ACKNOWLEDGEMENTS

First and foremost, I am very grateful to my supervisor Prof. Dr. Doğan İbrahim for providing his overall supervision, direction, encouragement and assistance throughout the duration of my thesis.

I would like to acknowledge gratefully all the staff members in the Department of Computer Engineering of the Near East University for their support and valuable advice given throughout the duration of my studies for the Degree of Master of Science.

And a big thank to my brothers and sisters for their constant love and support throughout the years. I would like to express my deepest gratitude for the constant support, understanding and love that I received from my wife Najat during the past years.

Finally, I will not forget the people who supported me in North Cyprus and made me enjoy the last two years I spent studying.

ABSTRACT

A fundamental aspect of signal processing is filtering. Filtering involves the manipulation of the spectrum of a signal by passing or blocking certain portions of the spectrum, depending on the frequency of those portions.

A key element in processing digital signals is the filter. Filters are designed according to what kind of manipulation of the signal is required for a particular application. There are two basic types of digital filters: Infinite Impulse Response Filter (IIR) and the Finite Impulse Response Filter (FIR).

A microcontroller is a computer on a chip. Because they have on-chip memory and I/O circuitry and other circuitries that enable them to function as small standalone computers without other supporting circuitry.

The Microchip PICmicro PIC16F87X family of microcontrollers are popularly known for their logic and controlling functions. These features make PIC16F87X microcontrollers a competent choice for applications where logic and controlling functions are combined with signal processing applications.

This thesis describes the development of hardware and software for the actual realization of digital filters on PIC microcontroller. An IIR type 2^{nd} order Butterworth digital filter has been implemented on a PIC16F877 microcontroller. It is shown in the thesis that digital filters can easily be realized on microcontrollers if a high-level programming language is used.

The PIC microcontroller has been used for the digital filter realization since it is a low-cost, widely available and a popular microcontroller.

The thesis also describes the development of a Matlab based software program to design both IIR and FIR type digital filters. The program is user friendly with a graphical user interface. Using the program the user can design and obtain the parameters for IIR and FIR type lowpass, highpass, and bandpass digital filters. In addition, the frequency and the phase responses of the designed filters can be plotted.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

Several specific terms are used in this thesis in conjunction with hardware and software components, or related to the general theory of digital signal processing. These different abbreviations are listed below.

A/D: Analog to Digital
BPF: Band-Pass Filter
D/A: Digital to Analog
DFT: Discrete Fourier Transform
DSP: Digital Signal Processing or Digital Signal Processor
FFT: Fast Fourier Transform
FIR: Finite Impulse Response
HPF: High-Pass Filter
IDFT: Inverse Discrete Fourier Transform
IIR: Infinite Impulse Response
I/O: Input/Output
LPF: Low-Pass Filter
LTI: Linear time-invariant

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Digital signal processing (DSP) is concerned with the representation of signals in digital form, and with the processing of these signals and the information that they carry. Since the early 1970's, when the first DSP chips were introduced, the field of digital signal processing has evolved dramatically. Digital signal processing has become an integral part of many commercial products and applications, and is becoming a commonplace term. DSP is useful in almost any application that requires the high-speed processing of a large amount of numerical data. The data can be anything from position and velocity information for a closed-loop control system, to two-dimensional video images, to digitized audio and vibration signals.

In signal processing, signals are often encountered that contain unwanted information, such as random noise or interference, or there is a need to selectively extract a signal of interest merged with several other signals. Filters are used in these situations to separate the signals of interest from others.

Filters can be analog or digital. Analog filters use electronic circuits to produce the required filtering effect, while a digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general purpose computing machine, such as a PIC microcontroller or a specialized DSP chip. There are two basic types of digital filters: Infinite Impulse Response Filter (IIR) and the Finite Impulse Response Filter (FIR).

Today's microcontrollers are fast, cheap and low power machines that can handle just about any control or signal processing application. The microcontroller is a direct descendent of the CPU, in fact every microcontroller has a CPU as the heart of the device. It is therefore important to understand the CPU in order to ultimately understand the microcontroller. The central processor unit is the brain of the microcontroller. The CPU controls all functions and uses the program that resides in RAM, EEPROM or EPROM to function.

Microcontrollers have traditionally been programmed using the native assembly language of the target processor. It is very common nowadays to use high-level languages such as Basic, Pascal, and C in programming microcontrollers. Assembly language has the advantage that the execution speed is very fast. On the other hand, developing an assembly language based program is a complex task. High-level

languages have the advantage that it is much easier to develop and maintain programs developed using these languages.

The objectives of the work presented within this thesis are to develop a microcontroller based digital filter. PIC16F87X series of microcontrollers are used in the thesis for the hardware realization of the filter. The high-level programming language C is used for software implementation on the PIC microcontroller.

In addition, it is to develop a Matlab based software for the design of FIR and IIR digital filters with given filter specifications, and to plot the frequency and the phase responses of the designed filters.

This thesis is organized into five chapters. The first three chapters present background information on the signal processing, analog and digital filters, FIR and IIR filter structures and their design. The final two chapters describe the details of the developed software program for designing and plotting the FIR and IIR digital filters and the microcontroller based digital filter realization.

Chapter 1 is an introduction to analog and digital signal processing. Advantages, disadvantages and applications of digital signal processing are also presented in this chapter.

In Chapter 2 the theory of filters in general, analog filter types, specifications and filter responses are discussed. Digital filters are introduced.

Chapter 3 discusses the digital FIR and IIR filter structures and design. First, the structure of the filters using different forms is given. The design of the FIR filters based on truncated Fourier series and the design based on frequency sampling approach are derived. The design of the IIR filters by bilinear transformation is presented.

Chapter 4 describes the software program developed by the author for the design of digital filters. In this chapter the interfaces and the functions of each component of the program are explained.

Chapter 5 describes the developed microcontroller hardware and software for the implementation of digital filters on a PIC microcontroller using a high-level programming language.

CHAPTER 1

SIGNALS AND SIGNAL PROCESSING

1.1 Overview

Signals play an important role in our daily life. Examples of signals that we encounter frequently are speech, music, pictures, and video signals.

This chapter provides background information about signals and signal processing. Also the advantages and disadvantages of digital signal processing will be presented in this chapter.

1.2 What Are Signals?

A signal is a function of an independent variable such as time, distance, position, temperature, pressure. For example speech and music signals represent air pressure as a function of time at a point in space. A black-and-white picture is a representation of light intensity as a function of two spatial coordinates. The video signal in television consists of a sequence of images, called frames, and is a function of three variables: two spatial and time. [1]

Most signals we encounter are generated by natural means. However, a signal can also be generated synthetically or by computer simulation. Signal caries information and the objective of signal processing is to extract the information carried by the signal. The method of information extraction depends on the type of signal and the nature of information being carried by the signal.

1.3 Classification of Signals

A signal is also a time-varying measurable quantity whose variation normally conveys information. The quantity is often a voltage obtained from some transducer e.g. a microphone. It is useful to define two types of signals, analog and discrete-time signals (digital signals). [7]

1.3.1 Analog Signals

Analog signals, which are continuous functions of time (t) measured in seconds, and exist for all values of time in the range -∞ to +∞. An example of analog signal is shown in Figure 1.1.

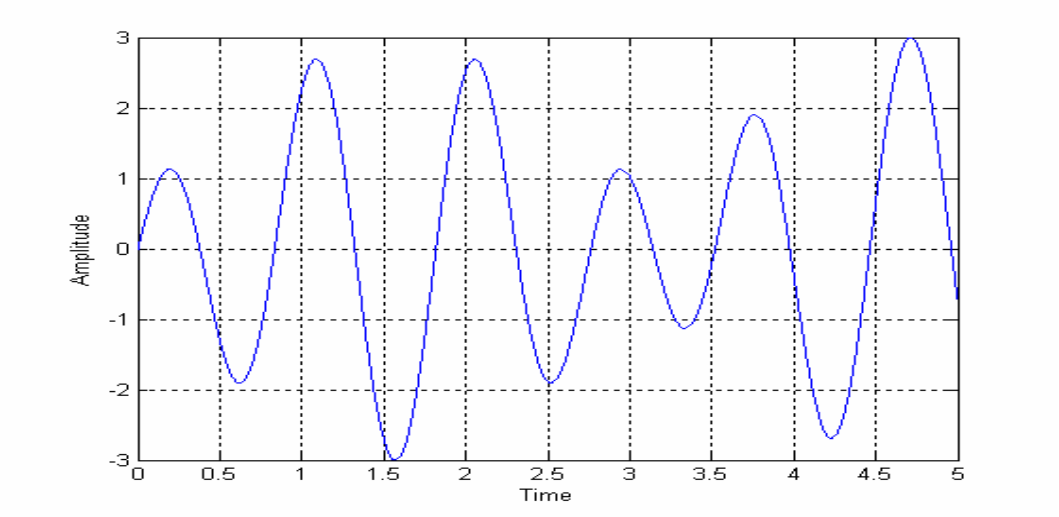

Figure 1.1 Analog Signals

1.3.2 Discrete-Time Signals

Discrete-time signals exist only at discrete points in time. Such a signal is often obtained by sampling an analog signal, i.e. measuring its value at discrete points in time. Sampling points are usually separated by equal intervals of time, say T seconds. Given an analog signal x(t) and denoting by x[n] the value of x(t) when t=nT, the sampling process produces a sequence of numbers: { .., x[-2], x[-1], x[0], x[1], x[2], .. } which is referred to as {x[n]} or ' the sequence x[n] '. The sequence exists for all integer values of n in the range -∞ to ∞.

Examples of discrete time signals are:

1. {..., -4, -2, 0, 2, 4, 6, ...} i.e. a sequence whose nth element, x[n], is defined by the formula x[n] = 2n. It is useful to underline the sample corresponding to n = 0.

2. {..., 0, ..., 0, 0, 1, 1, 1, ..., 1, ...} i.e. a " unit step " sequence whose nth element is:

$$u[n] = \begin{cases} 1 , & n \geq 0 \\ 0 , & n < 0 \end{cases}$$

Discrete time signals are represented graphically as shown in Figure 1.2

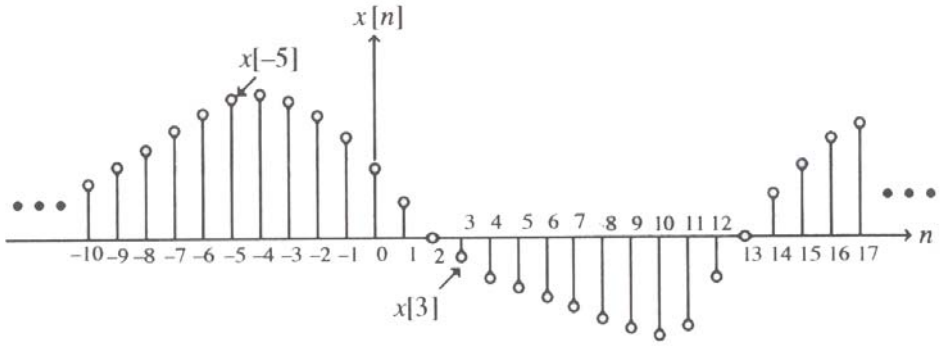

Figure 1.2 Discrete-time Signal

Discrete time signals are often generated by 'analog to digital conversion' (ADC) devices which produce binary numbers to represent sampled voltages or currents. The accuracy of conversion is determined by the 'word-length ' of the ADC device, i.e. the number of bits available for each binary number.

The process of truncating or rounding the sampled value to the nearest available binary number is termed ' quantization ' and the resulting sequence of quantized numbers is termed a 'digital signal'. A digital signal is therefore a discrete time signal with each sample digitized for arithmetic processing.

1.4 Analog Signal Processing

Analog signals may be "processed" in various ways by circuits typically consisting of resistors, capacitors, inductors, transistors and operational amplifiers. Examples of the type of processing operations that may be carried out are:

1. **Scaling (amplification or attenuation)**

Scaling is simply the multiplication of the signal by a positive or a negative constant. In the case of analog signals this operation is usually called amplification if the magnitude of the multiplying constant called gain is greater than one. If the magnitude of the multiplying constant is less than one, the operation is called attenuation.

2. **Modulation and Demodulation**

For transmission of signals over long distances, a transmission media such as a cable, optical fiber, or the atmosphere is employed. Each such medium has a bandwidth that is more suitable for the efficient transmission of signals in the high-frequency range.

As a result, for the transmission of a low-frequency signal over a channel, it is necessary to transform the signal to high-frequency signal by means of a modulation operation. At the receiving end, the modulated high-frequency signal is demodulated, and the desired low-frequency signal is then extracted by further processing.

3. **Multiplexing and Demultiplexing**

For an efficient utilization of a wideband transmission channel, many narrow-bandwidth low-frequency signals are combined to form a composite wideband signal that is transmitted as a single signal.

The process of combining these signals is called multiplexing which is implemented to ensure that a replica of the original narrow-bandwidth low-frequency signals can be recovered at the receiving end. The recovery process is called demultiplexing.

4. **Filtering**

One of the most widely used complex signal processing operation is filtering. Filtering is used to pass certain frequency components in a signal through the system without any distortion and to block other frequency components. The system implementing this operation is called a filter. [1]

For example, imagine a device for measuring the electrical activity of a baby's heart (EKG) while still in the womb. The raw signal will likely be corrupted by the

breathing and heartbeat of the mother. A filter might be used to separate these signals so that they can be individually analyzed.

There are many filter types, but the most common are lowpass, highpass, bandpass, and bandstop. A lowpass filter allows only low frequency signals (below some specified cutoff) through to its output, so it can be used to eliminate high frequencies.

A lowpass filter is handy, in that regard, for limiting the uppermost range of frequencies in an audio signal; it's the type of filter that a phone line resembles.

A highpass filter does just the opposite, by rejecting only frequency components below some threshold. An example highpass application is cutting out the audible 60Hz AC power "hum", which can be picked up as noise accompanying almost any signal in the U.S.

The designer of a cell phone or any other sort of wireless transmitter would typically place an analog bandpass filter in its output RF stage, to ensure that only output signals within its narrow, government-authorized range of the frequency spectrum are transmitted.

Engineers can use bandstop filters, which pass both low and high frequencies, to block a predefined range of frequencies in the middle. [2]

1.5 Digital Signal Processing

Most – though by no means all – of the signals that we will encounter will finally be processed in digital form. A signal will start life as an analog quantity which will be continuously variable. The processing will be often done on these signals when they have been turned into digital format. Digital signals may be "processed" using programmed computers, microcomputers or special purpose digital hardware.

Digital Signal Processing (DSP) is distinguished from other areas in computer science by the unique type of data it uses: signals. In most cases, these signals originate as sensory data from the real world: seismic vibrations, visual images, sound waves, etc.

DSP is the mathematics, the algorithms, and the techniques used to manipulate these signals after they have been converted into a digital form. This includes a wide variety of goals, such as: enhancement of visual images, recognition and generation of speech, compression of data for storage and transmission, etc. [2]

1.6 Digital Signal Processing Applications

Digital Signal Processing has applications in many areas in science and engineering like Medical, Commercial, Communication, etc. Table 1-1 illustrates a few of these varied applications.

Table 1-1 Application of Digital Signal Processing [2]

Area	Application
Space	-Space photograph enhancement -Intelligent sensory analysis by remote space probes
Medical	-Diagnostic imaging (CT, MRI, ultrasound, and others) -Electrocardiogram analysis -Medical image storage/retrieval
Commercial	-Image and sound compression for multimedia presentation -Movie special effects -Video conference calling
Communication	-Voice and data compression -Echo reduction -Signal multiplexing -Filtering
Military	-Radar -Sonar -Ordnance guidance -Secure communication
Industrial	-Oil and mineral prospecting -Process monitoring & control -CAD and design tools

1.7 Why Process Signals Digitally?

Signals are naturally analog and need to be converted to digital form for them to be processed digitally. Figure 1.3 illustrates this process

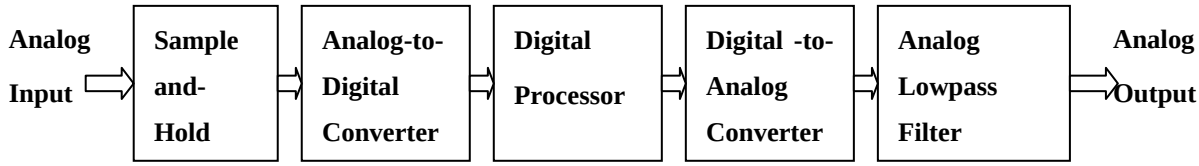

Figure 1.3 Digital Processing of an Analog Signal

The question might be why process signals digitally, since it may seem easier to process them as analog signals. [8]

There are more advantages to processing signals digitally than there are disadvantages. A list of a few is given with here and discussion will be done on an individual basis. These are:

- Programmability
- Stability
- Repeatability
- Adaptability

1.7.1 Programmability

The most important reason why Digital Signal Processing is favored over analog signal processing is that it is possible to design one hardware configuration that can be programmed to perform a very wide variety of signal processing tasks, simply by loading in different software. For example, a digital filter may be reprogrammed from a low pass to a high pass with no change in the hardware, which in an analog system would result in a complete change of circuit components.

This programmability of these devices makes them more suitable because they can be easily upgraded by simply changing the software in the system.

1.7.2 Stability

Performance is one thing we look into very critically when it comes to the design of any system. In analog systems the individual components (resistors, capacitors etc.) change their characteristics with changes in temperature.

Digital circuits will show no variation with temperature throughout their guaranteed operating range. Another form of variability that affects analog circuits is component aging. DSP circuits can be programmed to detect and compensate for changes in the aging of analog and mechanical parts of the system.

1.7.3 Repeatability

This refers to the ability to produce the same output with different systems with the identical specifications. This is one great advantage of a digital system, because analog circuit components have a tolerance specification which causes a spread of performance in analog systems. Resistors can have a tolerence of 5% of their value, depending on the prize of the component.

1.7.4 Adaptivity

This is the ability to change its parameters according to a change in the environment. An example of this is the noise cancellation system in a car. In this case the noise that is cancelled is originally caused by the engine and the resonances set up in the body panels by engine vibrations.

The noise cancellation system takes the engine speed as a reference and attempts to produce an “anti-noise“ signal to cancel the cockpit noise. There are microphones in each headrest that determine the success of the attempt. Based on the changes detected by the microphones, the system changes the characteristics of the anti-noise until the best noise reduction is achieved. When the engine speed changes, the systems adapts once more to the new engine speed.

This answers the question asked earlier about why convert from analog to digital. Only a few of the advantages of digital signal processing over analog are listed. The list goes on and can be found in books that cover the subject of digital signal processing.

1.8 Disadvantages of Digital Signal Processing

Since no system is perfect there are also some disadvantages of digital processing, some of them being:

- DSP designs can be expensive especially for high bandwidth signals where fast analog/digital conversion is required.
- The design of DSP systems can be extremely time-consuming and a highly complex and specialized activity. There is an acute shortage of computer science and electrical engineering graduates with the knowledge and skill required.
- The power requirements for DSP devices can be high, thus making them unsuitable for battery powered portable devices such as mobile telephones. Fixed point processing devices (offering integer arithmetic only) are available which are simpler than floating point devices and less power consuming. However the ability to program such devices is a particularly valued and difficult skill.

However the advantages outweigh the disadvantages and with the cost of digital processor hardware constantly decreasing, there is an increase in the use of DSPs. [8]

1.9 Summary

This chapter presented information about signal definition, classification, and some examples of signals. Advantages, disadvantages and applications of digital signal processing have also been listed in this chapter.

CHAPTER 2

ANALOG AND DIGITAL FILTER

2.1 Overview

Filters are signal conditioners. They function by accepting an input signal, blocking pre-specified frequency components, and passing the original signal minus those components to the output.

In this chapter the theory of Filters in general, analog filter design and filter responses will be discussed. Digital filtering will be introduced.

2.2 Filter

Filtering is a process of selecting, or suppressing, certain frequency components of a signal. A coffee filter allows small particles to pass while trapping the larger grains. A filter does a similar thing, but with more subtlety.

In signal processing, the function of a filter is to remove unwanted parts of the signal, such as random noise, or to extract useful parts of the signal, such as the components lying within a certain frequency range. The following block diagram illustrates the basic idea. [8]

Figure 2.1 Basic idea of how filters work

There are two main kinds of filters, analog and digital. They are quite different in their physical makeup and in how they work. An analog filter uses analog electronic circuits made up from components such as resistors, capacitors and op amps to produce the required filtering effect. Such filter circuits are widely used in such applications as noise reduction, video signal enhancement, graphic equalizers in hi-fi systems, and many other areas.

There are well-established standard techniques for designing an analog filter circuit for a given requirement. At all stages, the signal being filtered is an electrical voltage or current which is the direct analogue of the physical quantity (e.g. a sound or video signal or transducer output) involved.

A digital filter uses a digital processor to perform numerical calculations on sampled values of the signal. The processor may be a general-purpose computer such as a PC, or a specialized Digital Signal Processor chip. [5]

2.2.1 Analog Filter Types

The most common filter Types are the Butterworth, Chebyshev, and Bessel types. Many other types are available, but 90% of all applications can be solved with one of these three.

Butterworth ensures a flat response in the passband and an adequate rate of rolloff. A good "all rounder" the Butterworth filter is simple to understand and suitable for applications such as audio processing.

The Chebyshev gives a much steeper rolloff, but passband ripple makes it unsuitable for audio systems. It is superior for applications in which the passband includes only one frequency of interest (e.g., the derivation of a sinewave from a square wave, by filtering out the harmonics).

The Bessel filter gives a constant propagation delay across the input frequency spectrum.

2.2.2 Standard Analog Filter Blocks

The generic filter structure (Figure 2.2) lets us realize a highpass or lowpass implementation by substituting capacitors or resistors in place of components G1-G4. Considering the effect of these components on the op-amp feedback network, one can easily derive a lowpass filter by making G2/G4 into capacitors and G1/G3 into resistors. Making G2/G4 into resistors and G1/G3 into capacitors yields the highpass implementation.

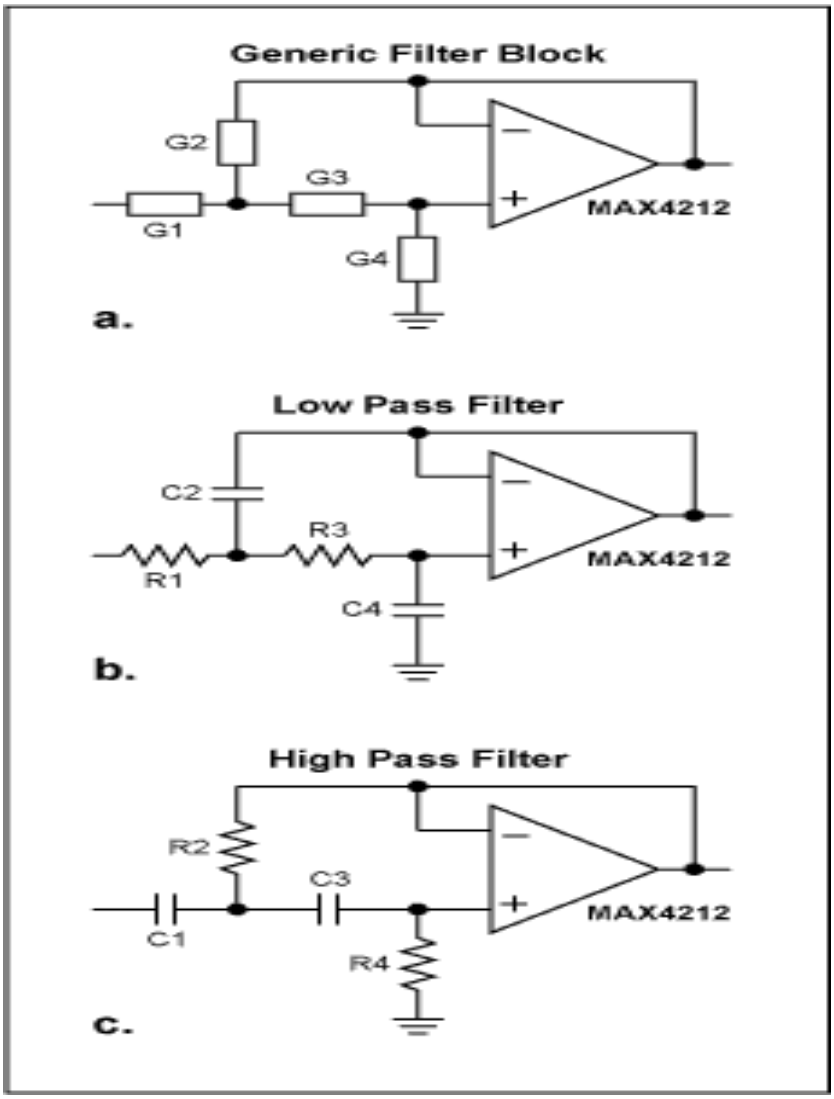

Figure 2.2 Standard Analog Filter Blocks

2.2.3 Filtering Definitions

The range of frequencies that is allowed to pass through is called the passband, and the range of frequencies that is blocked by the filter is called the stopband. Various types of filters can be defined depending on the nature of the filtering operation. In most cases the filtering operation for analog signals is linear and is described by the convolution integral

$$y(t) = \int_{-\infty}^{\infty} h(t-\tau)x(\tau)d\tau \qquad (2.1)$$

Where x(t) is the input signal and y(t) is the output of the filter characterized by an impulse response h(t).

A lowpass filter passes all low-frequencies components below a certain specified frequency f_c called the cutoff frequency, and blocks all high-frequency components above f_c. A highpass filter passes all high-frequencies components above a certain cutoff frequency f_c, and blocks all high-frequency components below f_c.

A bandpass filter passes all frequency components between two cutoff frequencies f_{c1} and f_{c2}, and blocks all frequency below the frequency f_{c1} and above the frequency f_{c2}.

A bandstop filter blocks all frequency components between two cutoff frequencies f_{c1} and f_{c2}, and passes all frequency below the frequency f_{c1} and above the frequency f_{c2}. A bandstop filter designed to block a single frequency component is called a notch filter.

A signal may get corrupted unintentionally by an interfering signal called interference or noise. In many applications the desired signal occupies a low-frequency band from dc to some frequency f_L Hz, and its corrupted by a high–frequency noise with frequency components above f_H Hz with $f_H > f_L$. In such cases, he desired signal can be recoverd from the noise-corrupted signal by passing the latter through a lowpass filter with a cutoff frequency f_c where $f_L < f_c < f_H$. A common source of noise is power lines radiating electric and magnetic fields. The noise generated by power lines appears as 60-Hz sinusoidal signal corrupting the desired signal and can be removed by passing the corrupted signal through a notch filter with a notch frequency at 60 Hz. [1]

2.2.4 Ideal Filters

The design of a classical analog filter, in many respects, remains today as it was practiced during the early days of radio. The design objective of the radio engineer was one of shaping the frequency spectrum of a received or transmitted signal using modulators, demodulators, and frequency selective filters. Filter designs were based, to various degrees, on lowpass, highpass, bandpass, bandstop, and all-pass models. The frequency response of an analog filter system is defined by H(jΩ), where Ω is called the analog frequency and is measured in radians per second. The mathematical specifications of classic ideal filters are summarized below: [6]

Table 2-1 mathematical specifications of classic ideal filters

Filter type	Mathematical specifications
Ideal Lowpass	$\|H(j\Omega)\| = \begin{cases} 1 & \text{for } \Omega \in [-\Omega_p, \Omega_p] \\ 0 & \text{otherwise} \end{cases}$
Ideal Highpass	$\|H(j\Omega)\| = \begin{cases} 0 & \text{for } \Omega \in [-\Omega_p, \Omega_p] \\ 1 & \text{otherwise} \end{cases}$
Ideal Bandpass	$\|H(j\Omega)\| = \begin{cases} 1 & \text{for } \Omega \in [-\Omega_H, -\Omega_L] \text{ or } \Omega \in [\Omega_L, \Omega_H] \\ 0 & \text{otherwise} \end{cases}$
Ideal Bandstop	$\|H(j\Omega)\| = \begin{cases} 0 & \text{for } \Omega \in [-\Omega_H, -\Omega_L] \text{ or } \Omega \in [\Omega_L, \Omega_H] \\ 1 & \text{otherwise} \end{cases}$
All-pass	$\|H(j\Omega)\| = 1 \; \forall \Omega$

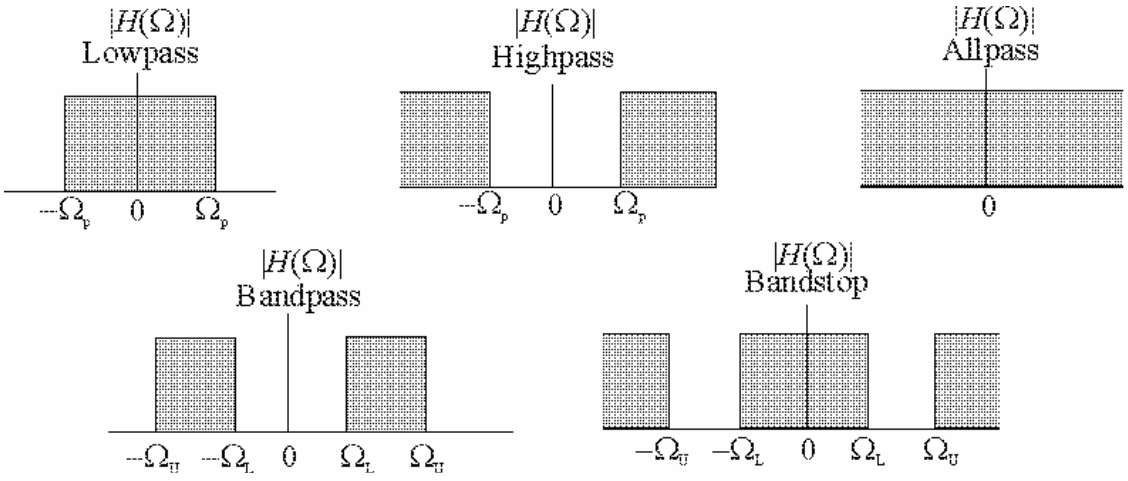

Figure 2.3 Classical ideal filter models

2.3 Analog Lowpass Filter Design

2.3.1 Filter Specification

In practice, the magnitude response characteristic of a lowpass filter in the passband and in the stopband cannot be constant and are therefore specified with some acceptable tolerances. A transition band is specified between the passband and the stopband to permit the magnitude to drop off smoothly. [1]

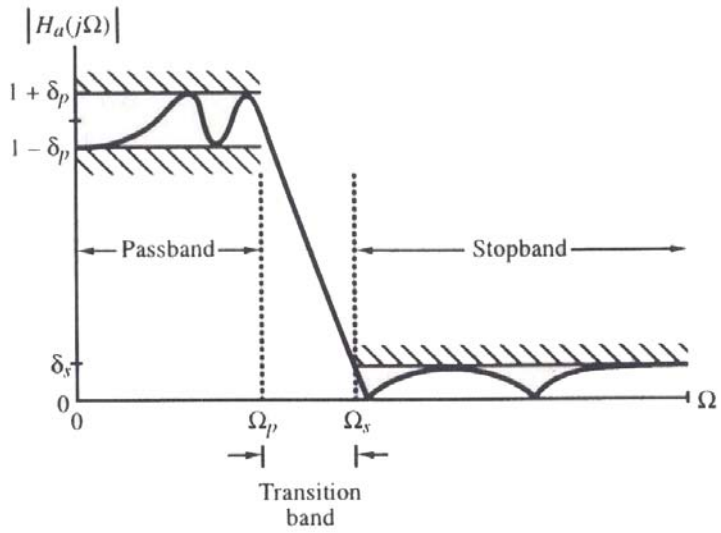

Figure 2.4 Normalized magnitude specification for an analog lowpass filter

As indicated in figure 2.4, in the *passband* defined by $0 \le \Omega \le \Omega_p$, we require

$$1-\delta_p \le |H_a(j\Omega)| \le 1+\delta_p, \quad \text{for } |\Omega| \le \Omega_p. \tag{2.2}$$

implying that the magnitude approximate unity within an error of $\pm\delta_p$.In the stopband, defined by $\Omega_s \le \Omega \le \infty$, we require

$$|H_a(j\Omega)| \le \delta_s, \qquad \text{for } \Omega_s \le |\Omega| \le \infty. \tag{2.3}$$

implying that the magnitude approximate zero within an error of δ_s. The frequency Ω_p and Ω_s are, respectively, called the *passband edge frequency* and the *stopband edge frequency*. The limits of the tolerances in the passband and the stopband, δ_p and δ_s, are called *ripples*.

Usually these ripples are specified in dB in term of the *peak passband ripple* α_p and the *minimum stopband attenuation* α_s, defined by

$$\alpha_p = -20\log_{10}(1-\delta_p) \text{ dB} \tag{2.4}$$

$$\alpha_s = -20\log_{10}(\delta_s) \text{ dB} \tag{2.5}$$

Often, the filter specifications are given in term of the *loss function* or *attenuation function* $a(\Omega)$ in dB, which is defined as the negative of the gain in dB, i.e., $-20\log_{10}|H_a(j\Omega)|$.

In analog filter theory two additional parameters are defined. The first one, called the *transition ratio* or *selectivity parameter*, is defined by the ratio of the passband edge frequency Ω_p and the stopband edge frequency Ω_s, and is usually denoted by

$$k = \Omega_p / \Omega_s \tag{2.6}$$

Note that for a lowpass filter, $k < 1$. The second one, called the *discrimination parameter* and defined as

$$k_1 = \varepsilon / \sqrt{A^2 - 1} \tag{2.7}$$

2.4 Butterworth Filter

The Butterworth filter is one of the most basic electronic filter designs. It is designed to have a frequency response which is as flat as mathematically possible in the passband.

It was first described by the British engineer S. Butterworth. The most basic Butterworth filter is the standard first-order low-pass filter, which can be modified into a high-pass filter, or placed in series with others to form band-pass and band-stop filters, and higher order versions of these. [2]

2.4.1 Butterworth Approximation

The magnitude-squared response of an analog lowpass Butterworth filter of N-th order is given by

$$|H_a(j\Omega)|^2 = \frac{1}{1+(\Omega/\Omega_c)^{2N}} \tag{2.8}$$

It can be easily shown that the first 2*N*-1 derivatives of $|H_a(j\Omega)|^2$ at $\Omega = 0$ are equal to zero, and as a result, the Butterworth filter is said to have maximally flat magnitude at $\Omega = 0$. The gain in dB is equal to zero, and at $\Omega = \Omega_c$, the gain is approximately –3 dB. Therefore, Ω_c is often called the 3-dB cutoff frequency. [1]

As mentioned, the frequency response of the Butterworth filter is maximally flat (i.e. no ripples) in the passband, and a frequency response which slopes off towards zero in the stopband. As shown in figure 2.5 on a logarithmic Bode plot, the cut band slopes off linearly towards negative infinity.

For a first-order filter, the cut line slopes off at -6 dB per octave, for second-order, -12 dB per octave, etc. All first-order filters are actually the same filter and so have the same frequency response. The Butterworth is the only filter that maintains this same shape for higher orders stopband. [4]

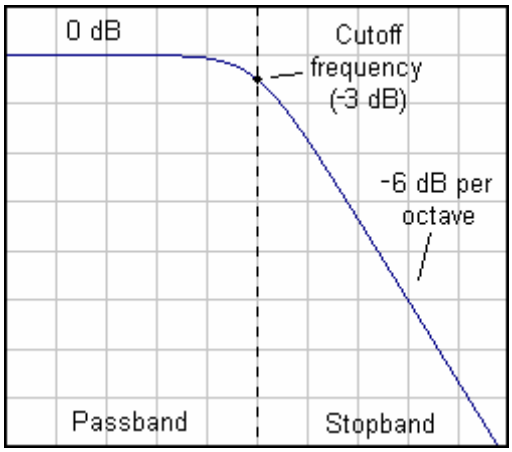

Figure 2.5 A logarithmic Bode plot for Butterworth filter

The two parameters completely characterizing a Butterworth filter are therefore the 3-db cutoff frequency Ω_c and the order *N*. The order *N* is given by

$$N = \frac{1}{2}\frac{\log_{10}\left[(A^2-1)/\varepsilon^2\right]}{\log_{10}(\Omega_s/\Omega_p)} = \frac{\log_{10}(1/k_1)}{\log_{10}(1/k)} \tag{2.9}$$

where $k_1 = \varepsilon/\sqrt{A^2-1}$, $k = \Omega_p/\Omega_s$

The transfer function of the Butterworth lowpass filter is given by

$$H_a(s) = \frac{C}{D_N(s)} = \frac{\Omega_C^N}{s^N + \sum_{l=0}^{N-1} d_l s^l} = \frac{\Omega_C^N}{\prod_{l=1}^{N}(s-p_l)} \tag{2.10}$$

where $p_\ell = \Omega_c e^{j[\pi(N+2l-1)/2N]}$, $\ell = 1,2,...,N$

Figure 2.6 shows magnitude response of the normalized Butterworth lowpass filter with Ω_c = 1 for some typıcal values of N.

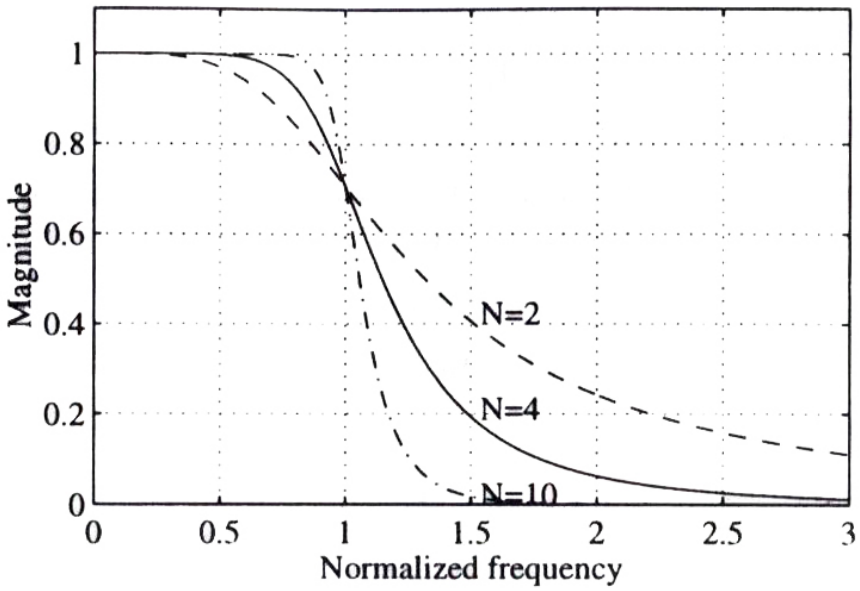

Figure 2.6 Typical butterworth lowpass filters responses

2.5 Chebyshev Filters

Chebyshev filters are used to separate one band of frequencies from another. They are more than adequate for many applications. The primary attribute of Chebyshev filters is their speed. These filters [2] are named from their use of the Chebyshev polynomials, developed by the Russian mathematician Pafnuti Chebyshev (1821-1894).

2.5.1 Chebyshev Approximation

In this case, the approximation error, defined as the difference between the ideal brickwall characteristic and the actual response, is minimized over a prescribed band of frequencies. In fact the magnitude error is equiripple in the band.

There are two types of Chebyshev transfer functions. In the type 1 Approximation, the magnitude characteristic is equiripple in the passband and monotonic in the stopband, as shown in figure 2.7 [1]

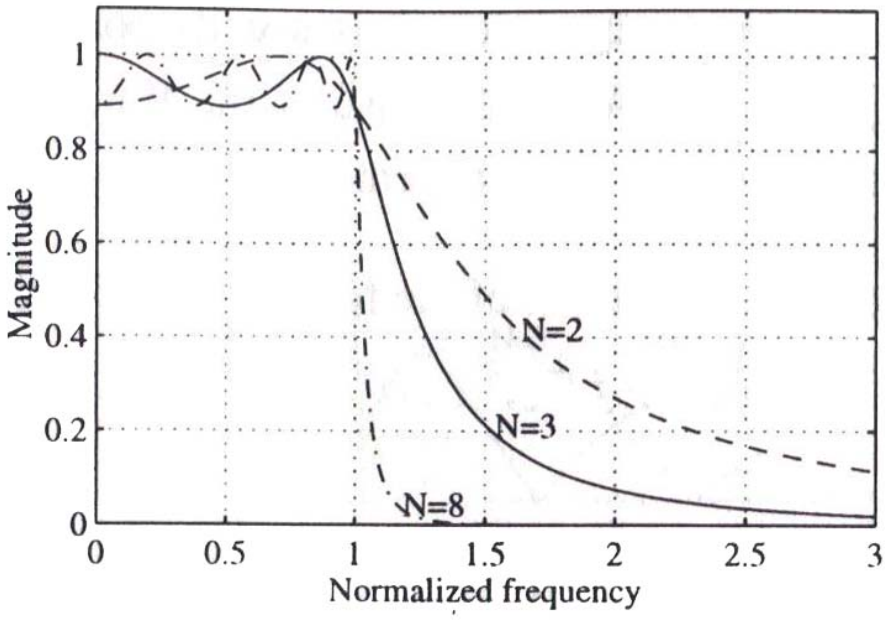

Figure 2.7 Typical Type 1 Chebyshev lowpass filter response

From the figure 2.7 it is seen that the squared-magnitude response is equiripple between $\Omega = 0$ and $\Omega = 1$, and it decreases monotonically for all $\Omega > 1$.

In the type 2 approximation, the magnitude response is monotonic in the passband and equiripple in the stopband, as shown in figure 2.8

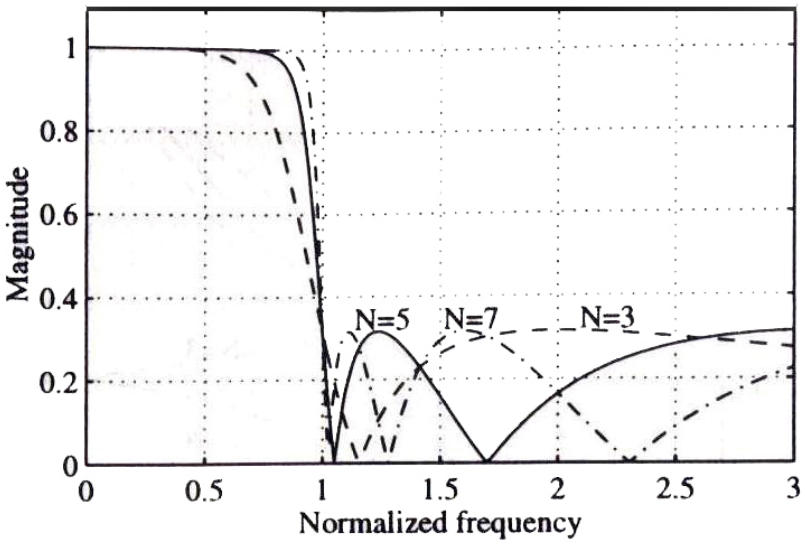

Figure 2.8 Typical Type 2 Chebyshev lowpass filter responses

The type1 Chebyshev transfer function H_a(s) has a magnitude response given by

$$|H_a(j\Omega_s)|^2 = \frac{1}{1+\varepsilon^2 T_N^2(\Omega/\Omega_p)}, \tag{2.11}$$

where $T_N(\Omega)$ is the Chebyshev polynomial of order N:

$$T_N(\Omega) = \begin{cases} \cos(N\cos^{-1}\Omega), & |\Omega| \le 1 \\ \cosh(N\cosh^{-1}\Omega), & |\Omega| > 1 \end{cases} \tag{2.12}$$

The order *N* is given by

$$N = \frac{\cosh^{-1}\left[(A^2-1)/\varepsilon\right]}{\cosh^{-1}(\Omega_s/\Omega_p)} = \frac{\cosh^{-1}(1/k_1)}{\cosh^{-1}(1/k)}, \tag{2.13}$$

where $k_1 = \varepsilon/\sqrt{A^2-1}$, $k = \Omega_p/\Omega_s$

The type 2 Chebyshev magnitude response, also known as the inverse Chebyshev response, exhibits a monotonic behavior in the passband with maximally flat response at $\Omega = 0$ and an equiripple behavior in the stopband. The square- magnitude response expression here is given by

$$|H_a(j\Omega)|^2 = \frac{1}{1+\varepsilon^2\left[\dfrac{T_N(\Omega_s/\Omega_p)}{T_N(\Omega_s/\Omega)}\right]^2} \tag{2.14}$$

For the type 2 Chebyshev the order *N* is also given by

$$N = \frac{\cosh^{-1}\left[(A^2-1)/\varepsilon\right]}{\cosh^{-1}(\Omega_s/\Omega_p)} = \frac{\cosh^{-1}(1/k_1)}{\cosh^{-1}(1/k)}, \tag{2.15}$$

where $k_1 = \varepsilon/\sqrt{A^2-1}$, $k = \Omega_p/\Omega_s$

2.6 Elliptic Approximation (Cauer filter)

An Elliptic filter, also known as a Cauer filter, has an equiripple passband and an equiripple stopband magnitude response, as shown in Figure 2.9 for typical elliptic lowpass filters. [1]

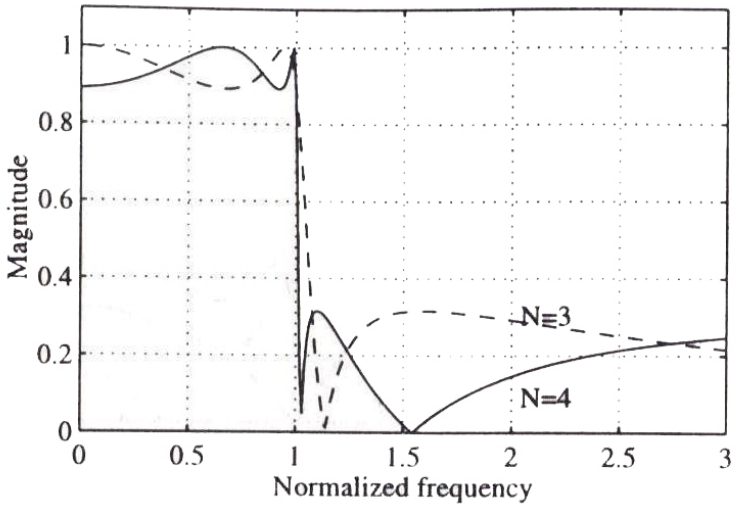

Figure 2.9 Typical elliptic lowpass filter responses

The transfer function of an elliptic filter meets a given set of filter specification, passband edge frequency Ω_p and stopband edge frequency Ω_s, passband ripple and a minimum stopband attenuation A, with the lowest filter order N. The square- magnitude response of an elliptic lowpass filter is given by

$$|H_a(j\Omega)|^2 = \frac{1}{1+\varepsilon^2 R_N^2(\Omega/\Omega_p)}, \tag{2.16}$$

where $R_N(\Omega)$ is rational function of order N that satisfies the property $R_N(1/\Omega) = 1/R_N(\Omega)$ with the roots of its numerator lying within the interval $0 < \Omega < 1$ and the roots of its denominator lying in the interval $1 < \Omega < \infty$.

For most applications, the filter order meeting a given set of specifications of passband edge frequency Ω_p, passband ripple ε, stopband edge frequency Ω_s, and the minimum stopband ripple A can be estimated using the formula :

$$N \cong \frac{\log_{10}(4/k_1)}{\log_{10}(1/\rho)}, \tag{2.17}$$

where $k_1 = \varepsilon / \sqrt{A^2 - 1}$,

$k = \Omega_p / \Omega_s$,

$k' = \sqrt{1 - k^2}$,

$$\rho_0 = \frac{1 - \sqrt{k'}}{2(1 + \sqrt{k'})},$$

$$\rho = \rho_0 + 2\rho_0^5 + 15\rho_0^9 + 150\rho_0^{13}$$

2.7 Bessel Filters

The term Bessel refers to a type of filter response, which features flat group delay in the passband as shown in figure 2.10. This is the characteristic of Bessel filters that makes them valuable to designers. [3]

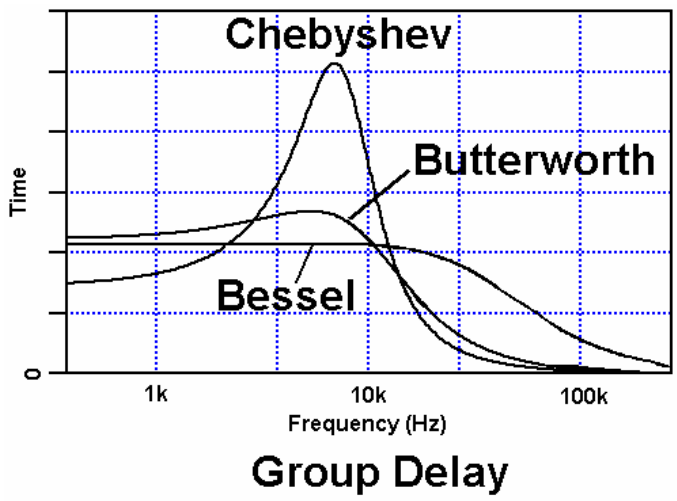

Figure 2.10 Group delay characteristic of Bessel filters

The Bessel approximation has a smooth passband and stopband response, like the Butterworth. For the same filter order, the stopband attenuation of the Bessel approximation is much lower than that of the Butterworth approximation as shown in figure 2.11

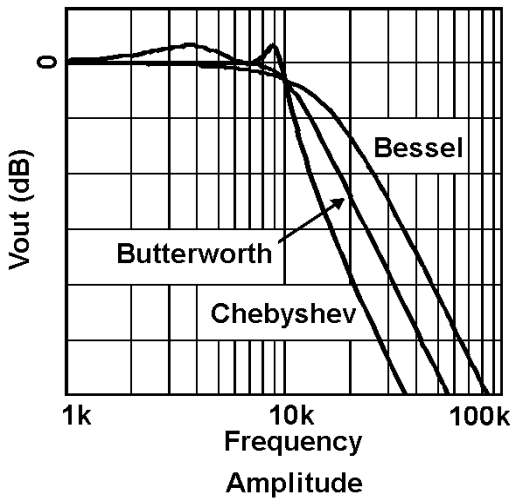

Figure 2.11 Typical elliptic lowpass filter responses

It can be seen that there is no ripple in the passband of a Bessel filter, and that it does not have as much rejection in the stop band as a Butterworth filter.The phase response of the three filter types is shown in figure 2.12. The Bessel response has the slowest rate of change of phase. [3]

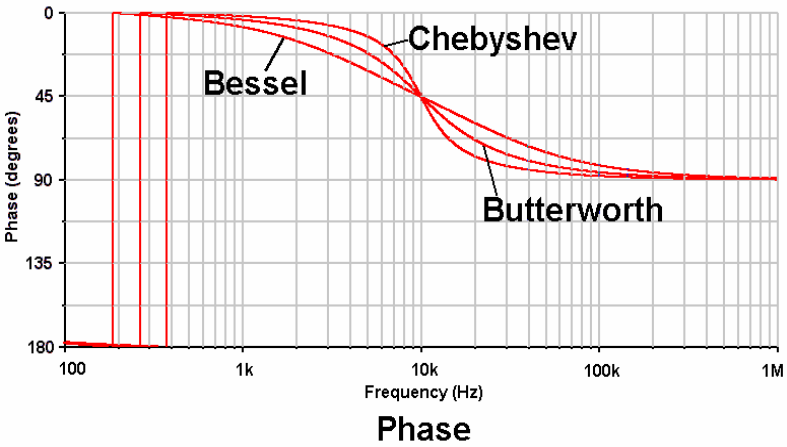

Figure 2.12 phase response of the three filter types

2.8 Filter Comparison

The following table gives a summary of Advantages and Disadvantages for the three filter types

Table 2-2 Advantages and Disadvantages for the three filter types [4]

Butterworth response	**Advantages:** Maximally flat magnitude response in the passband. Good all-around performance. Pulse response better than Chebyshev. Rate of attenuation better than Bessel.
	Disadvantages: Some overshoot and ringing in step response.
Chebyshev type 1 response	**Advantages:** Better rate of attenuation beyond the pass-band than Butterworth
	Disadvantages: Ripple in pass-band. Considerably more ringing in step response than Butterworth.
Chebyshev type 2 response	**Advantages:** Flat magnitude response in passband with steep rate of attenuation in transition-band.
	Disadvantages: Ripple in stopband. Some overshoot and ringing in step response
Bessel response	**Advantages:** Best step response-very little overshoot or ringing.
	Disadvantages: Slower initial rate of attenuation beyond the pass-band than Butterworth.

2.9 Digital Filters

Digital filters are used for two general purposes [5]: separation of signals that have been combined, and restoration of signals that have been distorted in some way. Analog (electronic) filters can be used for these same tasks; however, digital filters can achieve far superior results. Digital filters are a very important part of DSP. In fact, their extraordinary performance is one of the key reasons that DSP has become so popular.

2.9.1 Principal of digital filter

A digital filter takes a digital input, gives a digital output, and consists of digital components. In a typical digital filtering application, software running on a digital signal processor reads input samples from an A/D converter, performs the mathematical manipulations dictated by theory for the required filter type, and outputs the result via a D/A converter. An analog filter, by contrast, operates directly on the analog inputs and is built entirely with analog components, such as resistors, capacitors, and inductors.

The analog input signal must first be sampled and digitized using an A/D converter. The resulting binary numbers, representing successive sampled values of the input signal, are transferred to the processor, which carries out numerical calculations on them.

These calculations typically involve multiplying the input values by constants and adding the products together. If necessary, the results of these calculations, which now represent sampled values of the filtered signal, are output through a DAC (digital to analog converter) to convert the signal back to analog form. Note that in a digital filter, the signal is represented by a sequence of numbers, rather than a voltage or current.

The block diagram in figure 2.13 shows the basic setup of such a system

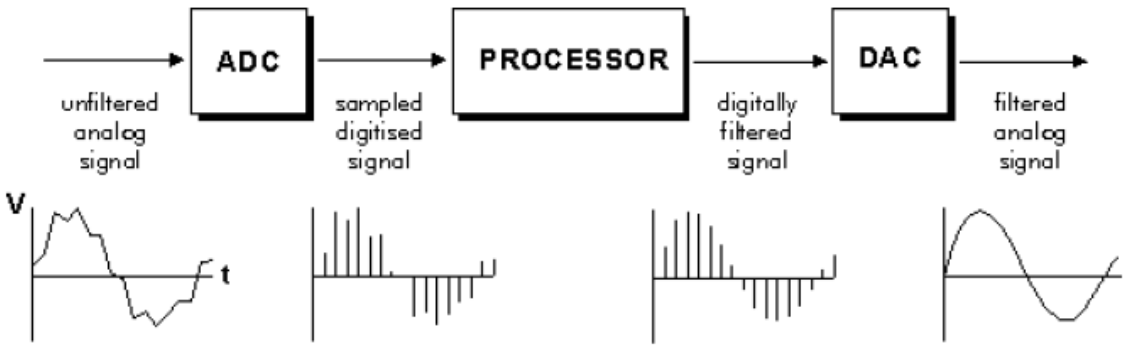

Figure 2.13 Principal of digital filter

2.9.2 Advantages of using digital filters

The following list gives some of the main advantages of digital filters over analog filters. [5]

1. A digital filter is *programmable*, i.e. its operation is determined by a program stored in the processor's memory. This means the digital filter can easily be changed without affecting the circuitry (hardware). An analog filter can only be changed by redesigning the filter circuit.
2. Digital filters are easily *designed*, *tested* and *implemented* on a general-purpose computer or workstation.
3. The characteristics of analog filter circuits (particularly those containing active components) are subject to drift and are dependent on temperature. Digital filters do not suffer from these problems, and so are extremely *stable* with respect both to time and temperature.
4. Unlike their analog counterparts, digital filters can handle *low frequency* signals accurately. As the speed of DSP technology continues to increase, digital filters are being applied to high frequency signals in the RF (radio frequency) domain, which in the past was the exclusive preserve of analog technology.
5. Digital filters are very much more *versatile* in their ability to process signals in a variety of ways; this includes the ability of some types of digital filter to adapt to changes in the characteristics of the signal.

6. Fast DSP processors can handle complex combinations of filters in parallel or cascade (series), making the hardware requirements relatively *simple* and *compact* in comparison with the equivalent analog circuitry.

2.9.3 Types of digital filters

A digital filter, in its most general form, takes in an input sequence of numbers x[n], performs computations on these numbers and outputs results of these computations as another sequence of numbers y[n]. Generally, y[n] is computed as the sum of weighed present and previous input samples and previous output samples, as shown in Equation 2.17. [13]

$$\begin{aligned} y[n] = a_0 x[n] + a_1 x[n-1] + ... + a_M x[n-M] + \\ b_1 y[n-1] + b_2 y[n-2] + ... + b_N y[n-N] \end{aligned} \tag{2.17}$$

where a_0, a_1, ... a_M and b_1, b_2, ... b_N are constants and referred to as filter coefficients. M+1 and N are the number of input and output samples used for computation. [13]

2.9.4 FIR and IIR filter

There are two basic types of digital filters Infinite Impulse Response (IIR) filter and the Finite Impulse Response (FIR) filter. As indicated in Equation 2.17, if b_1 through b_N are all zeros, then y[n] does not depend on the previous output samples (i.e., there is no feedback). In this case, this type of filter is termed as a Finite Impulse Response (FIR) filter.

Since there is no feedback term if the input sequence stops (i.e., x[n]'s become zeros), then y[n]'s also will become zeros after some delay. If any one of the coefficients b_1 through b_N are non-zero, the filter is called an Infinite Impulse Response (IIR) filter. For the FIR filter, the sequence of coefficients a_0, a_1, .. a_M also represent the response of the filter for a unit impulse (also called an impulse response).

The design of the IIR filters is similar to that of an analog filter whereas the design of the FIR filters is unique to digital filtering. The order of a FIR to meet the desired filter specifications is much greater then that of an IIR filter. This may be so, but FIR filters possess characteristics unknown to IIR filters. The most important of these are linear phase and constant group delay. This makes FIR filters a necessity in applications which demand little phase distortion. [8]

The advantages of FIR filters are:

- They can be designed to have linear phase response with respect to frequency, whereas IIR filters do not have linear phase response.
- They are always stable, unlike IIR filters.

The disadvantages of FIR filters over IIR filters are:

- FIR filters take relatively more memory and computation time.
- FIR filters cannot give sharper cut-off than IIR filters for the same number of filter coefficients.

2.10 Summary

This chapter discussed analog filter definitions, types and specifications. Filter responses were discussed and compared. An introduction to digital filtering and digital filter types was given.

CHAPTER 3

DIGITAL FILTER STRUCTURES AND DESIGN

3.1 Overview

After the description of digital filters have been introduced in the last chapter, the Finite Impulse Response (FIR) filters and the Infinite Impulse Response (IIR) filters will be discussed in this chapter. It will include the block diagram, realization, specifications, and basic approaches to filter design.

3.2 Digital Filter Structure

The actual implementation of a digital filter could be either in software or hardware form, depending on applications. A structural representation using interconnected basic building blocks is the first step in the hardware or the software implementation of a digital filter. The structural representation provides the relations between some pertinent internal variables with the input and the output that in turn provide the keys to the implementation. [1]

3.2.1 Block Diagram Representation

The I/O relations of a digital filter can be expressed in the time domain by the convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{3.1}$$

or by the linear constant coefficient difference equation

$$y[n] = -\sum_{k=1}^{N} d_k y[n-k] + \sum_{k=0}^{M} p_k x[n-k] \tag{3.2}$$

or by the state equations

$$\mathrm{s[n+1]} = \mathrm{As[n]} + \mathrm{Bx[n]} \tag{3.3a}$$

$$\mathrm{y[n]} = \mathrm{Cs[n]} + \mathrm{Dx[n]} \tag{3.3b}$$

An *N* point FIR digital filter is given by the following transfer function

$$H(z) = \sum_{k=0}^{N-1} h(k) z^{-k} \tag{3.4}$$

For a given input sequence X(z), the output from the filter Y(z) is

$$Y(z) = H(z)X(z) = \sum_{k=0}^{N-1} h(k) z^{-k} X(z) \tag{3.5}$$

A digital filter can be implemented on a general-purpose digital computer in software form or with special-purpose hardware. To this end, it is necessary to describe the I/O relationship by means of a computational algorithm.

3.2.1.1. Basic Building Blocks

The computational algorithm of an LTI digital filter can be conveniently represented in block diagram form using the basic building blocks representing the unit delay, the multiplier, and the adder as shown in Figure 3.1. In addition, this figure shows a pick-off node that splits a single incoming signal into multiple identical outgoing signals. Also the symbol of delay is represented by z^{-1}.

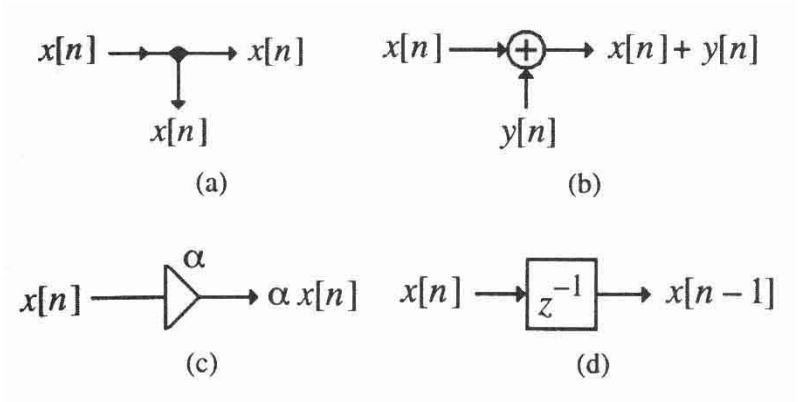

Figure 3.1 (a) pick-off node, (b) adder, (c) multiplier and (d) unit delay

3.2.2 Analysis of Block Diagram

Digital filter structures represented in block diagram form can often be analyzed by writing down the expressions for the output signal of each adder as a sum of its inputs, developing a set of equations relating the filter input and output signals in terms of all internal signals. Eliminating the unknown internal variables then results in the

expression for the output signal as a function of the input signal and the filter parameters that are the multiplier coefficients.

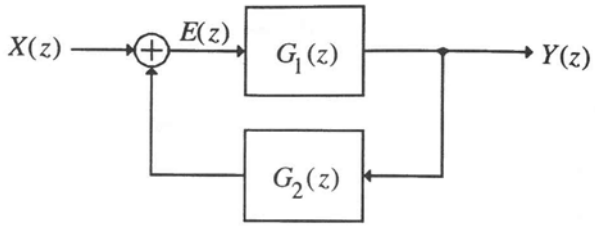

Figure 3.2 Single loop digital filter structure

3.2.2.1. The Delay-Free Loop Problem

For physical realizability of the digital filter structure, it is necessary that the block diagram representation contains no delay-free loop, i.e., feedback loops without any delay element. Illustration of a typical delay-free loop that appears unintentionally in a specific structure is shown in Figure 3.3. Analysis of this structure yields

$$y[n] = B\{A(w[n] + y[n]) + v[n]\}$$

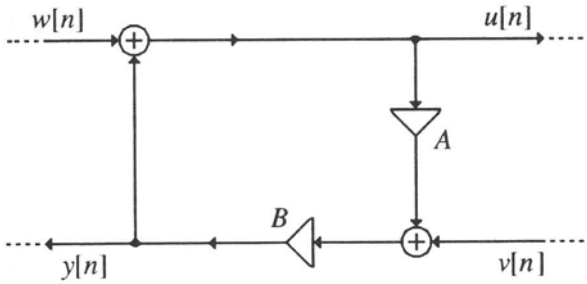

Figure 3.3 An example of a delay free loop

As noticed the determination of the current value of $y[n]$ requires the knowledge of the same value which is physically impossible.

A simple graph-theoretic-based method has been proposed to detect the presence of delay-free loops in an arbitrary digital filter structure, along with the methods to locate and remove these loops without altering the overall input-output relations. The

removal is achieved by replacing the portion of the structure containing the delay-free loops by an equivalent realization with no delay-free loops as shown in figure 3.4

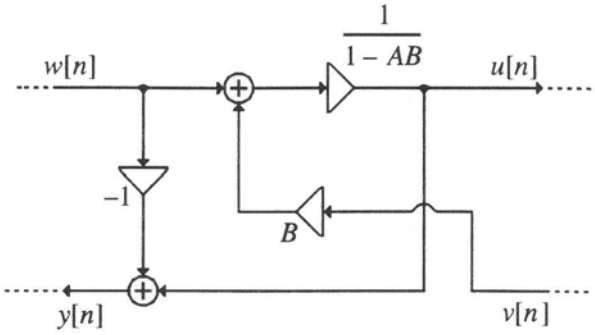

Figure 3.4 Realization of Figure 3.3 with no delay-free loop

3.2.2.2. Canonic and Noncanonic Structures

A digital filter structure is said to be *canonic* if the number of delays in the block diagram representation is equal to the order of the difference equation (i.e., the order of the transfer function). Otherwise, it is a *noncanonic* structure.

3.3 Basic FIR Digital Filter Structures

A causal FIR filter of order N is characterized by a transfer function,

$$H(z) = \sum_{k=0}^{N-1} h[k] z^{-k} \tag{3.6}$$

The I/O relation of the FIR filter is given by

$$y[n] = \sum_{k=0}^{N-1} h[k] x[n-k] \tag{3.7}$$

where y[n] and x[n] are the output and input sequences, respectively. Several methods of realization are outlined below.

3.3.1 Direct Form

An FIR filter of order N is characterized by N coefficients and requires N multipliers and N *-1* two-input adders for implementation. Structures in which the multiplier coefficients are precisely the coefficients of the transfer function are called *direct form* structures. A direct form realization on an FIR filter can be readily developed from Eq. (3.7). Figure 3.5 is an indication of this structure in two ways for N = 5, where:

$$y[n] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + h[3]x[n-3] + h[4]x[n-4]$$

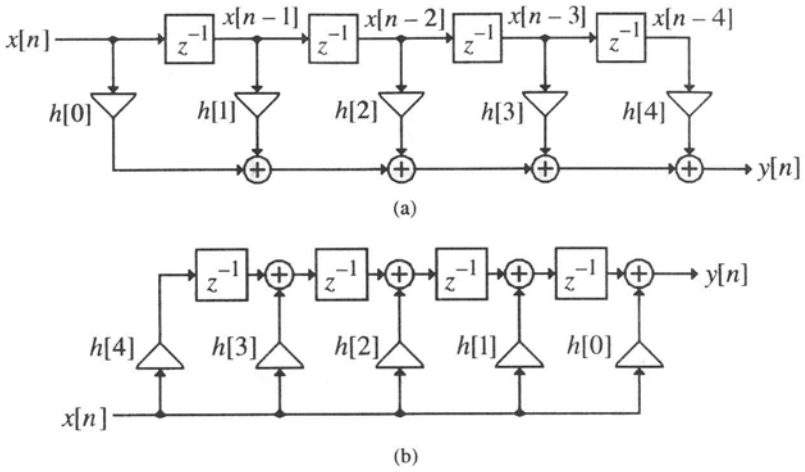

Figure 3.5 Direct form for FIR structure

3.3.2 Cascade Form

A higher-order FIR transfer function can also be realized as a cascade if FIR sections with each section characterized by either a first-order or a second-order transfer function. For example, Eq (3.6) can be factorized and be written in the form:

$$H(z) = h[0]\prod_{k=1}^{K}(1 + \beta_{1k}z^{-1} + \beta_{2k}z^{-2}) \tag{3.8}$$

where K=N/2 if N is even, and K=(N-1)/2 if N is odd, with $\beta_{2k} = 0$. A cascade realization of Eq. (3.8) for N=6 is shown in Figure 3.6 requiring three second-order sections. Note that the structure of Figure 3.6 is canonic form.

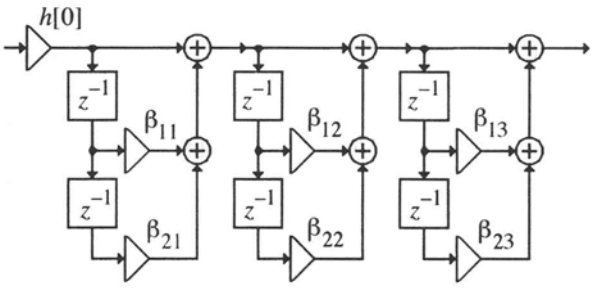

Figure 3.6 Cascade form FIR filter structure for a sixth order filter

3.3.3 Linear-Phase FIR Structure

Consider the realization of a length-7 Type 1 FIR transfer function with a symmetric impulse response**:**

$$\begin{aligned} H(z) &= h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[2]z^{-4} + h[1]z^{-5} + h[0]z^{-6} \\ &= h[0]\left(1 + z^{-6}\right) + h[1]\left(z^{-1} + z^{-5}\right) + h[2]\left(z^{-2} + z^{-4}\right) + h[3]z^{-3} \end{aligned} \tag{3.9}$$

A similar decomposition can be applied for the realization of a length- 8 Type 2 FIR transfer function as follows:

$$H(z) = h[0]\left(1 + z^{-7}\right) + h[1]\left(z^{-1} + z^{-6}\right) + h[2]\left(z^{-2} + z^{-5}\right) + h[3]z^{-3}\left(z^{-3} + z^{-4}\right) \tag{3.10}$$

Figure 3.7(a,b) shows the two types realization. [1]

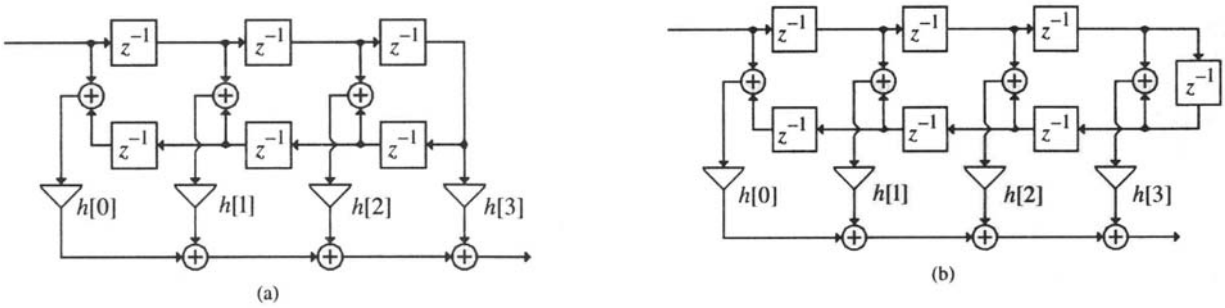

Figure 3.7 Linear-phase FIR structures: (a) Type 1, and (b) Type 2

3.4 Digital Filter Specifications

The magnitude response specification of a digital filter in the *passband* and in the *stopband* is given with some acceptable tolerance. In addition, a *transition band* is specified between the passband and the stopband to permit the magnitude to drop off smoothly. Figure 3.8 shows the specification for a lowpass filter**.**

In the passband:

$$1-\delta_p \le \left|G(e^{j\omega})\right| \le 1+\delta_p, \text{ for } |\omega| \le \omega_p \tag{3.11}$$

In the stopband:

$$|G(e^{j\omega})| \le \delta_s, \text{ for } \omega_s \le |\omega| \le \pi \tag{3.12}$$

Where ω_p and ω_s are respectively the *passband edge frequency* and *stopband edge frequency*. The limits of the tolerance in the passband and stopband, δ_p and δ_s, are usually called the *peak ripple values*.

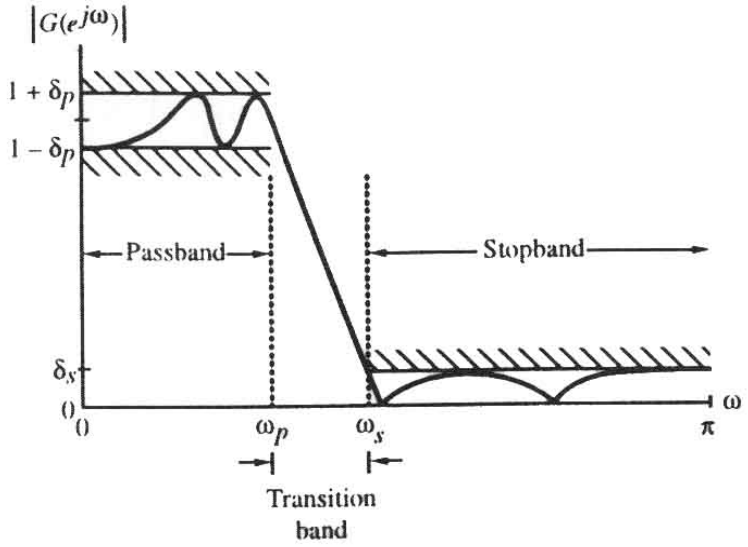

Figure 3.8 Typical magnitude specifications of digital filter

Digital filter specifications are often given in terms of the loss function,

$$A(\omega) = -20\log_{10}|G(e^{j\omega})|, \text{ in dB.}$$

Here the peak passband ripple and the minimum stopband attenuation are given in dB. i.e., the loss specifications of a digital filter are given by

$$\alpha_p = -20\log_{10}(1-\delta_p) \text{ dB,} \tag{3.13}$$

$$\alpha_s = -20\log_{10}(\delta_s) \text{ dB} \tag{3.14}$$

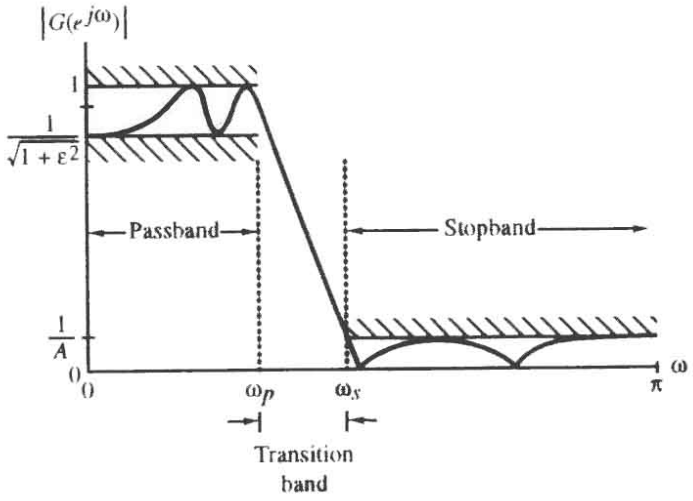

Figure 3.9 Alternative magnitude specifications for a digital filter

The *maximum passband attenuation* $\alpha_{\max} = -20\log_{10}(\sqrt{1+\varepsilon^2})$ dB (3.15)

For $\delta_p << 1$, as is typically the case, it can be shown that

$$\alpha_{\max} = -20\log_{10}(1-2\delta_p) \cong 2\alpha_p \tag{3.16}$$

Let F_T denote the sampling frequency in Hz, and F_p and F_s denote, respectively, the passband and stopband edge frequency in Hz. Then the normalized angular edge frequencies in radian are given by **:**

$$\omega_p = \frac{\Omega_p}{F_T} = \frac{2\pi F_p}{F_T} = 2\pi F_p T \tag{3.17}$$

$$\omega_s = \frac{\Omega_s}{F_T} = \frac{2\pi F_s}{F_T} = 2\pi F_s T \tag{3.18}$$

3.5 Estimation of FIR Filter Order

For the design of FIR lowpass digital filters, several formulas had been introduced by researchers, like Kaiser and Park [10], [11], to estimate the minimum value of the filter length N directly from the filter's specifications. A rather simple approximation formula developed by Kaiser is

Kaiser approximation:

$$N \cong \frac{-20\log_{10}(\sqrt{\delta_p \delta_s}) - 13}{14.6(\omega_s - \omega_p)/2\pi} \quad \text{(Moderate passband)} \tag{3.19}$$

Park approximation:

$$N \cong \frac{-20\log_{10}(\delta_s) + 0.22}{(\omega_s - \omega_p)/2\pi} \quad \text{(Narrow passband)} \tag{3.20}$$

$$N \cong \frac{-20\log_{10}(\delta_p) + 5.94}{27(\omega_s - \omega_p)/2\pi} \quad \text{(Wide passband)} \tag{3.21}$$

Note that in the above formulas the filter length of the FIR filter is inversely proportional to the transition band width and does not depend on the actual location of the transition band. This implies that a sharp cutoff FIR filter with narrow transition band would be of very long length, while an FIR filter with a wide transition band will have a very short length.

3.6 Design of FIR Filters

Two methods for designing FIR filters will be discussed in this section. The direct and straightforward method is based on truncating the Fourier series representation of the prescribed frequency response. The other method is based on the observation that, for a length-N FIR filter, N distinct equally spaced frequency samples of its frequency response constitute the N-point DFT of its impulse response, hence, the impulse response sequence can be readily computed by applying an inverse DFT of these frequency samples.

3.6.1 FIR Filter Design Based on Truncated Fourier series

The Fourier coefficients $\{h_d[n]\}$ are impulse response samples of the desired frequency response function $H_d[e^{j\omega}]$, where

$$H_d[e^{j\omega}] = \sum_{n=-\infty}^{\infty} h_d[n] e^{-j\omega n} \tag{3.22}$$

and it is given by:

$$h_d[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} H_d(e^{j\omega}) e^{j\omega n} d\omega, \qquad -\infty \le n \le \infty \tag{3.23}$$

Because the corresponding impulse response sequence $\{h_d[n]\}$ is of infinite length and noncausal, it is objective to find a finite duration impulse response sequence $\{h_d[n]\}$ of length 2N+1 whose DTFT $Ht[e^{j\omega}]$ approximate the desired DTFT $H_d[e^{j\omega}]$ in some sense.

3.6.1.1. Impulse Response of Ideal Filter

For an ideal lowpass filter has a zero-phase frequency response

$$H_{LP}(e^{j\omega}) = \begin{cases} 1 & |\omega| \le \omega_c \\ 0 & \omega_c \le \omega \le \pi \end{cases} \tag{3.24}$$

The corresponding impulse response coefficients are given by:

$$h_{LP}[n] = \frac{\sin \omega_c n}{\pi n}, \qquad -\infty \le n \le \infty \tag{3.25}$$

As the impulse response is doubly infinite, the coefficients outside the range –M ≤ n ≤ M is setting equals to zero. The new length of the LPF N = 2M + 1. The new coefficients when shifting to the right will be:

$$\widehat{h}_{LP}[n] = \begin{cases} \dfrac{\sin(\omega_c(n-M))}{\pi(n-M)}, & 0 \le n \le N-1 \\ 0, & \text{othewise} \end{cases} \tag{3.26}$$

Likewise, the impulse response coefficients $h_{HP}[n]$ of the ideal highpass filter are given by:

$$h_{HP}[n] = \begin{cases} 1 - \dfrac{\omega_c}{\pi} & for \;\; n = 0 \\ -\dfrac{\sin(\omega_c n)}{\pi n} & for \;\; |n| > 0 \end{cases} \tag{3.27}$$

Also, the impulse response coefficients $h_{BP}[n]$ of the ideal bandpass filter with cutoffs at ω_{c1} and ω_{c2} are given by:

$$h_{BP}[n] = \frac{\sin(\omega_{c2} n)}{\pi n} - \frac{\sin(\omega_{c1} n)}{\pi n}, \qquad |n| \geq 0 \tag{3.28}$$

And that of an ideal bandstop filter with cutoffs at ω_{c1} and ω_{c2} are given by:

$$h_{BS}[n] = \begin{cases} 1 - \dfrac{(\omega_{c2} - \omega_{c1})}{\pi} & for \;\; n = 0 \\ \dfrac{\sin(\omega_{c1} n)}{\pi n} - \dfrac{\sin(\omega_{c2} n)}{\pi n} & for \;\; |n| > 0 \end{cases} \tag{3.29}$$

3.6.1.2. Gibbs Phenomenon

Since the Fourier series represents a continuous time signal as a linear combination of continuous function, therefore it should be expected that the Fourier series is well suited for modeling smooth signals. Recall that the Fourier series is the infinite sum of weighted complex exponentials given by:

$$x(t) = \sum_{k=-\infty}^{\infty} a_k \exp(jk\omega_0 t) \tag{3.30}$$

In practice, it may be more partial to consider using only a finite sum to approximate $x(t)$. Using *2N+1* coefficients, the reconstruction approximation of $x(t)$ shall be defined to be:

$$X_N(t) = \sum_{k=-N}^{N} a_k \exp(jk\omega_0 t) \tag{3.31}$$

Any difference between $X_N(t)$ and $x(t)$ is attribute to the use of a finite number of terms (harmonics) to reconstruct $x(t)$. Defining the approximation error to be:

$$e_N(t) = x(t) - X_N(t) = x(t) - \sum_{k=-N}^{N} a_k \exp(jk\omega_0 t) \tag{3.32}$$

The number of harmonics required to produce an error that does not exceed a given mean error is signal-dependant.

Figure 3.10 shows the convergence of the Fourier series representation of a square wave. The finite series approximation of a square wave by different values of N has been produced. The behavior of partial sum in the vicinity of discontinuity exhibits ripples. As N increases, the ripple in the partial sum becomes compressed toward the discontinuity, but any finite values of N, the peak amplitude of these ripples remain constant. This behavior is called the Gibbs phenomenon. [3]

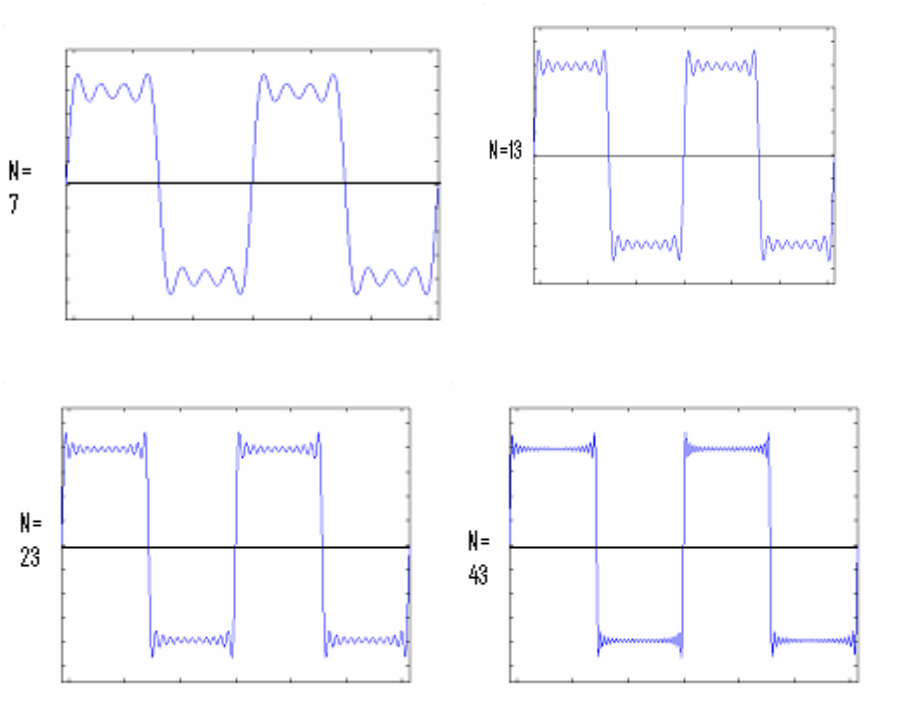

Figure 3.10 Convergence of the Fourier series representation of square wave

Now, if an ideal filter is considered the impulse response of it will be infinite length. To achieve simple truncation of ideal infinite length impulse response a window should be used. First, a rectangular window will be used, and it is given by:

$$w_R[n] = \begin{cases} 1 & 0 \le |n| \le M \\ 0, & otherwise \end{cases} \tag{3.33}$$

The presence of the oscillatory behavior in the Fourier transform of a truncated Fourier series representation of an ideal filter is basically due two reasons. First, the impulse response of an ideal filter is infinitely long and not absolutely summable, and as a result, the filter is unstable. Second, the rectangular window has an abrupt transition to

zero. The oscillatory behavior can be explained by examining the Fourier transform $\psi_R(e^{j\omega n})$ of the rectangular window function in Eq. (3.33):

$$\psi_R(e^{j\omega n}) = \sum_{n=-M}^{M} e^{-j\omega n} = \frac{\sin([2M+1]\omega/2)}{\sin(\omega/2)} \tag{3.34}$$

A plot of the above is shown in Figure 3.11 for M = 4 and 10. The frequency response $\psi_R(e^{j\omega n})$ has a narrow main lobe centered at ω = 0. As M increases, the width of the main lobe decreases. This implies that with increasing M, ripples in $H_t(e^{j\omega n})$ around the point of discontinuity occur more closely but with no decrease in amplitude.

The rectangular window has an abrupt transition to zero outside the range –M ≤ n ≤ M, which is the reason behind the Gibbs phenomenon in the magnitude response of the window ideal filter impulse response sequence. It can be reduced by either using a window that tapers smoothly to zero at each end or by providing a smooth transition from the passband to the stopband. [1]

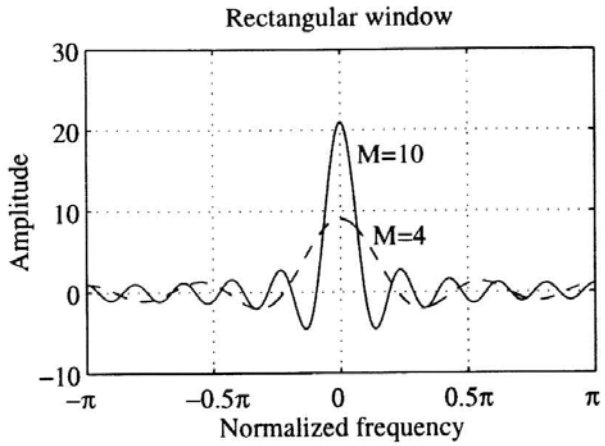

Figure 3.11 Frequency response of rectangular window

3.6.2 Fixed and Adjustable Window Functions

There are many proposed tapered windows. In this research three fixed tapered windows which are Hanning window, Hamming window, and Blackman window will be introduced. [12]

Hanning:

$$w[n] = \frac{1}{2}\left[1 + \cos\left(\frac{2\pi n}{2M+1}\right)\right], \qquad -M \le n \le M \tag{3.35}$$

Hamming:

$$w[n] = 0.54 + 0.46\cos\left(\frac{2\pi n}{2M+1}\right), \qquad -M \le n \le M \tag{3.36}$$

Blackman:

$$w[n] = 0.42 + 0.5\cos\left(\frac{2\pi n}{2M+1}\right) + 0.08\cos\left(\frac{4\pi n}{2M+1}\right), \quad -M \le n \le M \tag{3.37}$$

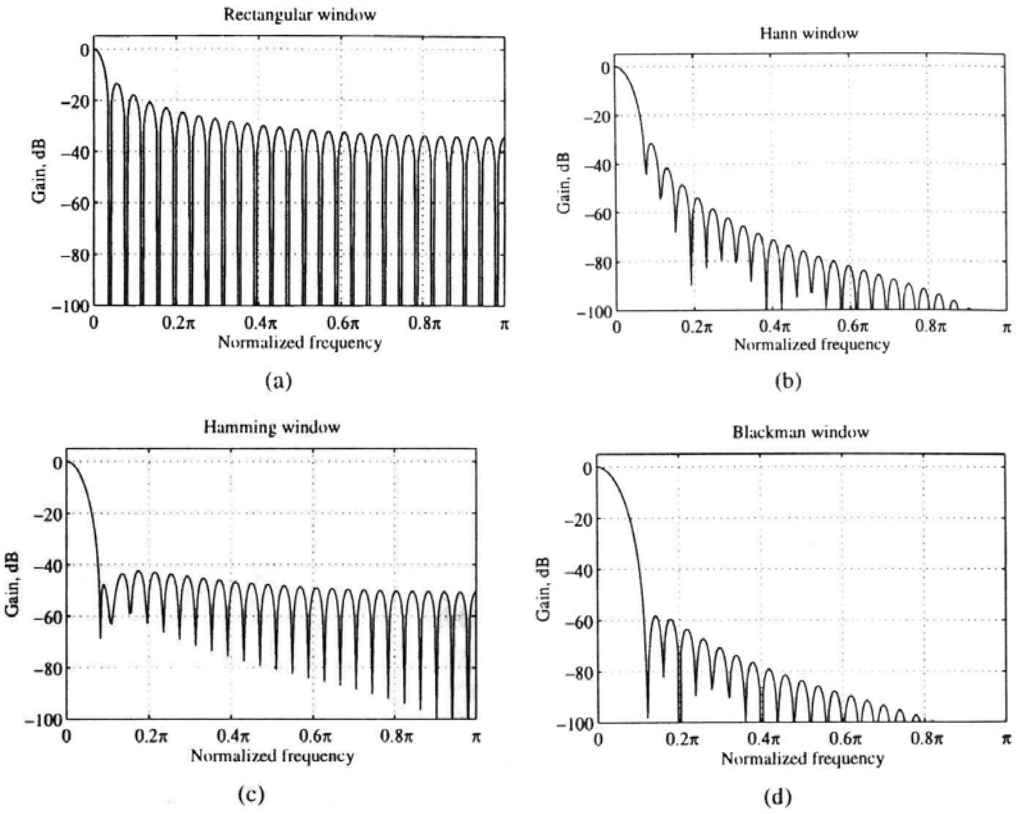

Figure 3.12 Gain response of the fixed window functions

A plot of the magnitude of the Fourier transform of each of the above windows in the dB scale is shown in Figure 3.12 for *M = 25*. Table 3.1 summarizes the essential properties of the above windows. In the case of these windows the value of the ripple δ does not depend on the filter length, and the cutoff frequency ω_c, and is essentially constant. Also, the transition bandwidth is approximately given by:

$$\Delta\omega \approx \frac{c}{M} \tag{3.38}$$

where *c* is a constant for most practical purposes.

Table 3.1 Properties of Some Fixed Windows [14]

Name of window function	Transition width/ sample frequency C	Passband ripple (dB)	Main lobe relative to side lobe (dB)	Maximum stopband attenuation (dB)
Rectangular	0.9 / N	0.75	13	21
Hanning	3.1/N	0.055	31	44
Hamming	3.3/N	0.019	41	53
Blackman	5.5/N	0.0017	57	74
Kaiser (β=4.54)	2.93/N	0.0274		50
Kaiser (β=8.96)	5.71/N	0.000275		90

For designing an FIR filter using one of the above windows, first the cutoff frequency ω_c is determined from the specified passband and stopband edges, ω_p and ω_s, by setting:

$$\omega_c = (\omega_p + \omega_s) / 2$$

Next, *M* is estimated using Eq. (3.38) , where the value of the constant *c* is obtained from Table 3.1 for the window chosen. Figure 3.13 shows the gain response of a lowpass FIR filter using the fixed window functions Hanning, Hamming, and Blackman.

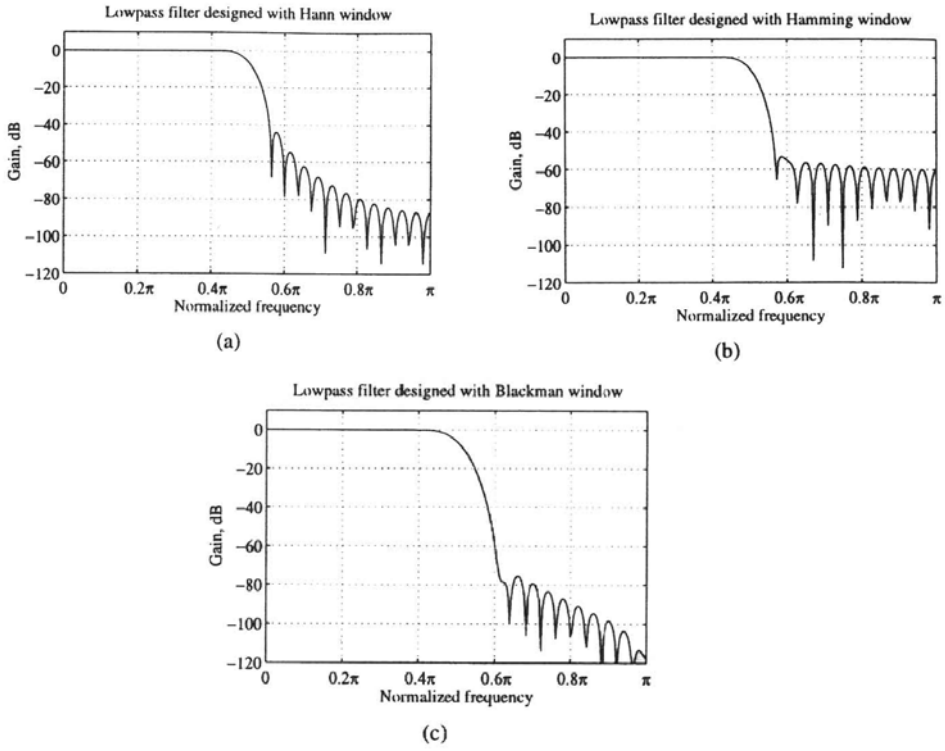

Figure 3.13 Gain response of LP FIR filter using fixed window functions

Also, an adjustable window which is the most widely used, Kaiser window, is given by:

$$w[n] = \frac{I_0\left\{\beta\sqrt{1-\left(n/M\right)^2}\right\}}{I_0(\beta)}, \qquad -M \le n \le M \tag{3.39}$$

where β is an adjustable parameter and $I_0(u)$ is the modified zeroth-order Bessel function, which can be expressed in a power series form:

$$I_0(u) = 1 + \sum_{r=1}^{\infty}\left[\frac{\left(u/2\right)^r}{r!}\right]^2 \tag{3.40}$$

The parameter β controls the minimum attenuation α_s i.e. the ripple δ_s, in the stopband of the windowed filter response. The parameter β can be computed from:

$$\beta = \begin{cases} 0.1102(\alpha_s - 8.7), & for\ \alpha_s > 50, \\ 0.5842(\alpha_s - 21)^{0.4} + 0.07886(\alpha_s - 21) & for\ 21 \le \alpha_s \le 50, \\ 0 & for\ \alpha_s < 21 \end{cases} \tag{3.41}$$

The filter length N can be estimated using the formula:

$$N \approx \begin{cases} \dfrac{\alpha_s - 7.95}{14.36\Delta f} + 1 & for \alpha_s > 21 \\ \dfrac{0.9222}{\Delta f} + 1 & for \alpha_s \le 21 \end{cases} \tag{3.42}$$

It should be noted that the Kaiser window provides no independent control over the passband ripple δ_p. In practice δ_p is approximately δ_s.

3.6.3 FIR Filter Design Based on Frequency Sampling Approach

In this approach, the specified frequency response $H_d[e^{j\omega}]$ is first uniformly sampled at N equally spaced points $\omega_k = 2\pi k / N$, $k = 0,1,.....,N-1$, providing N frequency samples. These frequency samples constitute an *N*-point DFT *H[k]* whose N-point inverse-DFT thus yields the impulse response coefficient *h*[*n*] of the FIR filter of length N. The basic assumption here is that the specified frequency response is uniquely characterized by the N frequency samples.

Now, $H_d[e^{j\omega}]$ is a periodic function of ω with a Fourier series representation given by Eq. 3.22. Its Fourier coefficients $h_d[n]$ are given by Eq. 3.23. It is instructive to develop the relation between $h_d[n]$ and *h*[*n*].

From Eq. 3.22,

$$H[k] = H_d(e^{j\omega k}) = H_d\left(e^{j(2\pi k/N)}\right) = \sum_{l=-\infty}^{\infty} h_d[l] W_N^{kl} \tag{3.43}$$

where $W_N = e^{-j(2\pi k/N)}$.

Taking the inverse-DTF of *H*[*k*] yields

$$h[n] = \frac{1}{N} \sum_{k=0}^{N-1} H[k] W_N^{-kn} \tag{3.44}$$

Then, substituting Eq. 3.43 in Eq. 3.44 yields:

$$h[n] = \frac{1}{N}\sum_{k=0}^{N-1}\sum_{l=-\infty}^{\infty} h_d[l] W_N^{kl} W_N^{-kn} = \sum_{l=-\infty}^{\infty} h_d[l]\left[\frac{1}{N}\sum_{k=0}^{N-1} W_N^{-k(n-l)}\right] \tag{3.45}$$

Using the Equation $\frac{1}{N}\sum_{k=0}^{N-1} W_N^{-k(n-r)} = \begin{cases} 1 & for\ r = n + mN \\ 0 & otherwise \end{cases}$

in Eq. 3.45 yields the desired relation:

$$h[n] = \sum_{m=-\infty}^{\infty} h_d(n + mN) \qquad 0 \le n \le N-1 \tag{3.46}$$

The above relation indicates that $h[n]$ is obtained from $h_d[n]$ by adding an infinite number of shifted replicas of $h_d[n]$ to $h_d[n]$, with each replica shifted by an integer multiple of N sampling instants, and observing the sum only for the interval $0 \le n \le N-1$. Thus, if $h_d[n]$ is a finite-length sequence of length less than or equal to N, then $h[n] = h_d[n]$ for $0 \le n \le N-1$, otherwise there is a time-domain aliasing of samples bearing no resemblance to $h_d[n]$.

Now, the expression of the transfer function H(z) of the FIR filter designed using the frequency sampling approach will be introduced.

$$H(z) = \sum_{n=0}^{N-1} h[n] z^{-n} \tag{3.47}$$

Substituting Eq. 3.44 in Eq. 3.47 yields:

$$H(z) = \sum_{n=0}^{N-1} \frac{1}{N} \sum_{k=0}^{N-1} H[k] W_N^{-nk} z^{-n} \tag{3.48}$$

Using algebra results:

$$H(z) = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H[k]}{1 - W_N^{-k} z^{-1}} \tag{3.49}$$

3.7 Basic IIR Digital Filter Structures

The causal IIR digital filters are characterized by a real rational transfer function of the form

$$H(z) = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + \cdots + p_M z^{-M}}{d_0 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_N z^{-N}} \tag{3.50}$$

The computation of the *n*th output sample requires the knowledge of several past samples of the output sequences, or in other words, the realization of a causal IIR filter requires some form of feedback. Some realization forms are followed. [1]

3.7.1 Direct Form

An *N*th-order IIR digital filter transfer function is characterized by 2*N*+1 unique coefficients and requires 2*N*+1 multipliers and 2*N* two-input adders for implementation. Consider a third-order IIR filter characterized by

$$H(z) = \frac{P(z)}{D(z)} = \frac{p_0 + p_1 z^{-1} + p_2 z^{-2} + p_3 z^{-3}}{1 + d_1 z^{-1} + d_2 z^{-2} + d_3 z^{-3}} \tag{3.51}$$

The structure of the direct form realization of the transfer function *H(z)* in Eq. (3.51) is shown in figure 3.14.

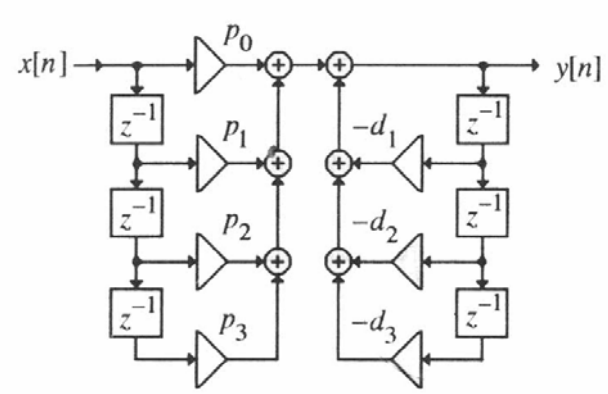

Figure 3.14 Direct form realization of a third-order IIR filter

3.7.2 Cascade Realizations

An IIR digital filter is often realized as a cascade of low-order filter sections as follows:

$$H(z) = \frac{P(z)}{D(z)} = \frac{P_1(z)P_2(z)P_3(z)}{D_1(z)D_2(z)D_3(z)} \tag{3.52}$$

Various different cascade realization of $H(z)$ can be obtained by different pole-zero polynomial pairings. Some examples of such realization are shown in figure 3.15.

Usually, the polynomials are factored into a product of first-order and second-order polynomials. In this case, $H(z)$ is expressed as

$$H(z) = p_0 \prod_k \left(\frac{1 + \beta_{1k} z^{-1} + \beta_{2k} z^{-2}}{1 + \alpha_{1k} z^{-1} + \alpha_{2k} z^{-2}} \right), \tag{3.53}$$

where for a first order factor, $\alpha_{2k} = \beta_{2k} = 0$.

Figure 3.16 shows a possible realization of a third-order transfer function

$$H(z) = p_0 \left(\frac{1 + \beta_{11} z^{-1}}{1 + \alpha_{11} z^{-1}} \right) \left(\frac{1 + \beta_{12} z^{-1} + \beta_{22} z^{-2}}{1 + \alpha_{12} z^{-1} + \alpha_{22} z^{-2}} \right)$$

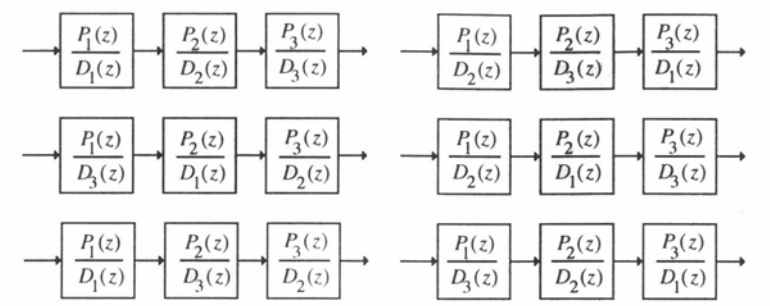

Figure 3.15 Different equivalent cascade IIR filter realizations

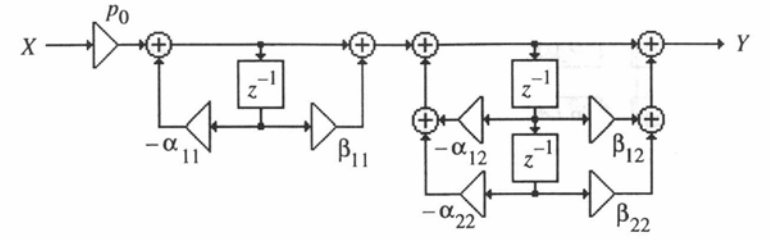

Figure 3.16 Cascade realization of a third-order IIR filter

3.8 Design of IIR Filters

Just as in the design of FIR filter, there are several methods that can be used to design IIR digital filters. IIR filters are usually designed by converting a prototype analog filter into a digital filter. Some methods of generating a digital filter from the analogue prototype are: [9]

- Impulse invariant design
- Step invariant design
- Design by Approximation of Derivatives
- Design by Bilinear transformation

The most useful method in practice is the bilinear transformation, which will be discussed in the following.

3.8.1 IIR Filter Design by Bilinear transformation

The bilinear transformation is a mathematical mapping of variables. In digital filtering, it is a standard method of mapping the s or analog plane into the z or digital plane. It transforms analog filters, designed using classical filter design techniques, into their discrete equivalents. [16]

The bilinear transformation from the s-plane into the z-plane is given by

$$s = \frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) \tag{3.54}$$

The bilinear transformation maps the s-plane into the z-plane by

$$G(z) = H_a(s)\Big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)} \tag{3.55}$$

For $T = 2$, the inverse transformation is given by $z = \dfrac{1+s}{1-s}$ (3.56)

For $s = j\Omega_0$ is $z = \dfrac{1+j\Omega_0}{1-j\Omega_0}$ which has a unity magnitude. This implies that a point on the imaginary axis in the s-plane is mapped onto a point on the unit circle in the z-plane.

In general for $s = \sigma_0 + j\Omega_0$

$$z = \frac{1 + (\sigma_0 + j\Omega_0)}{1 - (\sigma_0 + j\Omega_0)} = \frac{(1 + \sigma_0) + j\Omega_0}{(1 - \sigma_0) - j\Omega_0} \tag{3.57}$$

Therefore $$|z|^2 = \frac{(1 + \sigma_0)^2 + {\Omega_0}^2}{(1 - \sigma_0)^2 + {\Omega_0}^2}$$

Thus a point in the left-half s-plane with $\sigma_0 < 0$ is mapped to a point inside the unit circle in the z-plane $|z| < 1$. Likewise a point in the right-half s-plane with $\sigma_0 > 0$ is mapped to a point outside the unit circle in the z-plane $|z| > 1$. Any point in the s-plane is mapped onto a unique point in the z-plane as shown in Figure 3.17.

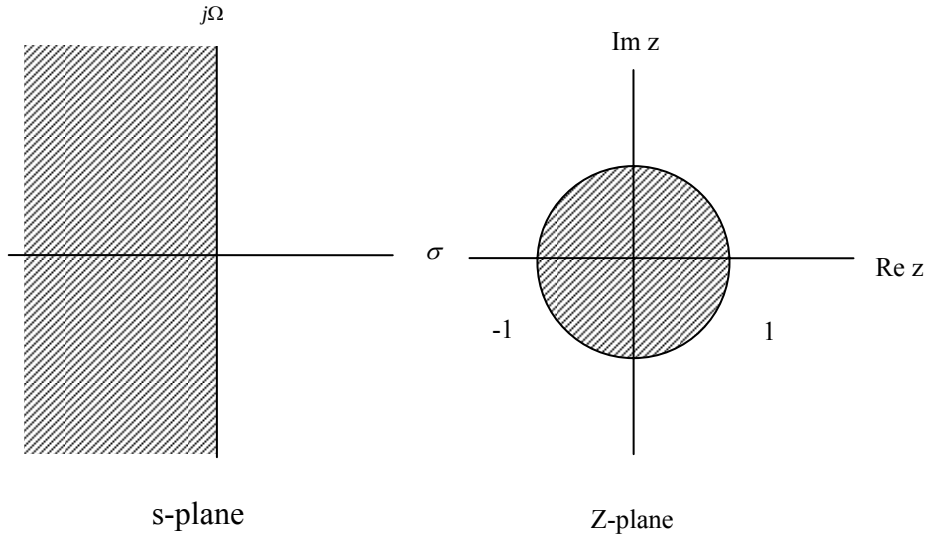

Figure 3.17 The bilinear transformation mapping

3.9 Summary

This chapter discussed the digital FIR and IIR filter structures and design. First, the structure of the filters using different forms was introduced. The design of the FIR filters based on truncated Fourier series and the design based on frequency sampling approach are derived. The design of the IIR filters by bilinear transformation was discussed.

CHAPTER 4

DEVELOPMENT OF A MATLAB-BASED DIGITAL FILTER DESIGN PROGRAM

4.1 Overview

This chapter describes the software program developed by the author for the design of digital filters. The chapter shows the interfaces and the functions of each component of the program. The complete source code of this application is included in Appendix A.

4.2 The Developed Design Program

The design program developed by the author was built to design different types of FIR and IIR digital filters. To design a specific digital filter, filter parameter must be given as input through graphical user interface and the desired filter gain and phase response will be plotted as a graph on the screen. Also the filter coefficients will be calculated and displayed.

4.3 Software Development Environment

The MATLAB software was used in developing the application program in this thesis. Packages such as MATLAB include many of features that a standard language includes, for example the support of graphical user interface (GUI). The principle advantage of MATLAB is that it enables sophisticated signal processing to be carried out using a simpler set of commands than a programming language typically uses.

4.3.1 Important Parts of Matlab System

The MATLAB system includes important parts for developing programs, which are listed below:

1. **Development Environment**

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

2. The MATLAB Mathematical Function Library

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

3. The MATLAB Language.

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

4.3.2 GUI Development Environment in Matlab

A graphical user interface (GUI) is a user interface built with graphical objects, such as buttons, text fields, and menus. In general, these objects already have meanings to most computer users. For example, when you press an OK button, your settings are applied and the dialog box is dismissed. Applications that provide GUIs are generally easier to learn and use since the person using the application does not need to know what commands are available or how they work. The action that results from a particular user action can be made clear by the design of the interface.

GUIDE, MATLAB's Graphical User Interface development environment, provides a set of tools for creating GUI's. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs. [17]

4.4 The User Application Interface

The main user interface of the application developed in this thesis consists of a window. This window contains a menu list for FIR filter design and another one for IIR filter design. Figure 4.1 shows the application interface which will be described in more detail within this chapter.

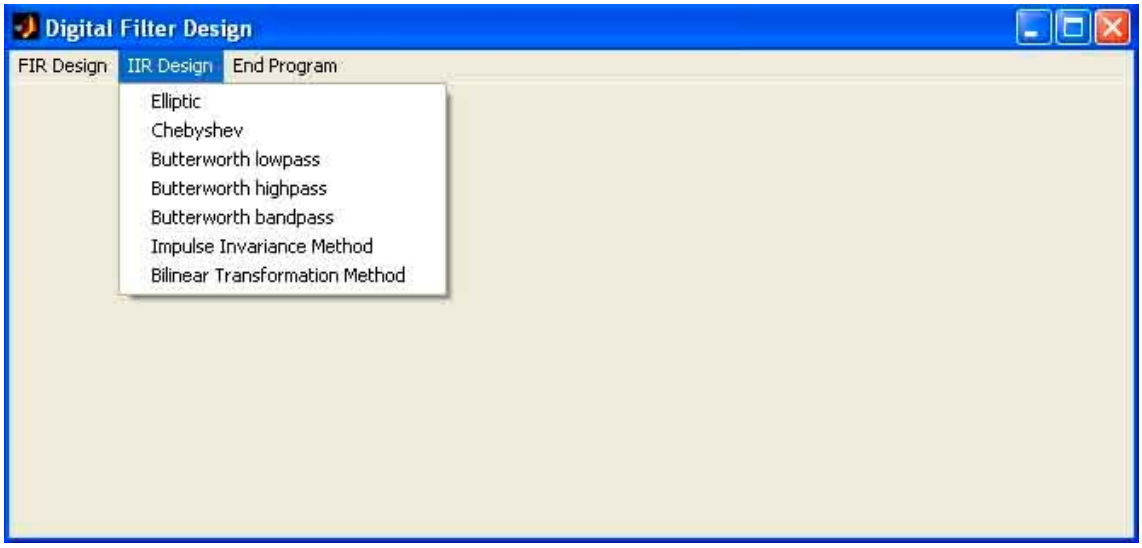

Figure 4.1 Main user interface of the application developed by the author

4.5 Functional Description of the Filter Design Interface

When a filter type is selected to be designed, the user interface shown in Figure 4.2 is obtained. On the Filter design interfaces window the following components exist:

1. Depending on filter type number of edit text boxes to get the filter parameter needed for the design.
2. One command button to give the order to design the filter.
3. Depending on filter type one or two list boxes to display filter coefficients obtained by the design of the filter.
4. Depending on filter type one or two command button to give the order to display filter coefficients.

In the following sections the components and functions of the program will be discussed.

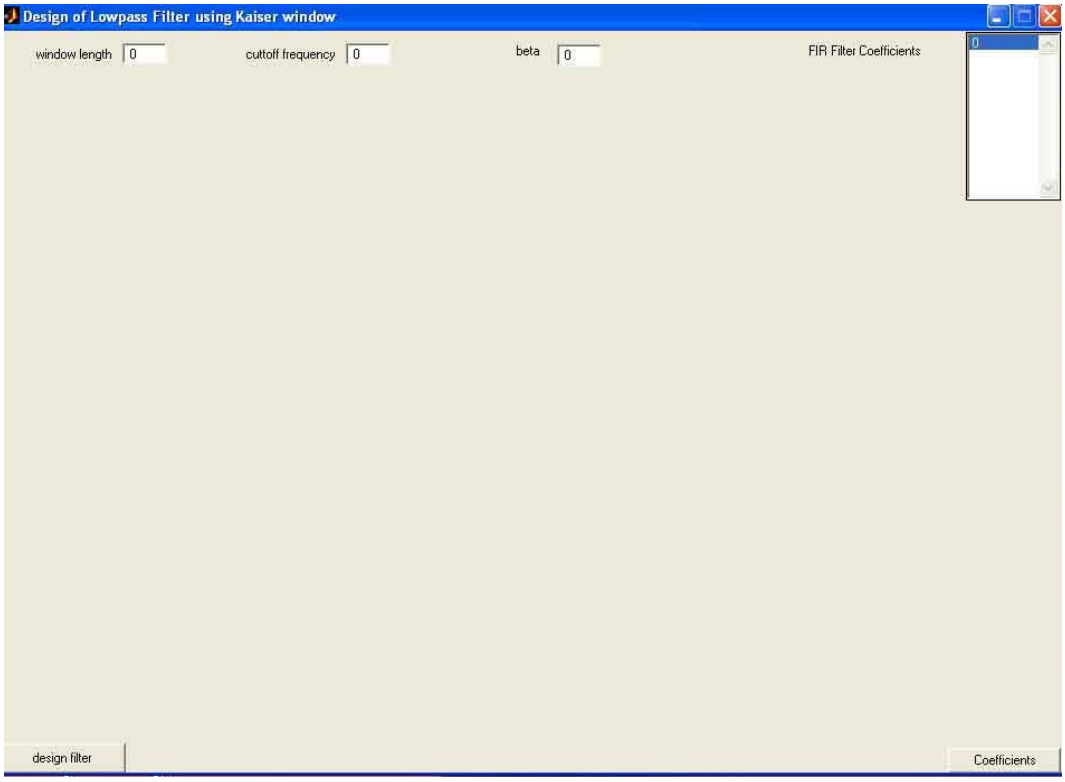

Figure 4.2 FIR filter design interface window

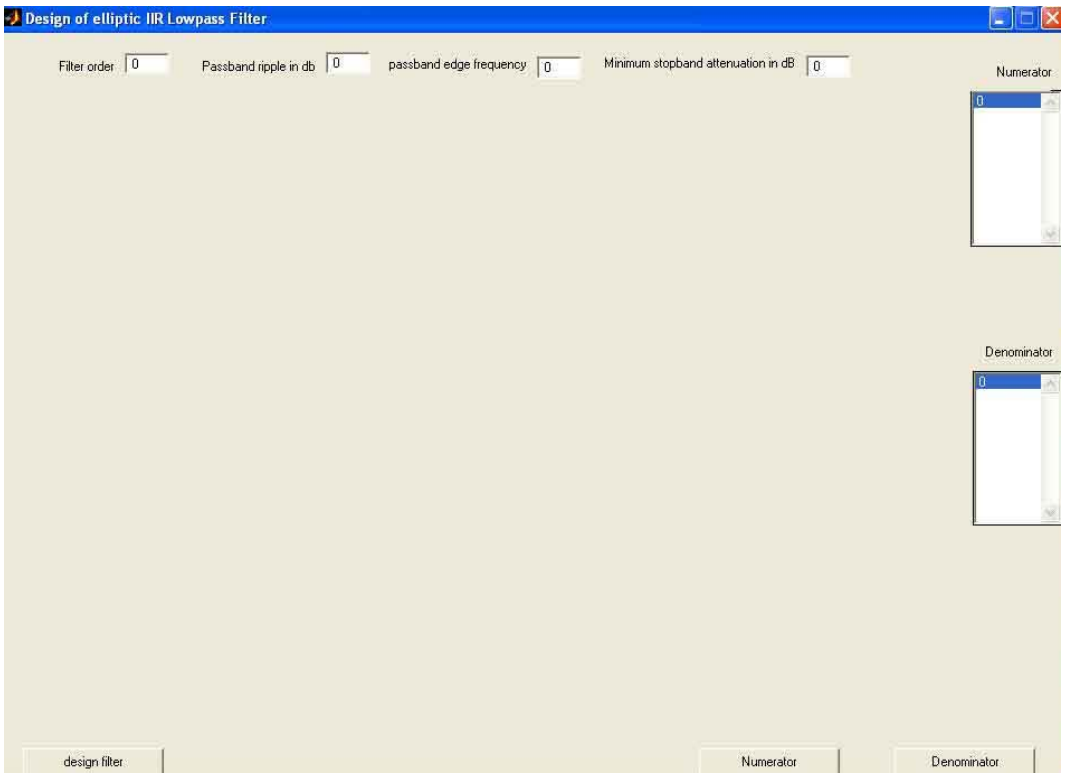

Figure 4.3 IIR filter design interfaces window

4.5.1 Edit Text Boxes

After selecting a filter type from menu list in the main user interface another filter design interface gives the possibility to enter the needed filter parameter like filter order, cutoff frequency and other parameter in labeled text boxes.

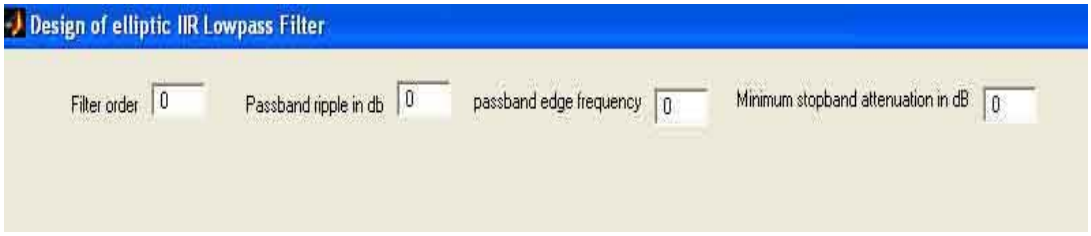

Figure 4.4 Text boxes for entering filter parameter

4.5.2 Command Buttons

After entering the needed filter parameter a command button ***design filter*** activate the design of the filter and show the filter responses. Depending on filter type another command button ***Coefficients***, ***Numerator*** or ***Denominator*** make it possible to display Filter coefficients in a list box.

Figure 4.5 Command buttons

4.5.3 List boxes

The computed FIR filter Coefficients are displayed on the list box ***FIR filter Coefficients***. For IIR filter the Numerator Coefficients are displayed on the list box ***Numerator*** and the Denominator Coefficients are displayed on the list box ***Denominator*** .

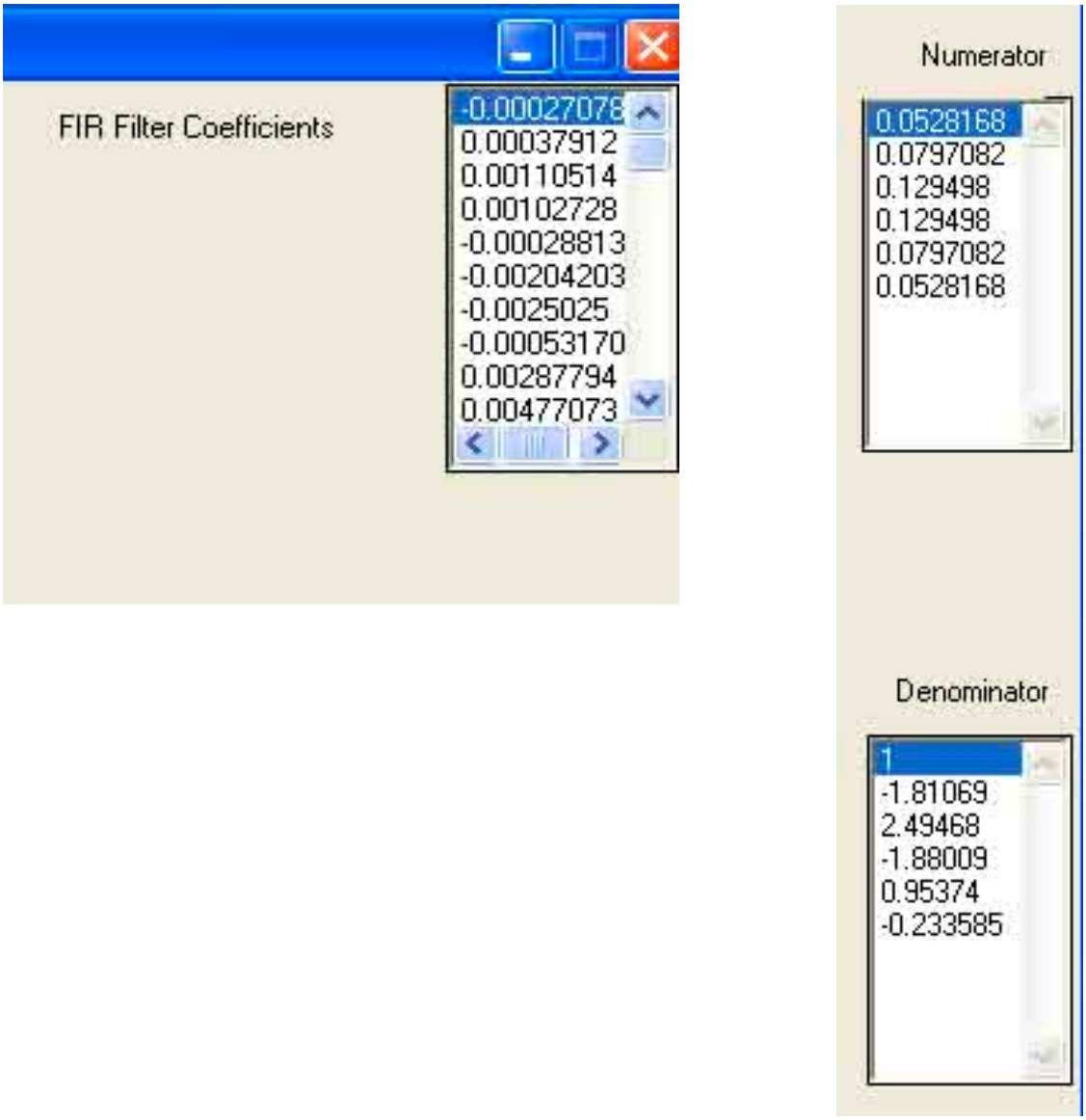

Figure 4.6 List boxes display filter Coefficients

4.6 Digital Filter Design Matlab Statements

In the developed filter design program a number of Matlab Statements are used, which are listed in the following sections with their syntax. [16], [17]

- **Kaiser statement:**

Kaiser statement computes a Kaiser window to design an FIR filter.

Syntax

w = kaiser(n,beta)

Description

The statement kaiser(n,beta) computes a length n Kaiser window with parameter beta. w = kaiser(n,beta) returns an n-point Kaiser window in the column vector w. beta is the Kaiser window β parameter that affects the sidelobe attenuation of the Fourier transform of the window.

- **hann statement:**

hann statement Compute a Hann (Hanning) window to design an FIR filter

Syntax

w = hann(n)

Description

w = hann(n) returns an n-point symmetric Hann window in the column vector w. n must be a positive integer.

- **Blackman statement:**

Blackman statement Compute a Blackman window to design an FIR filter.

Syntax

w = blackman(n)

Description

w = blackman(n) returns the n-point symmetric Blackman window in the column vector w, where n is a positive integer.

- **fir1 statement:**

fir1 statement is used to design a window-based finite impulse response filter. fir1 implements the classical method of windowed linear-phase FIR digital filter design. It designs filters in standard lowpass, highpass, bandpass, and bandstop configurations.By default the filter is normalized so that the magnitude response of the filter at the center frequency of the passband is 0 dB.

Syntax

b = fir1(n,Wn)

b = fir1(n,Wn,window)

Description

b = fir1(n,Wn) returns row vector b containing the n+1 coefficients of an order n lowpass FIR filter. This is a Hamming-window based, linear-phase filter with normalized cutoff frequency Wn. The output filter coefficients, b, are ordered in descending powers of z. Wn is a number between 0 and 1, where 1 corresponds to the Nyquist frequency.

If Wn is a two-element vector, Wn = [w1 w2], fir1 returns a bandpass filter with passband w1 $< \omega <$ w2.

- **butter statement:**

Butter statement is used for Butterworth analog and digital filter design. Butter designs lowpass, bandpass, highpass, and bandstop digital and analog Butterworth filters.

Syntax

[b,a] = butter(n,Wn)

Description

In digital Domain [b,a] = butter(n,Wn) designs an order n lowpass digital Butterworth filter with cutoff frequency Wn. It returns the filter coefficients in length n+1 row vectors b and a, with coefficients in descending powers of z..

For butter, the normalized cutoff frequency Wn must be a number between 0 and 1, where 1 corresponds to the Nyquist frequency, Л radians per sample. If Wn is a two-element vector, Wn = [w1 w2], butter returns an order 2*n digital bandpass filter with passband w1 < ω < w2.

- **cheby1 statement:**

Cheby1 statement is used for Chebyshev Type I filter design (passband ripple).Cheby1 designs lowpass, bandpass, highpass, and bandstop digital and analog Chebyshev Type I filters.

Syntax

[b,a] = cheby1(n,Rp,Wn)

Description

In Digital Domain [b,a] = cheby1(n,Rp,Wn) designs an order n Chebyshev lowpass digital Chebyshev filter with cutoff frequency Wn and Rp dB of peak-to-peak ripple in the passband. It returns the filter coefficients in the length n+1 row vectors b and a, with coefficients in descending powers of z.

Cutoff frequency is the frequency at which the magnitude response of the filter is equal to -Rp dB. For cheby1, the cutoff frequency Wn is a number between 0 and 1, where 1 corresponds to the Nyquist frequency, Л radians per sample. Smaller values of passband ripple Rp lead to wider transition widths (shallower rolloff characteristics).

- **Ellip statement:**

Ellip statement is used for Elliptic (Cauer) filter design. ellip designs lowpass, bandpass, highpass, and bandstop digital and analog elliptic filters.

Syntax

[b,a] = ellip(n,Rp,Rs,Wn)

Description

In digital Domain [b,a] = ellip(n,Rp,Rs,Wn) designs an order n lowpass digital elliptic filter with cutoff frequency Wn, Rp dB of ripple in the passband, and a stopband Rs dB down from the peak value in the passband. It returns the filter coefficients in the length n+1 row vectors b and a, with coefficients in descending powers of z.

The cutoff frequency is the edge of the passband, at which the magnitude response of the filter is -Rp dB. For ellip, the cutoff frequency Wn is a number between 0 and 1, where 1 corresponds to half the sampling frequency (Nyquist frequency).

Impinvar statement:

Impinvar statement is used to design IIR digital filter using Impulse invariance method for analog-to-digital filter conversion.

Syntax

[bz,az] = impinvar(b,a,fs)

Description

[bz,az] = impinvar(b,a,fs) creates a digital filter with numerator and denominator coefficients bz and az, respectively, whose impulse response is equal to the impulse response of the analog filter with coefficients b and a, scaled by 1/fs. If the argument fs, is leaved out or specified as the empty vector [], it takes the default value of 1 Hz.

- ## Bilinear statement:

Bilinear statement is used to design IIR digital filter using Bilinear transformation method for analog-to-digital filter conversion.

Syntax

[numd,dend] = bilinear(num,den,fs)

Description

The bilinear transformation is a mathematical mapping of variables. In digital filtering, it is a standard method of mapping the s or analog plane into the z or digital plane. It transforms analog filters, designed using classical filter design techniques, into their discrete equivalents.

4.7 Parts of the program listing

The complete source code of the application program is given in Appendix A. In this section, only the important parts of the source code for some FIR and IIR digital filter are given.

4.7.1 Source Code for FIR Digital Filters

- **FIR filter Design using Kaiser window**

```
data = getappdata(gcbf, 'UserData');  %Get filter parameter
w=kaiser(data.N+1,data.beta);          %Kaiser window
b = fir1(data.N,data.cf,w)             % Design the FIR filter using Kaiser window
freqz(b,1,512);                        % Plot the frequency and phase response
```

- **FIR filter Design using Blackman window**

```
data = getappdata(gcbf, 'UserData');  %Get filter parameter
w=blackman(data.N+1);                  %Blackman window
b = fir1(data.N,data.cf,ftype,w)       %Design the FIR filter using Blackman window
freqz(b,1,1024);                       % Plot the frequency and phase response
```

- **FIR filter Design using Hanning window**

```
data = getappdata(gcbf, 'UserData');  %Get filter parameter
w=hann(data.N+1);                      %Hanning window
b = fir1(data.N,data.cf,ftype,w)       %Design the FIR filter using Hanning window
freqz(b,1,1024);                       % Plot the frequency and phase response
```

4.7.2 Source Code for IIR Digital Filter

- **IIR Butterworth Bandpass Filter Design**

```
data = getappdata(gcbf, 'UserData');
M=data.N/2;
Wn=[data.W1 data.W2]
[b,a] = butter(M,Wn);
w= 0:0.01/pi:pi;
h = freqz(b,a,w);
gain = 20*log10(abs(h));
plot (w/pi,gain);grid;
xlabel('Normalized frequency');
ylabel('Gain, dB');
```

- **IIR Elliptic Lowpass Filter Design**

```
data = getappdata(gcbf, 'UserData');
[b,a] = ellip(data.N,data.Rp,data.Rs,data.Wn);
w= 0:0.01/pi:pi;
h = freqz(b,a,w);
plot (w/pi,20*log10(abs(h)));grid;
xlabel('\omega/\pi');
ylabel('Gain, dB');
```

- **IIR Type I Chebyshev Highpass Filter Design**

```
data = getappdata(gcbf, 'UserData');
[b,a] = cheby1(data.N,data.Rp,data.Wn,'high');
w= 0:0.01/pi:pi;
h = freqz(b,a,w);
plot (w/pi,20*log10(abs(h)));grid;
xlabel('\omega/\pi');
ylabel('Gain, dB');
```

- **IIR Butterworth LP Filter Design using Impulse Invariance Method**

```
data = getappdata(gcbf, 'UserData');
[num,den]=butter(data.N,data.Wn,'s');
%       Convert analogue filter into Discrete IIR filter
[b, a]=impinvar(num, den, data.Fs);
subplot(2,1,2)
[h, omega]=freqz(b, a, 512);
mag = 20*log10(abs(h))
plot(omega/pi,mag);grid;
xlabel('Normalized Frequency ')
ylabel('Gain dB')
```

4.8 Design and Results examples using the developed program

In the following a design example using the developed program and other results examples are given.

4.8.1 Transfer function general form of the digital filter

The general form of an FIR digital filter transfer function is given by

$$H(z) = \sum_{k=0}^{N-1} h[k] z^{-k} \tag{4.1}$$

The general form of IIR digital filter transfer function is given by

$$G(z) = \frac{B(z)}{A(z)} = \frac{b(1) + b(2)z^{-1} + ... + b(N+1)z^{-N}}{1 + a(2)z^{-1} + ... + a(N+1)z^{-N}} \tag{4.2}$$

4.8.2 Design example using the developed program:

A second order IIR butterworth lowpass digital filter is to design with the following specification:

Cuttoff frequency f_c = 100 Hz

Sampling frequency f_s = 1000 Hz

Solution:

1- Calculating filter coefficients

- Using the developed program ***lowpass butterworth*** from menu list ***IIR design*** is to choose.
- Filter parameter are to input

- Command button ***design filter*** plots the filter response and produce a file blp.cff containing the filter coefficients.
- Command buttons Numerator, Denominator displays the filter coefficients on the screen.

2- Plotting frequency response of designed filter

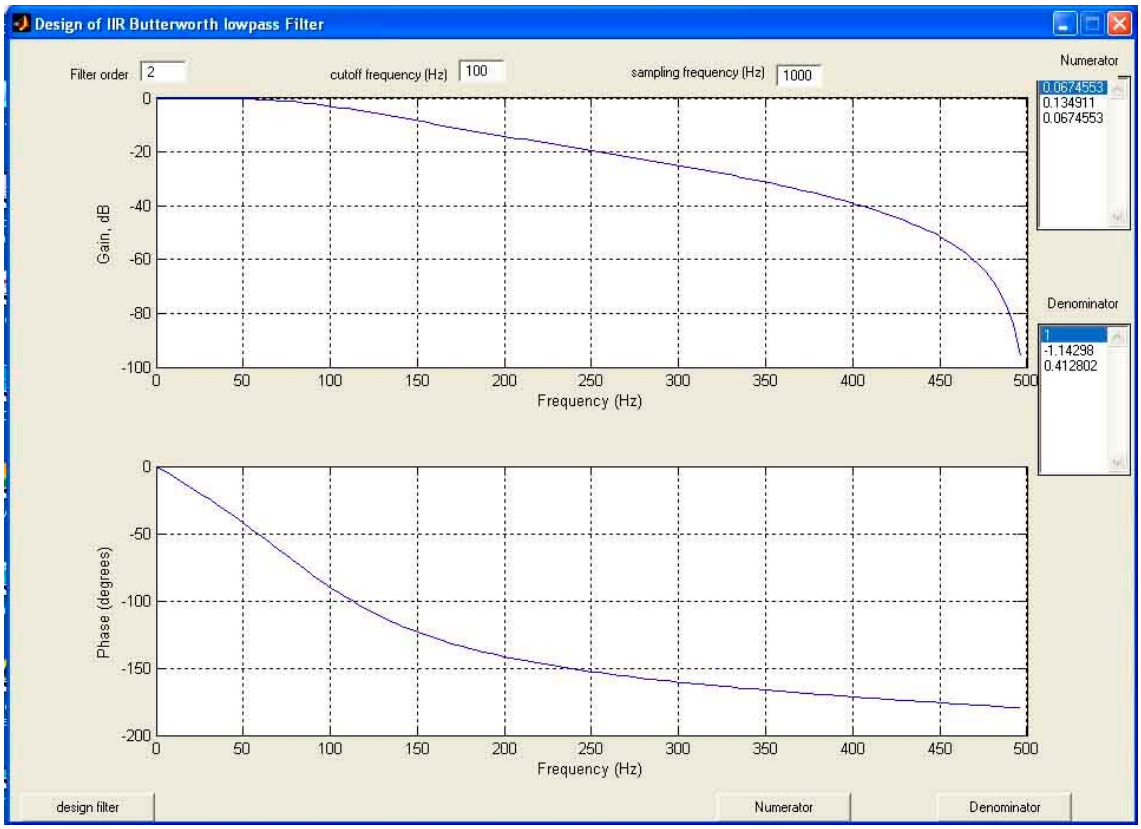

Figure 4.7 IIR butterworth lowpass filter

3- Writing transfer function of the designed filter

$$G(z) = \frac{0.0674552 + 0.13491054z^{-1} + 0.06745527z^{-2}}{1 - 1.14298050z^{-1} + 0.41280159z^{-2}}$$

4.8.3 Additional Results Examples

The following figures show some design examples obtained using the MATLAB program developed by the author:

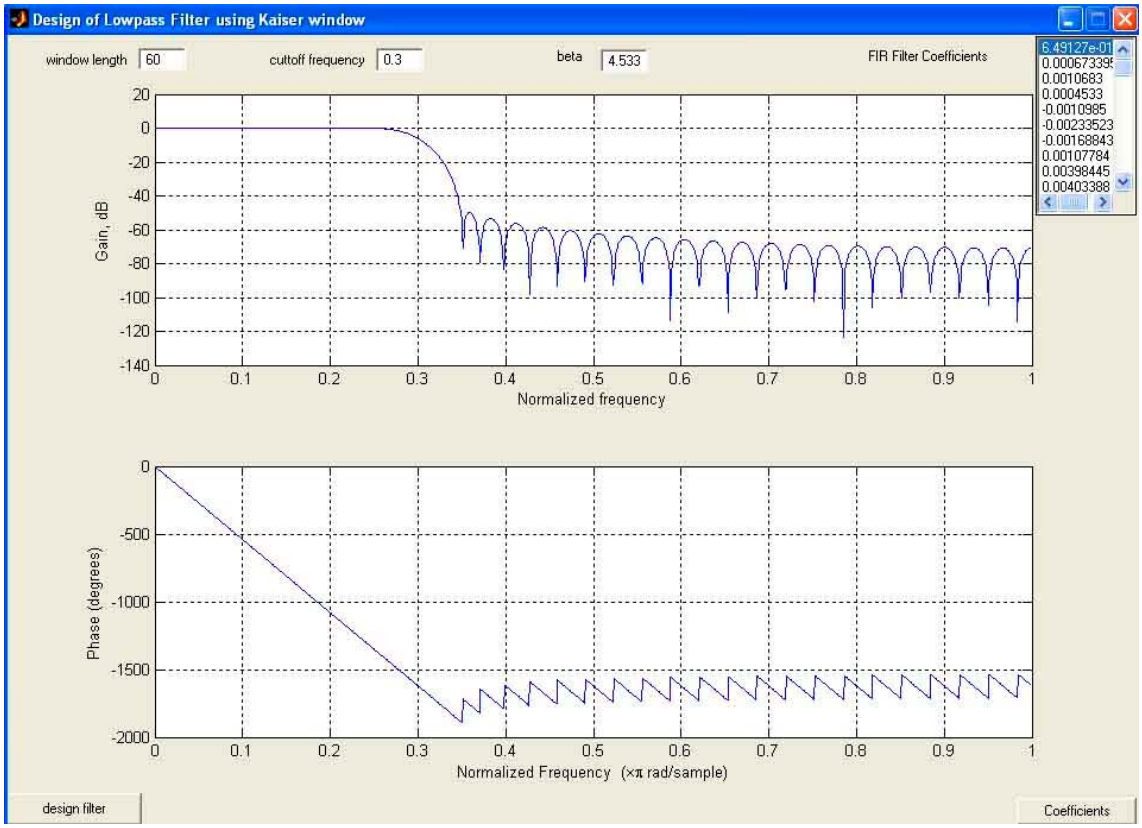

Figure 4.8 FIR lowpass filter using Kaiser window

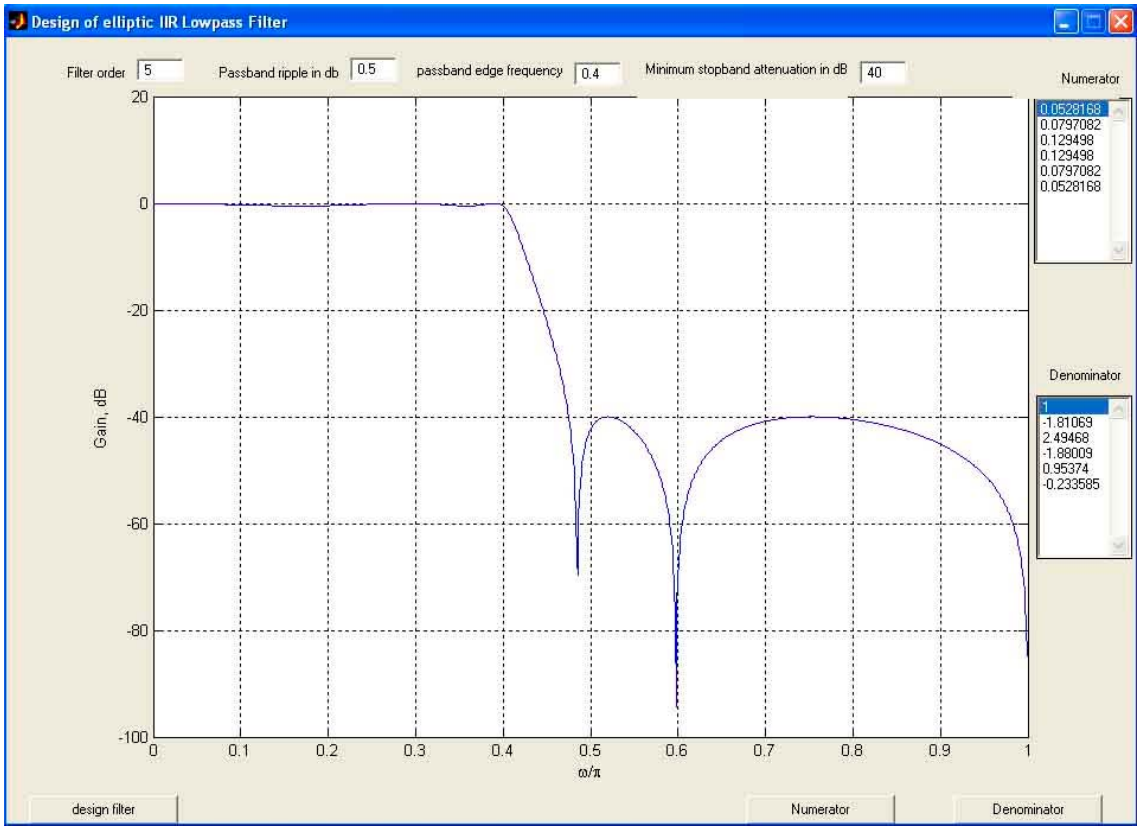

Figure 4.9 IIR elliptic lowpass filter

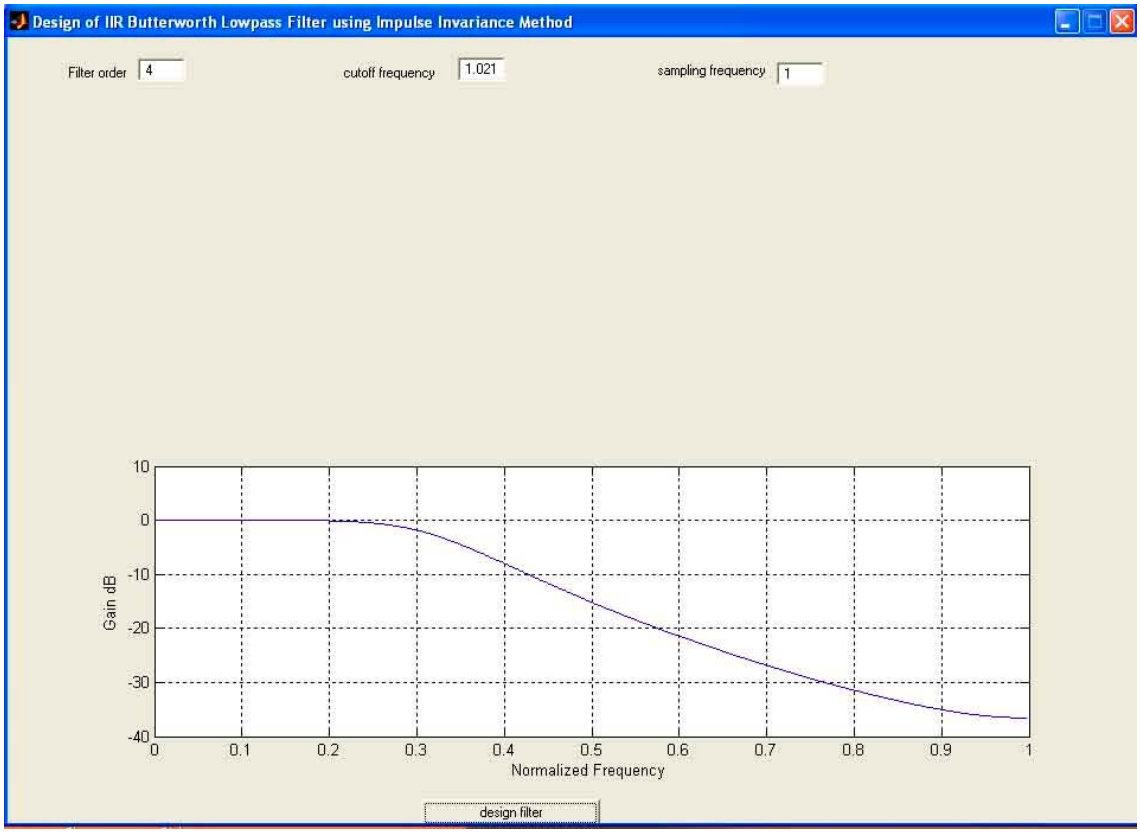

Figure 4.10 IIR Butterworth lowpass filter using impulse invariance method

4.9 Summary

This chapter has described the development of a MATLAB program for the design of FIR and IIR type digital filters. The program accepts the filter type and the required filter specifications from the user and then calculates the filter coefficients. The program is GUI based, easy to use, and is user friendly.

Another advantage of the program is that the frequency response and the phase response of the designed filter are also plotted by the program.

CHAPTER 5

DIGITAL FILTER IMPLEMENTATION

5.1 Overview

This chapter is about the implementation of a physical digital filter using a microcontroller. A PIC type microcontroller is used as the processing element. In this chapter a 2nd order IIR type Butterworth filter has been designed and implemented on the microcontroller.

5.2 Implementation

The block diagram of the digital filter implemented in this section is shown in Figure 5.1. The overall system consists of: A/D converter, Microcontroller, D/A converter.

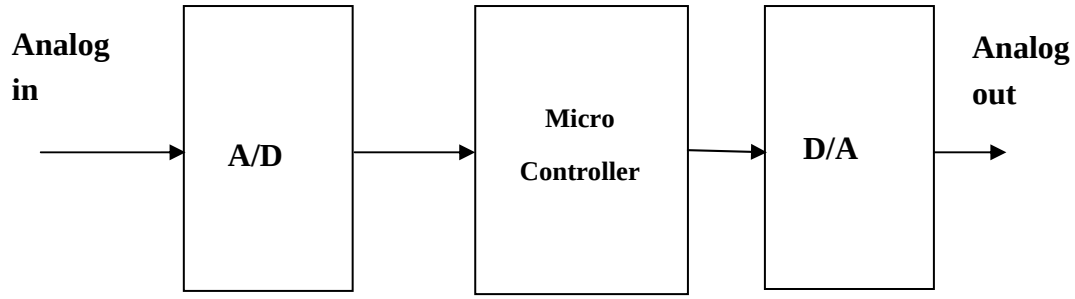

Figure 5.1 Block diagram of the digital filter

The operation of the system is as follows:

A/D Converter: Analog input signal is read from the A/D converter. In this project an 8-bit bipolar A/D converter is used. The input signal voltage level is ±5V and the converter provides a quantized digital signal. The converter used is the AD673 8-bit A/D converter manufactured by Analog Devices. This converter has a 20μs conversion time, accepts both unipolar and bipolar input voltages, and operates with two power supplies: +5V and -12V. [21]

The A/D converter has a built-in zener regulated reference voltage source which is used by the chip. The functional block diagram of the AD673 A/D converter is shown in Figure 5.2.

Basically the device has:

- An analog input pin
- 8 digital output pins
- A CONVERT input pin
- A DATA READY output pin

Conversion starts by applying the analog signal to the input pin and then pulsing the CONVERT pin. The high-to-low transition of the CONVERT signal starts the A/D conversion and the DATA READY pin goes to logic HIGH. After about 20μs the conversion is complete and the digital data is available at the 8 output pins. The end of conversion is signaled by the DATA READY output pin going low. This output is used to inform the microcontroller that the conversion is complete and the converted digital data can be read from the output of the A/D.

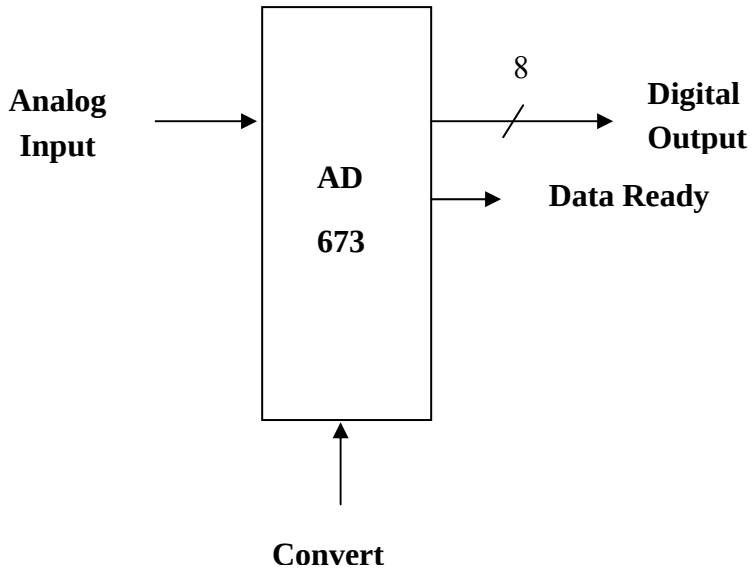

Figure 5.2 Functional block diagram of the A/D converter

The AD673 A/D converter provides offset-binary type output for bipolar signals. Offset-binary is similar to the standard 2s-complement numbering notation, but the most-significant-bit (MSB) is inverted. Some offset-binary numbers and their 2s-complement equivalents in 8-bit code are given below:

Table 5.1 offset-binary numbers and their 2s-complement

SCALE	OFFSET BINARY	2s COMPLEMENT
+Full scale	11111111	01111111
+0.75 Full scale	11100000	01100000
+0.5 Full scale	11000000	01000000
+0.25 Full scale	10100000	00100000
0	10000000	00000000
-0.25 Full scale	01100000	11100000
-0.5 Full scale	01000000	11000000
-0.75 Full scale	00100000	10100000
-Full scale	00000000	10000000

2s-complement is the commonly used numbering system in signal processing applications. Numbers represented in 2s-complement notation can easily be added, subtracted, multiplied, and divided. A given offset-binary number can be converted to its 2s-complement equivalent by simply inverting the MSB bit.

Microcontroller: In the practical part of this thesis a popular PIC16F877 type microcontroller, operating with 20MHz crystal is used. With 20MHz clock, the basic instruction cycle time is 0.2µs (the clock frequency is divided by 4 to obtain the instruction cycle time). This is a 40-pin microcontroller with the following features:

- 8K program memory
- 368 byte RAM
- 256 byte EEPROM
- 33 I/O ports
- 14 interrupt sources
- 8-bit timer
- 2x16-bit timers
- 10-bit 8 channel A/D converter
- USART
- Pulse width modulation (PWM) output
- 25mA output current
- Up to 20MHz operation
- Power on reset

The pin layout of the PIC16F877 microcontroller is shown in Figure 5.3. Notice that the I/O port pins are multi-function and shared between different functionalities.

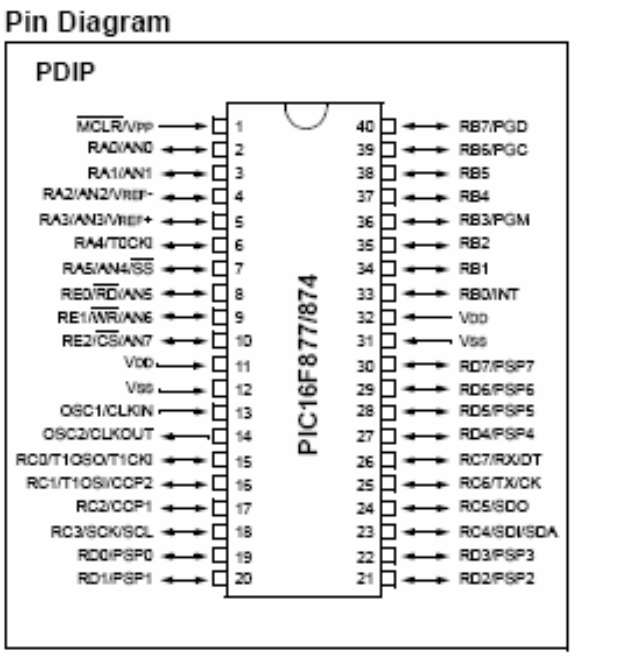

Figure 5.3 PIC16F877 pin layout

D/A Converter: The D/A converter receives digital signals from the microcontroller and provides an analog output signal. In the practical part of this thesis an AD7302 type D/A converter is used. This converter is manufactured by Analog Devices and is a dual, 8-bit buffered voltage output converter that operates from a single 2.7V to 5.5V supply. The on-chip output amplifier allows rail-to-rail output swing to be achieved. The device accepts parallel digital input data, converts this data to analog format and provides analog output. The functional block diagram, of the AD7302 converter is shown in Figure 5.4. [21]

Basically the device has:

- 8 digital inputs
- WR write input
- Channel select input
- Reference input
- Vo analog output

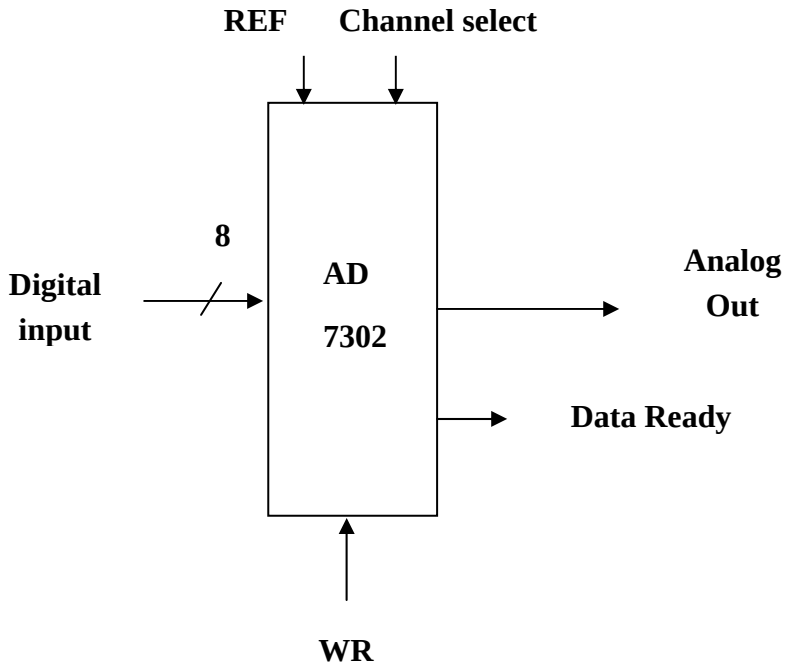

Figure 5.4 Functional block diagram of the D/A converter

Digital data is sent to the converter in 8-bit parallel format and then the WR write control input is pulsed low and then high. After about 2μs the analog data is available at the output of the device.

The input of the D/A converter should be in offset-binary. Thus, it is necessary to complement the MSB bit before the data is sent to the converter. Normally the output of AD7302 is in unipolar mode and an operational amplifier is connected to obtain bipolar signals in the range ±5V. The operational amplifier used in this project is the OP295 manufactured by Analog Devices. This is a quad operational amplifier in a small 8-pin package. The device operates from 3V to 36V either unipolar or with dual supply. The gain-bandwidth product of this amplifier is 75 kHz which is more than sufficient for our application. One of the advantages of using OP295 is that this operational amplifier provides rail-to-rail output voltage swing. Thus, the output voltage is accurate. Another advantage of the OP295 is its large current drive capability. The device can provide in excess of 20mA to coaxial cables and to capacitive loads.

The circuit diagram of the digital filter implementation is shown in Figure 5.5. Power is provided to the circuit using LM7805 for the +5V supply, LM7905 for -5V supply, and LM7912 for the -12V supply. All the power supplies are rated at a maximum current of 100mA.

The PIC microcontroller is at the centre of the design. The AD673 A/D converter is connected to Port B pins of the microcontroller. The converter is controlled from Port C pins of the microcontroller as follows:

PORT C pin	**AD673 pin**
RC0	DATA OUTPUT
RC1	CONVERT input

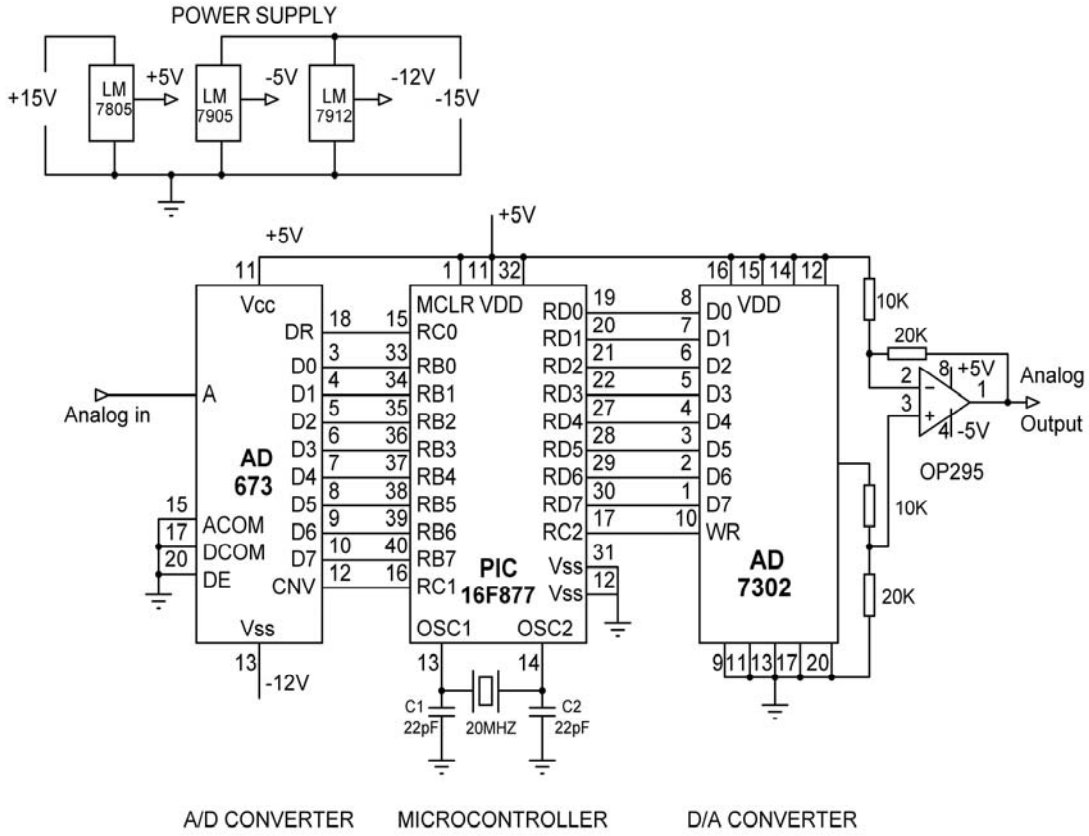

Figure 5.5 Circuit diagram of the digital filter designed

Analog signal is directly applied to pin 14 of the AD673. Pin 11 is connected to +5V supply, and pin 13 is connected to -12V supply.

Port D output of the microcontroller is connected to the data inputs of the AD7302 D/A converter. The WR control input of the converter is directly controlled from port RC2 of the microcontroller. The output of the D/A converter is connected to the positive input of the OP295 operational amplifier through a series of resistors. The negative input of the operational amplifier is driven from the +5V supply. The output voltage of the operational amplifier is ±5V.

5.3 Digital Filter Parameters

The digital filter was designed with the following parameters:

- 2^{nd} order Butterworth IIR filter
- Sampling frequency = 2.5 kHz
- Cut-off frequency = 250 Hz

The filter coefficients were obtained by running the MATLAB program developed earlier. The coefficients are:

b_0= 0.06745 b_1= 0.13491 b_2= 0.06745

a_0=1 a_1= -1.14298 a_2= 0.412801

Based on these parameters and the general form

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} + a_2 z^{-2}}$$

of the digital filter transfer function, we can write the transfer function as follows:

$$H(z) = \frac{0.06745 + 0.13491 z^{-1} + 0.06745 z^{-2}}{1 - 1.14298 z^{-1} + 0.412801 z^{-2}}$$

$$H(z) = \frac{0.06745(1 + 2z^{-1} + z^{-2})}{1 - 1.14298 z^{-1} + 0.412801 z^{-2}}$$

We can now derive the necessary equations for the implementation of this second order section as follows:

Writing the above transfer function as:

$$H(z) = \frac{y(z)}{u(z)} = \frac{K(1 + 2z^{-1} + z^{-2})}{1 + Bz^{-1} + Cz^{-2}}$$

where, K = 0.06745 B = -1.14298 C = 0.412801

Let,

$$\frac{y(z)}{u(z)} = \frac{y(z)}{q(z)} \frac{q(z)}{u(z)}$$

where,

$$\frac{q(z)}{u(z)} = \frac{K}{1 + Bz^{-1} + Cz^{-2}} \tag{5.1}$$

and

$$\frac{y(z)}{q(z)} = 1 + 2z^{-1} + z^{-2} \tag{5.2}$$

from (5.1) : $q(z) = Ku(z) - Bq(z)z^{-1} - Cq(z)z^{-2}$ (5.3)

writing (5.2) in terms of $q(z)$: $y(z) = q(z) + 2q(z)z^{-1} + q(z)z^{-2}$ (5.4)

Let, x_1 and x_2 be two state variables, where $x_1 = q(z)z^{-1}$ and $x_2 = q(z)z^{-2} = x_1 z^{-1}$

Then we can write,

$$q(z) = Ku(z) - Bx_1 - Cx_2 \tag{5.5}$$

$$y(z) = q(z) + 2x_1 + x_2 \tag{5.6}$$

Thus, the following operations will be required to implement the second order filter section:

$$\begin{aligned} Ku - Bx_1 - Cx_2 &\rightarrow q \\ q + 2x_1 + x_2 &\rightarrow y \\ x_1 &\rightarrow x_2 \\ q &\rightarrow x_1 \end{aligned}$$

The block diagram of the 2[ed] order filter implementation is shown in Figure 5.6.

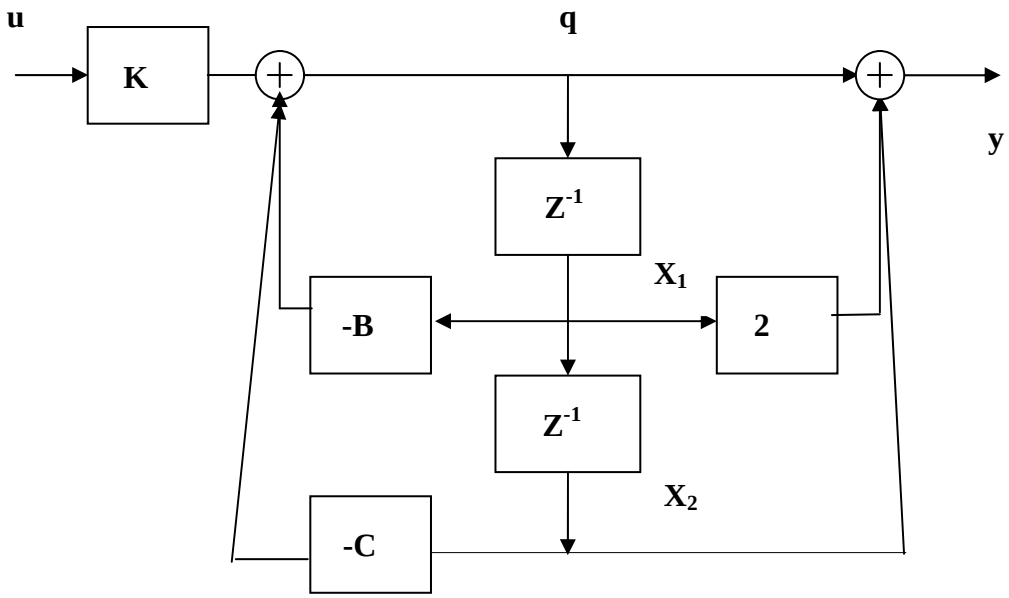

Figure 5.6 Second order filter implementation

The steps for a second order filter implementation are summarized below:

- Input u from A/D converter
- Calculate $Ku - Bx_1 - Cx_2$
- Store result in q
- Calculate $q + 2x_1 + x_2$
- Output result to D/A converter
- Perform $x_1 \rightarrow x_2$

  $q \rightarrow x_1$
- Enable interrupts
- If interrupt occurs repeat this process

5.4 Software of the Digital Filter

The microcontroller was programmed using the C language. The compiler used was the PICC Lite, manufactured by Hi-Tech and distributed free for a few types of microcontrollers.

The program is interrupt driven. The timer TMR0 of the microcontroller was programmed to generate an interrupt at every sampling time i.e. at every 400μs (sampling frequency = 2.5 kHz which has a period of 400μs). The program normally waits for an interrupt to occur and then reads a sample from the analog input channel, processes the input and then sends a signal to the analog output channel. The operation of the software is described below in simple steps:

- Initialize the program
- Perform digital filtering

The initialization consists of the following:

- Initialize program variables
- Initialize filter states
- Initialize port directions
- Load TMR0 timer register
- Enable interrupts
- Wait for interrupts

The digital filtering section of the program is entirely implemented in an interrupt service routine (ISR) so that the required sampling frequency is obtained accurately. The ISR consists of the following operations:

- Re-load TMR0 timer register
- Read a sample from A/D converter
- Perform filtering action
- Send output data to D/A converter
- Return from interrupt

On return from the ISR routine, the program goes back to the main code and waits for the next interrupt to occur. This way, the filtering action is guaranteed to occur at every sampling interval. i.e. at every 400µs.

The full listing of the digital filter program is given in Figure 5.7. At the beginning of the program the program variables are declared, filter states cleared to zero and the filter coefficients K, B and C are initialized. Then, the microcontroller registers are initialized. Timer TMR0 is set to interrupt at every 400 microseconds, and interrupts are enabled. The main program then waits in a loop for the occurrence of timer interrupts.TMR0 timing is controlled with the following formula: [20]

Time_to_interrupt (microsecond) = (4*clock_period)*pre_scaler*(256-TMR0)

where,

Time_to_interrupt:	is the required interrupt interval (400 microsecond)
clock _period:	is the crystal clock period (0.05us for 20MHz)
Pre_scaler:	is the timer pre-scaler value
TMR0:	is the number to be loaded into timer register TMR0

The pre-scaler can be programmed between 2 and 256 using the OPTION_REG register of the PIC microcontroller. We need a 400 microsecond interrupt time. Assuming a pre-scaler value of 8, with a 20MHz clock, the value to be loaded into the timer register TMR0 at every iteration can be calculated as:

TMR0 = 256 - 400 / (4*0.05*8)

or,

TMR0 = 6

Thus the timer register TMR0 should be loaded with decimal number 6 at the beginning of every interrupt loop. When a timer interrupt occurs program jumps to the interrupt service routine. This routine starts with procedure name “filter”.

The following variables are used in the program:

Variable	Description
x1	Filter state
x2	Filter state
q	Temporary state
fu	Floating point A/D input value
u	Integer A/D value
y	D/A value
K	Filter coefficient
B	Filter coefficient
C	Filter coefficient
temp	Temporary floating point variable

Integer offset binary variables are converted to floating point 2s complement format at the input. Similarly, 2s complement floating point variables are converted into offset binary format just before output to the D/A converter.

```
/*********************************************************
        2nd Order Butterworth IIR Digital Filter
        ========================================
This is the program for a 2nd order Butterworth IIR digital filter.
The filter is implemented on a PIC16F877 microcontroller, operating
At 20 MHz clock. The basic instruction cycle time is 0.2 microsecond.

The filter parameters are as follows:
        Sampling frequency = 2.5 kHz
        Cut-off frequency = 250 Hz

The filter is implemented in a timer interrupt routine (ISR). The ISR
is called every 400 microsecond (2.5 kHz) and the filter algorithm is
implemented.  The steps at every 400 microseconds are basically:

        Read analog input
        Process data
        Send out analog output
File:   FILTER.C                        Date:   February 2006
*********************************************************/
```

```
        #include <pic.h>
// Declare variables
        float x1, x2, q, fu, temp, K, B, C;
        unsigned char u, y, s, j;
// Declare symbols
        #define DATA_READY      RC0
        #define CONVERT         RC1
        #define WR              RC2
/*************** INTERRUPT SERVICE ROUTINE **************
   This is the interrupt service routine where the filter algorithm is implemented.
   This routine is called at every 400 microseconds. Inside the routine the
   analog data is received from the A/D converter, the digital filter algorithm
   is implemented, and the data is sent to the D/A converter */

void interrupt filter(void)
{
        TMR0 = 6;                       // Re-load TMR0
        // Start A/D conversion
        CONVERT = 1;
        for(j = 0; j<10; j++) CONVERT = 1; // Wait for DATA_READY to rise
        CONVERT = 0;
        while(DATA_READY);              // Wait until DATA_READY=0
        // Read converted analog data
        u = PORTB;                      // Get converted data
        // Convert to 2s complement floating point
        s = u & 0x80;
        u = u ^ 0x80;
        if(s == 0)
        {
                u = u ^ 0xFF;   u++;
        }
        fu = u;                         // A/D data in floating point, fu
        // Perform the digital filtering
        q = K*fu – B*x1 – C*x2;
        temp = q + 2*x1+x2;
        y = (int)temp;
        // Convert to offset binary
        if(s != 0)
        {
         y = y | 0x80;
```

```
else
{
        y = y ^ 0xFF;   y++;   y = y & 0x7F;
        }
        PORTD = y;
        WR = 0;
        WR = 1;                                 // Send out digital data to D/A
                x2 = x1;                                // Swap registers
        x1 = q;
        // Re-enable TMR0 interrupts
        T0IF = 0;                               // Re-enable TMR0 interrupts
}
/* ------------------- Start of MAIN program ---------------------- */
main()
{
// Initialize filter parameters
        x1 = 0;                 x2 = 0;
        K = 0.06745;            B = -1.14298;           C = 0.412801;
// Initialize registers
        TRISB = 0xFF;                                   // PORT B is all input
        TRISC = 1;                                      // RC0 is input
        TRISD = 0;                                      // PORT D is all output
        CONVERT = 0;                                    // A/D in idle mode
        WR = 1;                                         // D/A in idle mode
        T0CS = 0;   PSA = 0;                            // Select TMR0
        PS0 = 0;   PS1 = 1;   PS2 = 0;                  // Set timer pre-scaler to 8
        TMR0 = 6;                                       // Set for 400us interrupt
        T0IE = 1;                                       // Enable TMR0 interrupts
        T0IF =0;                                        //  Enable  TMR0  interrupt
flag
        ei();                                           // Enable global interrupts
WAIT:       goto WAIT;                                  // Wait for timer interrupt
}
/*------------------------- END OF PROGRAM -------------------------*/
```

Figure 5.7 Program listing of the digital filter

A part of the object code of the program is given in Figure 5.8. This code can be used to load the microcontroller memory using a programmer device.

```
:10000000830100308A00DF28FC0003088301C2005E
:100010000408C3007008C7007108C8007208C60051
:100030007708C9007808CA008330831 ...
```

Figure 5.8 Program object code

5.5 Experimental Setup

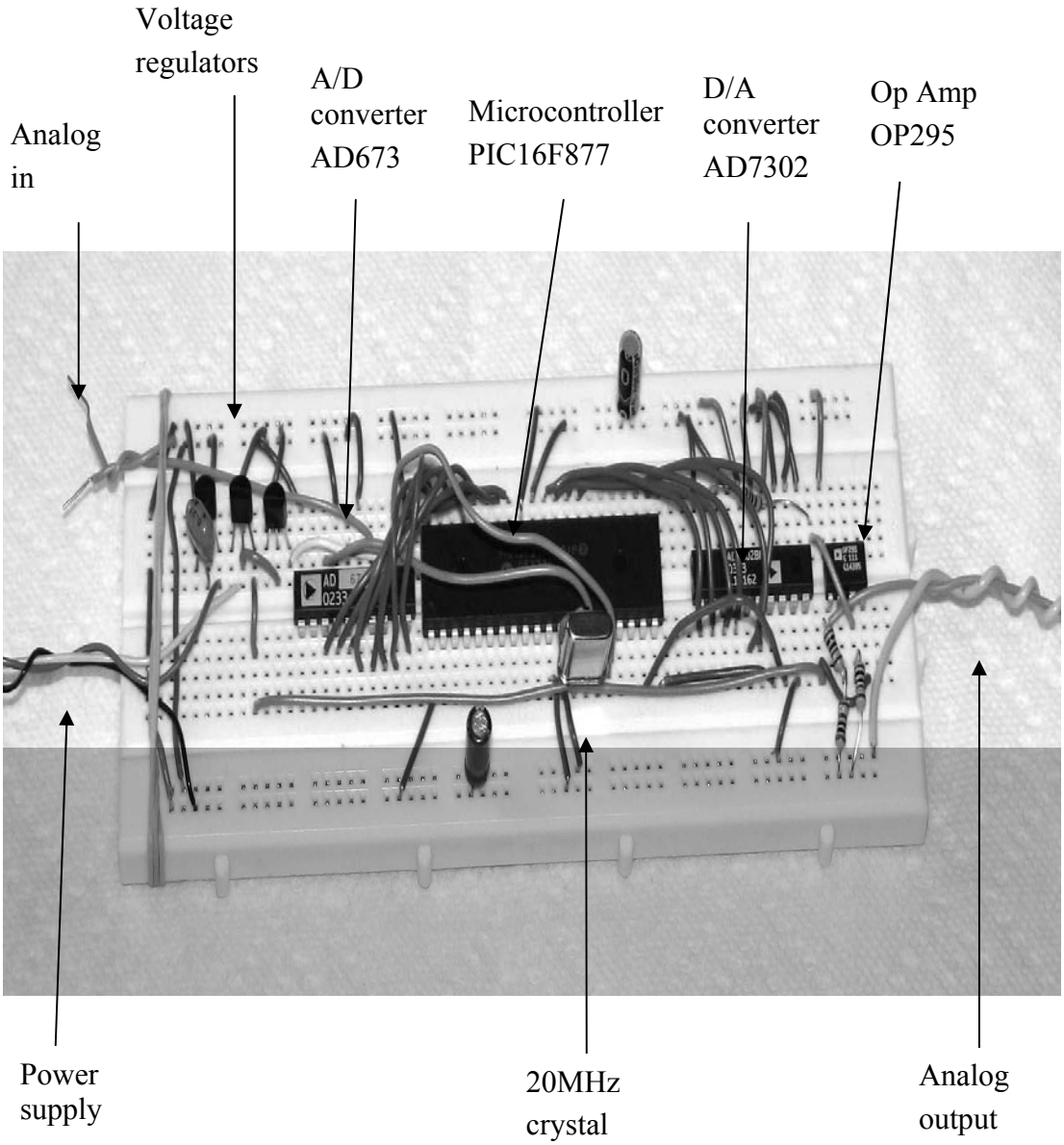

Figure 5.9 Digital filter implemented on a breadboard

The construction of the digital filter hardware on a breadboard is shown in Figure 5.9. The connections between various components are carried out using strip wires. The +5V, -5V, and -12V voltage regulators are on the breadboard but the ±15V main power supply is external to the breadboard. All the major components on the breadboard have been indicated with arrows in Figure 5.9.

5.5.1 Testing the Digital Filter

In order to test the digital filter designed, a sine-wave frequency generator with a sweeping range of 0-5kHz was connected to the analog input of the filter. Also, one channel of an oscilloscope was connected to this input so that the magnitude and the frequency of the input sine wave could be observed and also measured. Similarly, the second channel of the oscilloscope was connected to the analog output of the filter so that the magnitude and the shape of the output signal could be observed and also measured. Figure 5.10 and figure 5.11 show the experimental setup.

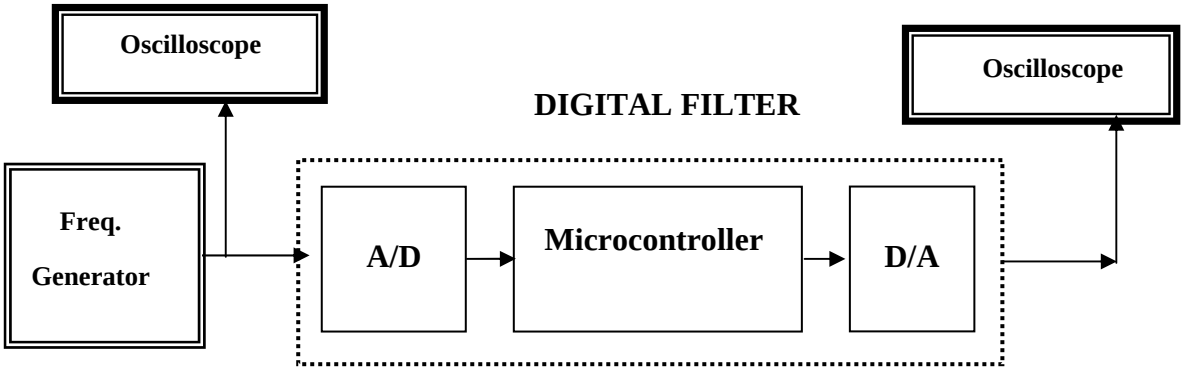

Figure 5.10 Experimental setup block diagram

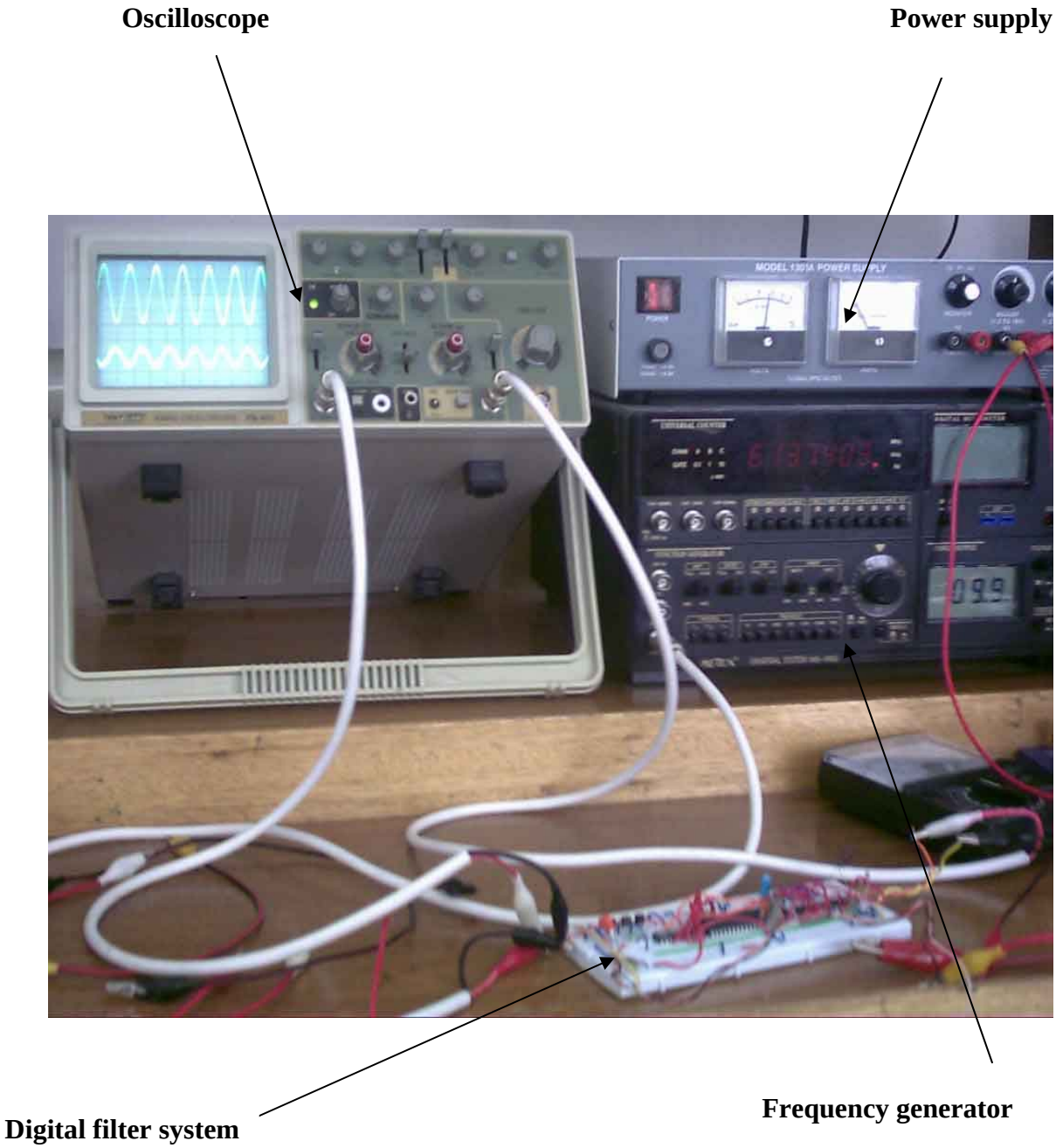

Figure 5.11 Experimental setup

5.5.2 Input/Output oscilloscope observation

The figures 5.12 (a, b, c, d) show the magnitude and the frequency of the input sine wave which could be observed and also measured on the oscilloscope. Similarly, through the second channel of the oscilloscope the analog output of the filter could be displayed. The magnitude and the shape of the output signal could be observed and also measured.

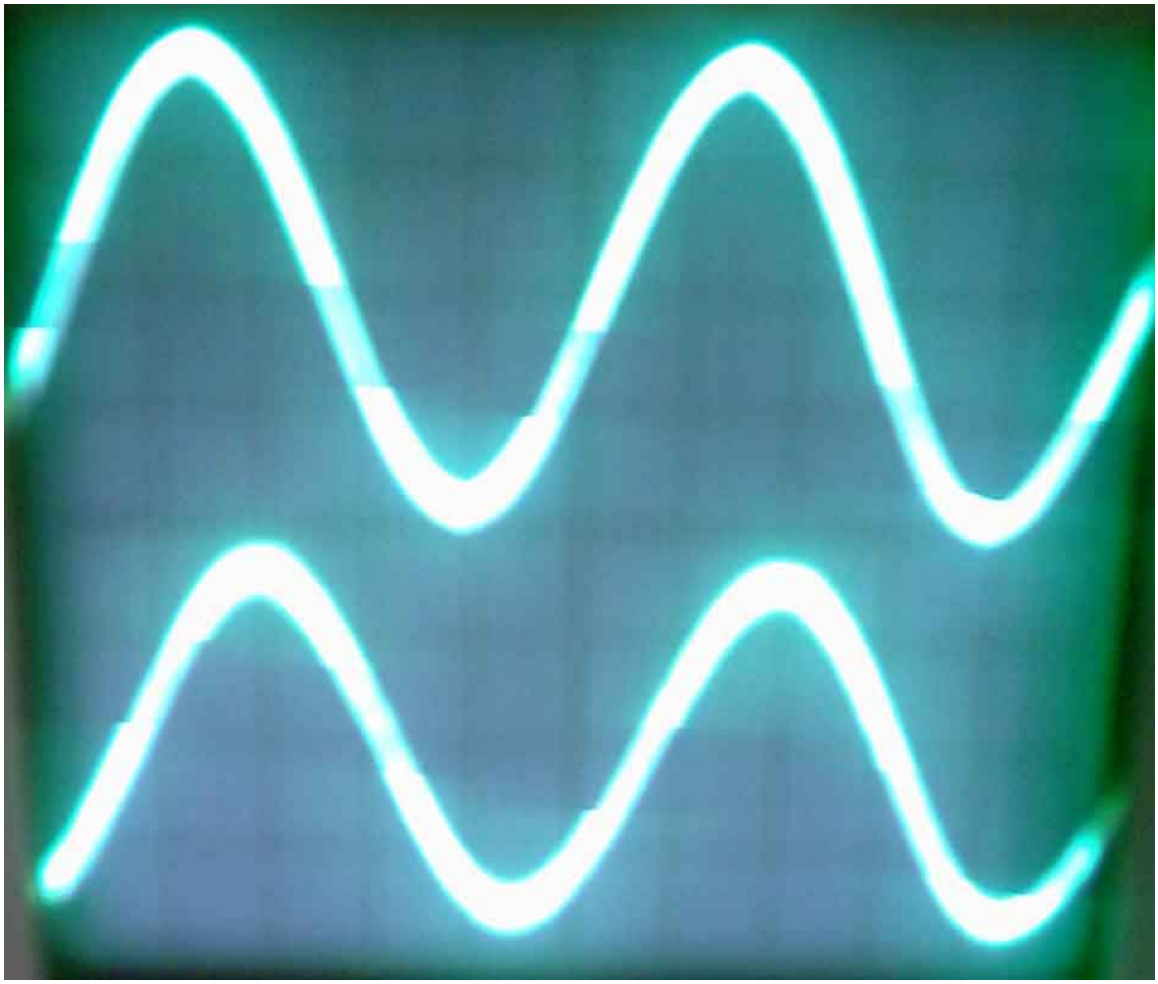

Figure 5.12a Magnitude and frequency response by 100 Hz

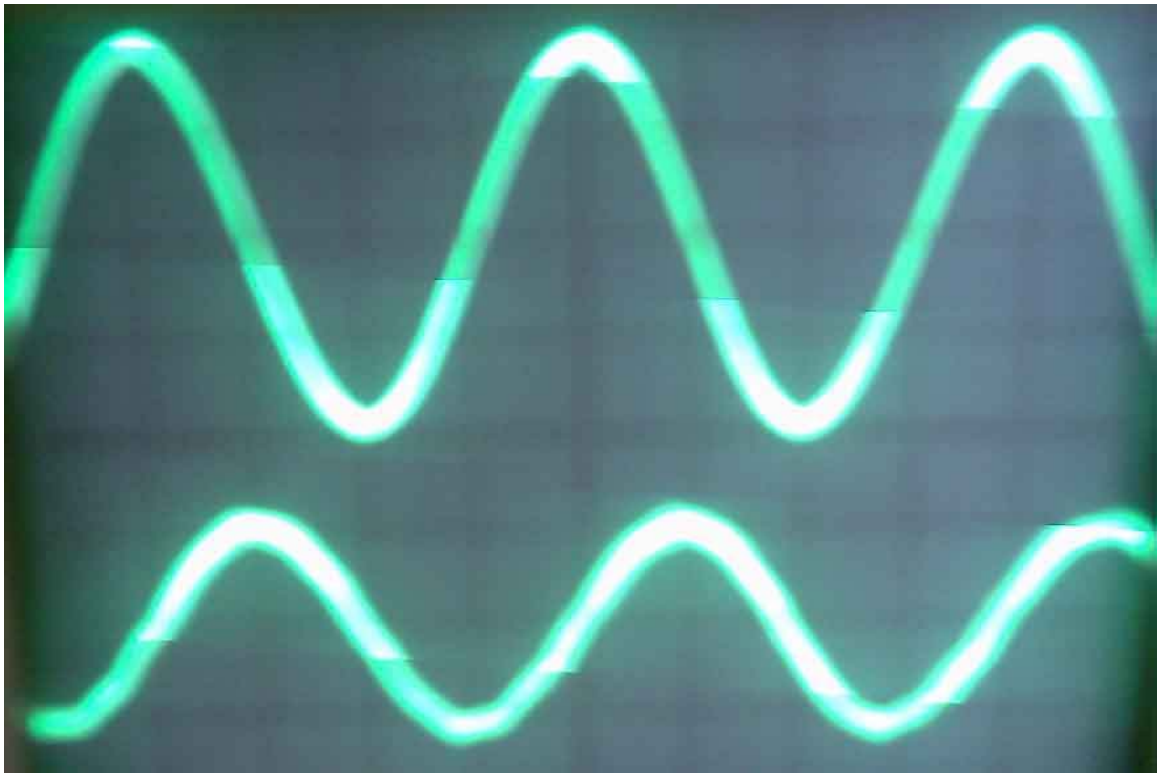

Figure 5.12b Magnitude and frequency response by 250 Hz

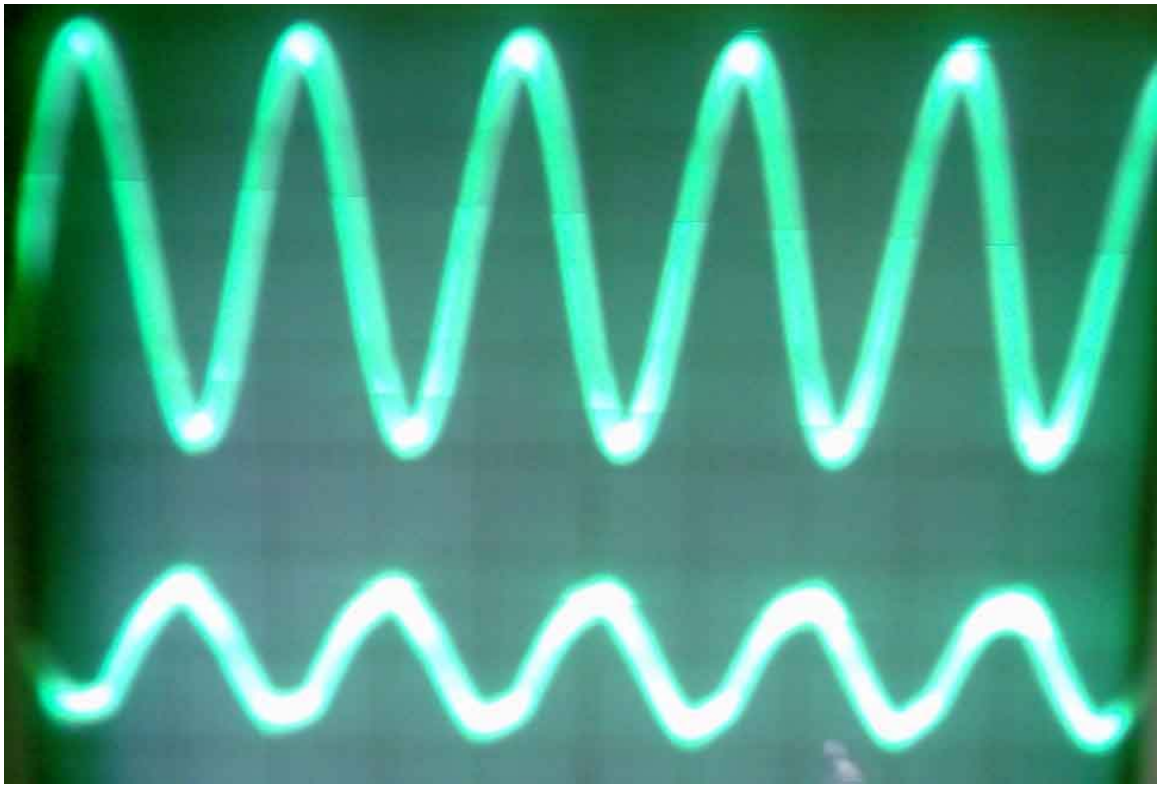

Figure5.12c Magnitude and frequency response by 500 Hz

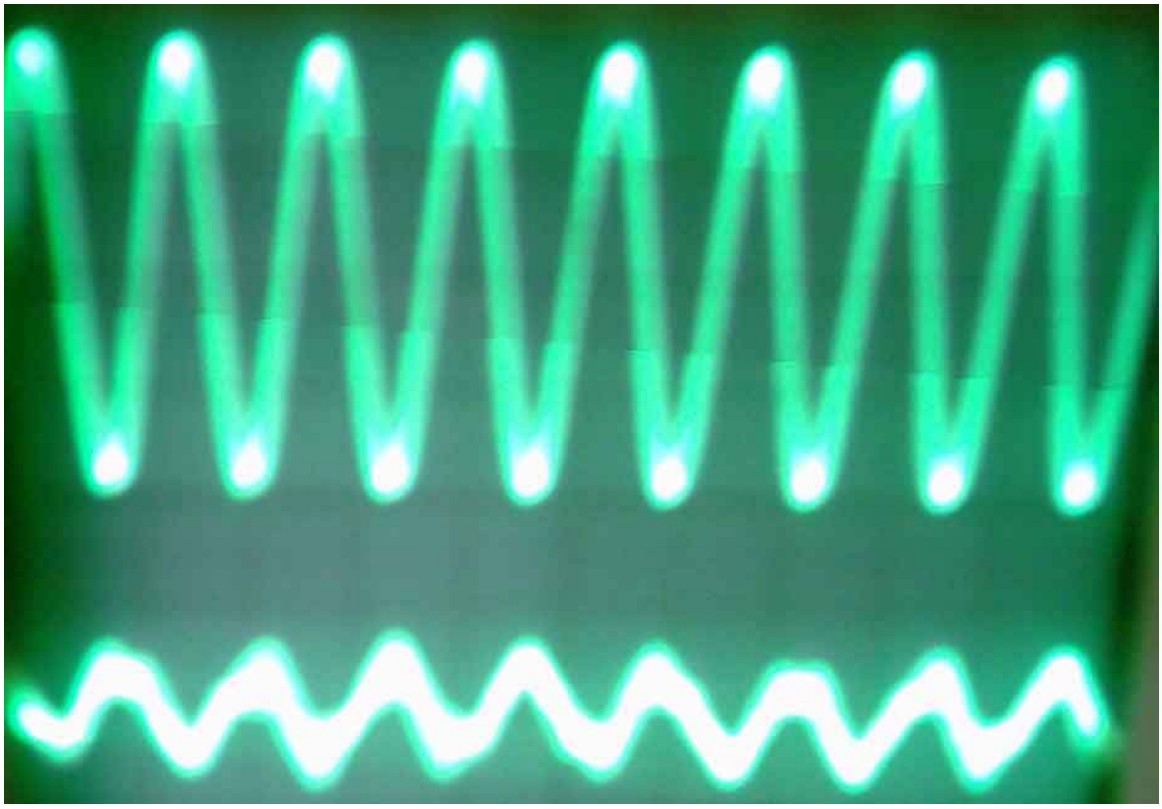

Figure 5.12d Magnitude and frequency response by 750 Hz

5.6 Experimental Results

The frequency values of the input sine wave, the value of input magnitude and the measured output magnitude values are listed in table 5.2.

Table 5.2 Input-output gain response of the filter

FREQUENCY (Hz)	**INPUT MAGNITUDE (V)**	**OUTPUT MAGNITUDE (V)**	**20*log V_o/V_i (dB)**
10	6	6	0
50	6	6	0
100	6	6	0
250	6	4.2	-3
500	6	3	-6
750	6	2	-9

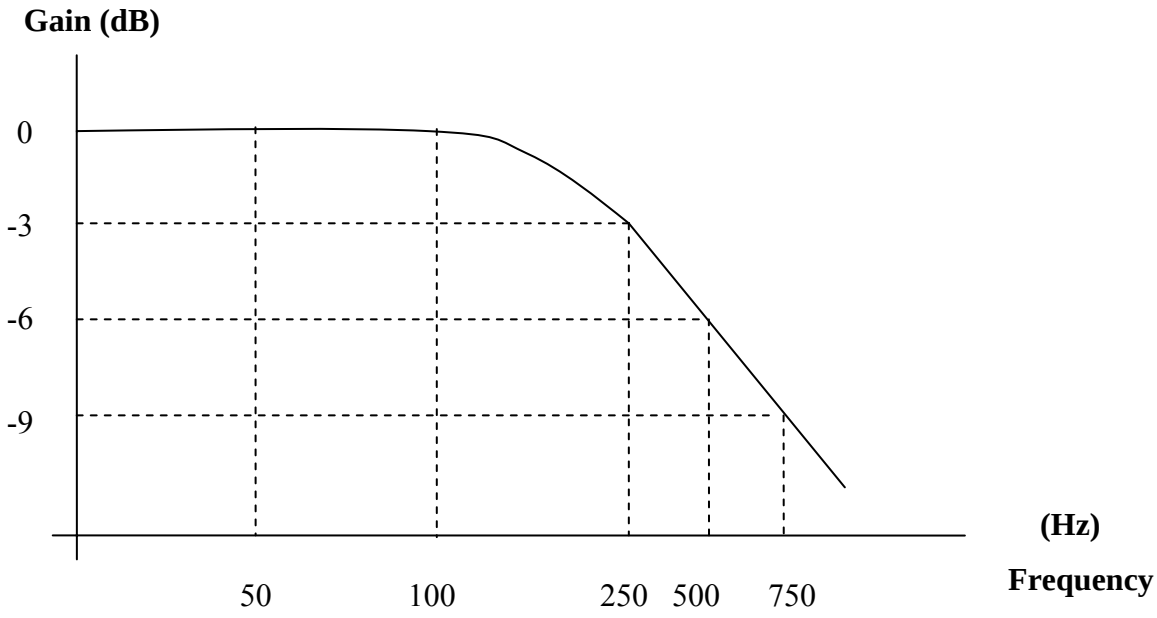

Figure 5.13 Frequency response of the developed digital filter

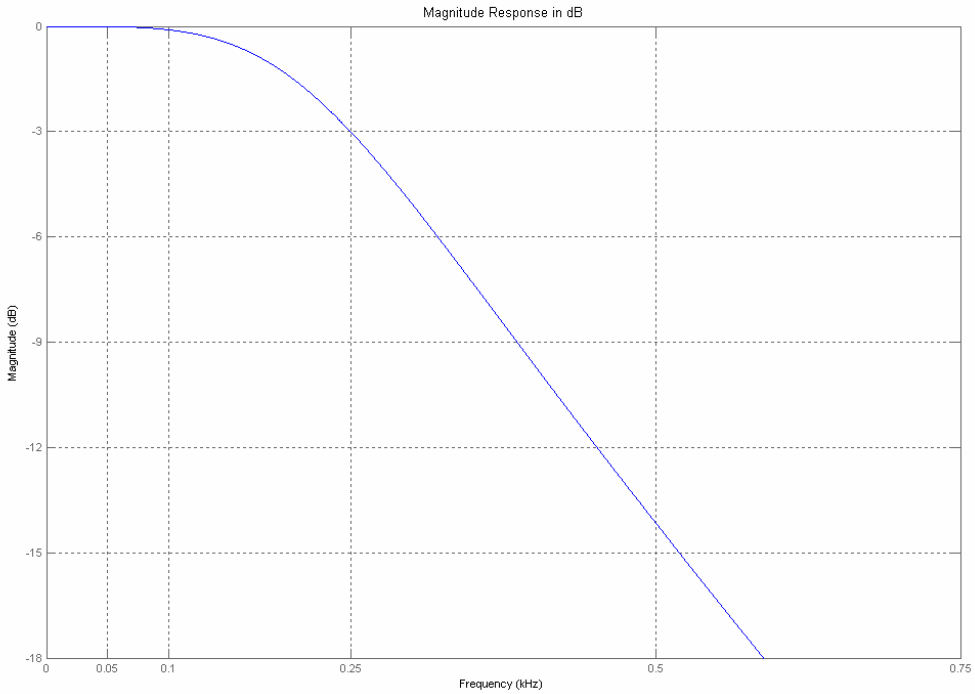

Figure 5.14 Frequency response of the digital filter using Matlab

When the frequency response plot of the digital filter obtained from microcontroller digital filter design (figure 5.13) is compared with the theoretical frequency plot obtained using Matlab GUI design (figure 5.14), it can be seen that the cut-off frequency of the hardware design matched very well with theory, the attenuation levels of the two designs occurring for frequencies higher than the cut-off frequency are, on the other hand different. Possible reasons for this are:

- Optimal standard theoretical software implementation by Matlab
- Hardware components (8-bit A/D and D/A converter) with small data widths.

5.7 Summary

This chapter has described the developed microcontroller based hardware and software for the implementation of an IIR 2nd order Butterworth digital filter. The popular PIC microcontroller has been used as the hardware. The software of the filter algorithm is based on the high-level PIC C language.

CONCLUSION

Digital filters are increasingly used in many fields such as speech analysis and processing, biomedical analysis and processing, telecommunications, etc.

This thesis focused on the development of a microcontroller based hardware and software system for the implementation of an IIR 2nd order Butterworth digital filter.

The PIC microcontroller has been used as the processing element since it is a low-cost, widely available and a popular microcontroller. The software of the filter algorithm is based on the high-level PIC C language. Digital filters can easily be realized on microcontrollers if a high-level programming language is used. High-level languages have the advantage that it is much easier to develop and maintain programs developed using these languages.

The digital filter designed and implemented behaved as expected. The filter was designed for a cut-off frequency of 250 Hz and a sampling frequency of 2.5 kHz. The results obtained were satisfactory related to the cut-off frequency compared with theoretical results. The obtained attenuation levels for frequencies higher than the cut-off frequency are different from theoretical results.

The output response of the filter could be improved if an A/D and a D/A converter with higher data widths are used. E.g. a much better response could be obtained if 16-bit converters are used.

Digital filters of higher order can easily be implemented by simply cascading the second order sections.

However, the ideas presented in this thesis can be applied to other types of digital filter. In addition, digital filters realized using PIC microcontroller can be used for a variety of applications, such as noise filtering, as well as detecting some selected frequency components present in a speech signal.

The thesis focused also on the development of a program for the design of FIR and IIR digital filters. The developed program is based on MATLAB and is GUI based, easy to use, and is user friendly. It can be used as learning tool for undergraduate students without Matlab experience

REFERENCES

[1] Mitra, S.K. Digital Signal Processing – A Computer Based Approach. McGraw-Hill, 1998.

[2] Smith S.W., The Scientist and Engineer's Guide to Digital Signal Processing. California Technical Publishing, 1999.

[3] Mamedov, F. Signals, Systems and Transformers. Nicosia: Near East University, 2002.

[4] http://www.answers.com/topic/butterworth-filter

[5] http://www.netrino.com/Publications/Glossary/

[6] A. Williams and F. Taylor, Electronic Filter Design Handbook, 3rd Edition, McGraw-Hill, 1995

[7] http://www.cs.man.ac.uk/CS3291/Notes0506/ University of Manchester. Department of Computer Science

[8] Marven, C., and Ewers G. A Simple Approach to Digital Signal Processing, Wiley, 1996.

[9] Proakis, J.G., and Manolakis D.G. Digital Signal Processing Principles, Algorithms, and Applications. New Jersy: Prentice Hall, 1996.

[10] Kaiser, J.F. Nonrecursive digital filter design using I_0-sinh window function. In Proc. 1974 IEEE International Symposium on Circuits & Systems, pages 20-23, San Francisco CA, 1974

[11] Parks, T.W., and Burrus C.S. Digital Filter Design. New York: John Wiley & Sons, 1987.

[12] Saramaki T. Finite impulse response filter design. In S. K. Mitra and J. F. Kaiser, editors, Handbook for Digital Signal Processing, chapter 4, pages 155-278. Wiley-Interscience, 1993.

[13] http://www.microchip.com/downloads/en/AppNotes/00852a.pdf

[14] Ifeachor, E.C., and Jervis B.W. Digital Signal Processing – A Practical Approach. Pearson/Prentice Hall, 2002.

[15] Oppenheim, A.V., and Willsky A.S. Signals and Systems. Englewood Cliffs NJ: Prentice-Hall, 1999.

[16] Oppenheim, A.V., and Schafer R.W. Discrete-Time Signal Processing. Upper Saddle River, NJ: Prentice-Hall, 1999.

[17] http://www.mathworks.com/access/helpdesk/help/toolbox/filterdesign/

[18] Taylor, F.J. Principles of Signals and Systems. New York: McGraw-Hill,1992.

[19] Chassaing, R. Digital Signal Processing and Applications with the C6713 and C6416 DSK. Wiley-Interscience, 2005.

[20] http://www.microchip.com/

[21] http://www.analog.com/

[22] http://www.ruhr-uni-bochum.de/dv/lehre/seminar/digi-filter/digi-filter.pdf

[23] http://www.dspguru.com/info/faqs/iirfaq.htm

[24] http://instruct1.cit.cornell.edu/courses/bionb441/GUIdesign/guiprog.html

[25] http://www.mrc-cbu.cam.ac.uk/cnbh/aimmanual/tools/parameter/

[26] http://www-sigproc.eng.cam.ac.uk/

[27] http://cnx.rice.edu/content/m10127/latest/

[28] http://www.maxim-ic.com/appnotes.cfm/appnote_number/1795