time of many plans and selects the faster. In this current implementation plans can also be saved to disk and used at a later time.
A simple example of FFTW's use is shown below.

```
Fftw_plan plan;
COMPLEX A[n], B[n]
/*Plan the computation */
plan = fftw_create _plan (n);
/*Execute the plan */
Fftw(plan, A)
/*The plan can be reused for other inputs of size n*/
Fftw (plan);
```

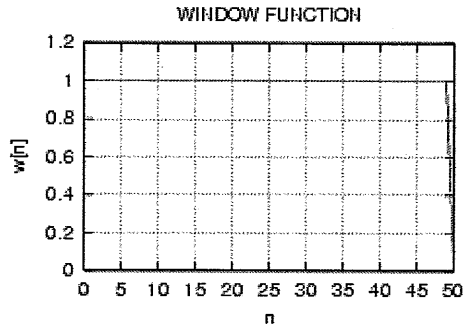
Realization of FIR Filters Using Software

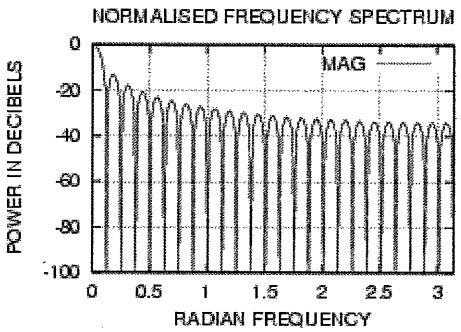
7. REALIZATION OF FIR FILTERS USING SOFTWARE

SPECTRUM OF THE VARIOUS WINDOWS

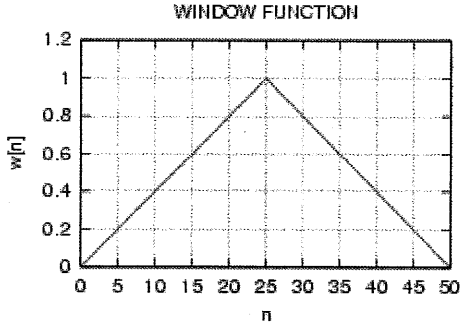
INPUT: TAPLENGTH = 50

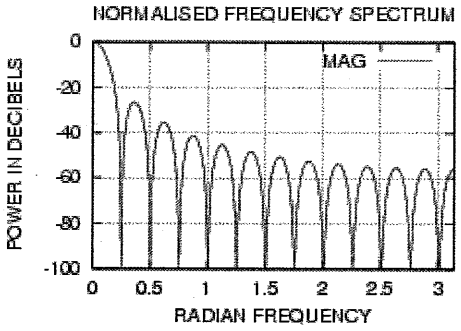
RECTANGLULAR WINDOW



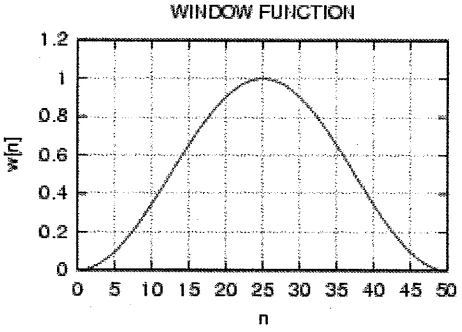


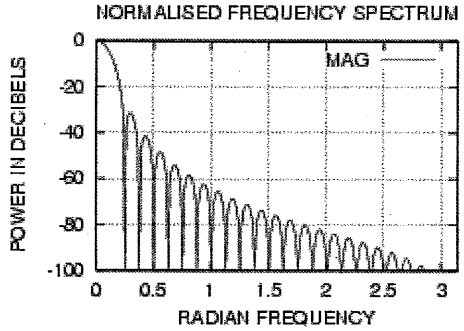
BARTLETT WINDOW



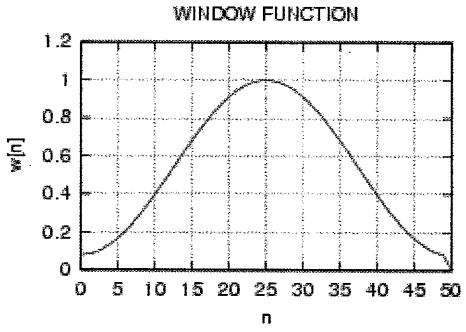


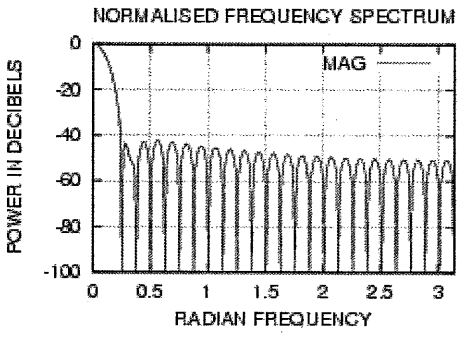
HANNING WINDOW



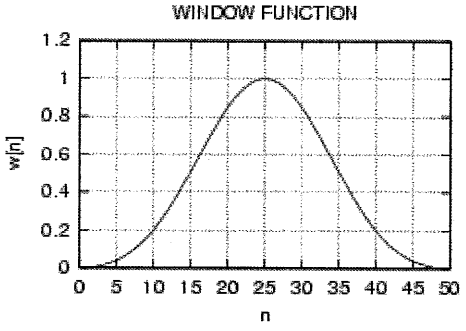


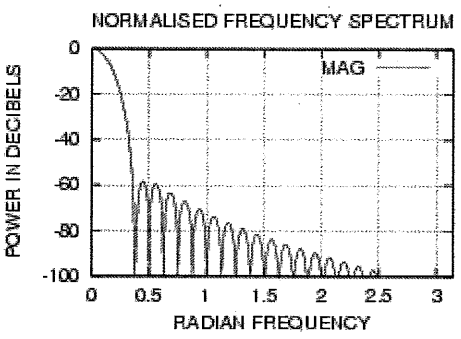
HAMMING WINDOW





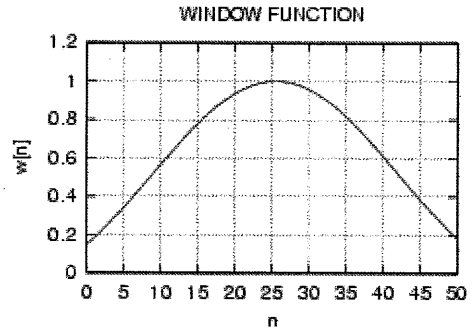
BLACKMANN WINDOW

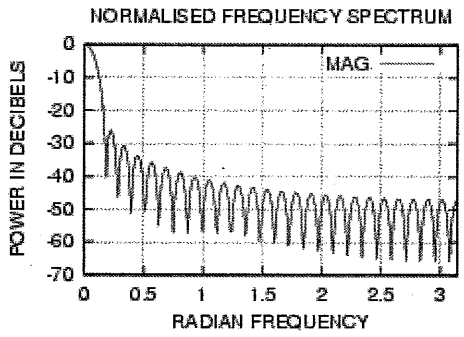




KAISER WINDOW

INPUT: ROLLOFF TRANSTION WIDTH = 0.28radians/sec
PEAK APPROXIMATION ERROR = 0.01
(The parameters of the Kaiser window were selected to obtain a tap length of 50)



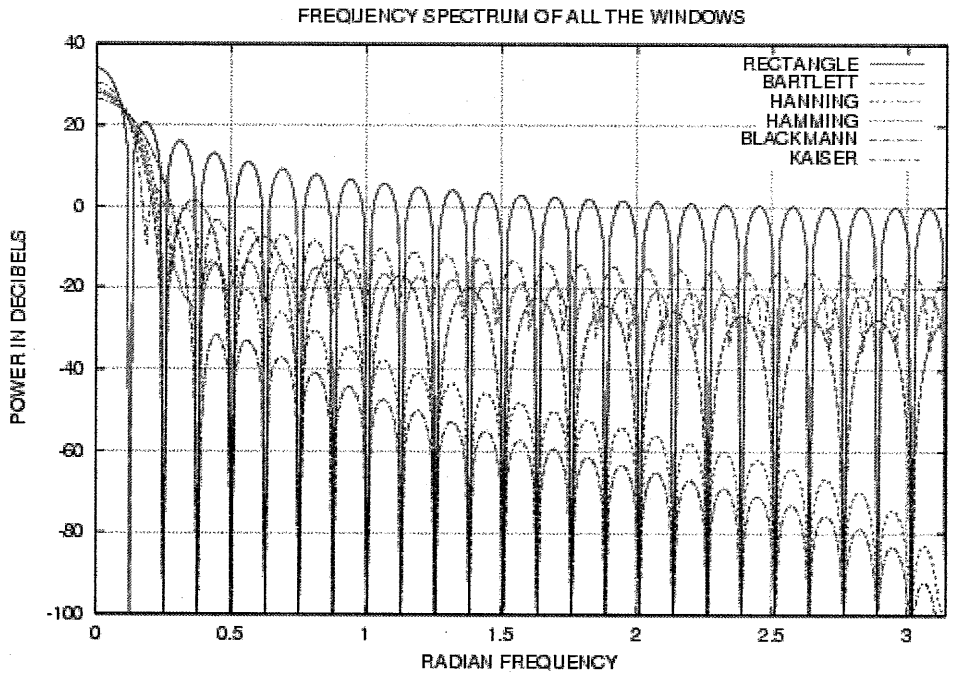


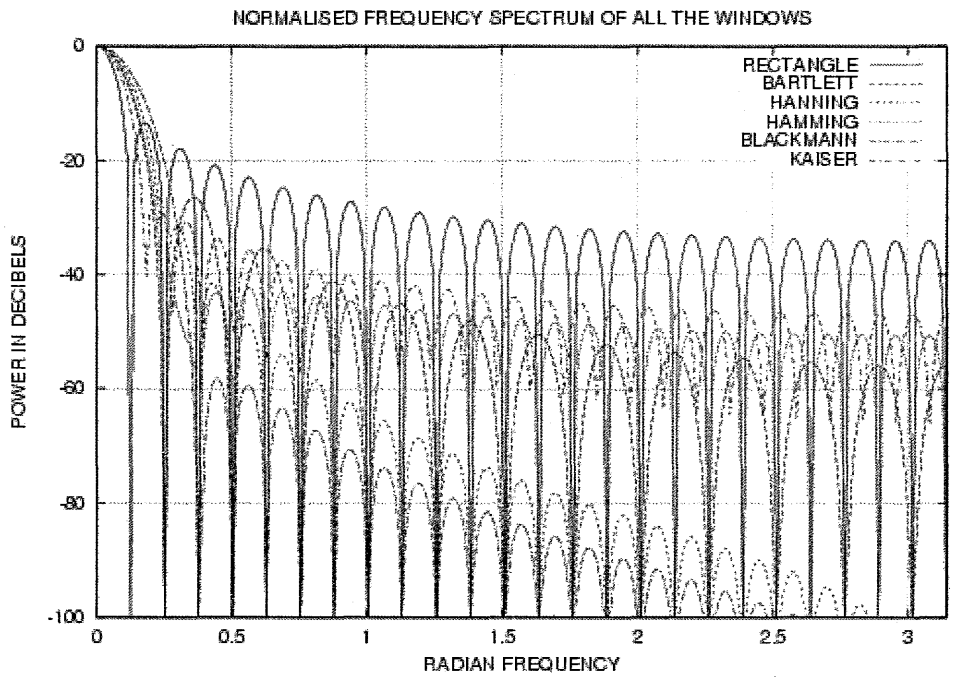
7.1 QUANTITATIVE COMPARISON OF VARIOUS WINDOWS

Windows are basically used to reduce the ripples (that occur due to the truncation of a signal) in the passband and stopband. The desirable window characteristics are

- Small width of the mainlobe of the frequency response of the window containing as much of the total energy as possible.
- Sidelobes of the frequency response that decrease in energy rapidly as ω tends to π.

There are various windows that can be employed to achieve this. The various windows we have studied are the Rectangular window, Bartlett window, Hamming window, Hanning window, Blackman window and the Kaiser window. A comparison of these windows is done using the output obtained with respect to the gain, approximate mainlobe width, 3-dB bandwidth, 10-dB bandwidth, peak sidelobe amplitude and number of sidelobes for a given filter tap length.





Type of Window	Gain (dB)	Approximate mainlobe width (Radian frequency)	3-dB bandwidth (Radian frequency)	10-dB bandwidth (Radian frequency)	Number of side-lobes	Peak sidelobe amplitude (relative) (dB)
Rectangular	34	0.08 π	0.016 π	0.03π	24	-13.5
Bartlett	27.8	0.17 π	0.025π	0.045π	11	-26
Hanning	28	0.16 π	0.026π	0.05π	21	-32
Hamming	28.5	0.16 π	0.025π	0.046 π	23	-44
Blackman	29.5	0.24 π	0.03π	0.058π	17	-58.3
Kaiser	30	0.12 π	0.023π	0.04π	24	-27

Rectangular window is characterized by a narrow mainlobe and provides a 3-dB bandwidth of 0.03π radian frequency .The first sidelobe is about 13.5dB below the mainlobe peak resulting in poor stopband attenuation. The remaining sidelobes are almost rejected to the same level. Thus we see that a rectangular window provides a narrow bandwidth but provides insufficient sidelobe rejection. The system response results in overshoot and undershoot at the edge, which is known as **Gibbs** phenomena.

To reduce Gibbs effect and provide the required characteristics, windows are tapered smoothly to zero as in the case of other window functions.

Hanning window is a raised cosine window with a platform of 0.5. The mainlobe has a 3-dB bandwidth of 0.053π radian frequency. The first sidelobe is attenuated by 32dB. As the frequency increases, the sidelobe peaks fall in amplitude rapidly. Since the amplitude is zero at the edges, there is no overshoot or undershoot at the edges. Hence Gibbs phenomenon is reduced. The gain provided by Hanning window is less than that of the rectangular window.

Hamming window is another raised cosine window with a platform of 0.54. The mainlobe has a 3-dB bandwidth of 0.05π radian frequency. The first sidelobe is attenuated to 44dB. As the frequency increases, the sidelobe peaks remain constant. The overall bandwidth of Hamming window is more than that of Hanning window. However the signal amplitude at the output is greater than that of Hanning window but less than that of Rectangular window.

Bartlett window is a triangular window. The 3-dB bandwidth is 0.05π radian frequency. The first sidelobe is attenuated by 26dB. As the frequency increases, the sidelobe peaks falls continuously and then becomes constant. The system results in a mainlobe width same, as that of Hanning window but the amplitude is less than the Hanning window.

Blackman window provides a wide mainlobe width of 0.06π radian frequency. The very first sidelobe is rejected to 58dB. The successive sidelobes are further rejected. The system results in a 3dB bandwidth nearly same as that of Hanning but the amplitude is slightly more than Hanning window.

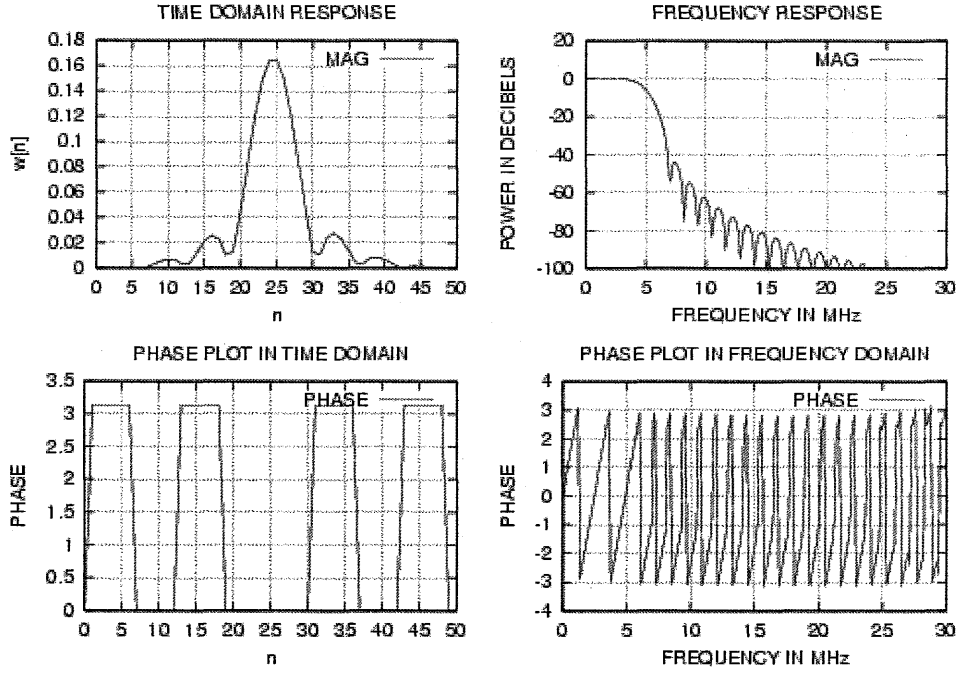
Kaiser window provides a slightly larger mainlobe than the rectangular window. This window is nearly optimum. It provides a 3-dB bandwidth of 0.12π radian frequency. The very first sidelobe is attenuated to 27dB for a tap length of 50. Kaiser window response has small mainlobe width and also higher sidelobe suppression, which makes it a desirable window.

From the above comparison, we observe that rectangular window has a narrow mainlobe but low sidelobe suppression. On the other extreme, Blackman window has wider bandwidth and higher sidelobe rejection. The intermediate response characteristics of narrow mainlobe width and considerable stopband attenuation are obtained by the use of Hanning and Hamming windows.

7.2 QUANTITATIVE COMPARISON OF LOWPASS FILTERS DESIGNED USING VARIOUS WINDOWS

Lowpass filter was designed using the above-mentioned windows to obtain a comparison of performance characteristics. The response of a lowpass filter designed using Hanning window is shown below. (Hanning window has been chosen as it has small mainlobe width, provides faster rolloff and sufficient stopband attenuation). From the phase spectra we observe that the filter has linear phase in the pass band. However, the filter can be designed to have constant or zero phase by suitably shifting in the time domain.

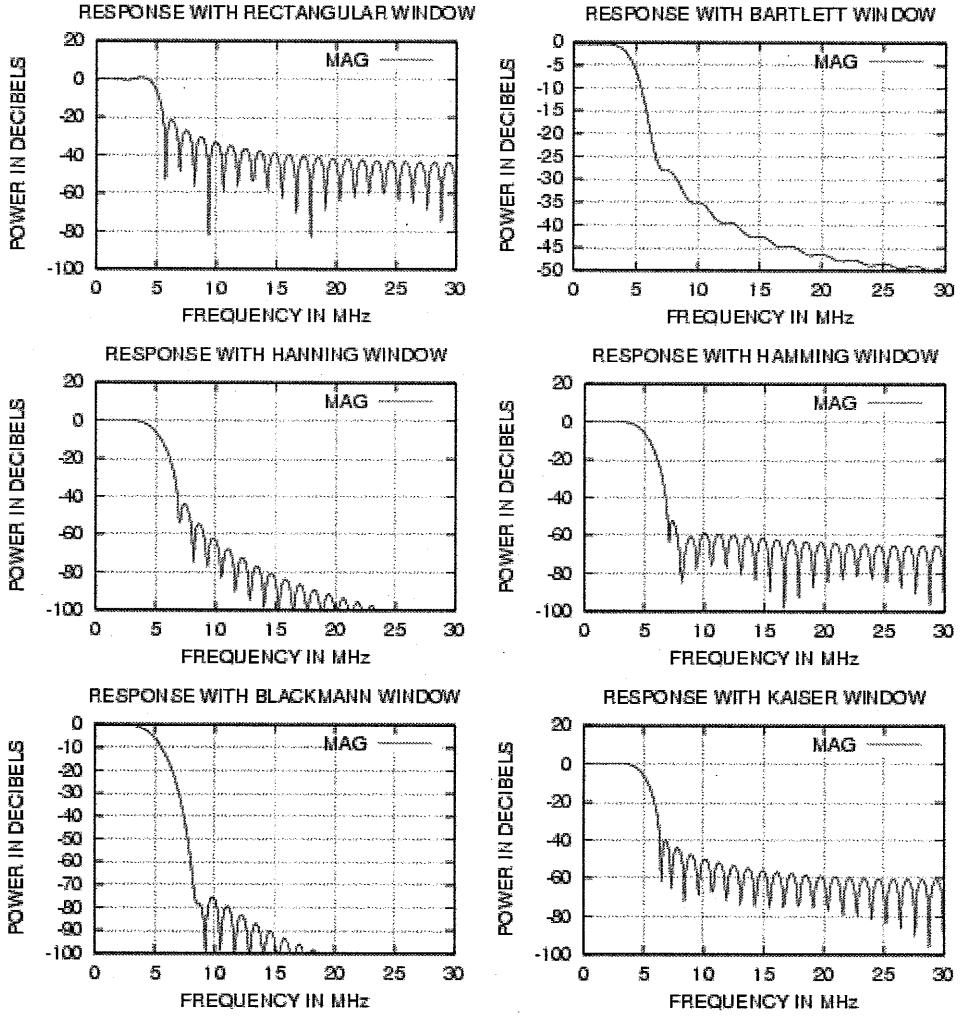
INPUT: FILTER TAP LENGTH = 50
TOTAL BANDWIDTH = 30MHz
CUTOFF FREQUENCY = 5MHz

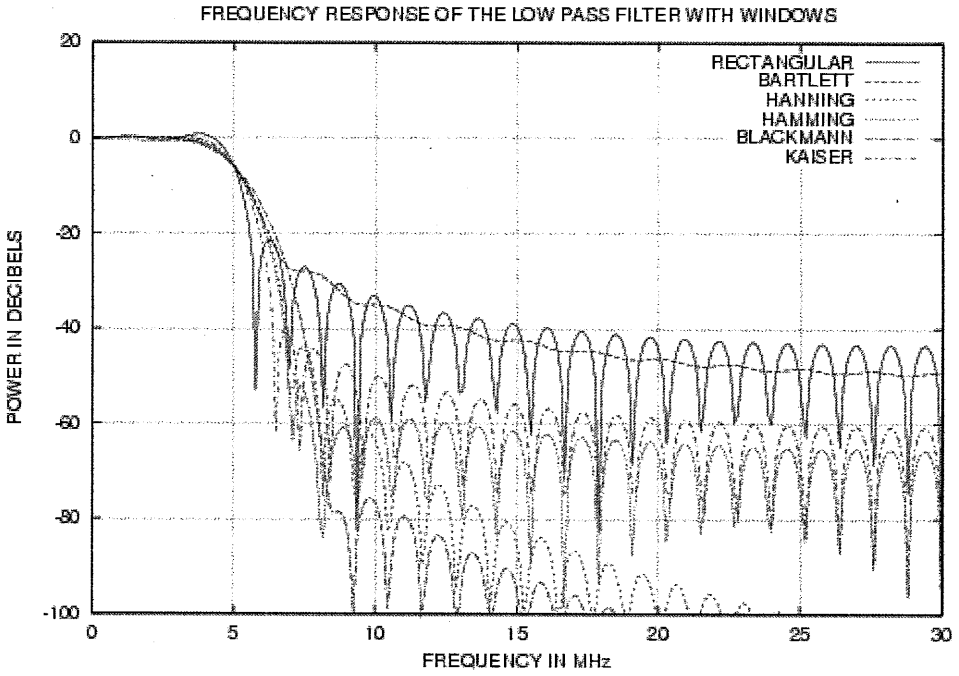


LOWPASS FILTER RESPONSE

INPUT: TOTAL BANDWIDTH = 30MHz
CUTOFF FREQUENCY = 5MHz
FILTER TAP LENGTH = 50

FOR KAISER WINDOW ONLY: ROLLOFF TRANSTION WIDTH = 2.72MHZ
PEAK APPROXIMATION ERROR = 0.01
(The parameters were selected to obtain a tap length of 50)





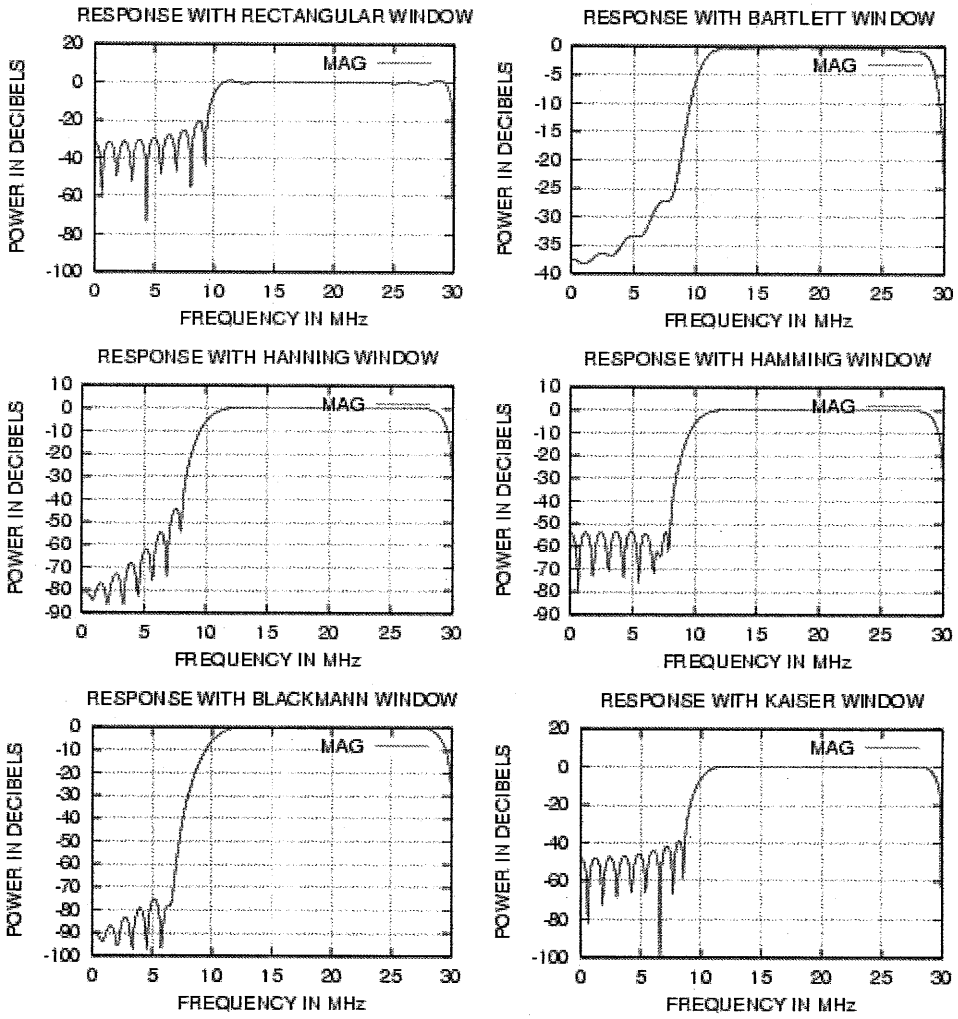
Type of Window	3-dB bandwidth (MHz)	10-dB bandwidth (MHz)	Rolloff rate of the mainlobe (dB/MHz)	Number of side-lobes	First sidelobe suppression (dB)	Last sidelobe suppression (dB)	Peak ripple (dB)
Rectangular	4.71	5.2	17.6	20	-21.8	-45	0.7
Bartlett	4.52	5.5	7.25	-	-27.5	-50	0.45
Hanning	4.5	5.45	7	14	-45	-100	0.05
Hamming	4.5	5.42	9.5	19	-52	-65	0.01
Blackman	4.37	5.51	6.5	8	-75	-100	0.01
Kaiser	4.6	5.35	16.5	20	-40	-60	0.08

A comparison of lowpass filters designed using different windows was made and results obtained are tabulated as shown above. The Kaiser window provides the designer considerable flexibility in meeting the filter specifications. Hence it is usually preferred for design of filters.

HIGHPASS FILTER RESPONSE

INPUT: TOTAL BANDWIDTH = 30MHz
CUTOFF FREQUENCY = 10MHz
FILTER TAP LENGTH = 50

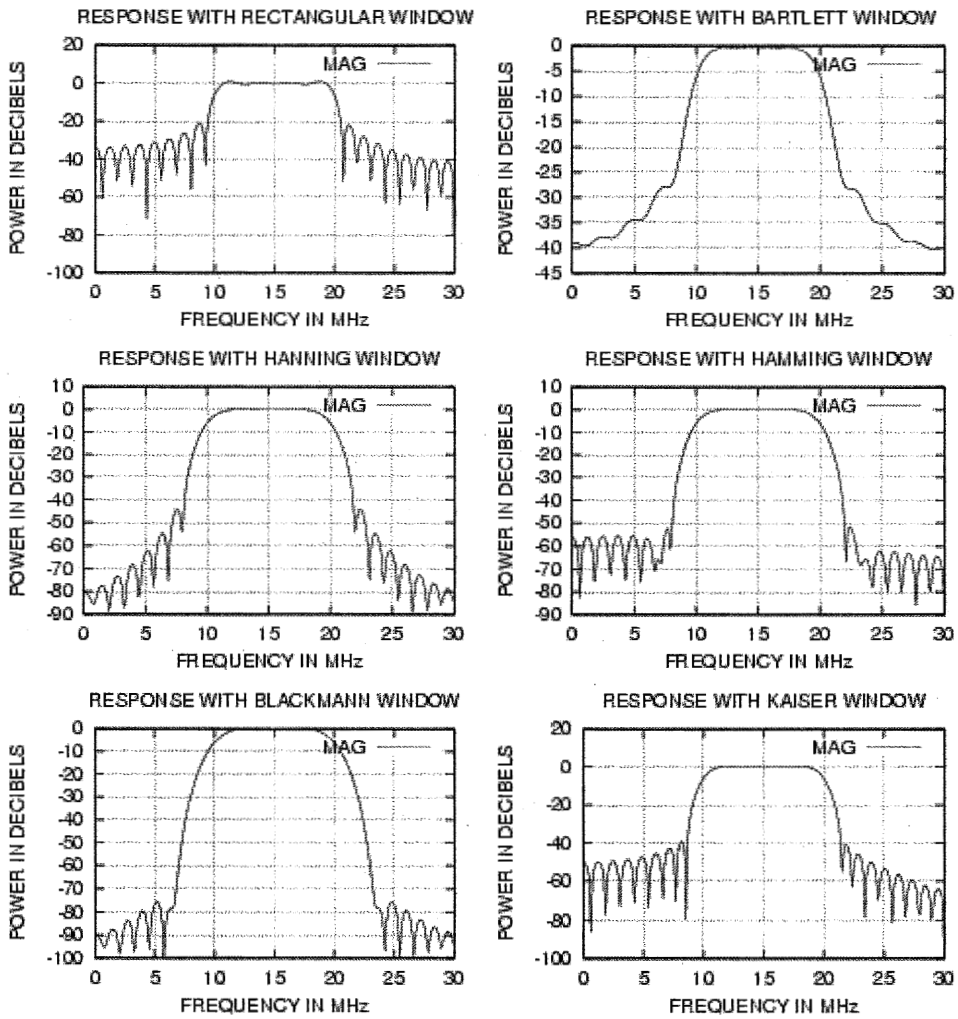
FOR KAISER WINDOW ONLY: ROLLOFF TRANSTION WIDTH = 2.72MHZ
PEAK APPROXIMATION ERROR = 0.01
(The parameters were selected to obtain a tap length of 50)



BANDPASS FILTER RESPONSE

INPUT: TOTAL BANDWIDTH = 30MHz
LOWER CUTOFF FREQUENCY = 10MHz
UPPER CUTOFF FREQUENCY = 20MHz
FILTER TAP LENGTH = 50

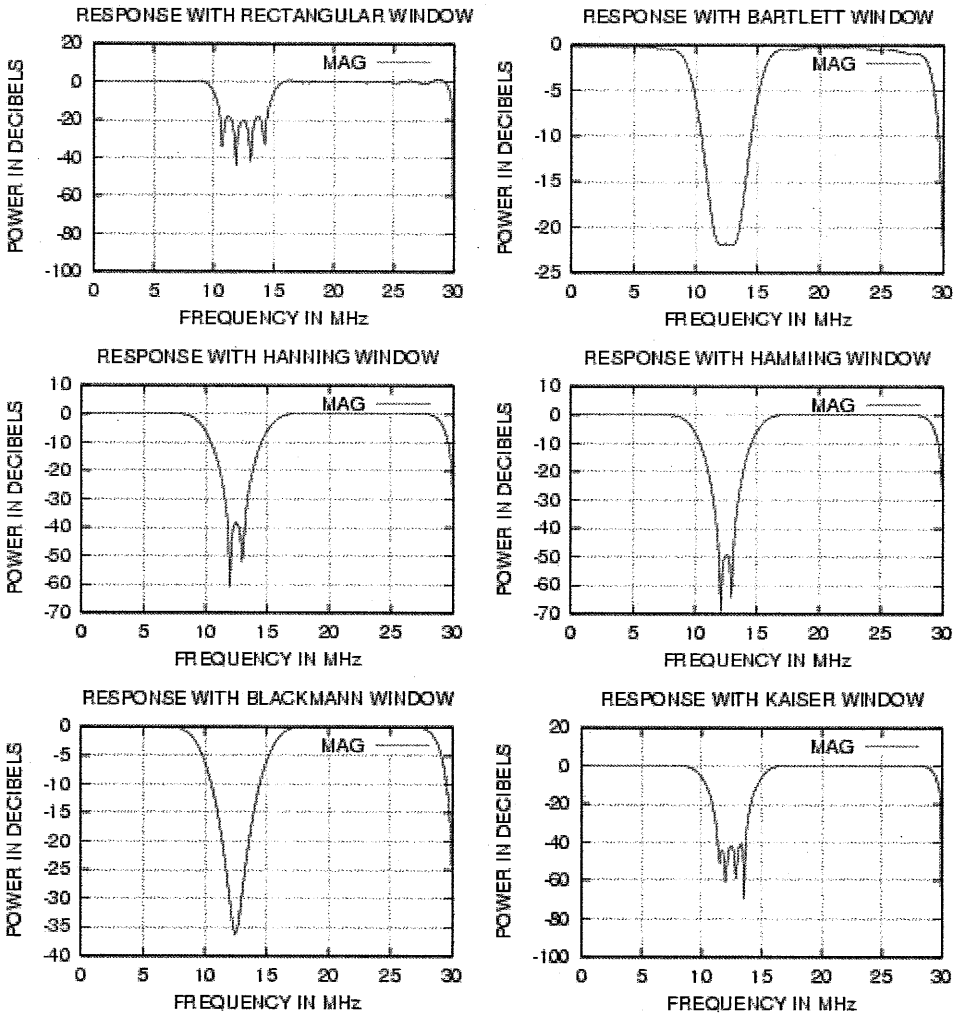
FOR KAISER WINDOW ONLY: ROLLOFF TRANSTION WIDTH = 2.72MHZ
PEAK APPROXIMATION ERROR = 0.01
(The parameters were selected to obtain a tap length of 50)



BANDSTOP FILTER RESPONSE

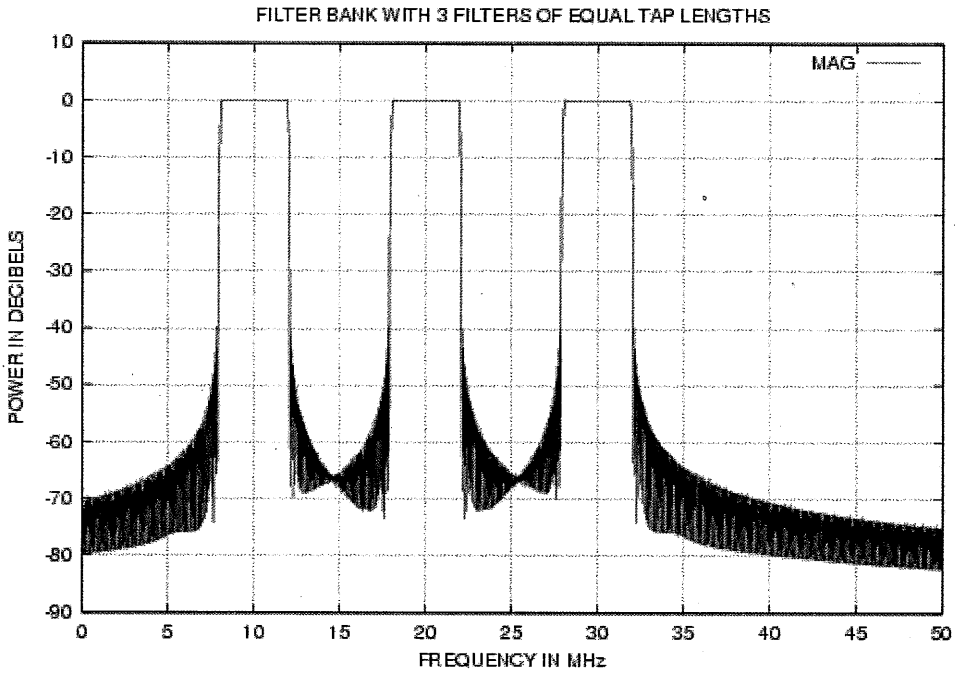
INPUT: TOTAL BANDWIDTH = 30MHz
LOWER CUTOFF FREQUENCY = 10MHz
UPPER CUTOFF FREQUENCY = 15MHz
FILTER TAP LENGTH = 50

FOR KAISER WINDOW ONLY: ROLLOFF TRANSTION WIDTH = 2.72MHZ
PEAK APPROXIMATION ERROR = 0.01
(The parameters were selected to obtain a tap length of 50)



FILTER BANK WITH 3 FILTERS OF EQUAL TAP LENGTHS (USING KAISER WINDOW)

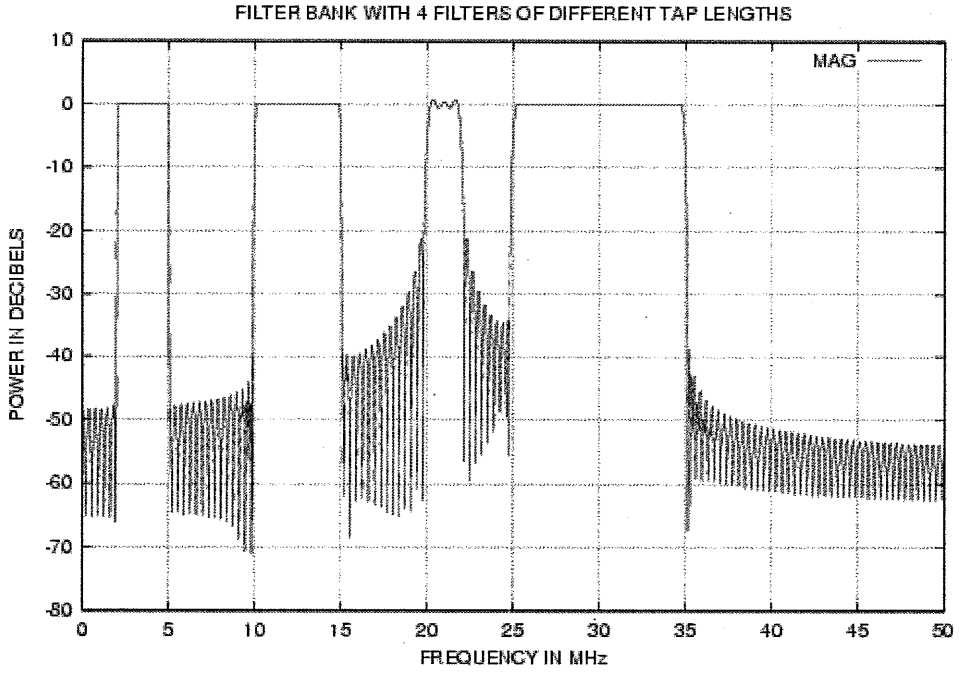
INPUT: TOTAL BANDWIDTH OF THE FILTER BANK = 50MHz
CUTOFF FREQUENCY = 5MHz
FIRST CARRIER FREQUENCY = 10MHz
SECOND CARRIER FREQUENCY = 20MHz
THIRD CARRIER FREQUENCY = 30MHz
ROLLOFF TRANSISTION WIDTH = 0.2MHz
PEAK APPROXIMATION ERROR = 0.01



FILTER BANK WITH 4 FILTERS OF DIFFERENT TAP LENGTH (USING KAISER WINDOW)

INPUT: TOTAL BANDWIDTH OF THE FILTER BANK = 50MHz

	Lower cutoff freq. (MHz)	Upper cutoff freq. (MHz)	Rolloff transition width (MHz)	Peak appro. error (delta)
1st filter	2	5	0.1	0.001
2nd filter	10	15	0.2	0.01
3rd filter	20	22	0.3	0.1
4th filter	25	35	0.4	0.001



Application in Radio Astronomy

8. APPLICATION IN RADIO ASTRONOMY

8.1 INTRODUCTUON TO RADIO ASTRONOMY

Radiation from celestial objects have been observed over a wide range of electromagnetic spectrum, ranging from the very longest radio waves to γ rays which have extremely short wavelengths. However the part of the radiation that can pass through the four-ionosphere with limited absorption is largely limited to two frequency bands one in the optical and the other in the radio frequency band.

Since our eyes are sensitive to the optical region (0.4μ to 0.8μm) we have to use other means of detecting the rich store of information in the celestial objects. It was only in the middle of last century that man found out that sources in the sky emit radio waves. The branch of astronomy, which deals with observation of celestial objects in the frequency range 30MHz to 200GHz, is called *Radio Astronomy*. Ground breaking work was done in this field by **Karl Guthe Jansky**, a radio engineer at BELL telescope laboratory. In 1932 while studying the direction of thunderstorms static, Jansky was able to detect the origin of a HISS type static, which was previously unaccounted for. He observed that the HISS is due to radio waves of extra-terrestrial origin, thus laying the foundation for radio astronomy. Though the importance of this result took a while to be recognized, it lead to development of radio astronomy which has now become a major branch of astronomy.

Radio waves are typically about a million times longer than those in the optical range. Regions of space which are opaque to light waves because of interstellar dust are generally transparent to radio waves due to their long wavelengths.

As radio astronomy progressed, the need for radio telescopes with better resolution came up. This led to several innovations both in antenna and electronics engineering complementing each other. Many pioneering high tech developments are due to research done in this field. Spin-offs from this area find applications in various fields including satellite communication, space research, image-processing and bio-medical sciences.

8.2 METHANOL MASERS

The word **MASER** is an acronym of **Microwave Amplification by Stimulated Emission of Radiation**. In masers the radiations are emitted by the transition of molecules between rotational energy levels.

While masers are quite difficult to produce on Earth, they occur naturally in some regions of interstellar space. Maser emission has so far been found in the molecular gas surrounding regions of large star formation, the atmosphere of some types of red giant stars, in the circumnuclear discs surrounding dusty galaxies and in the tail of Halley comet. A variety of molecules are known to exhibit maser emission including the OH (Hydroxyl) radical, water, methanol and formaldehyde.

The masers occur in different parts of the galaxy, some are moving towards us, and others away. This means that due to the Doppler effect, the radiation from these masers will appear to us, to be bluer and redder, respectively than its true color. Because we know what color the maser should be, we can use the color we observe it to be, to calculate the velocity at which the maser is moving with respect to us.

Methanol maser is the first signature of massive star formation. The formation of a massive star is one of the most spectacular complex events in the sky. It occurs hidden from view in the core of giant, dusty clouds of molecular Hydrogen. These clouds collapse under the weight of their own gravity, compressing the gas to densities and temperatures high enough for the nuclear fusion to occur. The nuclear reactions generate energy and stop the collapse. A star is born.

If the newly formed star has more than eight times the mass of the sun, it produces sufficient radiation to induce very strong stimulated emission at radio wavelengths, the maser, from the molecules of OH, Methanol etc in the surrounding clouds. This intense emission from molecules of methanol, methanol maser, often accompanies massive star formation, and is found nearby compact radio nebulas. Moreover, many methanol masers don't seem to accompany any radio nebula and have hypothesized that they represent an earlier stage of star formation.

Methanol maser arises from several transitions. The 6.7GHz transition of methanol is the second strongest centimeter masing transition of any molecule (after the 22 GHz water transition) and is commonly found toward star formation regions.

8.3 RADIO TELESCOPE

A radio telescope is used to receive and measure the radio emissions from celestial sources and also certain vital functions. It may be required to measure the intensity of radiation from the celestial object, find out whether this varies with time, to determine the spectrum of the radiation over a certain frequency range and it may

be used to produce a detailed map of the brightness distribution over some region of the sky and also measure polarization properties. The measurements are made in the presence of emissions coming from other sources in the sky, manmade interference and additional noise signals generated by the measuring equipment itself. Radio telescopes are much larger than optical telescopes and can achieve resolutions comparable to that obtained in optical telescopes. The basic elements of a radio telescope are the reflector, subreflector, feed, a receiving system and a recording device.

Raman Research Institute (RRI) has research facilities in several areas, which includes a fully equipped Radio Astronomy Laboratory (RAL) and an on campus millimeter wave radio telescope with a 10.4m diameter to track the radiation from a celestial source. It is presently equipped with receivers to observe in the centimeter wave band.

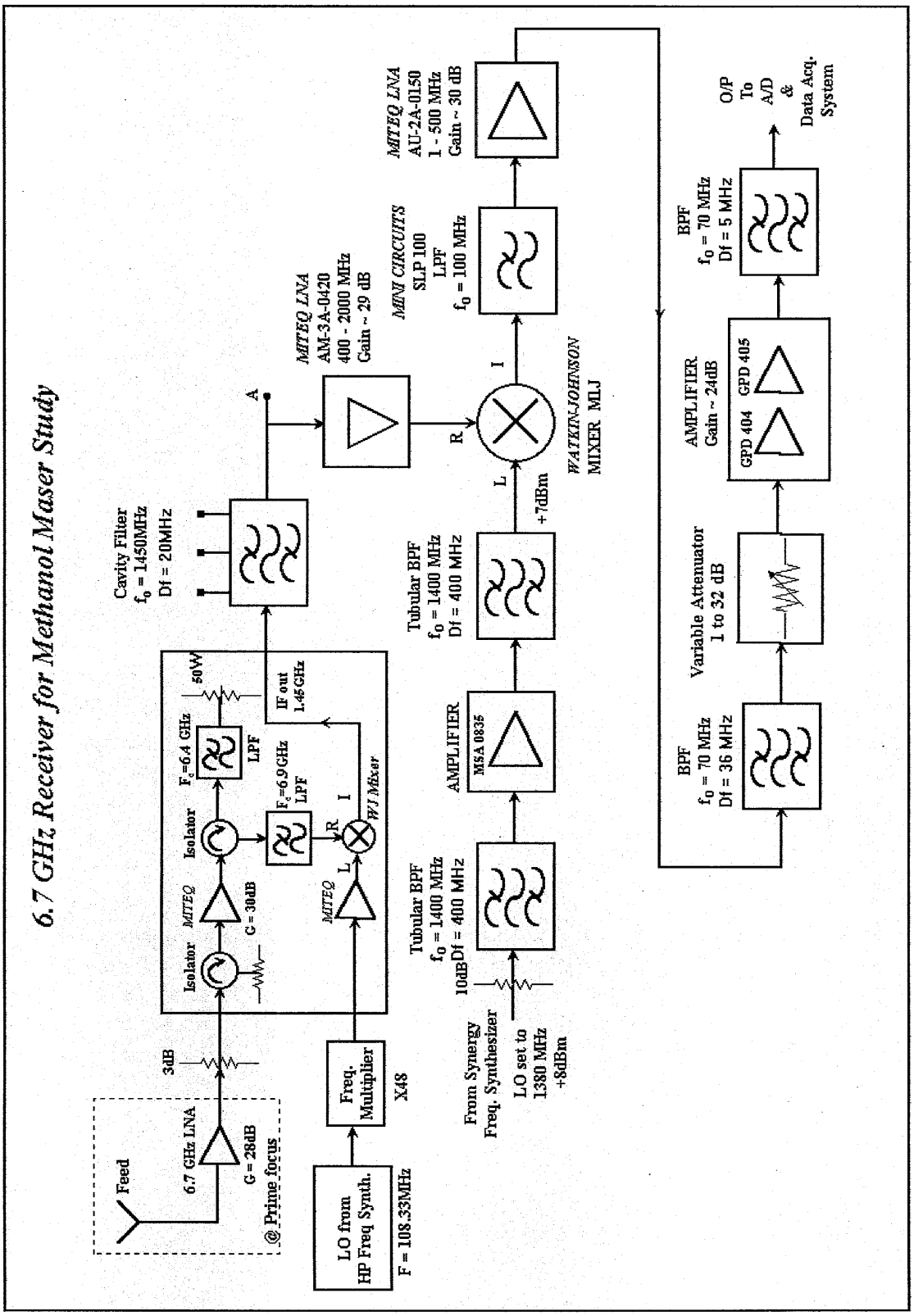
8.4 APPLICATION OF THE FIR FILTER

A radio telescope tracks radiation from celestial sources. The combination of a reflector and a feed converts the incident electromagnetic radiation to an electrical signal. The signal received by the telescope is fed to a low noise amplifier. An isolator prevents reflection of the signal, which could otherwise result in the re-radiation of signal power and eventual interference effects. The signal is then boosted by 30-dB. This is followed by a filter, which limits the signal to the frequency range of interest. Since the signal frequencies are very high, it would be convenient to have the signal frequency converted to a convenient intermediate frequency, as hardware is easier to build at lower frequencies. Heterodyning technique is used for down conversion. Heterodyning has the advantage of providing tunability over a wide range. Also, much of the required electronics is independent of the radio frequencies. The master oscillator is a Rubidium oscillator, which is a very stable oscillator and provides the reference for the synthesizer. The synthesizer is tunable and provides the local oscillator signal required for heterodyning. The signal frequency obtained from the synthesizer is multiplied by a factor of 48. The signal from the local oscillator and the signal from the source are multiplexed and the IF signal brought down to an intermediate frequency of 1450MHz. The intermediate frequency is always maintained at 1450MHz irrespective of the source being observed. This is done by suitably tuning the synthesizer.

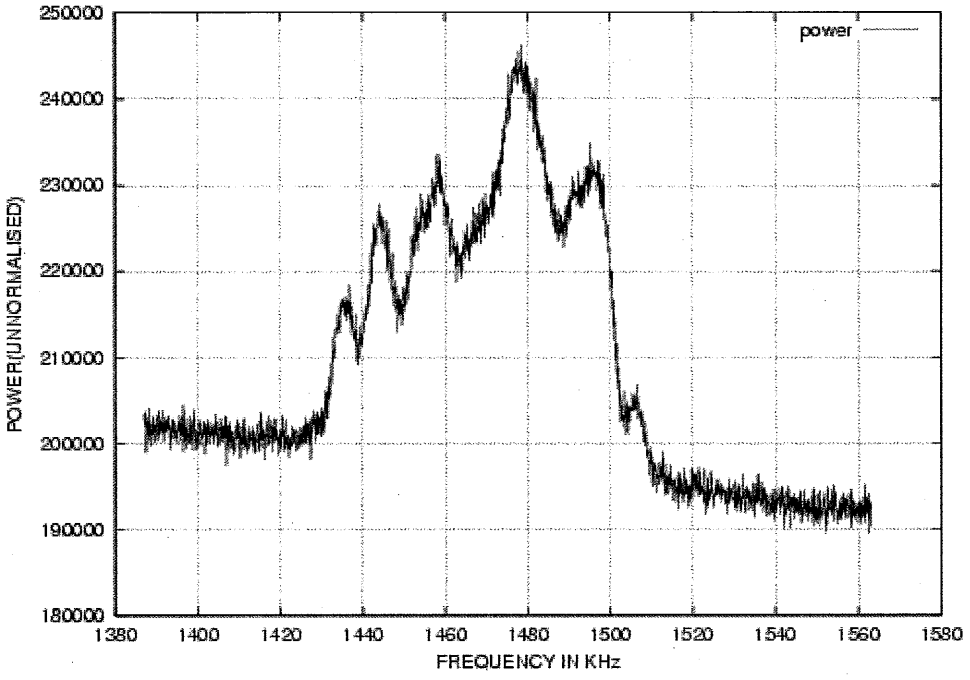
The subsequent bandpass filter has a center frequency of 1450MHz and a bandwidth of 20MHz. An amplifier with a gain factor of 29dB follows the filter

stage. This signal is further down converted in frequency using a local oscillator–mixer setup. The following filter stage is responsible for allowing only the 80-100MHz frequencies. There are subsequent amplifier stages to enhance the power levels. The final filter stage is a bandpass filter centered at 70MHz with a bandwidth of 5MHz.

70MHz IF is harmonically sampled at a sampling rate of 10MHz, instead of lowpass conversion, followed by sampling. This is possible because of the modern flash analog-to-digital converters, which have a very high conversion rate. The AD9059 converter accomplishes this analog-to-digital conversion. It has an analog bandwidth of 120MHz and can be used to sample upto 60MHz bandwidth signals. The block diagram of the receiver system is shown below.


6.7 GHz Receiver for Methanol Maser Study
Feed
@ Prime focus
6.7 GHz LNA
G = 28dB
3dB
LO from HP Freq Synth.
F = 108.33MHz
Freq. Multiplier
X48
Isolator
MITEQ
G = 30dB
Isolator
F_c=6.4 GHz LPF
50W
F_c=6.9 GHz LPF
WJ Mixer
IF out 1.45 GHz
MITEQ
Cavity Filter
f_o = 1450MHz
Df = 20MHz
A
MITEQ LNA
AM-3A-0420
400 - 2000 MHz
Gain ~ 29 dB
From Synergy Freq. Synthesizer
LO set to 1380 MHz +8dBm
10dB
Tubular BPF
f_o = 1400 MHz
Df = 400 MHz
AMPLIFIER
MSA 0835
Tubular BPF
f_o = 1400 MHz
Df = 400 MHz
+7dBm
WATKIN-JOHNSON
MIXER MLJ
MINI CIRCUITS
SLP 100
LPF
f_o = 100 MHz
MITEQ LNA
AU-2A-0150
1 - 500 MHz
Gain ~ 30 dB
BPF
f_o = 70 MHz
Df = 36 MHz
Variable Attenuator
1 to 32 dB
AMPLIFIER
Gain ~ 24dB
GPD 404
GPD 405
BPF
f_o = 70 MHz
Df = 5 MHz
O/P To A/D & Data Acq. System

A case study was performed by observing a spectral line arising due to methanol maser emission at 6.7GHz using the above setup. The signal received at the final stage of the receiver system, centered at 70MHz having a bandwidth of 5MHz, is harmonically sampled with a sampling frequency of 10MHz. A 128K-point FFT was performed on the data obtained from the observation and integrated for 6.7 minutes. It was observed that the signal appears in the frequency band ranging from 0 to 5MHz, which can be attributed to sampling which results in replicas of the original frequency band at harmonics of the sampling frequency. The resulting response is shown below.



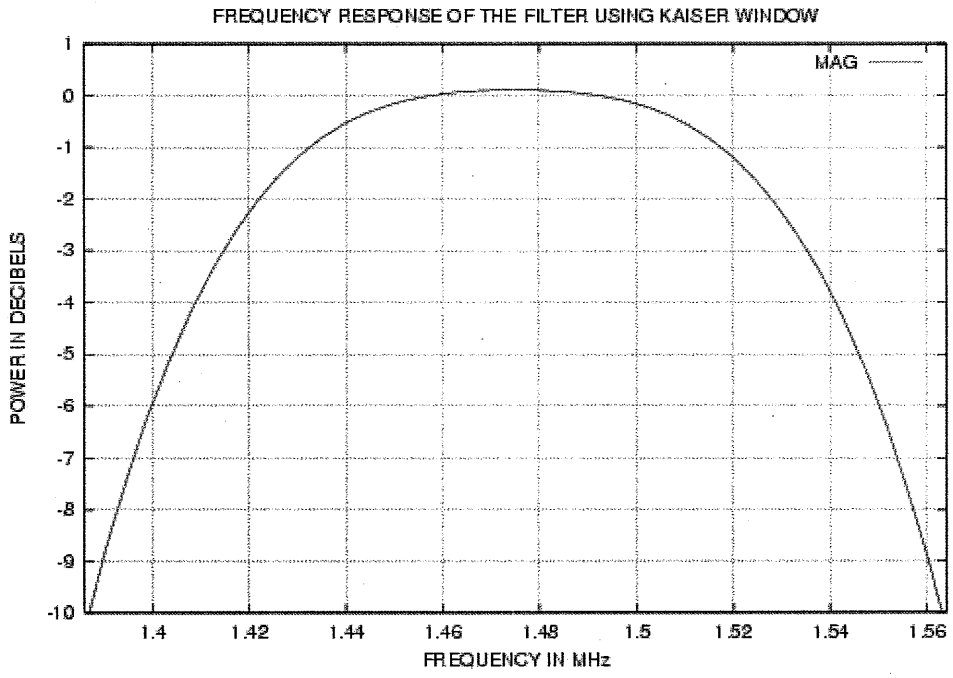
Response of the input signal after applying 128k-point FFT

The spectrum of the received signal has many features over a range of 250KHz. The center frequency of emission varies due to the Doppler effect resulting from the motion of the earth around the sun and also the intrinsic motion of the celestial source. So, to study the emission, we should be able to tune to the frequencies in this range. This requires dynamically tunable filters and is easily achieved by digital filters. FIR filters have an added advantage of finite impulse duration. With changing frequencies of sources, the digital filters can be easily reconfigured by changing the filter coefficients in a lookup table. The data coming at high speed can be easily processed.

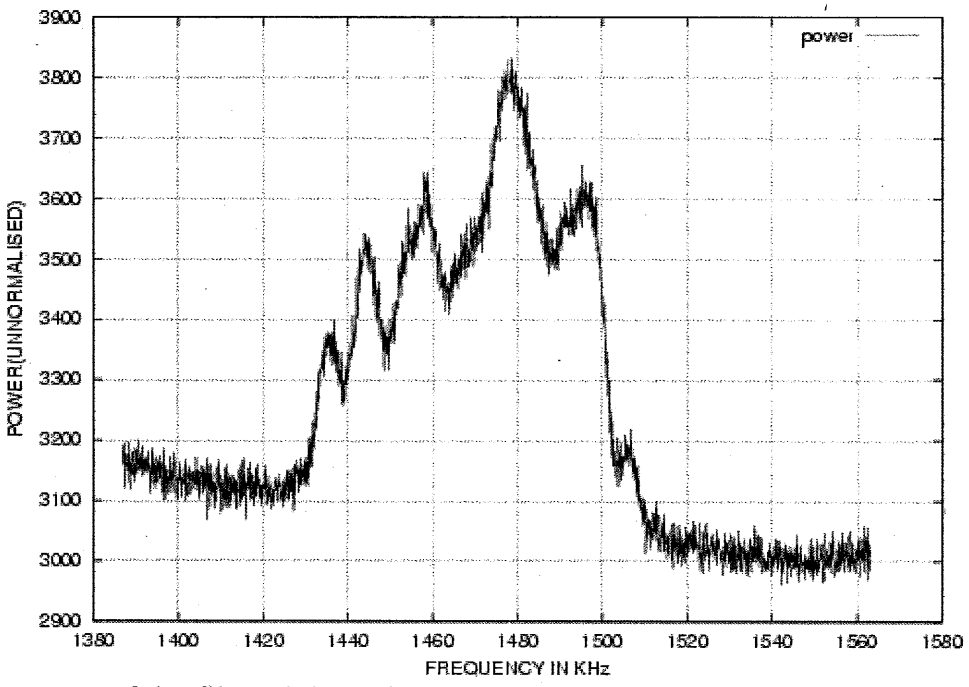
FIR filter designed was implemented on the digitized data appearing at different frequency in the IF bandpass. The bandpass filter (BPF) was designed to meet the required specifications of:

Total bandwidth: 5MHz
Sampling Frequency (fs): 10MHz
Passband width: 250KHz.
Lower cutoff frequency (f_l): 1.4MHz
Upper cutoff frequency (f_u): 1.55MHz
Rolloff transition: 0.1MHz
Peak approximation error: 40dB

The frequency response of the designed bandpass filter in the required range is as shown below.



The FIR filter designed for these specifications has a tap length of 224 and is applied to the sampled data of 4-bit word size. A decimation 8 16k-point FFT was applied and integrated over a long interval (6.7min). The response is shown below.



Response of the filtered data after applying a decimation 8 16k point FFT

This was compared with the response obtained after applying a 128k-point FFT on the data, which is integrated over same period of time, 6.7 minutes and the results are in conformity. Thus, the FIR filter was successfully implemented and tested.

Conclusion

9. CONCLUSION

9.1 ACHIEVEMENTS

This project has provided us an opportunity to comprehend the fundamental concepts involved in DSP in general and the design of FIR filters in particular. We studied the basic tools of filter design-the DFT and the FFT.

We have developed a software for the design of digital FIR filters. This software provides options to choose between lowpass, highpass, bandpass and bandstop filters using a window of user's choice. This software designs the filter given the specifications of cutoff frequency, bandwidth, tap length or rolloff transition width and peak ripple, and stores the filter coefficients for further processing. Once the filter is designed, the user can view the magnitude and phase plots. Discrete Fourier Transform (DFT) was used to analyze the frequency response of the filters. During the initial stages, the DFT was directly computed. In the later stages, FFTW, a library from public domain for Fast Fourier Transform (FFT) was used to increase the speed of computation.

This project has given us firsthand experience in programming concepts involved with the design and implementation of software using C and exposure to LINUX operating system. The C programs developed call "gnuplot" to display the magnitude and phase spectra of the filters. It is a command driven interactive function plotting program, a powerful tool on LINUX platform for graphics.

In the final phase of this project a bandpass filter was designed using this software and implemented for analyzing the data obtained from a Methanol Maser source observed using a 10.4m diameter radio telescope operating in the centimeter wave band.

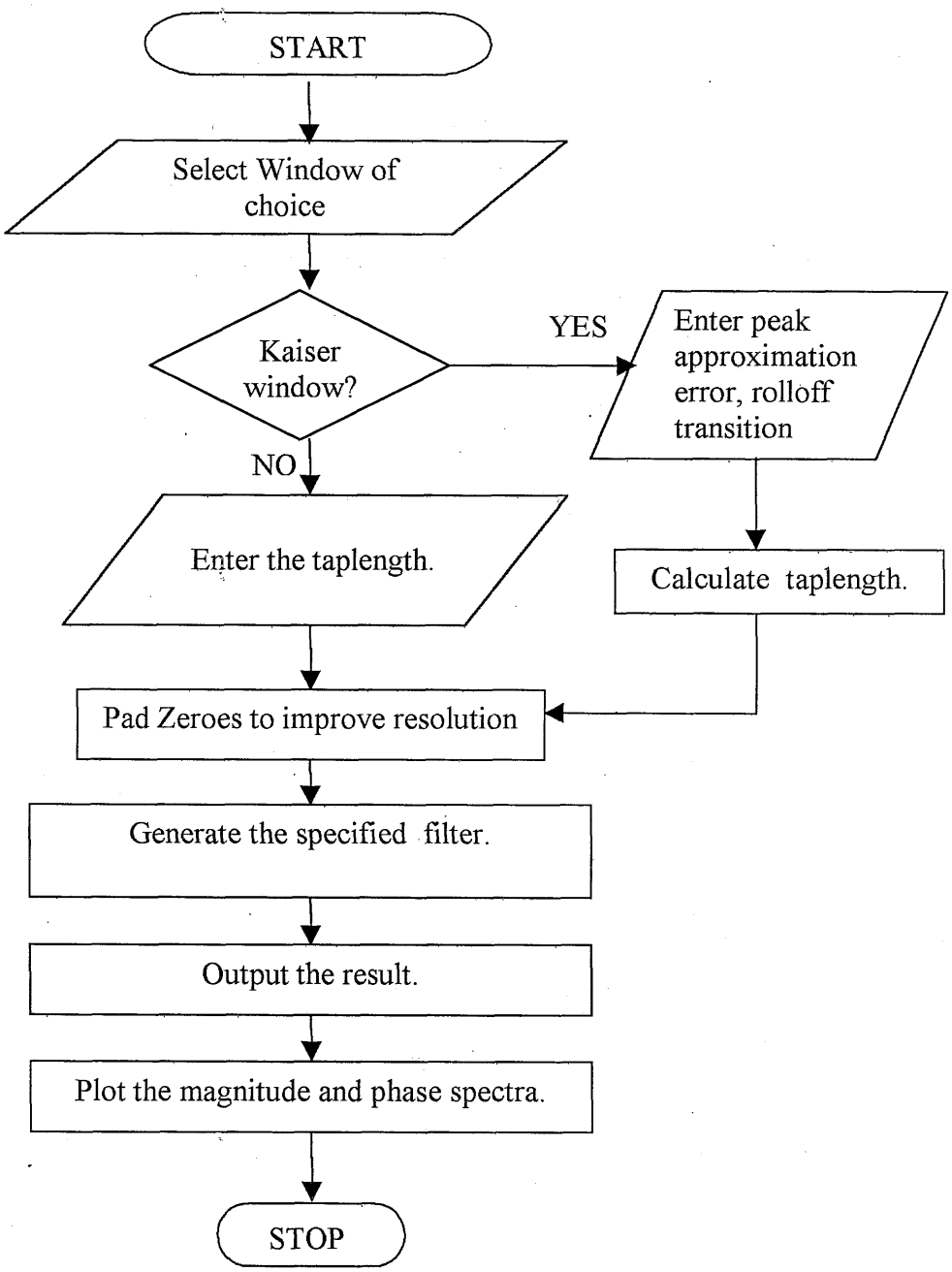
9.2 SCOPE FOR FURTHER WORK IN THIS FIELD

Digital filters can be employed in real time systems and can be implemented using high level descriptor languages like VHDL using Field Programmable Gate Arrays (FPGA). This would be an exciting project to undertake in future, which can be used in a radio telescope and in many other DSP related fields.

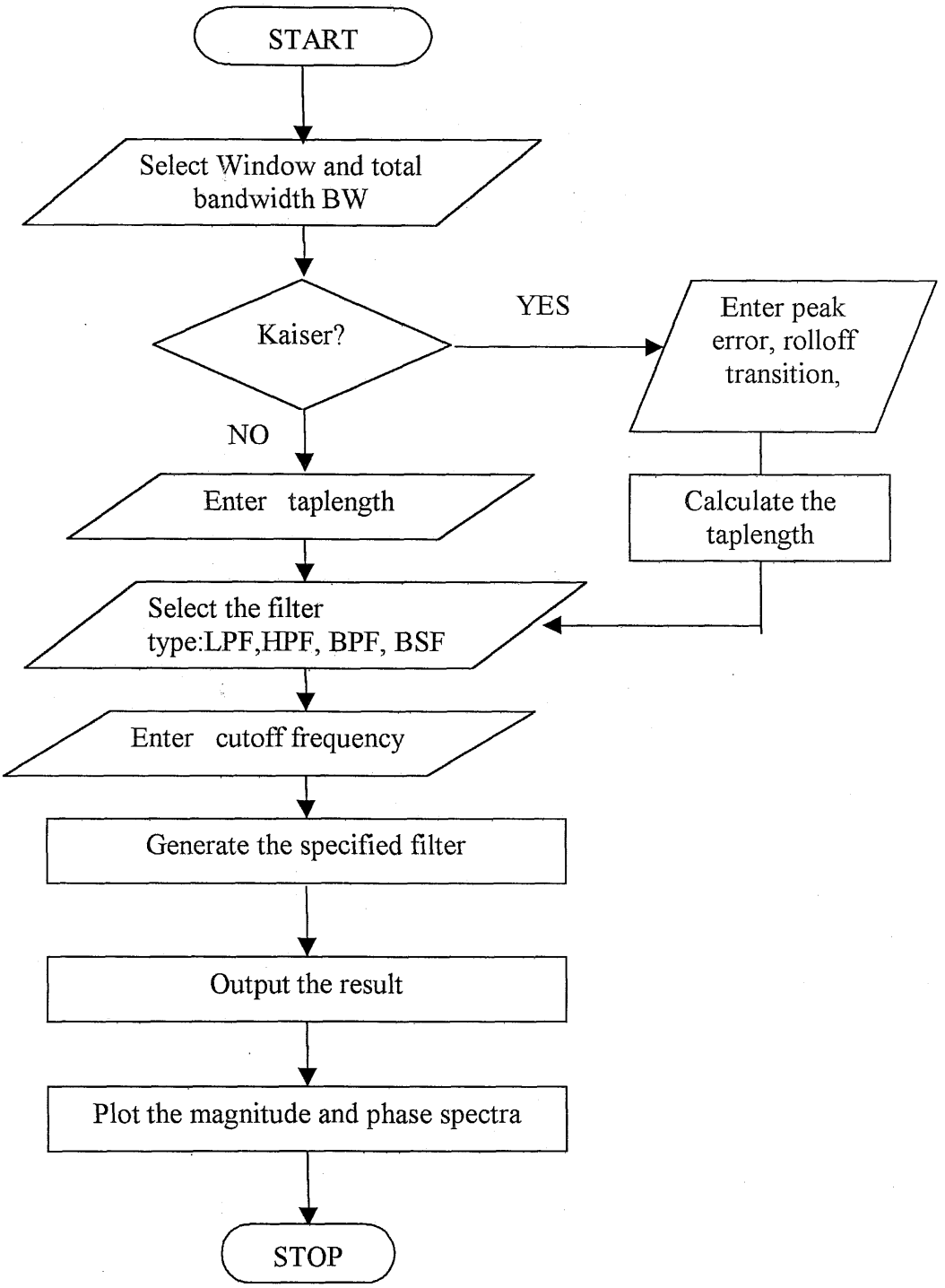
Appendix

Flowcharts

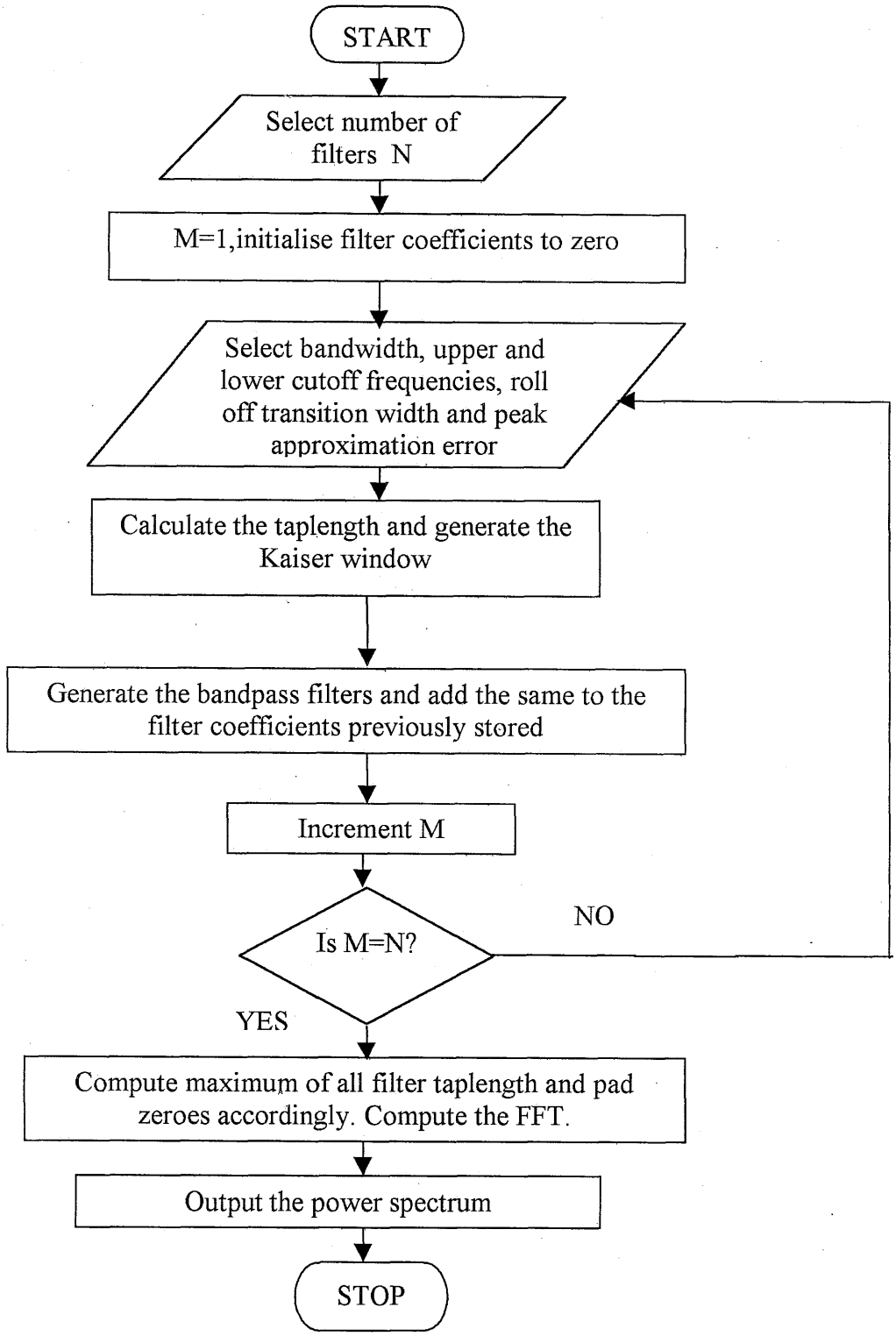
TO OBTAIN MAGNITUDE AND PHASE PLOTS OF VARIOUS WINDOWS



FILTER DESIGN USING WINDOWING TECHNIQUE



FILTER BANK


START
Select number of filters N
M=1,initialise filter coefficients to zero
Select bandwidth, upper and lower cutoff frequencies, roll off transition width and peak approximation error
Calculate the taplength and generate the Kaiser window
Generate the bandpass filters and add the same to the filter coefficients previously stored
Increment M
Is M=N?
NO
YES
Compute maximum of all filter taplength and pad zeroes accordingly. Compute the FFT.
Output the power spectrum
STOP

Programs

//TO FIND THE SPECTRUM OF THE VARIOUS WINDOWS USING DISCRETE FOURIER TRANSFORM

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

typedef struct     //structure defines complex data
{
   double re,im;
}complex;

int main(int argc,char **argv)
{
   enum {Rect=1,Bart,Han,Ham,Black,Kaiser} Window type;
   int i,ch,m,n;
   double u,alpha,beta,del,a,df,dw,x;
   double dft(complex *,complex *,int);
   double bet(double);
   double ino(double *);
   double power(complex *,complex *,int);
   complex *w,*g;
        //arrays to store window function and its DFT respectively
   FILE *fp;

   if(argc==1)
         fp=stdin;
   else
         fp=fopen(argv[1],"rt");

   fprintf(stderr,"Enter 1 for Rectangular,2 for Bartlett,3 for
Hanning,4 for Hamming,5 for Blackmann,6 for Kaiser : ");
   scanf("%d",&ch);     //allows selection of window of choice

   switch(ch)
   {
         case Rect:     //Rectangular Window

         fprintf(stderr,"Enter the filter tap length(m) : ");
         scanf("%d",&m);
         n=10*m;

         w=(complex *)malloc(n*sizeof(complex));
         g=(complex *)malloc(n*sizeof(complex));

         //Rectangular window function calculation
         for(i=0;i<m;i++)
         {
               w[i].re=1;
               w[i].im=0;
         }
         break;

         case Bart:     //Bartlett window
```

```
fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

w=(complex *)malloc(n*sizeof(complex));
g=(complex *)malloc(n*sizeof(complex));

//Bartlett window function calculation
for(u=0,i=0;u<=m/2;i++,u++)
{
      w[i].re=2*u/m;
      w[i].im=0;
}

for(i=(m/2)+1,u=(m/2)+1;u<=m;i++,u++)
{
      w[i].re= 2-(2*u/m);
      w[i].im=0;
}
break;

case Han:     //Hanning window

fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

w=(complex *)malloc(n*sizeof(complex));
g=(complex *)malloc(n*sizeof(complex));

//Hanning window function calculation
for(i=0;i<m;i++)
{

      w[i].re=0.5-0.5*cos(2*M_PI*i/m);
      w[i].im=0;
}
break;

case Ham:     //Hamming window

fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

w=(complex *)malloc(n*sizeof(complex));
g=(complex *)malloc(n*sizeof(complex));

//Hamming window function calculation
for(i=0;i<m;i++)
{
      w[i].re=0.54-0.46*cos(2*M_PI*i/m);
      w[i].im=0;
```

```
            }
            break;

            case Black:     //Blackmann window

            fprintf(stderr,"Enter the filter tap length(m) : ");
            scanf("%d",&m);
            n=10*m;

            w=(complex *)malloc(n*sizeof(complex));
            g=(complex *)malloc(n*sizeof(complex));

            //Blackmann window function calculation
            for(i=0;i<m;i++)
            {
                  w[i].re=0.42-0.5*cos(2*M_PI*i/m)+.08*cos(4*M_PI*i/m);
                  w[i].im=0;
            }
            break;

            case Kaiser:     //Kaiser window

                fprintf(stderr,"Enter the rolloff transition width(dw)
(radian frequency) : ");
                scanf("%lf",&dw);
                fprintf(stderr,"Enter the peak approximation
error(delta) : ");
                scanf("%lf",&del);

                a=-20*log10(del);     //expresses delta in decibels
                m=((int)((a-8)/(2.285*dw)+1.0));
                     //taplength calculation
                fprintf(stderr,"The filter tap length(m) is :
%d/n/n",m);
                n=10*m;
                if((m%2)==0)
                              alpha=(double)(m-1)/2;
                else
                              alpha=(double)m/2;

                beta=bet(a);

                w=(complex *)malloc(n*sizeof(complex));
                g=(complex *)malloc(n*sizeof(complex));

                //Kaiser window function calculation
                for(i=0;i<m;i++)
                  {

                  x=beta*sqrt(1-(pow(((i-alpha)/alpha),2)));
                  w[i].re=ino(&x)/ino(&beta);
                  w[i].im=0.0;
                  }
                  break;
```

```
        }

    for(i=m;i<n;i++)
          w[i]=(complex){0.0,0.0};
   //resolution factor improvement by padding zeroes from m to n

    dft(w,g,n);      //function call for DFT computation
    power(g,w,n,bw);      //function call for power calculation
    system("gnuplot pltwin-ka");
        //function call to plot the frequency & phase spectra
    return 0;
}


double bet(double x)
     //to calculate the shape parameter beta in Kaiser function
{
      double beta;
      if(x<21)
            beta=0.0;
      if ((x>=21) && (x<=50))
            beta=0.5842*pow((x-21),0.4)+0.07886*(x-21);
      if (x>50)
            beta=0.1102*(x-8.7);
      return beta;
}


double ino(double *x)
     //to calculate zeroth order modified Bessel's function
{
   double ret_val;
   static double e;
   static int i;
   static double t,y,de,xi,sde;

   y=*x/(double)2.;
   t=(double)1e-20;
   e=(double)1.;
   de=(double)1.;

   for (i=1;i<=25;++i)
        //calculation of Bessel function upto 25 terms
   {
         xi=(double) i;
         de=de*y/xi;
         sde=de*de;
         e+=sde;
         if (e*t>sde)
         break ;
   }

   ret_val=e;
   return ret_val;
```

```
}


double dft(complex *w,complex *g,int n)      //DFT COMPUTATION
{
   int k,j;
   complex s;
   double vi,vr,arg;

   for(j=0;j<n;j++)
   {
         arg=2*M_PI*j/n;
         vi=0;
         vr=0;
         for(k=0;k<n;k++)
               //computation of DFT of window
         {
               s.re=cos(arg*k);
               s.im=sin(arg*k);
               vr+=(w[k].re*s.re-w[k].im*s.im);
               vi+=(w[k].re*s.im+w[k].im*s.re);

         }
         g[j].re=vr;      //real part of DFT
         g[j].im=vi; );      //imaginary part of DFT
   }
   return 0;
}


double power(complex *g,complex *w,int n)
     //function to calculate the power
{
   double pow,pown,poww,db,dbn,x,phase;
   int i;
   FILE *fp1;

   fprintf(stderr,"\nTHE FILTER CO-EFFICIENTS ARE STORED IN THE FILE
win-ka.res\n\n");

   fp1=fopen("win-ka.res","w");

   fprintf(fp1,"Radian Freq. \tPower in dB\tNorm. power\tn\tWin-
coeff.\n");

   x=hypot(g[0].re,g[0].im);

   for(i=0;i<(int)n/2+1;i++)
   {
         pow=hypot(g[i].re,g[i].im);      //magnitude
         pown=hypot(g[i].re,g[i].im)/x;      //to normalise magnitude

         if(pow<1.0e-5)
               db=-100;
```

```
            else
                  db=20*log10(pow);

            if(pown<1.0e-5)
                  dbn=-100;
            else
                  dbn=20*log10(pown);
                 //to express magnitude in decibels

      fprintf(fp1,"\n%lf\t%lf\t%lf\t%d\t%lf",(2*i*M_PI/n),db,dbn,i,w[i].
re);
      }
      fclose(fp1);
      return 0;
   }
```

//program to plot

```
//the results are stored in a file called win-ka.res

set grid
set title 'FREQUENCY SPECTRUM'
set xlabel 'RADIAN FREQUENCY'
set ylabel 'POWER IN DECIBELS'
plot 'win-ka.res' u 1:2 t 'MAG' w l
pause -1 "hit return to continue"

set title 'NORMALISED FREQUENCY SPECTRUM'
set xlabel 'RADIAN FREQUENCY'
set ylabel 'POWER IN DECIBELS'
plot 'win-ka.rels' u 1:3 t 'MAG' w l
pause -1 "hit return to continue"

set title 'WINDOW FUNCTION'
set xlabel 'n'
set ylabel 'w[n]'
plot 'win-ka.res' u 4:5 t 'MAG' w l
pause -1 "hit return to continue"
```

//TO FIND THE RESPONSE OF THE VARIOUS FILTERS WITH DIFFERENT WINDOWS USING FAST FOURIER TRANSFORM

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include"fftw.h"

typedef fftw_complex complex;

int main(int argc,char **argv)
{
   enum {Rect=1, Bart, Han, Ham, Black,Kaiser} WindowType;
   enum {LPF=1,HPF,BPF,BSF} FilterType;
   int i,m,n,ch1,ch2;
   double fc,wc,bw,fl,wl,fu,wu,u,alpha,beta,a,del,df,dw,x;
   double power(complex *,complex *,int,double);
   double ino(double *);
   double bet(double);
   complex *w,*h,*f,*d;
   fftw_plan plan;
   FILE *fp;

   if(argc==1)
         fp=stdin;
   else
         fp=fopen(argv[1],"rt");

   fprintf(stderr,"Enter 1 for Rectangle,2 for Bartlett,3 for
Hanning,4 for Hamming,5 for Blackmann,6 for Kaiser : ");
   scanf("%d",&ch1);      //select window type of choice

   fprintf(stderr,"Enter 1 for LPF,2 for HPF,3 for BPF,4 for BSF :
");
   scanf("%d",&ch2);      //to select filter type of choice

   fprintf(stderr,"Enter the total bandwidth(bw) of the filter (in
MHz) : ");
   scanf("%lf",&bw);

   switch(ch1)
   {
         case Rect:     //Rectangular window

         fprintf(stderr,"Enter the filter tap length(m) : ");
         scanf("%d",&m);
         n=10*m;

         if((m%2)==0)
               alpha=(double)(m-1)/2;
         else
               alpha=(double)m/2;

         //allocation of arrays to store
```

```
w=(complex *)malloc(n*sizeof(fftw_complex));
h=(complex *)malloc(n*sizeof(fftw_complex));
f=(complex *)malloc(n*sizeof(fftw_complex));
d=(complex *)malloc(n*sizeof(fftw_complex));

//Rectangular window function generation
for(i=0;i<m;i++)
{
            w[i].re=1;
            w[i].im=0;
}
break;


case Bart:      //Bartlett window

fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

if((m%2)==0)
      alpha=(double)(m-1)/2;
else
      alpha=(double)m/2;

w=(complex *)malloc(n*sizeof(fftw_complex));
h=(complex *)malloc(n*sizeof(fftw_complex));
f=(complex *)malloc(n*sizeof(fftw_complex));
d=(complex *)malloc(n*sizeof(fftw_complex));

//Bartlett window function generation
for(u=0,i=0;u<=m/2;i++,u++)
{
      w[i].re=2*u/m;
      w[i].im=0;
}

for(i=(m/2)+1,u=(m/2)+1;u<=m;i++,u++)
{
      w[i].re= 2-(2*u/m);
      w[i].im=0;
}
break;


case Han:     //Hanning window

fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

if((m%2)==0)
      alpha=(double)(m-1)/2;
else
```

```
        alpha=(double)m/2;

w=(complex *)malloc(n*sizeof(fftw_complex));
h=(complex *)malloc(n*sizeof(fftw_complex));
f=(complex *)malloc(n*sizeof(fftw_complex));
d=(complex *)malloc(n*sizeof(fftw_complex));

//Hanning window function generation
for(i=0;i<m;i++)
{
        w[i].re=0.5-0.5*cos(2*M_PI*i/m);
        w[i].im=0;
}
break;


case Ham:       //Hamming window

fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

if((m%2)==0)
        alpha=(double)(m-1)/2;
else
        alpha=(double)m/2;

w=(complex *)malloc(n*sizeof(fftw_complex));
h=(complex *)malloc(n*sizeof(fftw_complex));
f=(complex *)malloc(n*sizeof(fftw_complex));
d=(complex *)malloc(n*sizeof(fftw_complex));

//Hamming window function generation
for(i=0;i<m;i++)
{
        w[i].re=0.54-0.46*cos(2*M_PI*i/m);
        w[i].im=0;
}
break;


case Black:     //Blackmann window

fprintf(stderr,"Enter the filter tap length(m) : ");
scanf("%d",&m);
n=10*m;

if((m%2)==0)
        alpha=(double)(m-1)/2;
else
        alpha=(double)m/2;

w=(complex *)malloc(n*sizeof(fftw_complex));
h=(complex *)malloc(n*sizeof(fftw_complex));
```

```
            f=(complex *)malloc(n*sizeof(fftw_complex));
            d=(complex *)malloc(n*sizeof(fftw_complex));

            //Blackmann window function generation
            for(i=0;i<m;i++)
            {
                  w[i].re=0.42-0.5*cos(2*M_PI*i/m)+.08*cos(4*M_PI*i/m);
                  w[i].im=0;
            }
            break;


            case Kaiser:      //Kaiser window

            fprintf(stderr,"Enter the rolloff transition width(df) (in
MHz) : ");
            scanf("%lf",&df);
            fprintf(stderr,"Enter the peak approximation error(delta) :
");
            scanf("%lf",&del);

            dw=df*M_PI/bw;
                 //rolloff transition in terms of sampling frequency
            a=-20*log10(del);      // peak approximation error in db
            m=((int)((a-8)/(2.285*dw)+1.0));     // taplength calculation
            fprintf(stderr,"The filter tap length(m) is : %d\n\n",m);
            n=10*m;

            if((m%2)==0)
                  alpha=(double)(m-1)/2;
            else
                  alpha=(double)m/2;

            beta=bet(a);

            w=(complex *)malloc(n*sizeof(complex));
            f=(complex *)malloc(n*sizeof(complex));
            h=(complex *)malloc(n*sizeof(complex));
            d=(complex *)malloc(n*sizeof(complex));

            //Kaiser window function generation
            for(i=0;i<m;i++)
            {
                  x=beta*sqrt(1-(pow(((i-alpha)/alpha),2)));
                  w[i].re=ino(&x)/ino(&beta);
                  w[i].im=0.0;
            }
            break;

           }


      switch(ch2)
```

```
        {
                case LPF:       //Lowpass filter

                fprintf(stderr,"Enter the cutoff frequency(fc) (in MHz) :
");
                scanf("%lf",&fc);
                wc=fc*M_PI/bw;

                for(i=0;i<m;i++)
                {
                    //Lowpass filter co-efficients
                    f[i].re=sin(wc*(i-alpha))/(M_PI*(i-alpha));
                    f[i].im=0.0;
                }
                break;


                case HPF:       //Highpass filter

                fprintf(stderr,"Enter the cutoff frequency(fc) (in MHz) :
");
                scanf("%lf",&fc);
                wc=fc*M_PI/bw;

                for(i=0;i<=m;i++)
                {           //Highpass filter co-efficients
                            f[i].re=((sin(M_PI*(i-alpha)))-(sin(wc*(i-
alpha))))/(M_PI*(i-alpha));
                            f[i].im=0.0;
                }
                break;


                case BPF:       //Bandpass Filter

                fprintf(stderr,"Enter the lower cutoff frequency(fl) (in
MHz) : ");
                scanf("%lf",&fl);
                fprintf(stderr,"Enter the upper cutoff frequency(fu) (in
MHz) : ");
                scanf("%lf",&fu);
                wl=fl*M_PI/bw;
                wu=fu*M_PI/bw;

                for(i=0;i<m;i++)
                {           //Bandpass filter co-efficients
                            f[i].re=(sin(wu*(i-alpha))-sin(wl*(i-
alpha)))/(M_PI*(i-alpha));
                            f[i].im=0.0;
                }

                break;
```

```
            case BSF:    //Bandstop Filter

            fprintf(stderr,"Enter the lower cutoff frequency(fl) (in
MHz) : ");
            scanf("%lf",&fl);
            fprintf(stderr,"Enter the upper cutoff frequency(fu) (in
MHz) : ");
            scanf("%lf",&fu);
            wl=fl*M_PI/bw;
            wu=fu*M_PI/bw;
                //cutoff frequencies in terms of sampling frequency

            for(i=0;i<m;i++)
                {      //Bandstop Filter co-efficients
                      f[i].re=(sin(wl*(i-alpha))+sin(M_PI*(i-alpha))-
sin(wu*(i-alpha)))/(M_PI*(i-alpha));
                      f[i].im=0.0;
                }
            break;
      }


      plan=fftw_create_plan((int)n,FFTW_BACKWARD,FFTW_ESTIMATE);

      for(i=0;i<m;i++)
      {
            h[i].re=(w[i].re*f[i].re)-(w[i].im*f[i].im);
            h[i].im=(w[i].re*f[i].im)+(w[i].im*f[i].re);
      }

      for(i=m;i<n;i++)
            h[i]=(complex){0.0,0.0};
      fftw_one(plan,h,d);
           //computes FFT of elements in array h and stores in d
      power(d,h,n,bw);
      system("gnuplot pltfil-ka");     //function call to plot
      return 0;
   }

   double bet(double x)     //function to calculate beta
   {
         double beta;
         if(x<21)
               beta=0.0;
         if ((x>=21) && (x<=50))
               beta=0.5842*pow((x-21),0.4)+0.07886*(x-21);
         if (x>50)
               beta=0.1102*(x-8.7);
         return beta;
   }
```

```
double ino(x)      //Zeroth order modified Bessel's function
double  *x;
{
      double ret_val;
      static double e;
      static int i;
      static double t,y,de,xi,sde;

      y=*x/(double)2.;
      t=(double)1e-20;
      e=(double)1.;
      de=(double)1.;

      //summation of 25 terms
      for (i=1;i<=25;++i)
      {
            xi=(double) i;
            de=de*y/xi;
            sde=de*de;
            e+=sde;
            if (e*t>sde)
            break ;
      }

      ret_val=e;
      return ret_val;
}


double power(complex *d,complex *h,int n,double bw)
//program to calculate power
{
   double pow,db,phase,powh,phaseh;
   int i;
   FILE *fp1;

     fprintf(stderr,"\nTHE FILTER CO-EFFICIENTS ARE STORED IN THE
FILE fil-ka.res\n\n");

     fp1=fopen("fil-ka.res","w");

     fprintf(fp1,"Freq. in MHz\tPower in dB\tPhase(freq.)\tn\tFil-
coeff\tPhase(time)\n");

   for(i=0;i<(int)n/2+1;i++)
   {
         pow=hypot(d[i].re,d[i].im);      //magnitude or power
         phase=atan2(d[i].im,d[i].re);    //phase in frequency domain
         phaseh=atan2(h[i].im,h[i].re);     //phase in time domain

         if(pow<1.0e-5)
               db=-100;
         else
```

```
                db=20*log10(pow); //to express magnitude in decibels

            fprintf(fp1,"\n%lf\t%lf\t%lf\t%d\t%lf\t%lf",(2*i*bw/n),db,ph
ase,i,h[i].re,phaseh);
            }
            fclose(fp1);
      return 0;
   }
```

//program to plot

```
//results are stored in file 'fil-ka.res'

set grid
set title 'FREQUENCY RESPONSE'
set xlabel 'FREQUENCY IN MHz'
set ylabel 'POWER IN DECIBELS'
plot 'fil-ka.res' u 1:2 t 'MAG' w l
pause -1 "hit return to continue"

set title 'PHASE PLOT IN FREQUENCY DOMAIN'
set xlabel 'FREQUENCY IN MHz'
set ylabel 'PHASE'
plot 'fil-ka.res' u 1:3 t 'PHASE' w l
pause -1 "hit return to continue"

set title 'TIME DOMAIN RESPONSE'
set xlabel 'n'
set ylabel 'fil[n]'
plot 'fil-ka.res' u 4:5 t 'MAG' w l
pause -1 "hit return to continue"

set title 'PHASE PLOT IN TIME DOMAIN'
set xlabel 'n'
set ylabel 'PHASE'
plot 'fil-ka.res' u 4:6 t 'PHASE' w l
pause -1 "hit return to continue"
```

//TO FIND THE RESPONSE OF THE FILTER BANK WITH FILTERS OF DIFFERENT TAP LENGTHS

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include"fftw.h"

typedef fftw_complex complex;

int main(int argc,char **argv)
{
   int i,q,t,n,m[]={0,0,0,0,0,0,0,0,0,0};
   double alpha[10],beta[10],a[10],del[10],dw[10],df[10];
   double mmax,bw,fl[10],fu[10],wl[10],wu[10],x[10],y[10];
   complex *h,*g,*s;
   double ino(double *);
   double bet(double);
   double power(complex *,int,double);
   double max(int *,int);
   FILE *fp;
   fftw_plan plan;

   if(argc==1)
         fp=stdin;
   else
         fp=fopen(argv[1],"rt");

   fprintf(stderr,"Enter the number of filters in the filter bank
(<10) : ");
   scanf("%d",&q);
   fprintf(stderr,"Enter the total bandwidth(bw) of the filter bank
(in MHz) : ");
   scanf("%lf",&bw);

   for(t=0;t<q;t++)
        //accept parameters of all BPFs in filter bank and compute
co-efficients of filter bank
   {
         fprintf(stderr,"Enter the lower cutoff frequency(fl) for
filter %d (in MHz) : ",t+1);
         scanf("%lf",&fl[t]);
         fprintf(stderr,"Enter the upper cutoff frequency(fu) for
filter %d (in MHz) : ",t+1);
         scanf("%lf",&fu[t]);
         fprintf(stderr,"Enter the rolloff transition width(df) for
filter %d (in MHz) : ",t+1);
         scanf("%lf",&df[t]);
         fprintf(stderr,"Enter the peak approximation error(delta)
for filter %d : ",t+1);
         scanf("%lf",&del[t]);

         dw[t]=df[t]*M_PI/bw ;
```

```
            a[t]=-20*log10(del[t]);
            m[t]=((int)((a[t]-8)/(2.285*dw[t])+1.0));
      //arrays store the parameters of all the filters in filter bank
            fprintf(stderr,"The tap length(m) for filter %d is :
%d\n\n",t+1,m[t]);

            if((m[t]%2)==0)
                  alpha[t]=(double)(m[t]-1)/2;
            else
                  alpha[t]=(double)m[t]/2;

            beta[t]=bet(a[t]);
            wl[t]=fl[t]*M_PI/bw;
            wu[t]=fu[t]*M_PI/bw;
      }
      mmax=max(m,q);
           //function to compute the maximum of taplenths of all the
filters in the filter bank.
      n=10*mmax;

      s=(complex *)malloc(n*sizeof(complex));
      g=(complex *)malloc(n*sizeof(complex));
      h=(complex *)malloc(n*sizeof(complex));

      for(t=0;t<q;t++)
      {
            for(i=0;i<m[t];i++)
                 //filter bank implemented using Kaiser window
            {
                  x[t]=beta[t]*sqrt(1-(pow(((i-alpha[t])/alpha[t]),2)));
                  y[t]=(sin(wu[t]*(i-alpha[t]))-sin(wl[t]*(i-
alpha[t])))/(M_PI*(i-alpha[t]));
                  h[i].re=(y[t]*ino(&x[t])/ino(&beta[t]));
                  h[i].im=0.0;
            }



            for(i=m[t];i<n;i++)
                  h[i]=(complex){0.0,0.0};

            for(i=0;i<n;i++)
            {
                  s[i].re+=h[i].re;
                  s[i].im+=h[i].im;
            }
      }

      plan=fftw_create_plan((int)n,FFTW_BACKWARD,FFTW_ESTIMATE);

      fftw_one(plan,s,g);      //function call to perform FFT calculation
      power(g,n,bw);
      system("gnuplot pltbank");      //function call to plot results
```

```
    return 0;
}


double max(int *x,int q)
     //function which calculates the maximum of filter lengths of all
the filters in the filter bank
{
    int k;
    double ma;

    ma=x[0];
    for(k=0;k<q;k++)
    {
          if(x[k]>ma)
                ma=x[k];
    }
    return ma;
}

double bet(double x)
     //function to compute the shape parameter beta
{
    double beta;

    if(x<21)
    {
          beta=0.0;
    }
    if ((x>=21) && (x<=50))
    {
          beta=0.5842*pow((x-21),0.4)+0.07886*(x-21);
    }
    if (x>50)
                beta=0.1102*(x-8.7);
    return beta;
}


double ino(x)
     // to calculate zeroth order modified Bessel's function
double  *x;
{
          double ret_val;
    static double e;
    static int i;
    static double t,y,de,xi,sde;

    y=*x/(double)2.;
    t=(double)1e-20;
    e=(double)1.;
    de=(double)1.;
    for(i=1;i<=25;++i)
```

```
        //summation of 25 terms
        {
                xi=(double)i;
                de=de*y/xi;
                sde=de*de;
                e+=sde;
                if(e*t>sde)
                        break ;
        }
        ret_val=e;
        return ret_val;
    }


    double power(complex *g,int n,double bw)
         //function to compute the power
    {
        double pow,db;
        int i;
        FILE *fp1;

        fprintf(stderr,"\nTHE FILTER CO-EFFICIENTS ARE STORED IN THE FILE
bank.res\n\n");

        fp1=fopen("bank.res","w");

        fprintf(fp1,"Freq. in MHz\tPower in dB\n");

        for(i=0;i<(int)n/2+1;i++)
        {
                pow=hypot(g[i].re,g[i].im);      // to find power of DFT

                if(pow<=1.0e-5)
                        db=-100;
                else
                        db=20*log10(pow);       //power expressed in db

                fprintf(fp1,"\n%lf\t%lf",(2*i*bw)/n,db);
        }
        fclose(fp1);
        return 0;
    }
```

//program to plot

```
//results are saved in file 'bank.res'

set grid
set title 'FILTER BANK WITH FILTERS OF DIFFERENT TAP LENGTHS'
set xlabel 'FREQUENCY IN MHz'
set ylabel 'POWER IN DECIBELS'
```

```
plot 'bank.res' u 1:2 t 'MAG' w l
pause -1 "hit return to continue"
```

BIBLIOGRAPHY

- Digital Signal Processing Principles, Algorithms, and Applications
  John G. Proakis
  Dimitris G. Manolakis

- Discrete–Time Signal Processing
  Alan V. Oppenheim
  Ronald W. Schafer

- Theory and Application of Digital Signal Processing
  Lawrence R. Rabiner
  Bernard Gold

- The Scientist and Engineer's Guide To Digital Signal Processing
  Steven W. Smith

- Signal Analysis
  Athanasios Papoulis

- The Fast Fourier Transform
  E. Oran Brigham