Analysis of Digital IIR Filter Using LabVIEW

A Thesis report

submitted towards the partial fulfillment of the

requirements of the degree of

Master of Engineering

in

Electronics Instrumentation and Control Engineering

Submitted by

Sweta Tripathi

Roll No-800851018

Under the supervision of

Dr. Yaduvir Singh
Associate Professor, EIED
and
Dr. Hardeep Singh
Assistant Professor, ECED

DEPARTMENT OF ELECTRICAL AND INSTRUMENTATION ENGINEERING

THAPAR UNIVERSITY

PATIALA – 147004

JULY 2010

DECLARATION

I hereby declare that the report entitled ***"Analysis of digital IIR Filter using Labview"*** *is an authentic record of my own work carried out as requirements for the award of degree of M.E. (Electronic Instrumentation & Control) at Thapar University, Patiala, under the guidance of Dr. Yaduvir Singh (Associate Professor, EIED) and Dr. Hardeep Singh (Assistant Professor, ECED) during January to July 2010.*

Sweta

Date: 06/07/2010

Sweta Tripathi

Roll No. - 800851018

It is certified that the above statement made by the candidate is correct to best of my knowledge and belief.

06/07/2010

Dr. Yaduvir Singh
Associate Professor, EIED,
(Supervisor)
Thapar University, Patiala

Dr. Hardeep Singh
Assistant Professor, ECED,
(Co-Supervisor)
Thapar University, Patiala

S. Ghosh
12/7/10

Dr. Smarajit Ghosh
Professor & Head, EIED,
Thapar University, Patiala

12.7.10

Dr. R.K. Sharma
Dean of Academic Affairs
Thapar University, Patiala

ii

ACKNOWLEDGEMENT

Words are often too less to reveals one's deep regards. An understanding of the work like this is never the outcome of the efforts of a single person. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis.

First of all I would like to thank the Supreme Power, one who has always guided me to work on the right path of the life. Without His grace this would never come to be today's reality.

This work would not have been possible without the encouragement and able guidance of my supervisor, Dr. Yaduvir Singh and co-Supervisor, Dr. Hardeep Singh. Their enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Their feedback and editorial comments were also invaluable for the writing of this thesis.

I would like to express my deep sense of gratitude towards Dr. Smarajit Ghosh, Professor and Head, EIED, Thapar University, Patiala who has been a constant source of inspiration for me throughout this work.

I would also like thank all the faculty members of the department and my friends who directly or indirectly helped me in completion of my thesis.

No words of thanks are enough for my dear parents whose support and care makes me stay on earth. Thanks to be with me.

Date: 06-07-2010
Place: Thapar University, Patiala *Sweta Tripathi*

ABSTRACT

In the recent years, LabVIEW has been used in scientific and engineering practice for a wide range of applications. For example, industrial purpose like a temperature sensor, level sensor, pressure sensor etc. Virtual Instruments (VIs) are used in LabVIEW instead of programs. LabVIEW enables easy creation of front panel user interface and interactive control of the system. Functionalities are realized by assembling block diagrams. Presently, LabVIEW is a fully featured programming language produced by National Instruments. It is a graphical language by which code is constructed and saved. There is no text-based code, but a diagrammatic view of how the data flows through the program. Thus, LabVIEW is a beloved tool of the scientists and engineers who can often visualize data flow rather than how a text based conventional programming language must be built to achieve a task. Through this dissertation, the efforts have been made to introduce the concept of filtering, compare analog and digital filters, describes finite impulse response (FIR) and infinite impulse response (IIR) filters, and how digital IIR filter can be designed using LabVIEW.

A Digital IIR Filter system is developed using National Instruments (NI) data LabVIEW software package. All the types of IIR filters like Butterworth filters, Chebyshev filters, inverse Chebyshev filters, and Elliptic filters are designed to generate their magnitude response and filter coefficients. The LabVIEW software is used to develop virtual instrument (VI) that includes a front panel and a functional diagram. The VI reads the desired parameters of the filters entered by the user on the front panel and determines its magnitude response and filter coefficients.

CONTENTS

ABBREVIATIONS

IIR	Infinite Impulse Response
FIR	Finite Impulse Response
VI	Virtual Instruments
NI	National Instruments
DAQ	Data Acquisition
G-Program	Graphical Program
HP	Highpass
LP	Lowpass
BP	Bandpass
BS	Bandstop
PB	Passband
SB	Stopband
dB	Decibels
PLC	Programmable Logic Controller
DLL	Dynamic Link Libraries
GPIB	General Purpose Interface Bus
A-D	Analog to Digital Converter
D-A	Digital to analog Converter
FFT	Fast Fourier Transform

SYMBOLS

f_c	Cutoff Frequency
x_i	Input
y_i	Output
a_i	Filter Coefficients
n	Discrete Time Index / Numbers of Feed forward Taps
τ	Time Constant
a_k	Set of Reverse Coefficients
b_j	Set of Forward Coefficients
b_k	k^{th} Feedforward Taps
N_a	Numbers of Reverse Coefficients
N_b	Numbers of Forward Coefficients
N	Order
ε	Ripple Parameters

LITERATURE SURVEY

A lot of literature is available related to this topic. Here is the literature survey that is relevant with the work carried out for this thesis work

According to S. Chen et.al. in 1991 paper **"***Minimum sensitivity IIR filter design using principal component approach"*, An IIR filter design algorithm in the state space model is presented. The methods of principal component analysis and balanced realization are employed to solve the design problem so that the designed filter in state space form achieves minimum sensitivity to parameter variation and/or round off noise. Two design examples are also presented to indicate the advantages. It was found that the new filter design which has a state space realization using a principal component approximation was more advantageous. This was computationally simple method to design a digital filter has minimum sensitivity to parameter variation and/or roundoff noise. The resulting design was always stable. Hence it is not necessary to find and modify the unstable poles of the filter. It is possible to predict the error between the desired filter and the designed filter. [7]

Later Mohammed Abo-Zahhad et.al. in 1996 presented a paper on *FILTER DESIGNER: A COMPLETE DESIGN AND SYNTHESIS PROGRAM FOR LUMPED, WAVE-DIGITAL, FIR AND IIR FILTERS* in which an interactive filter design program suitable for both experts and non-experts was described. It was written in C++ and consists of more than 15000 lines of source code. It can be used for the design and synthesis of 64 filter families which include: lumped, wave-digital (WD). FIR and IIR filters can be made where each may be lowpass. highpass, band rejection. bandpass or phase corrector. Four amplitude approximation functions were available for all these cases; namely Butterworth. Chebyshev, Inverse Chebyshev and Elliptic approximation functions. Although it operates in both batch and interactive modes, this paper dealt exclusively

with the interactive mode which was somewhat more general and very easy to use. The filter designer offers superior accuracy and flexibility in manipulating filters with different specifications and network realizations. The synthesized networks were in the form of LC ladder for lumped filters: lattice and bireciprocal for wave-digital filters: direct form for FIR filters as well as direct, cascade and parallel realizations for IIR filters. Analysis of amplitude, phase and group delay was possible for all kinds of filter designs. In addition, pole-zero patterns and network realization was available in both numerical and graphical format . In this paper several examples illustrating different design and synthesize capabilities were given.[14]

After that in 1999 a new method is proposed for designing IIR digital allpass filters with an equi-ripple phase response that can be proven to be optimal in the Chebyshev sense was proposed by Xi Zhang and Hiroshi Iwakura in his paper "*Design of IIR Digital Allpass Filters Based on Eigenvalue Problem"* The proposed procedure was based on the formulation of an eigen value problem by using the Remez exchange algorithm. Since there exists more than one eigen value in the general eigen value problem, he introduced a new and very simple selection rule for the eigen value to be searched for, where the rational interpolation was performed if and only if the real maximum eigen value was chosen. Therefore, the solution of the rational interpolation problem can be gotten by computing only one eigen vector corresponding to the real maximum eigen value, and the optimal filter coefficients are easily obtained through a few iterations without any initial guess of the solution. The design algorithm proposed in this correspondence not only retains the speed inherent in the Remez exchange algorithm but also simplifies the interpolation step because it has been reduced to the computation of the real maximum eigen value. [15]

Balbir Kumar and Ashwani Kumar in 1999 in his paper "*Design of Efficient FIR Filters for the Amplitude Response*", discussed an efficient design of a linear-phase, FIR structure yielding optimal amplitude response approximating mod 1/w for mid-band frequency range . Mathematical formulas for computation of weights were derived. These

weights turn out to be universal. Using this property, a versatile structure performing optimally for various orders was proposed. [23]

CHIA-NAN CHANG, HUI-KANG TENG, JUN-YUAN CHEN, AND HUANG-JEN CHIU of Department of Electronic Engineering,National Taiwan University of Science and Technology in his paper published in 2000, "*Computerized Conducted EMI Filter Design System Using LabVIEW and Its Application*" developed a novel computerized system for obtaining the conducted EMI measurement and systematic filter design of a switched-mode power supply . The measurement, control and filter design of the conducted EMI noises of the tested device were obtained and integrated using an automatic data acquisition system and a comprehensive virtual instrument, which was developed using the LabVIEW application software program. As an application of this system, conducted EMI noise measurement and filter design of a boost ac-dc converter with PFC (94 kHz, 100 W, 200 V) has been achieved while successfully satisfying the FCC Class A limit in the frequency range from 450 kHz to 30 MHz, which confirms the validity of the developed computerized system. This computerized system has the advantages of providing accurate measurement of different conducted EMI noises and fast determination of the associated filter corner frequencies and component values. Modification and enhancement of this computerized system are also easy due to the block diagram form of the source code in the virtual instrument. [24]

According to N. Kehtarnavaz and C. Gope in 2006 in his paper "DSP SYSTEM DESIGN USING LABVIEW AND SIMULINK:A COMPARATIVE EVALUATION" , presented a comparative evaluation between LabVIEW and Simulink in terms of a number of ease-of-use and functionality criteria. Twenty students taking a senior undergraduate DSP lab course were asked to perform the evaluation. The students' responses indicate that these two graphical environments provide more or less the same design features with LabVIEW having an edge over Simulink as far as graphical display/visualization and DSP hardware integration tools are concerned. [37]

Aimin Jiang and Hon Keung Kwan of Department of Electrical and Computer Engineering,University of Windsor in 2007 proposed a method for designing IIR digital filters with a novel stability criterion based on the argument principle in his paper, "*IIR Digital Filter Design with Novel Stability Criterion Based on Argument Principle"* Unlike the stability criteria used in some design algorithms, this stability condition is both sufficient and necessary. In the paper, the weighted leastsquares (WLS) design of IIR filters is first formulated as an iterative quadratic programming (QP) problem without any constraint. Then the stability criterion is incorporated in the quadratic form at each iteration. Two examples are presented to illustrate the effectiveness of the proposed approach. [38]

Adam Slowik and Michal Bialko of Department of Electronics and Computer Science, Technical University of Koszalin in his 2007 paper, " *Design and Optimization of IIR Digital Filters with Non-Standard Characteristics Using Particle Swarm Optimization Algorithm*" presented an application of a particle swarm optimization to the design of stable IIR digital filters with nonstandard amplitude characteristics. Particle swarm algorithms are newly elaborated technique for optimization of multi-modal functions, as for example functions describing the problem of calculation of an optimal set of transfer function coefficients in which the gradient algorithms can easily stick at local extreme. Design of IIR digital filters with non-standard amplitude characteristics which considerably differ from typical Butterworth, Chebyshev, and Cauer approximations, is possible using presented method. The IIR digital filter with linearly falling amplitude characteristics is designed with the use of proposed method. The filter is stable and fulfill all prescribed design assumptions. [39]

CHAPTER 1

INTRODUCTION

This Dissertation work involves the use of LabVIEW (Laboratory Virtual Instrument Engineering Workbench) software by National Instrument's to design a Digital IIR Filter. The Digital Filter Design problem involves the determination of a set of filter coefficients to meet a set of design specifications. These specifications typically consist of the width of the passband and the corresponding gain, the width of the stopband(s) and the attenuation therein; the band edge frequencies (which give an indication of the transition band) and the peak ripple tolerable in the passband and stopband(s) [34].

The LabVIEW based digital filter system involves the concept of Virtual Instrumentation. A virtual instrumentation system is computer software that a user would employ to develop a computerized test and measurement system, for controlling from a computer desktop an external measurement hardware device, and for displaying test or measurement data collected by the external device on instrument-like panels on a computer screen. Virtual instrumentation extends also to computerized systems for controlling processes based on data collected and processed by a computerized instrumentation system. Because their functionality is software defined by the user, virtual instruments are extremely flexible, powerful and cost effective. LabVIEW programs are called virtual instruments (VI) because their appearance and operation imitate actual instruments. However, behind the scene these are analogous to main programs, functions and subroutines from other languages [18]. LabVIEW contains application-specific libraries of code for data acquisition (DAQ), general purpose interface bus (GPIB) and serial instrument control, data analysis, data presentation and data storage. The analysis library contains a multitude if useful functions including signal generation, signal processing, filter windows, static regression, linear algebra, and array arithmetic. Because of LabVIEW's graphical nature, it is inherently a data presentation package. Output appears in the desired form. Charts, graphs and user defined graphics comprise just a fraction of available output options [16]. The LabVIEW program

development environment is different from others. While other programming system use text based languages to create line of codes, LabVIEW uses a graphical program called "G". The developed LabVIEW program called VI would perform the various tasks like generation of appropriate control signals, measurement and interpretation of data.

A VI has two main parts, the front panel and the block diagram. The front panel is the interactive user interface of a VI, so named because it simulates the front panel of a physical instrument. The front panel contains knobs, push buttons, graphs and many other controls which are user inputs and indicators which are program outputs. The block diagram is the VI's source code, constructed in LabVIEW's graphical language. The pictorial block diagram which corresponds to the source code constructed in LabVIEW's graphical programming language G is the actual executable program [16]. The front panel allows the user to enter input such as the cutoff frequencies, attenuations, and filter types etc. and to display an output such as the magnitude response, filter coefficients and the pole - zero plot of filter. When any signal is applied to the filter, the filter specifications are continuously read by the VI via the DAQ's analog input port.

Two types of digital filters exist – the IIR (Infinite Impulse Response) and the FIR (Finite Impulse Response). IIR filter possess certain properties, which make them the preferred design choices in numerous situations over FIR filters. Most notably, FIR filters (all zero system function) are always stable, with a realization existing for each FIR filter. Another feature exclusive to FIR filters is that of a linear phase response [13].

The content of this project is the design of IIR filters using LabVIEW. The design of IIR filters proceeds through a vastly different set of steps than those followed by FIR filter design algorithms. The design of IIR filters is closely related to the design of analog filters, which is a widely studied topic. An analog filter is usually designed and a transformation is carried out into the digital domain. Two transformations exist – the **impulse invariant** transformation and the **bilinear** transformation. In this project, the focus is on designing minimum order IIR filters to meet a set of specifications using

LabVIEW functions. Each design is accompanied by a plot of its frequency response, impulse response and pole-zero diagrams.

The responses of IIR filters using LabVIEW are compared with the responses from MATLAB with the same specifications. The main goal of this work is to obtain an optimized filter response along with the filter coefficients.

Organization of Work

Organization of the work is as follows:

- A brief introduction is presented in **chapter 1.**
- **Chapter 2** deals with the basic concepts of Filters. Advantages of digital filters over analog filters are discussed in details. The various types of filters on the basis of frequency characteristics and parameters are discussed. It also covers the comparative study of IIR and FIR filters.
- **Chapter 3** presents the basic idea of IIR filters. Here two approximation schemes of IIR filter design for all the types (LP, BP etc.) are discussed in detail.
- **Chapter 4** presents the concept of Virtual Instrumentation and their advantages over traditional instruments.
- **Chapter 5** deals with the implementation of VI concept using NI software called LabVIEW.
- **Chapter 6** presents the design of IIR filter using LabVIEW. The simulation study of IIR fitters and design methods are illustrated.
- **Chapter 7** presents results of simulation and a comparative study of responses with MATLAB responses.
- **Chapter 8** concludes the work and gives future scope of the work.

CHAPTER 2

DIGITAL FILTERING

2.1 INTRODUCTION

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 filtering process alters the frequency content of a signal. For example, the bass control on a stereo system alters the low-frequency content of a signal, while the treble control alters the high-frequency content . Two common filtering applications are removing noise and decimation. Decimation consists of lowpass filtering and reducing the sample rate.

The following block diagram illustrates the basic idea.

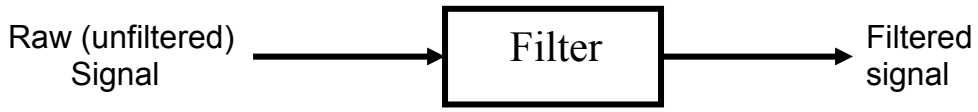


Figure 2.1 Block diagram of filter

2.2 Types of Filters

There are two main kinds of filter, analog and digital.

1. Analog Filters
2. Digital Filters

2.2.1 Analog Filter

An analog filter has an analog signal at both its input $x(t)$ and its output $y(t)$. Both $x(t)$ and $y(t)$ are functions of a continuous variable *t* and can have an infinite number of values. 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. 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 [5, 35].

2.2.2 Digital Filter

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 specialised DSP (Digital Signal Processor) chip. Digital filters are used in a wide variety of signal processing applications, such as spectrum analysis, digital image processing, and pattern recognition. Digital filters eliminate a number of problems associated with their classical analog counterparts and thus are preferably used in place of analog filters. The analog input signal must first be sampled and digitised using an ADC (analog to digital 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.

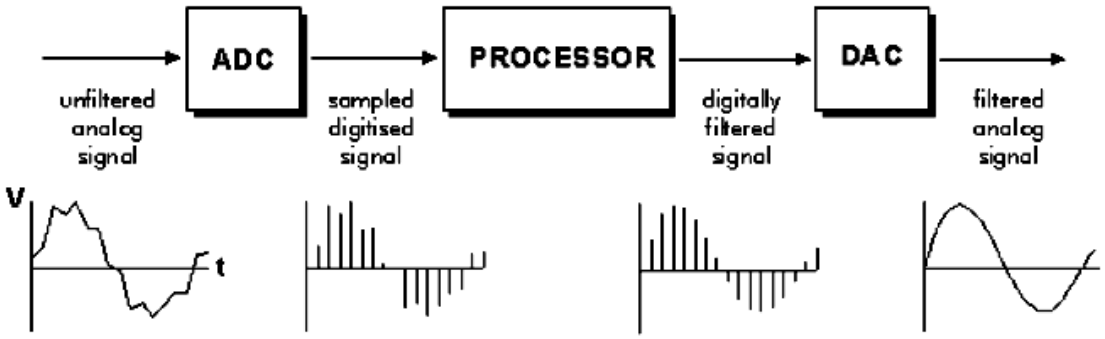


Figure 2.2 Block diagram of a digital filter

2.2.3 Advantages of Using Digital Filters

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

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 to both 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 [5, 6].

2.3 Characteristics of an Ideal Filter

Ideal filters allow a specified frequency range of interest to pass through while attenuating a specified unwanted frequency range. The filters are classified according to their frequency range characteristics. The following filter classifications are based on the frequency range a filter passes or blocks:

- Lowpass filters pass low frequencies and attenuate high frequencies.
- Highpass filters pass high frequencies and attenuate low frequencies.
- Bandpass filters pass a certain band of frequencies.
- Bandstop filters attenuate a certain band of frequencies.

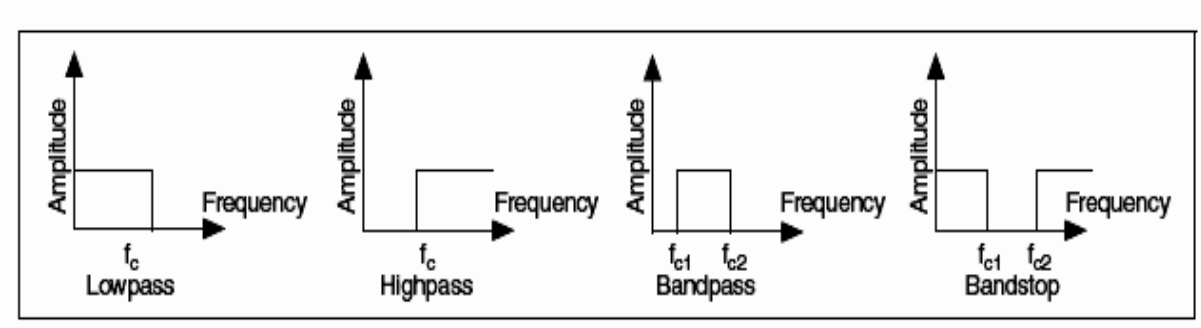


Figure 2.3 Ideal frequency characteristics

In Figure 2.3, the filters exhibit the following behavior:

• The lowpass filter passes all frequencies below *fc*.

• The highpass filter passes all frequencies above *fc*.

• The bandpass filter passes all frequencies between *fc*1 and *fc*2.

• The bandstop filter attenuates all frequencies between *fc*1 and *fc*2.

The frequency points *fc*, *fc*1, and *fc*2 specify the cut-off frequencies for the different filters. When designing filters, you must specify the cut-off frequencies. The passband of the filter is the frequency range that passes through the filter. An ideal filter has a gain of one (0 dB) in the passband so the amplitude of the signal neither increases nor decreases. The stopband of the filter is the range of frequencies that the filter attenuates. Figure 2.4 shows the passband (PB) and the stopband (SB) for each filter type .

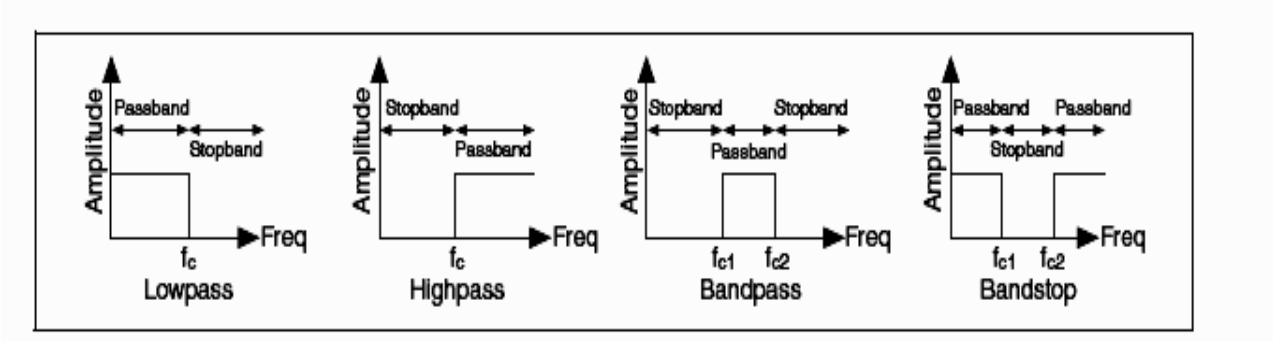


Figure 2.4 Passband and Stopband

2.4 Practical (Non ideal) Filters

In practical applications, ideal filters are not realizable. Ideally, a filter has a unit gain (0 dB) in the passband and a gain of zero (–∞ dB) in the stopband. However, real filters

cannot fulfill all the criteria of an ideal filter. In practice, a finite transition band always exists between the passband and the stopband. In the transition band, the gain of the filter changes gradually from one (0 dB) in the passband to zero (–∞ dB) in the stopband.

2.4.1Transition Band

Figure 2.5 shows the passband, the stopband, and the transition band for each type of practical filter. In each plot in Figure 2.5, the *x*-axis represents frequency, and the *y*-axis represents the magnitude of the filter in dB. The passband is the region within which the gain of the filter varies from 0 dB to –3 dB.

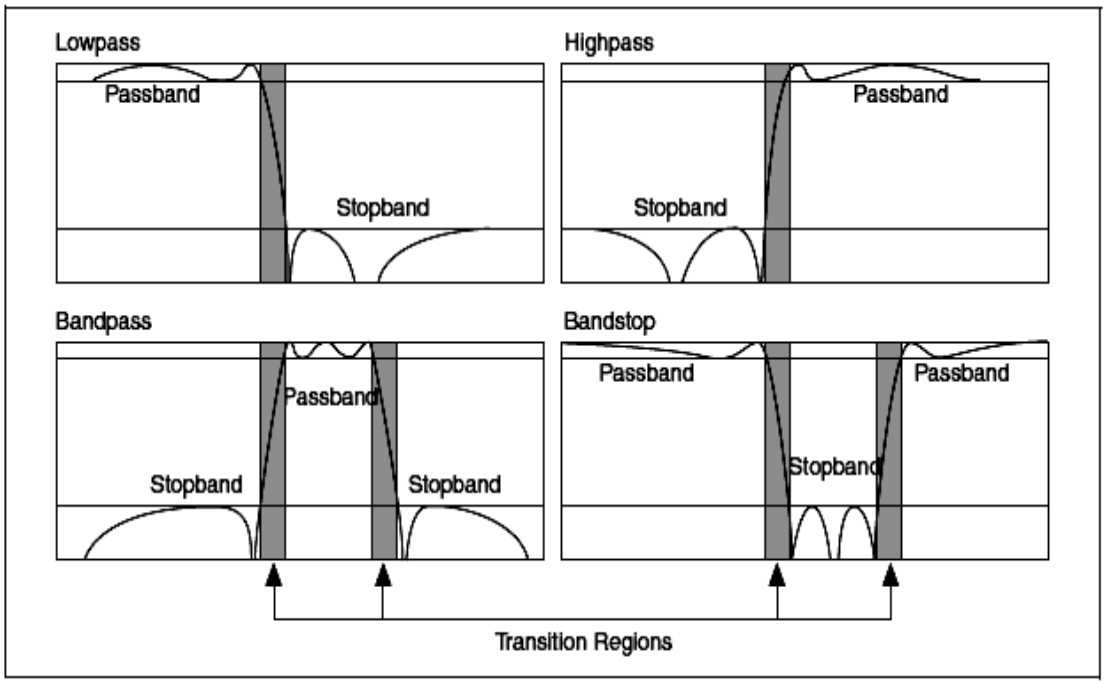


Figure 2.5 Response of nonideal filters

2.4.2 Passband Ripple and Stopband Attenuation

In many applications, you can allow the gain in the passband to vary slightly from unity. This variation in the passband is the passband ripple, or the difference between the actual gain and the desired gain of unity. In practice, the stopband attenuation cannot be infinite, and you must specify a value with which you are satisfied. Measure both the passband ripple and the stopband attenuation in decibels (dB).

2.4.3 Sampling Rate

The sampling rate is important to the success of a filtering operation. The maximum frequency component of the signal of interest usually determines the sampling rate. In general, choose a sampling rate 10 times higher than the highest frequency component of the signal of interest .

2.4.4 Digital filter coefficients

All of the digital filter examples given above can be written in the following general forms:

Zero order: $y_n = a_0 x_n$

First order: $y_n = a_0 x_n + a_1 x_{n-1}$

Second order: $y_n = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2}$

Similar expressions can be developed for filters of any order.

The constants $a_0, a_1,$ and a_2 appearing in these expressions are called the filter coefficients.

It is the values of these coefficients that determine the characteristics of a particular filter.

2.5 Common Digital Filters

Traditional filter classification begins with classifying a filter according to its impulse response. These terms refer to the differing "impulse responses" of the two types of filter. Digital filter can be classified as one of the following types:

• Finite impulse response (FIR) filter, also known as non-recursive filters (in a non-recursive filter the current output is calculated solely from the current and previous input values).

• Infinite impulse response (IIR) filter, also known as recursive filter (a recursive filter is one which in addition to input values also uses previous output values).

The impulse response of a digital filter is the output sequence from the filter when a unit impulse is applied at its input.

2.5.1 Impulse Response

An impulse is a short duration signal that goes from zero to a maximum value and back to zero again in a short time. Equation 3-1 provides the mathematical definition of an impulse. The impulse response of a filter is the response of the filter to an impulse and

$$x_0 = 1$$
$$x_I = 0 \qquad for\ all\ I \neq 0 \tag{3.1}$$

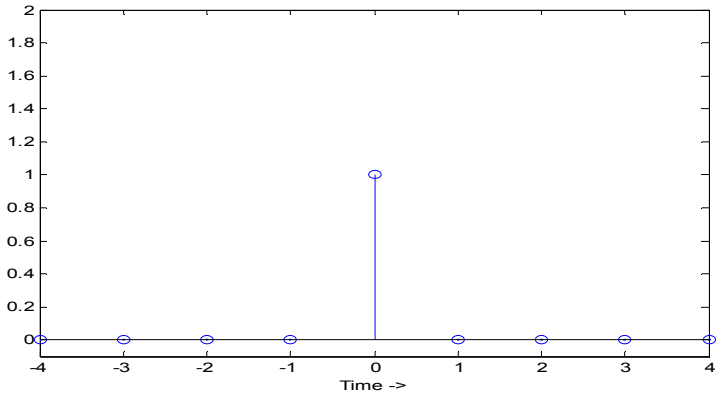


Figure 2.6 Impulse response

depends on the values upon which the filter operates. The Fourier transform of the impulse response is the frequency response of the filter. The frequency response of a filter provides information about the output of the filter at different frequencies. In other words, the frequency response of a filter reflects the gain of the filter at different frequencies. For an ideal filter, the gain is one in the passband and zero in the stopband. An ideal filter passes all frequencies in the passband to the output unchanged but passes none of the frequencies in the stopband to the output.

2.5.2 FIR Filters

Finite impulse response (FIR) filters are digital filters that have a finite impulse response. FIR filters operate only on current and past input values and are the simplest filters to design. FIR filters also are known as nonrecursive filters.

This can be stated mathematically as

$$h(n) = \{0, \quad n \leq \tau_1 \qquad -\infty < \tau_1 < \tau_2 < +\infty$$
$$\{0, \quad n \geq \tau_2 \tag{3.2}$$

where h(n) denotes the impulse response of the digital filter, n is the discrete time index, and τ_1 and τ_2 are constants. A difference equation is the discrete time equivalent of a continuous time differential equation. The general difference equation for a FIR digital filter is

$$y(n) = \sum_{K=0}^{n-1} b_K \; x(n-k) \qquad \textbf{(3.3)}$$

where y(n) is the filter output at discrete time instance n, bk is the k-th feedforward tap, or filter coefficient, and x(nk) is the filter input delayed by k samples. The Σ denotes summation from k = 0 to k = n-1 where n is the number of feedforward taps in the FIR filter. The FIR filter output depends only on the previous n inputs. FIR filters are the simplest filters to design. If a single impulse is present at the input of an FIR filter and all subsequent inputs are zero, the output of an FIR filter becomes zero after a finite time. Therefore, FIR filters are finite. The time required for the filter output to reach zero equals the number of filter coefficients [35]. Equation 3-3 describes the behavior of the filter only in terms of the current and past values of input. So FIR filters are also known as a nonrecursive filters.

2.5.3 IIR Filters

Infinite impulse response (IIR) filters, also known as recursive filters operate on current and past input values and current and past output values. Theoretically, the impulse response of an IIR filter never reaches zero and is an infinite response. A recursive filter is one which in addition to input values also uses previous output values [5, 35]. The expression for a recursive filter therefore contains not only terms involving the input values $\left(x_n, x_{n-1}, x_{n-2}, \cdots\right)$ but also terms involving the past output values $y_{n-1}, y_{n-2}, \cdots$.
The following general difference equation characterizes IIR filters

$$y_i = \frac{1}{a_0}\left(\sum_{j=0}^{N_b-1} b_j x_{i-j} - \sum_{k=1}^{N_a-1} a_k y_{i-k} \right) \qquad \textbf{(3.4)}$$

where b_j is the set of forward coefficients, N_b is the number of forward coefficients, a_k is the set of reverse coefficients, and N_a is the number of reverse coefficients. Where x_i is the current input, x_{i-j} is the past inputs, and y_{i-k} is the past outputs.

From this explanation, recursive filters require more calculations to be performed, since there are previous output terms in the filter expression as well as input terms. In fact, the reverse is usually the case: to achieve a given frequency response characteristic using a recursive filter generally requires a much lower order filter (and therefore fewer terms to be evaluated by the processor) than the equivalent nonrecursive filter. IIR filters might have ripple in the passband, the stopband, or both. IIR filters have a nonlinear-phase response.

2.5.4 Comparing FIR and IIR Filters

Because designing digital filters involves making compromises to emphasize a desirable filter characteristic over a less desirable characteristic, comparing FIR and IIR filters can help in selecting the appropriate filter design for a particular application.

IIR filters have the advantages of providing the higher selectivity for a particular order. IIR filters can achieve the same level of attenuation as FIR filters but with far fewer coefficients. Therefore, an IIR filter can provide a significantly faster and more efficient filtering operation than an FIR filter. FIR filters provide a linear-phase response [38]. IIR filters provide a nonlinear-phase response. FIR filters are used for applications that require linear-phase responses like high quality audio systems. IIR filters are used for applications that do not require phase information, such as signal monitoring applications. Compared to IIR filters, FIR filters sometimes have the disadvantage that they require more memory and/or calculation to achieve a given filter response characteristic. Also, certain responses are not practical to implement with FIR filters. FIR filters are always stable because they are implemented using an all-zero transfer function. Since no poles can fall outside the unit circle, the filter will always be stable [15]. But because of this, the order of FIR filter is much higher than the IIR filter which has the comparable magnitude response. The higher order of the FIR filters lead to longer processing times and larger memory requirements.

CHAPTER 3

IIR DIGITAL FILTERS

Digital Filters are designed by using the values of both the past outputs and the present input, an operation brought about by convolution. If such a filter is subjected to an impulse then its output need not necessarily become zero. The impulse response of such a filter can be infinite in duration. Such a filter is called an Infinite Impulse Response filter or IIR filter [13]. The infinite impulse response of such a filter implies the ability of the filter to have an infinite impulse response. This indicates that the system is prone to feedback and instability.

The report studies several different types of IIR filters including the Butterworth Filter, Chebyschev I & II Filters and Elliptic Low, High and Bandpass filters. IIR filters are designed essentially by the Impulse Invariance or the Bilinear Transformation method.

3.1 Impulse Invariance

This procedure involves choosing the response of the digital filter as an equi-spaced sampled version of the analog filter.

1. Decide upon the desired frequency response
2. Design an appropriate analogue filter
3. Calculate the impulse response of this analogue filter
4. Sample the analogue filter's impulse response
5. Use the result as the filter coefficients

The impulse invariance method maps the left hand portion of the s-plane into the interior of the unit circle and the right hand portion of the s-plane to the exterior of the unit circle; hence each horizontal strip in the s-plane is overlayed onto the z-plane to form the digital system function from the analog system function. This is shown in Figure 3.1.

Since any practical analog filter can never be bandlimited interference is a major consideration. Due to the aliasing that arises in the sampling process the distortion in the frequency response is one of the major limiting factors of this implementation while its advantage lies in the fact that there is a linear relationship between the analog and digital

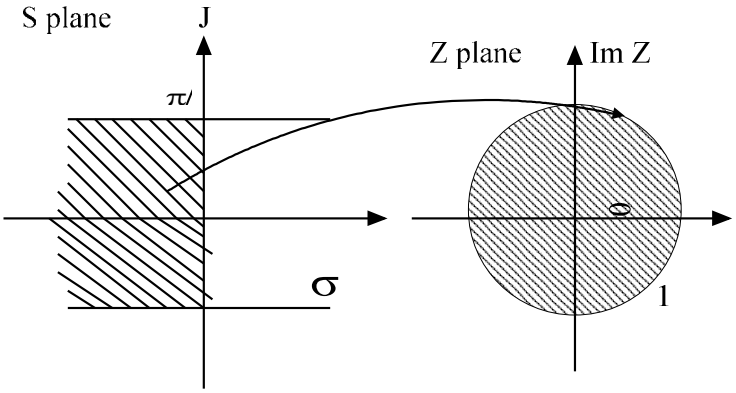


Figure 3.1 Comparison in s-plane and z-plane using impulse invariance method

frequency response. Hence in order to prevent sever distortion due to the band limiting this method is restricted to the design of Low and Bandpass Filters [35, 15].

3.2 Bilinear Transformation

There is another approximation technique of IIR digital filters from analog filters. It is called **bilinear transformation Method**. This method is one of the best currently available methods for designing IIR digital filters due to simplicity and similarity of the frequency response of IIR digital filters to that of reference analog filters. The Bilinear Transformation method overcomes the effect of aliasing that is caused due to the analog frequency response containing components at or beyond the Nyquist Frequency. The bilinear transform is a method of compressing the infinite, straight analogue frequency axis to a finite one long enough to wrap around the unit circle once only. This is also sometimes called frequency warping. This introduces a distortion in the frequency. This is undone by pre-warping the critical frequencies of the analog filter (cutoff frequency, center frequency) such that when the analog filter is transformed into the digital filter, the designed digital filter will meet the desired specifications.

Bilinear Transformation:

Consider an analog filter:

$$\mathrm{H}(s) = \frac{b}{s + a} \qquad \textbf{(3.1)}$$

This system can be characterized by a differential equation

$$\frac{d}{dt} y(t) + ay(t) = bx(t) \tag{3.2}$$

Suppose we approximate the integral rather than the derivative

$$y(t) = \int_{t_o} y'\tau \; d\tau + y(t_0) \tag{3.3}$$

We can approximate the integral by using the Trapezoidal formula

$$y(nT) = \frac{T}{2}[y'(nT) + y'(nT-T)] + y(nT-T) \tag{3.4}$$

From the differential equation we can substitute for y(t)

$$y'(nT) = -ay(nT) + bx(nT) \tag{3.5}$$

We can substitute this in the trapezoidal rule and write

$$\left(1+\frac{aT}{2}\right)y(n) - \left(1-\frac{aT}{2}\right)y(n-1) = \frac{bT}{2}[x(n) + x(n-1)] \tag{3.6}$$

The Z-transform of this gives:

$$\left(1+\frac{aT}{2}\right)Y(z) - \left(1-\frac{aT}{2}\right)z^{-1}Y(z) = \frac{bT}{2}\left(1+z^{-1}\right)X(z) \tag{3.7}$$

Which is simplified to

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b}{\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right) + a} \tag{3.8}$$

Clearly the mapping is as follows

$$H(z) = H(s)\Big|_{s=\frac{2}{T}\left(\frac{1-z^{-1}}{1+z^{-1}}\right)} \tag{3.9}$$

This mapping is known as bilinear transformation.

By solving this equation for y we obtain

$$s \Leftrightarrow \frac{2}{T}\left(\frac{z-1}{z+1}\right) \tag{3.10}$$

This transformation is known as the Bilinear or Tustin Transformation. The Laplace transforms in the filter expressions are replaced by the corresponding z-transforms.

Replacing s = σ + j Ω and performing algebraic manipulations, substituting $z = e^{jw}$ we get

$$\omega = 2 \tan^{-1}(\Omega T/2) \qquad \textbf{(3.11)}$$

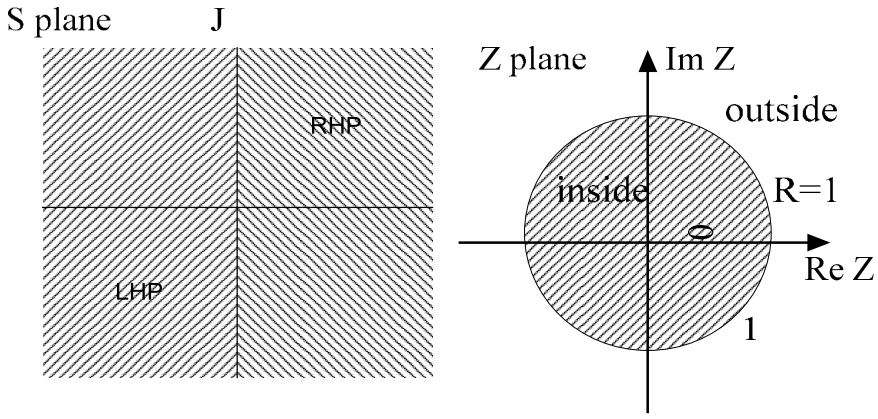


Figure 3.2 Comparison in s-plane and z-plane using bilinear transformation method

It can be seen that analog dc (s = 0) maps to digital dc (z = 1) and the highest analog frequency (s = ∞) maps to the highest digital frequency (z = -1). It is easy to show that the entire jw axis in the s plane is mapped exactly once around the unit circle in the z plane. Therefore, it does not alias.

With (2/T) real and positive, the left-half s plane maps to the interior of the unit circle, and the right-half s plane maps outside the unit circle. The constant provides one remaining degree of freedom that can be used to map any particular finite frequency the jw axis in the s plane to a particular desired location on the unit circle e^{jw} in the z plane. All other frequencies will be warped. In particular, approaching half the sampling rate, the frequency axis compresses more and more. Filters having a single transition frequency, such as lowpass or highpass filters, map beautifully under the bilinear transform; you simply map the cut-off frequency where it belongs, and the response looks great. In particular, ``equal ripple'' is preserved for optimal filters of the elliptic and

Chebyshev types because the values taken on by the frequency response are identical in both cases; only the frequency axis is warped [13, 35, 15].

3.3 Transfer function of an IIR Filter

Equation 3.12 defines the direct-form transfer function of an IIR filter.

$$H(z)=\frac{b_0+b_1z^{-1}+.....+b_{N_b-1}z^{-(N_b-1)}}{1+a_1z^{-1}+.....+a_{N_a-1}z^{-(N_a-1)}} \qquad \textbf{(3.12)}$$

A filter implemented by directly using the structure defined by Equation 3.12.Where a_n and b_n are the reverse and forward coefficients of the IIR filter.

It can be written in the form of general difference equation as follows

$$y_i=\frac{1}{a_0}\left(\sum_{j=0}^{N_b-1}b_j x_{i-j}-\sum_{k=0}^{N_a-1}a_k y_{i-k}\right) \qquad \textbf{(3.13)}$$

where b_j is the set of forward coefficients, N_b is the number of forward coefficients, a_k is the set of reverse coefficients, and N_a is the number of reverse coefficients.

Equation 3.13 describes a filter with an impulse response of theoretically infinite length for nonzero coefficients. However, in practical filter applications, the impulse response of a stable IIR filter decays to near zero in a finite number of samples.

In most IIR filter designs and all of the LabVIEW IIR filters, coefficient a_0 is 1. The output sample at the current sample index *i* is the sum of scaled current and past inputs and scaled past outputs, as shown by Equation 3.14

$$y_i=\left(\sum_{j=0}^{N_b-1}b_j x_{i-j}-\sum_{k=0}^{N_a-1}a_k y_{i-k}\right) \qquad \textbf{(3.14)}$$

where x_i is the current input, x_{i-j} is the past inputs, and y_{i-k} is the past outputs. IIR filters might have ripple in the passband, the stopband, or both.

3.4 IIR Filter Types

Digital IIR filter designs come from the classical analog designs and include the following filter types:

• Butterworth filters

• Chebyshev filters

• Chebyshev II filters, also known as inverse Chebyshev and Type II Chebyshev filters

• Elliptic filters, also known as Cauer filters

The IIR filter designs differ in the sharpness of the transition between the passband and the stopband and where they exhibit their various characteristics—in the passband or the stopband.

3.4.1Butterworth Filters

Butterworth filters have the following characteristics:

• Smooth response at all frequencies

• Monotonic decrease from the specified cut-off frequencies

• Maximal flatness, with the ideal response of unity in the passband and zero in the stopband

• Half-power frequency, or 3 dB down frequency, that corresponds to the specified cut-off frequencies.

The transfer function for Butterworth filter is given by

$$B(\omega) = \frac{1}{\left[1 + \left(\frac{\omega}{\omega_0}\right)^{2n}\right]^{1/2}} \qquad \textbf{(3.13)}$$

Where n is the order of filter.

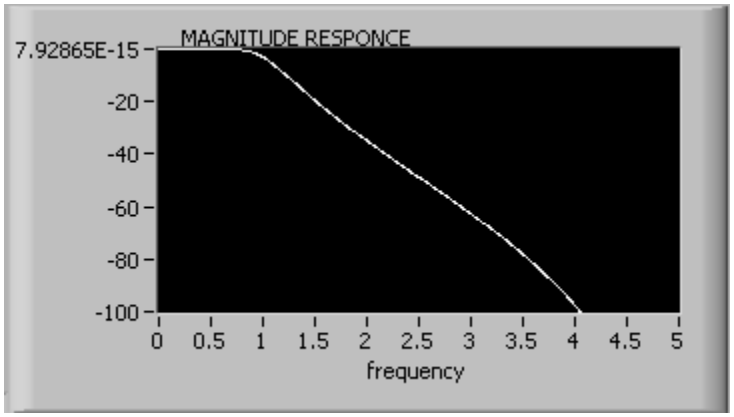


Figure 3.3 Response of Butterworth filter

As shown in Figure 3.3, after specifying the cut-off frequency of a Butterworth filter, LabVIEW sets the steepness of the transition proportional to the filter order. Higher order Butterworth filters approach the ideal lowpass filter response.

Butterworth filters do not always provide a good approximation of the ideal filter response because of the slow rolloff between the passband and the stopband.

3.4.2 Chebyshev Filters

Chebyshev filters have the following characteristics:

• Minimization of peak error in the passband

• Equiripple magnitude response in the passband

• Monotonically decreasing magnitude response in the stopband

• Sharper rolloff than Butterworth filters

Compared to a Butterworth filter, a Chebyshev filter can achieve a sharper transition between the passband and the stopband with a lower order filter. The sharp transition between the passband and the stopband of a Chebyshev filter produces smaller absolute errors and faster execution speeds than a Butterworth filter.

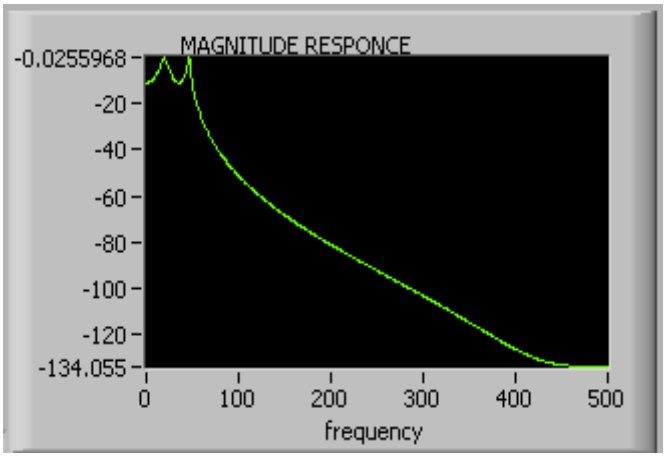


Figure 3.4 Response of Chebyshev filter

Figure 3.4 shows the frequency response of a lowpass Chebyshev filter. In Figure 3.4, the maximum tolerable error constrains the equiripple response in the passband. Also, the sharp rolloff appears in the stopband. The frequency response of the filter is given by

$$\left|H(\Omega)\right|^2 = \left(1+\varepsilon^2 \mathrm{T_N}^2\left(\Omega/\Omega_\mathrm{P}\right)\right)^{-1} \tag{3.14}$$

Where ε is a parameter of the filter related to ripple present in the passband and $\mathrm{T_N}(\mathrm{x})$ is the Nth- order Chebyshev polynomial defined as

$$\begin{aligned} T_\mathrm{N} &= \cos(N\cos^{-1}x) && |x| \le 1 \\ &\quad \cos(N\cosh^{-1}x) && |x| \ge 1 \end{aligned} \tag{3.15}$$

3.4.3 Chebyshev II filters or Inverse chebyshev filters

Chebyshev II filters have the following characteristics:

• Minimization of peak error in the stopband

• Equiripple magnitude response in the stopband

• Monotonically decreasing magnitude response in the passband

• Sharper rolloff than Butterworth filters

Chebyshev II filters are similar to Chebyshev filters. However, Chebyshev II filters differ from Chebyshev filters in the following ways:

• Chebyshev II filters minimize peak error in the stopband instead of the passband. Minimizing peak error in the stopband instead of the passband is an advantage of Chebyshev II filters over Chebyshev filters.

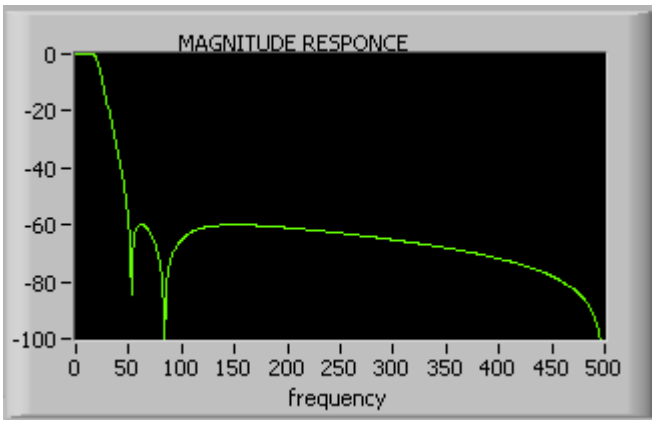


Figure 3.5 Response of Inverse Chebyshev filter

• Chebyshev II filters have an equiripple magnitude response in the stopband instead of the passband.

• Chebyshev II filters have a monotonically decreasing magnitude response in the passband instead of the stopband

In Figure 3.5, the maximum tolerable error constrains the equiripple response in the stopband. Also, the smooth monotonic rolloff appears in the stopband. Chebyshev II filters have the same advantage over Butterworth filters that Chebyshev filters have—a sharper transition between the passband and the stopband with a lower order filter, resulting in a smaller absolute error and faster execution speed.

3.4.4 Elliptic Filters

Elliptic filters have the following characteristics:

• Minimization of peak error in the passband and the stopband

• Equiripples in the passband and the stopband

Compared with the same order Butterworth or Chebyshev filters, the elliptic filters provide the sharpest transition between the passband and the stopband, which accounts for their widespread use.

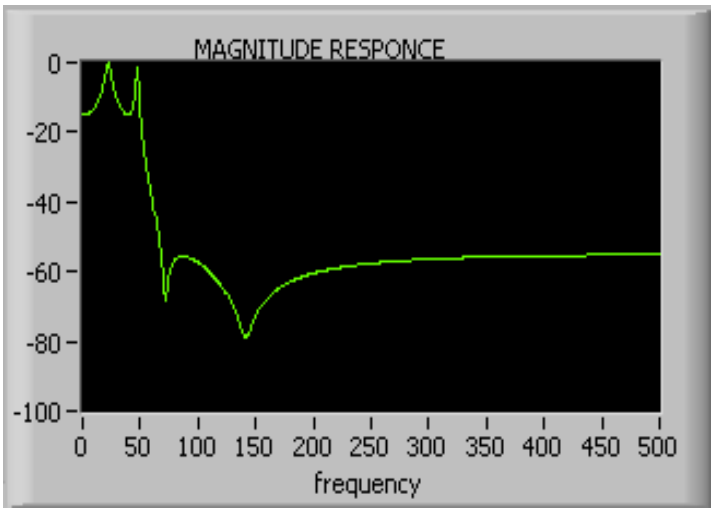


Figure 3.6 Response of Elliptic filter

In Figure 3.6, the same maximum tolerable error constrains the ripple in both the passband and the stopband. Also, even low-order elliptic filters have a sharp transition edge. The transfer function is given by

$$\left|H(\Omega)\right|^2 = \left(1+\varepsilon^2 U_N\left(\Omega/\Omega_c\right)\right)^{-1} \qquad \textbf{(3.16)}$$

where $U_N(x)$ is the Jacobian elliptic function of order N and ε is a constant related to passband ripple. They provide a realization with the lowest order for a particular set of conditions.

3.5 Selecting a Digital Filter Design

Digital filters are selected according to the following application:

• Does the analysis require a linear-phase response?

• Can the analysis tolerate ripples?

• Does the analysis require a narrow transition band?

Figure 3.7 works as a guideline for selecting the appropriate filter for an analysis application.

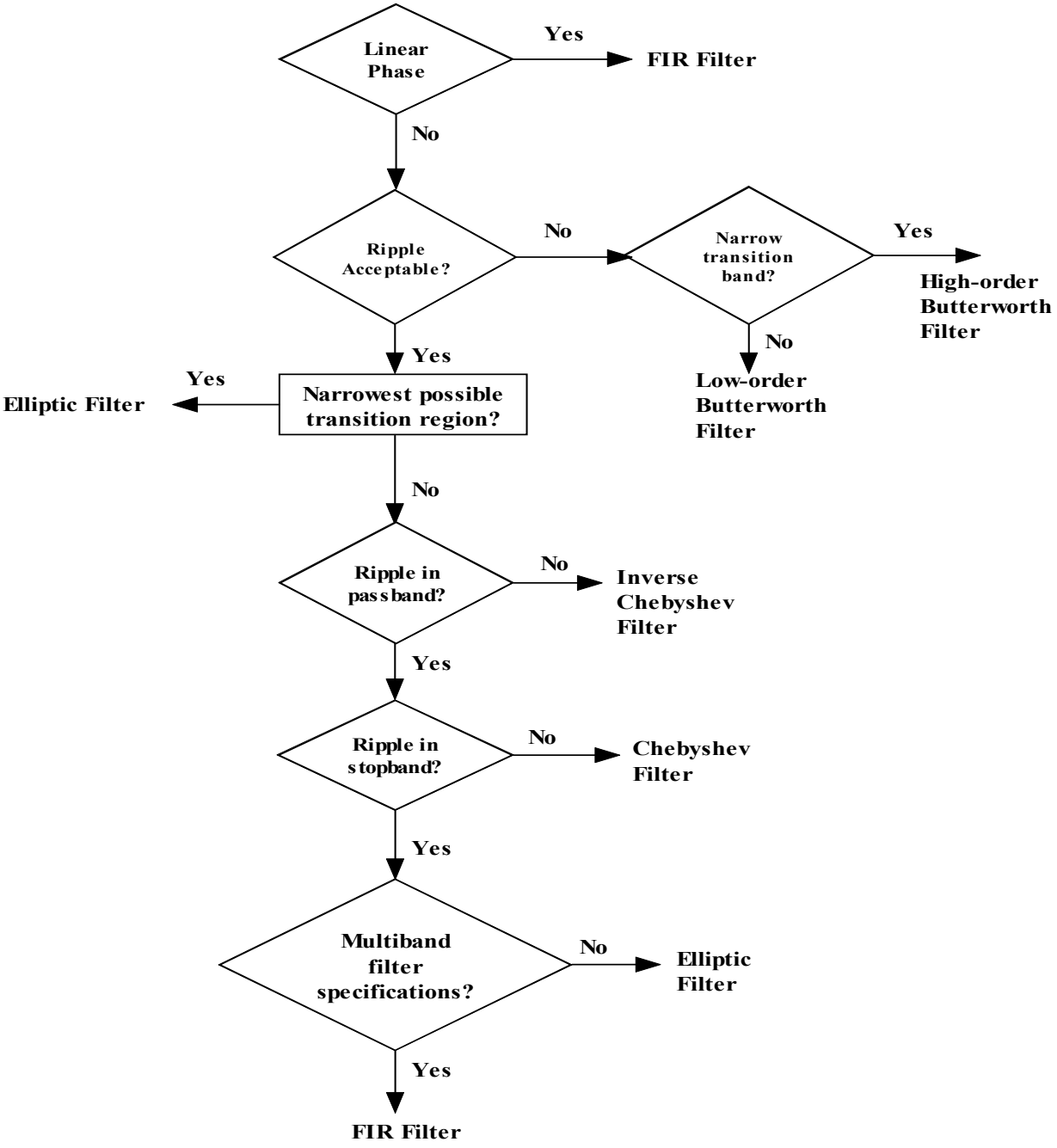


Figure 3.7 Flow chart for selecting filter

CHAPTER 4

VIRTUAL INSTRUMENTATION

4.1 Introduction

A virtual instrument consists of an industry-standard computer or workstation equipped with powerful application software, cost-effective hardware such as plug-in boards, and driver software, which together perform the functions of traditional instruments. Virtual instruments represent a fundamental shift from traditional hardware-centered instrumentation systems to software-centered systems that exploit the computing power, productivity, display, and connectivity capabilities of popular desktop computers and workstations. A virtual instrumentation system is computer software that a user would employ to develop a computerized test and measurement system, for controlling from a computer desktop an external measurement hardware device and for displaying test or measurement data collected by the external device on instrument-like panels on a computer screen. Virtual instrumentation extends also to computerized systems for controlling processes based on data collected and processed by a computerized instrumentation system .

An instrument is a device designed to collect data from an environment, or from a unit under test, and to display information to a user based on the collected data. Such an instrument may employ transducer to sense changes in a physical parameter, such as temperature or pressure, and to convert the sensed information into electrical signals, such as voltage or frequency variations. The term instrument may also cover, and for purposes of this description it will be taken to cover, a physical or software device that performs an analysis on data acquired from another instrument and then outputs the processed data to display or recording means. This second category of instruments would, for example, include oscilloscopes, spectrum analyzers and digital multimeters. The types of source data collected and analyzed by instruments may thus vary widely, including physical parameters such as temperature, pressure, distance, light, sound frequencies, amplitudes and electrical parameters including voltage, current, and frequency.

4.2 History of Instrumentation Systems

Historically, instrumentation systems originated in the distant past, with measuring rods, thermometers, and scales. In modern times, instrumentation systems have generally consisted of individual instruments, for example, an electro-mechanical pressure gauge comprising a sensing transducer wired to signal conditioning circuitry, outputting a processed signal to a display panel and perhaps also to a line recorder, in which a trace of changing conditions is inked onto a rotating drum by a mechanical arm, creating a time record of pressure changes. Even complex systems such as chemical process control applications typically employed, until the 1980s, sets of individual physical instruments wired to a central control panel that comprised an array of physical data display devices such as dials and counters, together with sets of switches, knobs and buttons for controlling the instruments.

The introduction of computers into the field of instrumentation began as a way to couple an individual instrument, such as a pressure sensor, to a computer, and enable the display of measurement data on a virtual instrument panel, displayed in software on the computer monitor and containing buttons or other means for controlling the operation of the sensor. Thus, such instrumentation software enabled the creation of a simulated physical instrument, having the capability to control physical sensing components .

4.3 Creation of Virtual Instrumentation

A large variety of data collection instruments designed specifically for computerized control and operation were developed and made available on the commercial market, creating the field now called “virtual instrumentation.” Virtual instrumentation thus refers to the use of general purpose computers and workstations in combination with data collection hardware devices and virtual instrumentation software to construct an integrated instrumentation system. In such a system the data collection hardware devices, which incorporate sensing elements for detecting changes in the conditions of test subjects are intimately coupled to the computer, whereby the operations of the sensors are controlled by the computer software and the output of the data collection devices is displayed on the computer screen .

Virtual instrumentation systems typically also comprise pure software "instruments", such as oscilloscopes and spectrum analyzers, for processing the collected sensor data and "massaging" it in ways useful to the users of the data. For example, software means may be employed to analyze collected data as needed to present the user with isolated information on "max" or "min" readings, averages, standard deviations, or combinations of results from related sensing points, etc.

4.4 Programming Requirements

Until the 1990's, the programming of virtual instrumentation systems was a task strictly for professional programmers, who wrote the required software programs using "textual" programming languages such as BASIC, C ++, or PASCAL. The development of instrumentation systems software by professionals using textual programming languages such as C++ is very time consuming and tedious and it typically results in the production of a program consisting of many pages of source code, written in a computer language that is virtually unreadable by non-programmers, and which thus cannot be modified by the typical users of such programs.

4.5 Drawbacks of current Approaches

In the last ten years, there have appeared several commercial software products for the development of virtual instrumentation systems using purely graphical programming methods. Each of these products provides users, typically including users who are not skilled software programmers, with a "graphical development environment" within which to design a custom virtual instrumentation system . Typically, the user is presented with a "design desktop" environment, generally having the look-and-feel familiar to users of Windows®-based graphical applications, in which a variety of software options and "tools" are accessible from toolbars and dialog boxes featuring drop-down menus, and may be accessed by manipulating an on-screen cursor using a computer mouse. However, these older software packages for developing virtual instrumentation systems, using graphical programming means, provide the user with tools for designing so-called "data flow" diagrams. The user of these software packages is thus required both to place icons representing desired system components onto a design desktop and then to effect

"wiring" connections between components. In order to design a data flow diagram that corresponds to a workable measurement system application, the user is required to have comparably deep knowledge and understanding of the specific data paths and element combinations that will be required to attain the user's objective, which is a "solution" to the user's measurement requirements. User designed systems developed using this type of software are also prone to errors, because they generally allow the user to unwittingly wire together components that are functionally incompatible.

4.6 The most recent Developments

The range of applications that may be made the subject of an instrumentation system spans the range of human activity. Therefore, a software development system that aims to provide a large cross-section of potential users with the tools to design their own customized instrumentation system must provide the user with a large range of development tools and options, including tools and options that may be or are mutually incompatible in a given application.

Ideally, such a system should also organize the software tools provided to the user in a way that enables the user to select easily the particular tools best suited for the pertinent application, and guide the user's selection of configuration options. Older software packages lack built-in safeguards against the construction of unworkable combinations of components, and provide inadequate intuitive guidance to the user seeking to develop a measurement application. An important objective in the development of Data Translation's DT Measure Foundry was to present the user with development tools in a manner that guides the user intuitively to choose system elements that are appropriate to the user's project, and that automatically precludes the user's selection of system elements that are mutually incompatible.

4.7 Virtual Instruments versus conventional Instruments

Stand-alone traditional instruments such as oscilloscopes and waveform generators are very powerful, expensive, and designed to perform one or more specific tasks defined by the vendor. However, the user generally cannot extend or customize them. The knobs and

buttons on the instrument, the built-in circuitry, and the functions available to the user, are all specific to the nature of the instrument. In addition, special technology and costly components must be developed to build these instruments, making them very expensive and slow to adapt.

Virtual instruments, by virtue of being PC-based, inherently take advantage of the benefits from the latest technology incorporated into off-the-shelf PCs. These advances in technology and performance, which are quickly closing the gap between stand-alone instruments and PCs, include powerful processors such as the Pentium 4 and operating systems and technologies such as Microsoft Windows XP, .NET, and Apple Mac OS X. In addition to incorporating powerful features, these platforms also offer easy access to powerful tools such as the Internet . Traditional instruments also frequently lack portability, whereas virtual instruments running on notebooks automatically incorporate their portable nature.

Engineers and scientists whose needs, applications, and requirements change very quickly, need flexibility to create their own solutions. One can adapt a virtual instrument to the particular needs without having to replace the entire device because of the application software installed on the PC and the wide range of available plug-in hardware.

4.8 Characteristics of Virtual Instrumentation systems

4.8.1 Flexibility

Except for the specialized components and circuitry found in traditional instruments, the general architecture of stand-alone instruments is very similar to that of a PC-based virtual instrument. Both require one or more microprocessors, communication ports (for example, serial and GPIB), and display capabilities, as well as data acquisition modules. What make one different from the other are their flexibility and the fact that one can modify and adapt the instrument for particular needs. A traditional instrument might contain an integrated circuit to perform a particular set of data processing functions; in a virtual instrument, these functions would be performed by software running on the PC processor. One can extend the set of functions easily, limited only by the power of the software used.

4.8.2 Lower Cost

By employing virtual instrumentation solutions, one can lower capital costs, system development costs, and system maintenance costs, while improving time to market and the quality of the products.

4.8.3 Plug-In and Networked Hardware

There is a wide variety of available hardware that can either plug into the computer or access through a network. These devices offer a wide range of data acquisition capabilities at a significantly lower cost than that of dedicated devices. As integrated circuit technology advances, and off-the-shelf components become cheaper and more powerful, so do the boards that use them. With these advances in technology comes an increase in data acquisition rates, measurement accuracy, precision and better signal isolation. Depending on the particular application, the hardware you choose might include analog input or output, digital input or output, counters, timers, filters, simultaneous sampling, and waveform generation capabilities. The wide gamut of boards and hardware could include any one of these features or a combination of them.

4.8.4 Connectivity and Instrument Control

Virtual instrumentation software productivity comes about because the software includes built-in knowledge of hardware integration. Designed to create test, measurement, and control systems, virtual instrumentation software includes extensive functionality for I/O of almost any kind. LabVIEW has ready-to-use libraries for integrating stand-alone instruments, data acquisition devices, motion control and vision products, GPIB/IEEE 488 and serial/RS-232 devices, and PLCs, among others, to build a complete measurement and automation solution. LabVIEW also incorporates major instrumentation standards such as VISA, an interoperable standard for GPIB, serial, and VXI instrumentation; PXI and software and hardware based on the PXI Systems Alliance Compact PCI standard; IVI interchangeable virtual instrument drivers; and VXI plug & play, a driver standard for VXI instruments .

4.8.5 Open Environment

Although LabVIEW provides the tools required for most applications, LabVIEW also is an open development environment. Standardization of software relies greatly on the ability of the package you select to work well with other software, measurement and control hardware, and open standards, which define interoperability between multiple vendors. In addition, conforming to open commercial standards reduces overall system cost. A large number of third-party hardware and software vendors develop and maintain hundreds of LabVIEW libraries and instrument drivers to help easily, use their products with LabVIEW. However, this is not the only way to provide connectivity to LabVIEW-based applications. LabVIEW offers simple ways to incorporate ActiveX software, dynamic link libraries (DLLs), and shared libraries from other tools. In addition, one can share LabVIEW code as a DLL, built executable, or using ActiveX. LabVIEW also offers a full range of options for communications and data standards, such as TCP/IP, OPC, SQL database connectivity, and XML data formats.

4.8.6 Multiple Platforms

The majority of computer systems use some variation of the Microsoft Windows operating system. Nevertheless, other options offer clear advantages for certain types of applications. Real-time and embedded development continues to grow rapidly in most industries, as computing power is packaged into smaller and more specialized packages. Minimizing losses resulting from changing to new platforms is important and choosing the right software for this purpose is a key factor. LabVIEW minimizes this concern, because it runs on Windows 2000, NT, XP, Me, 98, 95, and NT embedded, as well as Mac OS, Sun Solaris, and Linux. LabVIEW also compiles code to run on the Ventur Com ETS real-time operating system through the LabVIEW Real-Time Module. Given the importance of legacy systems, National Instruments continues to make available older versions of LabVIEW for Windows, Mac OS, and Sun operating systems. LabVIEW is platform independent; virtual instruments that you write in one platform can transparently be ported to any other LabVIEW platform by simply opening the virtual instrument. Because LabVIEW applications are portable across platforms . In addition, because one can create platform-independent virtual instruments by porting applications between

platforms, one can save development time and other inconveniences related to platform portability.

4.8.7 Analysis Capabilities

Virtual instrumentation software requires comprehensive analysis and signal processing tools, because the application does not just stop when the data is collected. High-speed measurement applications in machine monitoring and control systems usually require order analysis for accurate vibration data. Closed-loop, embedded control systems might need point-by-point averaging for control algorithms to maintain stability. In addition to the advanced analysis libraries already included in LabVIEW, National Instruments provides add-on software such as the LabVIEW Signal Processing Toolset, the LabVIEW Sound and Vibration Toolkit, and the LabVIEW Order Analysis Toolkit to complement analysis offerings.

4.8.8 Visualization Capabilities

LabVIEW includes a wide array of built-in visualization tools to present data on the user interface of the virtual instrument – for charting and graphing as well as 2D and 3D visualization. One can instantly reconfigure attributes of the data presentation, such as colors, font size, graph types, and more, as well as dynamically rotate, zoom, and pan these graphs with the mouse. Rather than programming graphics and all custom attributes from scratch, you can simply drag-and-drop these objects onto the instrument front panels.

4.8.9 Scalability

Engineers and scientists have needs and requirements that can change rapidly. They also need to have maintainable, extensible solutions that can be used for a long time. By creating virtual instruments based on powerful development software such as LabVIEW, one inherently designs an open framework that seamlessly integrates software and hardware. This ensures that the applications not only work well today but that one can easily integrate new technologies in the future as they become available, or extend

solutions beyond the original scope, as new requirements are identified. Moreover, every application has its own unique requirements that require a broad range of solutions.

4.9 Application of Virtual Instrumentation in the Engineering Process

Virtual instruments provide significant advantages in every stage of the engineering process, from research and design to manufacturing test.

4.9.1 Research and Design

In research and design, engineers and scientists demand rapid development and prototyping capabilities. With virtual instruments, one can quickly develop a program, take measurements from an instrument to test a prototype, and analyze results, all in a fraction of the time required to build tests with traditional instruments. When flexibility is needed, a scalable open platform is essential, from the desktop, to embedded systems, to distributed networks. The demanding requirements of research and development (R&D) applications require seamless software and hardware integration. Whether we need to interface stand-alone instruments using GPIB or directly acquire signals into the computer with a data acquisition board and signal conditioning hardware, LabVIEW makes integration simple. With virtual instruments, one also can automate a testing procedure, eliminating the possibility of human error and ensuring the consistency of the results by not introducing unknown or unexpected variables .

4.9.2 Development Test and Validation

With the flexibility and power of virtual instruments, one can easily build complex test procedures. For automated design verification testing, we can create test routines in LabVIEW and integrate software such as National Instruments Test Stand, which offers powerful test management capabilities. One of the many advantages these tools offer across the organization is code reuse. The code is developed in the design process, and then plugs these same programs into functional tools for validation, test, or manufacturing.

4.9.3 Manufacturing Test

Decreasing test time and simplifying development of test procedures are primary goals in manufacturing test. Virtual instruments based on LabVIEW combined with powerful test management software such as Test Stand deliver high performance to meet those needs. These tools meet rigorous throughput requirements with a high-speed, multithreaded engine for running multiple test sequences in parallel. Test Stand easily manages test sequencing, execution, and reporting based on routines written in LabVIEW. Test Stand integrates the creation of test code in LabVIEW. Test Stand also can reuse code created in R&D or design and validation.

CHAPTER 5

LabVIEW

5.1 Introduction

LabVIEW is a development system for industrial, experimental, and educational measurement and automation applications based on graphical programming, in contrast to textual programming - however, textual programming is supported in LabVIEW. LabVIEW has a large number of functions for numerical analysis and design and visualization of data. LabVIEW is a revolutionary graphical development environment with built in functionality for data acquisition, instrument control, measurement analysis, and data presentation. LabVIEW gives the flexibility of a powerful programming language without the complexity of traditional development environments. LabVIEW delivers extensive acquisition, analysis, and presentation capabilities in a single environment, so that one can seamlessly develop a complete on the platform of choice.

LabVIEW gives development environment for measurement control and automation. Unlike general purpose programming languages, LabVIEW provides functionality specifically tailored to the needs of measurements control, and automation applications, accelerating development process. From built in analysis capabilities to connectivity with a wide variety of I/O, LabVIEW delivers what engineers and scientists need to quickly build test and measurement. Data acquisition, embedded control. Scientific research and process monitoring systems.

The LabVIEW graphical development environment gives powerful tools to create applications without writing any lines of text based code. With LabVIEW, one can drag and drop pre-built objects to quickly and simply create user interfaces for the application. Then, one can specify system functionality by assembling block diagrams a natural design notation for scientist and engineers .

LabVIEW has tight Integration with thousands of Instruments and Measurement devices and delivered seamless connectivity with measurement hardware, so one can quickly configure and use virtually any measurement device, including everything from stand-alone instruments to plug-in data acquisition devices, motion controllers, image acquisition systems, and programmable logic controllers (PLCs). LabVIEW has open connectivity with Other Applications and furthermore, LabVIEW can allow you to connect to other applications and share data through ActiveX, the Web, DLLs, shared libraries, SQL, TCP/IP, XML, OPC, wireless communication and other methods. LabVIEW's open connectivity enables you to create open, flexible applications that can communicate with other applications across your organization. With LabVIEW, one can develop systems that meet even the most demanding performance requirements across a variety of platforms including Windows, Macintosh, UNIX, or real-time systems. LabVIEW also includes traditional program development tools. You can set breakpoints, animate program execution to see how the program executes, and single-step through the program to make debugging and program development easier.

5.1.1 Why to use LabVIEW?

LabVIEW empowers you to build your own solutions for scientific and engineering systems. LabVIEW gives you the flexibility and performance of a powerful programming language without the associated difficulty and complexity.

LabVIEW gives thousands of successful users a faster way to program instrumentation, data acquisition, and control systems. By using LabVIEW to prototype, design, test, and implement your instrument systems, system development time can be reduced and productivity increases by a factor of 4 to 10.

LabVIEW also gives you the benefits of a large installed user base, years of product feedback, and powerful add-on tools. Finally, National Instruments technical support and Developer Zone ensure successful development of your solutions .

5.2 How Does a LabVIEW Work?

LabVIEW programs are called virtual instruments, or VIs, because their appearance and operation imitate physical instruments, such as oscilloscopes and multimeters. Every VI

uses functions that manipulate input from the user interface or other sources and display that information or move it to other files or other computers .

A VI contains the following three components:

- **Front panel**— Serves as the user interface.
- **Block diagram**— Contains the graphical source code that defines the functionality of the VI.
- **Icon and connector pane**— Identifies the VI so that you can use the VI in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages.

This section overviews the LabVIEW front panel, block diagram, and palettes. It also explains the dataflow model for program execution that LabVIEW follows.

5.2.1 Front Panel

The front panel is the user interface of the VI. You build the front panel with controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, pushbuttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

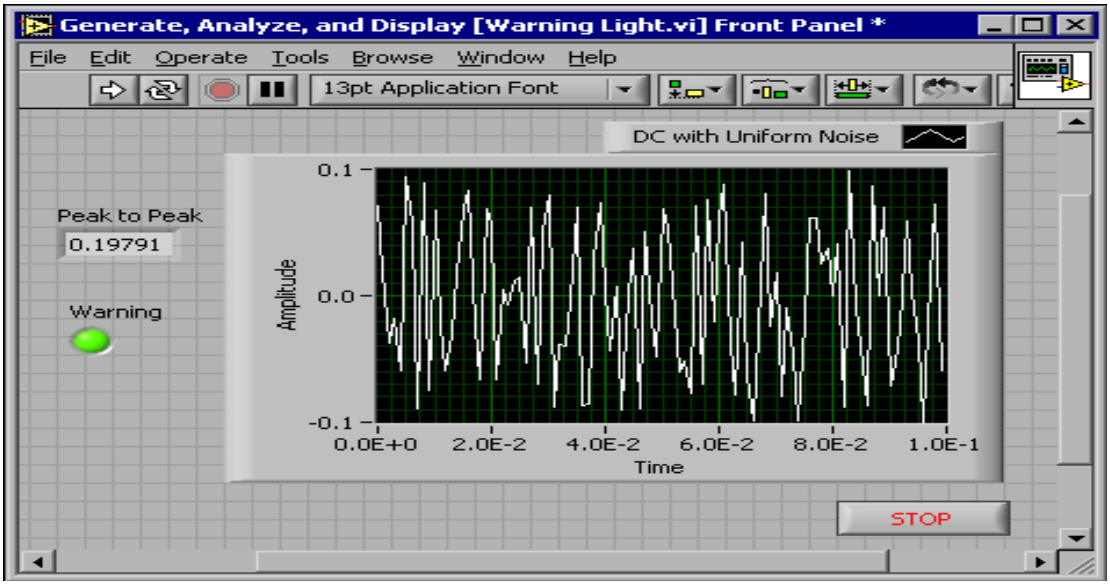


Figure 5.1 Front panel

The front panel can contain knobs, push buttons, graphs, and other controls and indicators.

1. A control (input).
2. An indicator (output).

Controls are knobs, push buttons, dials, and other input devices. Controls simulate the input devices on a physical instrument and supply data to the block diagram of the VI.

Indicators are graphs, LEDs, and other displays. Indicators simulate the output devices on a physical instrument and display data the block diagram acquires or generates.

5.2.2 Block Diagram

The block diagram contains the graphical source code of the VI. In the block diagram, one programs the VI to control and perform functions on the inputs and outputs created on the front panel. The block diagram can include functions and structures from the built-in LabVIEW VI libraries. It also can include terminals that are associated with controls and indicators created on the front panel.

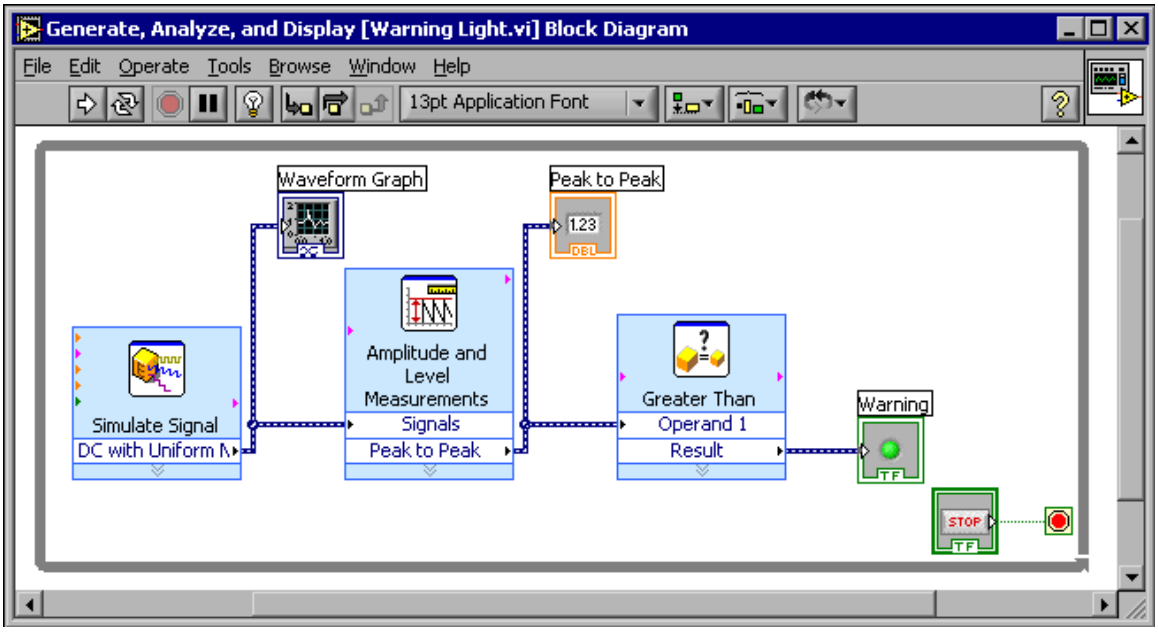


Figure 5.2 Block diagram

1. A function.
2. A structure.
3. Terminals from the front panel

5.2.3 Palettes

LabVIEW palettes give the options needed to create and edit front panel and block diagram.

5.2.3.1Tools Palette

The **Tools** palette is available on the front panel and the block diagram. A tool is a special operating mode of the mouse cursor. When you select a tool, the cursor icon changes to the tool icon. Use the tools to operate and modify front panel and block diagram objects.
Select **Window»Show Tools Palette** to display the **Tools** palette. You can place the **Tools** palette anywhere on the screen. If automatic tool selection is enabled and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the **Tools** palette .

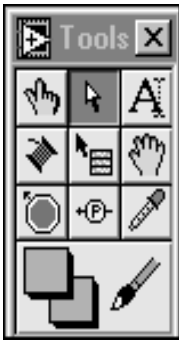


Figure 5.3 Tool palette

5.2.3.2 Controls Palette

The **Controls** palette is available only on the front panel. The **Controls** palette contains the controls and indicators you use to create the front panel. Select **Window»Show Controls Palette** or right-click the front panel workspace to display the **Controls** palette. You can place the **Controls** palette anywhere on the screen.

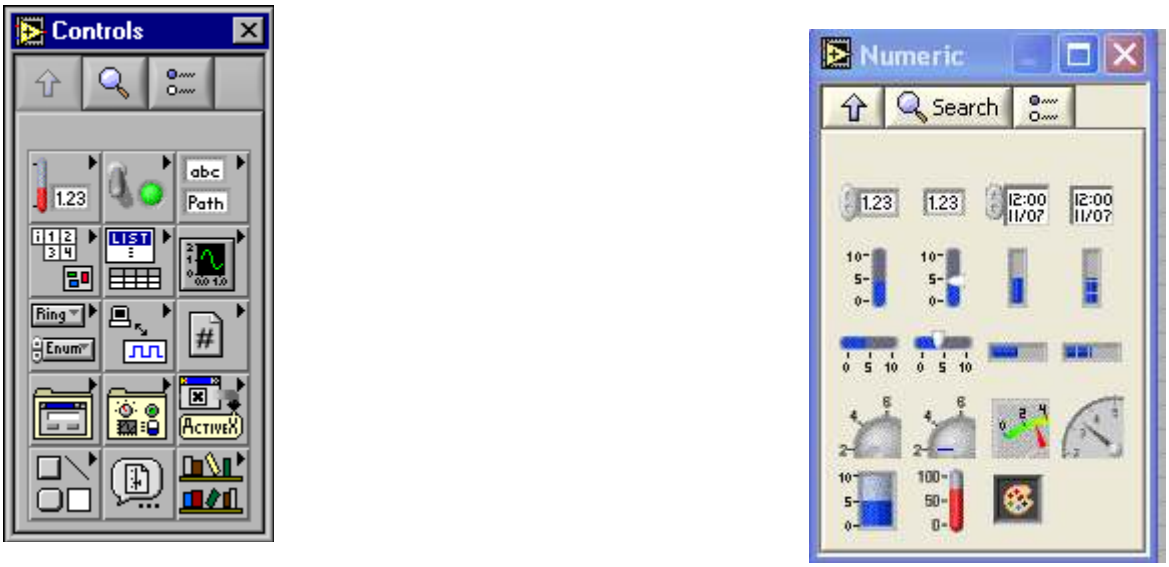


Figure 5.4 Control Palette

5.2.3.3 Functions Palette

The **Functions** palette is available only on the block diagram. The **Functions** palette contains the VIs and functions you use to build the block diagram. Select **Window»Show Functions Palette** or right-click the block diagram workspace to display the **Functions** palette. You can place the **Functions** palette anywhere on the screen.

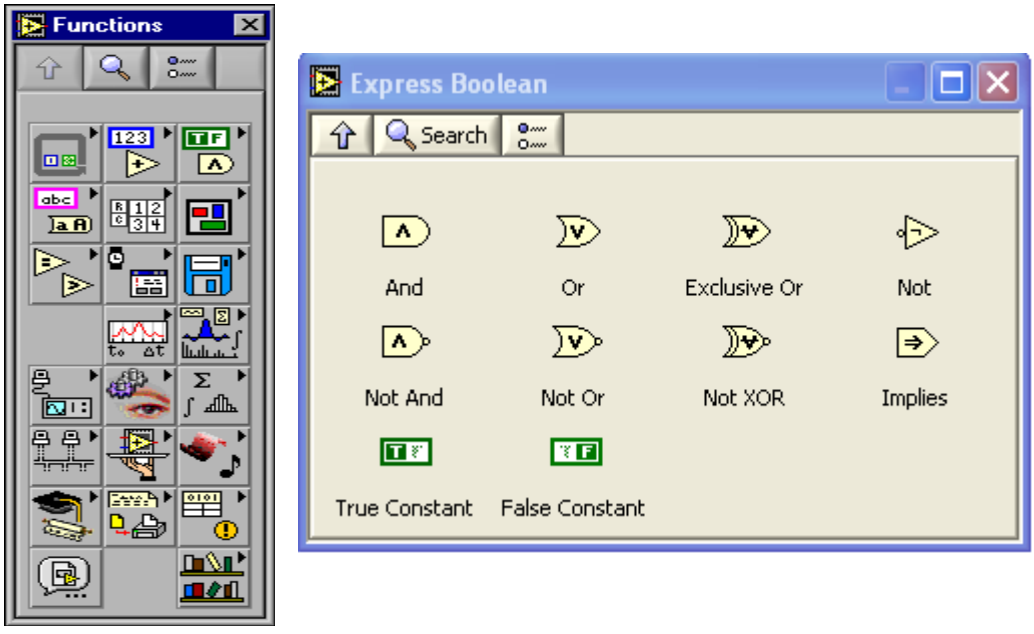


Figure 5.5 Function palette

5.3 Data Flow

LabVIEW VIs follow a dataflow model for program execution. The block diagram consists of nodes such as VIs, structures, and terminals from the front panel. These nodes are connected by *wires*, which define the flow of data through the program. The execution of a node occurs when all its inputs are available. When a node finishes executing, it releases all its outputs to the next node in the dataflow path.

5.4 Data Acquisition & Instrument Control in LabVIEW

PC-based data acquisition refers primarily to the acquisition of data by means of PC-based plug-in cards, designed as classical multifunction cards, and external box systems and by means of PC-based instruments. There are two basic ways to develop applications with DAQ hardware and software components: First, the conventional way- using the data acquisition libraries of LabVIEW. Second, is by using DAQ Wizards.

- **LabVIEW data acquisition libraries** - It is divided into several main groups, which are again divided into several subgroups. The main groups are important prerequisite to understand the application.
- **Analog input**- The Libraries contain functions contain functions concerning he A-D conversion. The choice ranges from single-point measurement to continuous multiple-point acquisition with sampling rates ranging from megahertz to the PCI bus.
- **Analog output** -The libraries are similar to the analog input, only different is the function they contain are used by D-A converter to output arbitrary or continuous curve forms.
- **Digital I/O** - The Digital I/O libraries allow you to set reset and read input and output ports in any port width. User can freely define the inputs and outputs.

CHAPTER 6

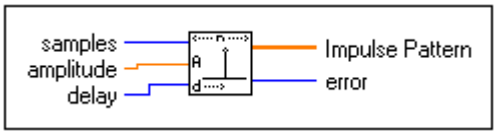
DEVELOPING THE DIGITAL IIR FILTER USING LabVIEW

6.1 Designing IIR Filters

When choosing an IIR filter for an application, you must know the response of the filter. We design IIR filters by approximating the desired magnitude response of a discrete-time system.

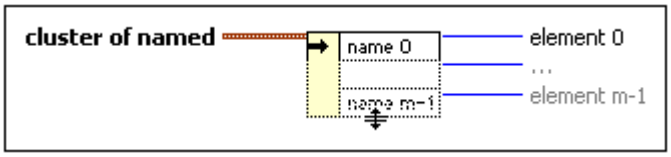
6.1.1 Functions Used in VIs

1. **Impulse Pattern:** Generates an array containing an impulse pattern.



Window>>Function palette>>All function>>Analyze>>Signal processing>>Signal generation>> impulse pattern.vi

2. **Unbundle By Name:** Returns the cluster elements whose names you specify.

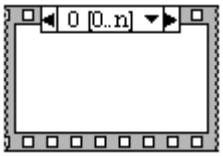


Window>>Function palette>>All function>>cluster>>Unbundle by name

3. **Numeric Constants:** Use the numeric constant to pass a numeric value to the block diagram.

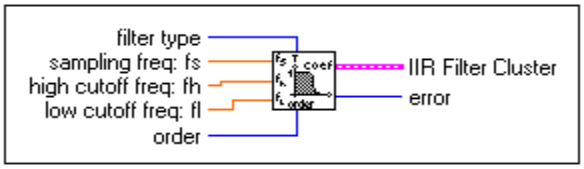
Window>>Function palette>>All function>>numeric>>numeric constant

4. **Stacked Sequence Structure:** Consists of one or more subdiagrams, or frames, that execute sequentially.



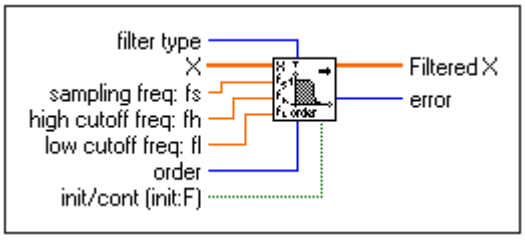
Window>>Function palette>>All function>>structures>> Stacked Sequence Structure

5. **Butterworth Coefficients:** Generates the set of filter coefficients to implement an IIR filter as specified by the Butterworth filter model.



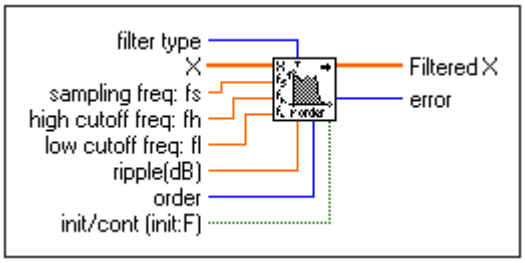
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> Butterworth Coefficients.vi

6. **Butterworth Filter:** Generates a digital Butterworth filter.



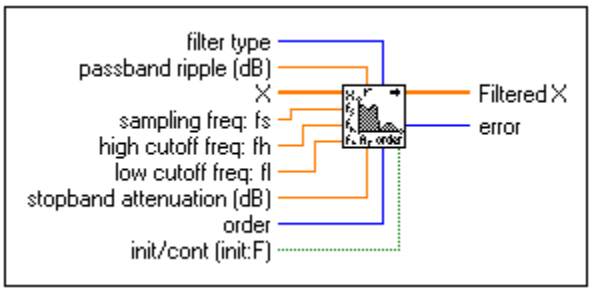
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> Butterworth filter.vi

7. **Chebyshev Filter:** Generates a digital Chebyshev filter.



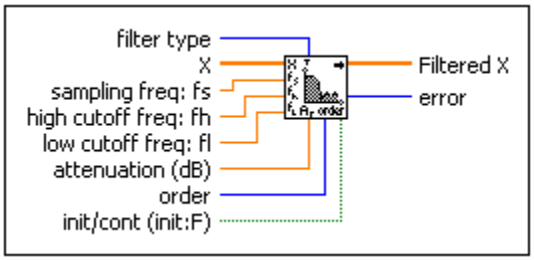
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> chebyshev filter.vi

8. **Elliptic Filter:** Generates a digital Elliptic filter.



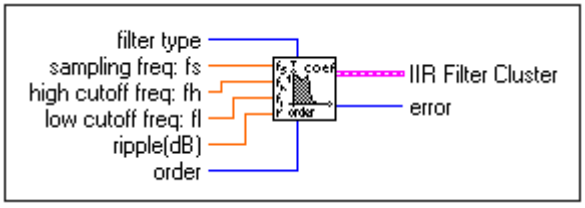
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> elliptic filter.vi

9. **Inverse Chebyshev Filter:** Generates a digital Chebyshev II filter.



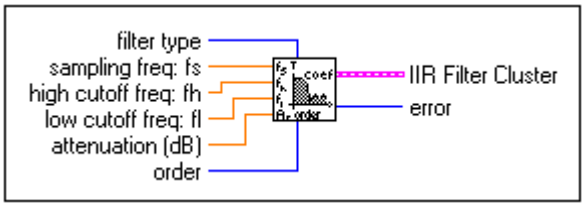
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> inverse chebyshev filter.vi

10. **Chebyshev Coefficients:** Generates the set of filter coefficients to implement an IIR filter as specified by the Chebyshev filter model.



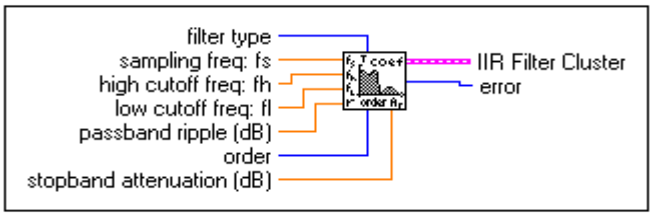
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> chebyshev Coefficients.vi

11. **Inverse Chebyshev Coefficients:** Generates the set of filter coefficients to implement an IIR filter as specified by the Chebyshev II Filter model.



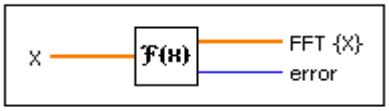
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> inverse chebyshev Coefficients.vi

12. Elliptic Coefficients: Generates the set of filter coefficients to implement a digital elliptic IIR filter.



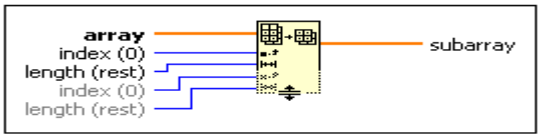
Window>>Function palette>>All function>>Analyze>>signal processing >> filters >> IIR >> Elliptic Coefficients.vi

13. FFT: Computes the fast Fourier transform (FFT) of the input sequence **X**.



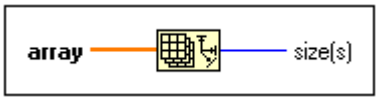
Window>>Function palette>>All function>>Analyze>>signal processing >> frequency domain>> FFT.vi

14. Array Subset: Returns a portion of array starting at index and containing length elements.



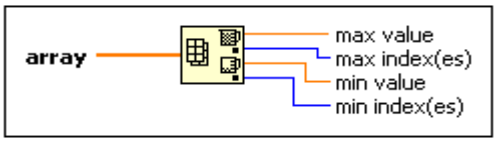
Window>>Function palette>>All function>>Arrays>> Array Subset

15. Array Size: Returns the number of elements in each dimension of array.



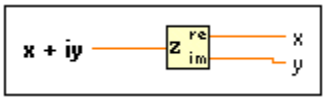
Window>>Function palette>>All function>>Arrays>> Array Size

16. Array Max & Min: Returns the maximum and minimum values found in array.



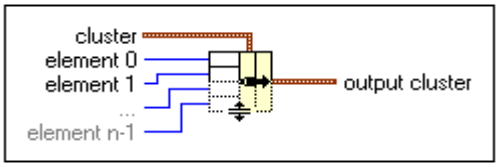
Window>>Function palette>>All function>>Arrays>> Array max & min

17. Complex To Re/Im: Breaks a complex number into its rectangular components.



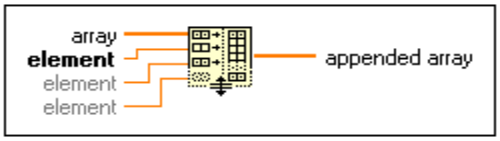
Window>>Function palette>>All function>>numeric>>complex>> Complex To Re/Im

18. Bundle: Assembles a cluster from individual elements**.**



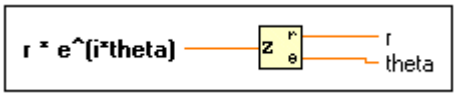
Window>>Function palette>>All function>>cluster>>bundle

19. Build Array: Concatenates multiple arrays or appends elements to an n-dimensional array.



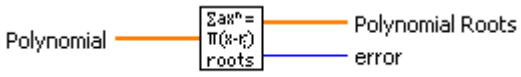
Window>>Function palette>>All function>>Arrays>> Build Array

20. Complex To Polar: Breaks a complex number into its polar components.



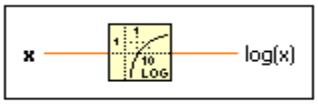
Window>>Function palette>>All function>>numeric>>complex>> Complex To Polar

21. Complex Polynomial Roots.vi: Finds the complex roots of a Complex Polynomial.



Window>>Function palette>>All function>>select VI>>vi.lib>> Complex Polynomial Roots.vi

22. Logarithm Base 10: Computes the base 10 logarithm of x.



Window>>Function palette>>All function>>numeric>>logarithmic>> Logarithm Base 10

6.1.2 Butterworth filter

Figure 6.1 shows the block diagram of a VI that returns the magnitude response of a butterworth IIR filter.

The VI in Figure 6.1 completes the following steps to compute the magnitude response, filter coefficients and pole-zero plots to find the stability of the filter.

1. Pass the all parameters (cutoff frequency, stop frequency, passband attenuation, stopband attenuation, sample rate) with filer type—lowpass, highpass, bandpass, or bandstop to the Case structure to calculate the order of filter.
2. Apply these parameters to the Butterworth Coefficient Function to generate the coefficients.
3. Divides these coefficient arrays into two separate parts named as forward and reverse coefficients.
4. Display these forward and reverse coefficients with coefficient size.
5. Pass these coefficients to Complex Polynomial Roots VI and apply its output to Complex To Re/Im Function to display the pole-zero plot.

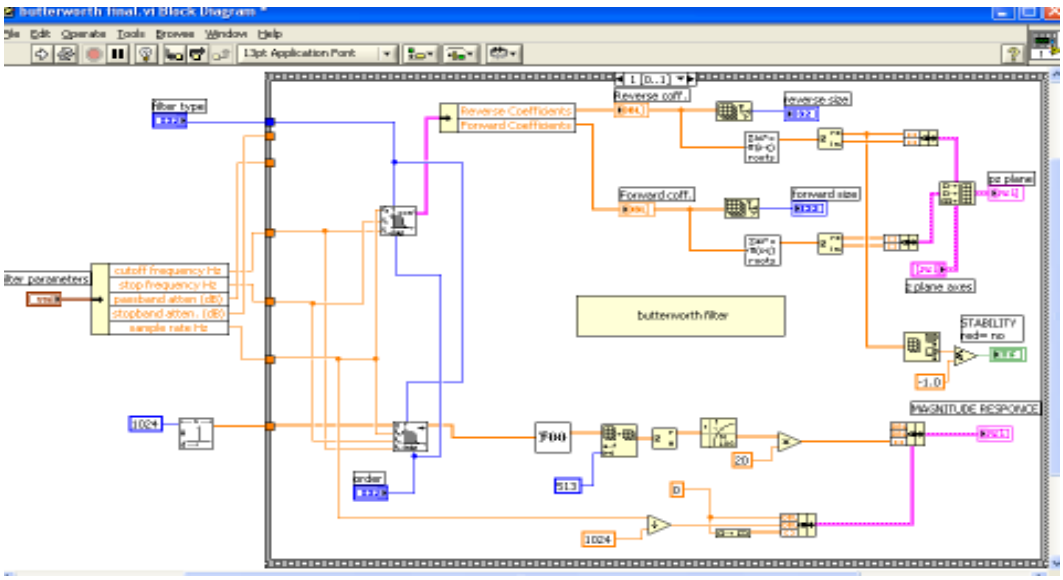


Figure 6.1 Butterworth.vi

6. Compare the output of Complex To Re/Im Function (only for reverse coefficients) with -1 to find out the stability. This is shown with the help of LED. If LED is red then the system is unstable otherwise stable.
7. Pass an impulse signal through the Butterworth filter Function.
8. Apply all the parameters and filter type—lowpass, highpass, bandpass, or bandstop to the Butterworth filter Function.
9. The signal passed out from this function is the impulse response of the filter.
10. Pass the filtered signal to the FFT VI. Use the FFT VI to perform a Fourier transform on the impulse response and to compute the frequency response of the filter, such that the impulse response and the frequency response comprise the Fourier transform pair $h(t) \Leftrightarrow H(f)$. h(t) is the impulse response. H(f) is the frequency response.
11. Use the Array Subset function to reduce the data returned by the FFT VI. Half of the real FFT result is redundant so the VI needs to process only half of the data returned by the FFT VI.
12. Use the Complex To Polar function to obtain the magnitude-and-phase form of the data returned by the FFT VI. The magnitude-and-phase form of the complex

output from the FFT VI is easier to interpret than the rectangular component of the FFT.

13. Convert the magnitude to decibels.
14. Display the magnitude response with the help of waveform graph.

6.1.3 Chebyshev Filter, Inverse Chebyshev Filter, and Elliptic Filter

Because the same mathematical theory applies to design other type of IIR filters, the block diagram in Figure 6.2, 6.3, and 6.4 of VI return the magnitude response of chebyshev, inverse chebyshev and elliptic IIR filter respectively.

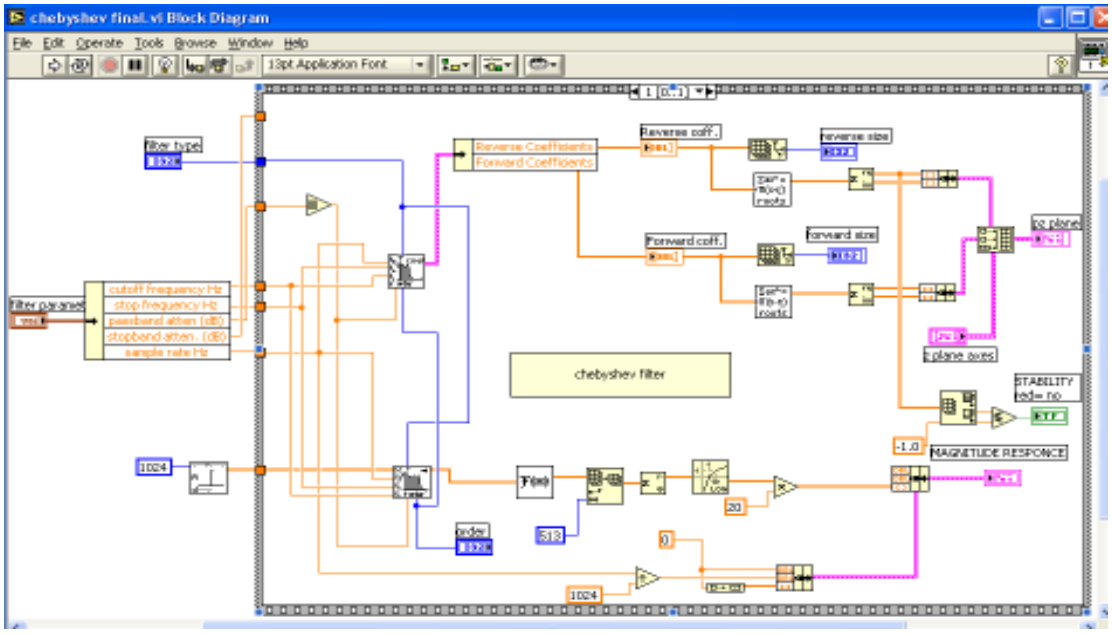


Figure 6.2 Chebyshev.vi

The design procedure is same for all type of filter as described for butterworth filter. The main difference between these VIs is that the chebyshev, inverse chebyshev and elliptic coefficient functions are used in step 2 in place of butterworth coefficient function and the chebyshev, inverse chebyshev and elliptic filter function in place of butterworth filter function in step 7 and step 8.

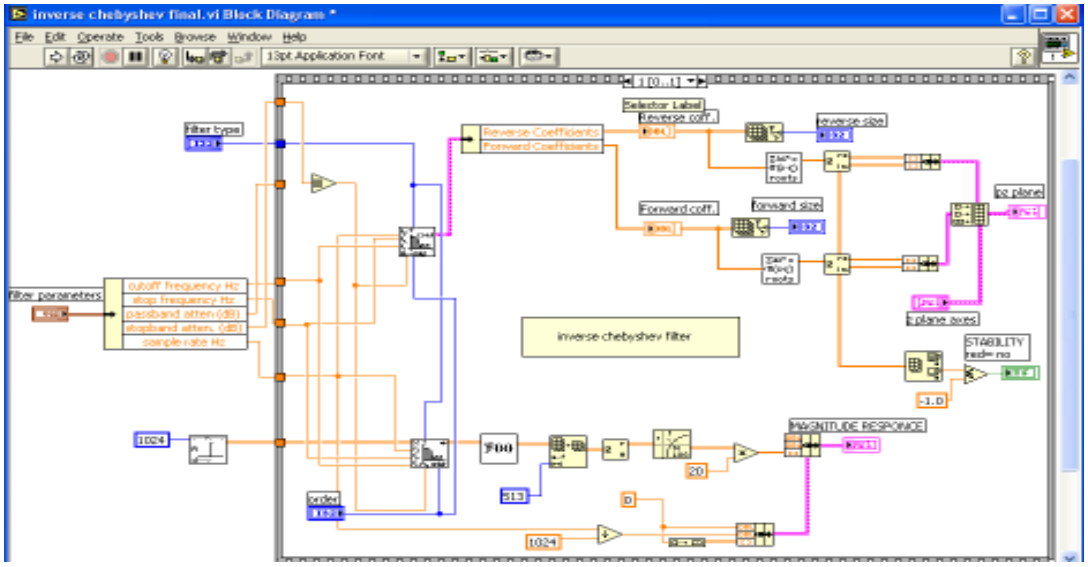


Figure 6.3 Inverse Chebyshev.vi

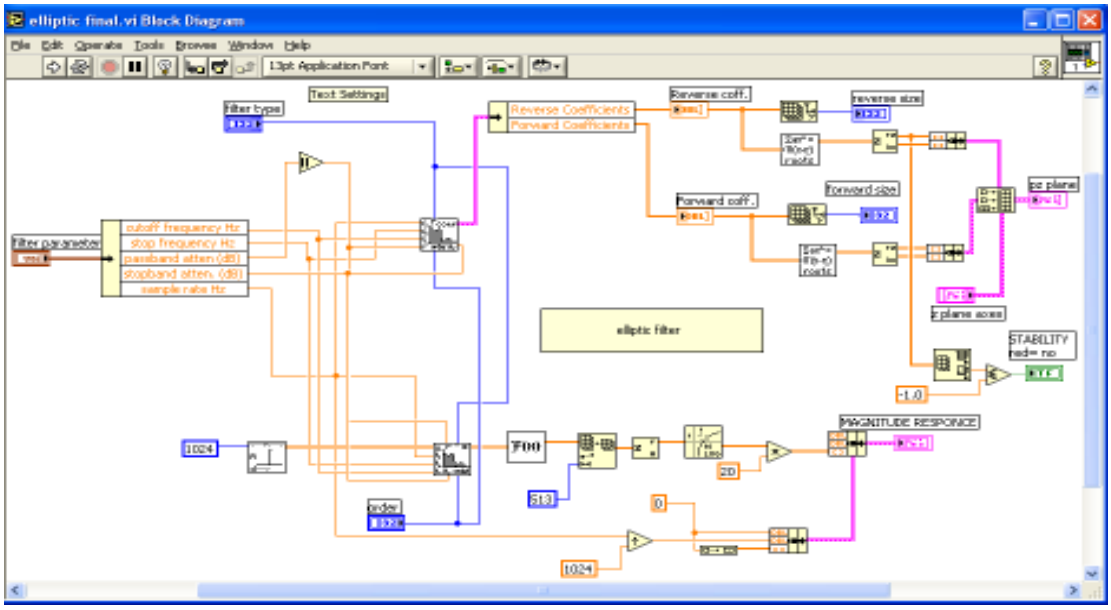


Figure 6.4 Elliptic.vi

CHAPTER 7

RESULTS AND COMPARITIVE STUDY

7.1 Butterworth Filter

Lowpass Filter

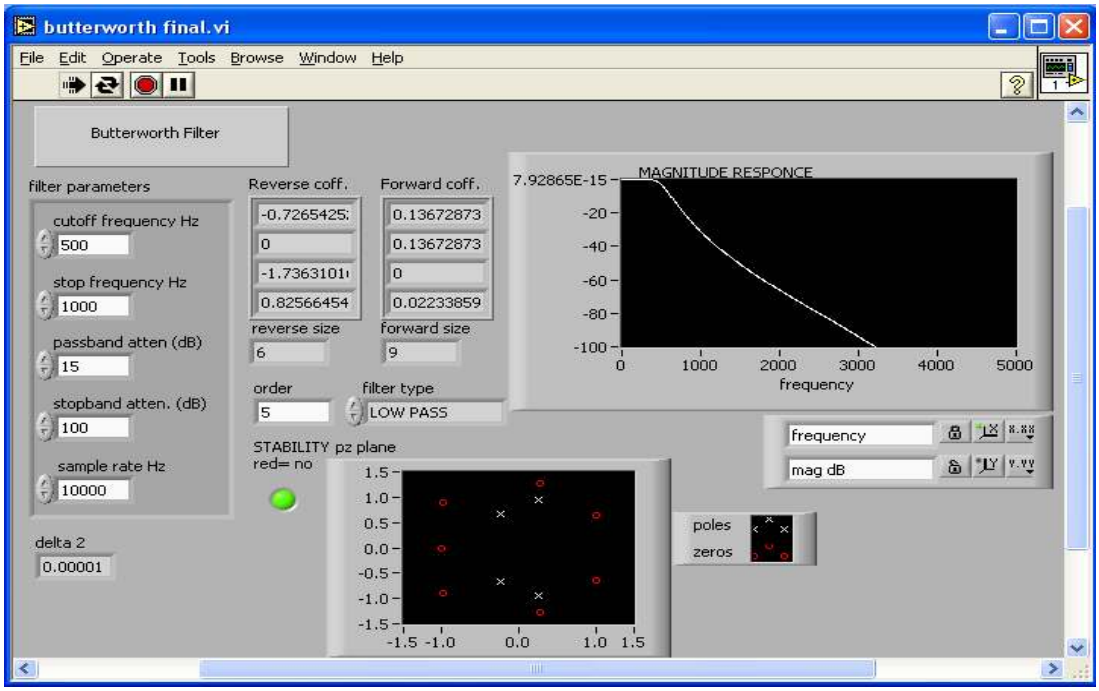


Figure 7.1 Response of Butterworth Lowpass Filter for 5th order

Figure 7.1 is response of butterworth lowpass filter. The specifications for this filter are given as :

cutoff frequency – 500 Hz,

stop frequency– 1000Hz,

passband attenuation- 15 dB,

stopband attenuation- 100 dB,

sample rate- 10000 Hz,

The Butterworth filter is one type of signal processing filter design which is designed to have a frequency response which is as flat as mathematically possible in the passband. Another name for it is maximally flat magnitude filter.The pole-zero plot the designedfilter for above specification is also shown in figure 7.1. If we increase the order of the filter keeping the same specifications, the magnitude response plot slope becomes steep.

7.2 Chebyshev Filter

Lowpass Filter

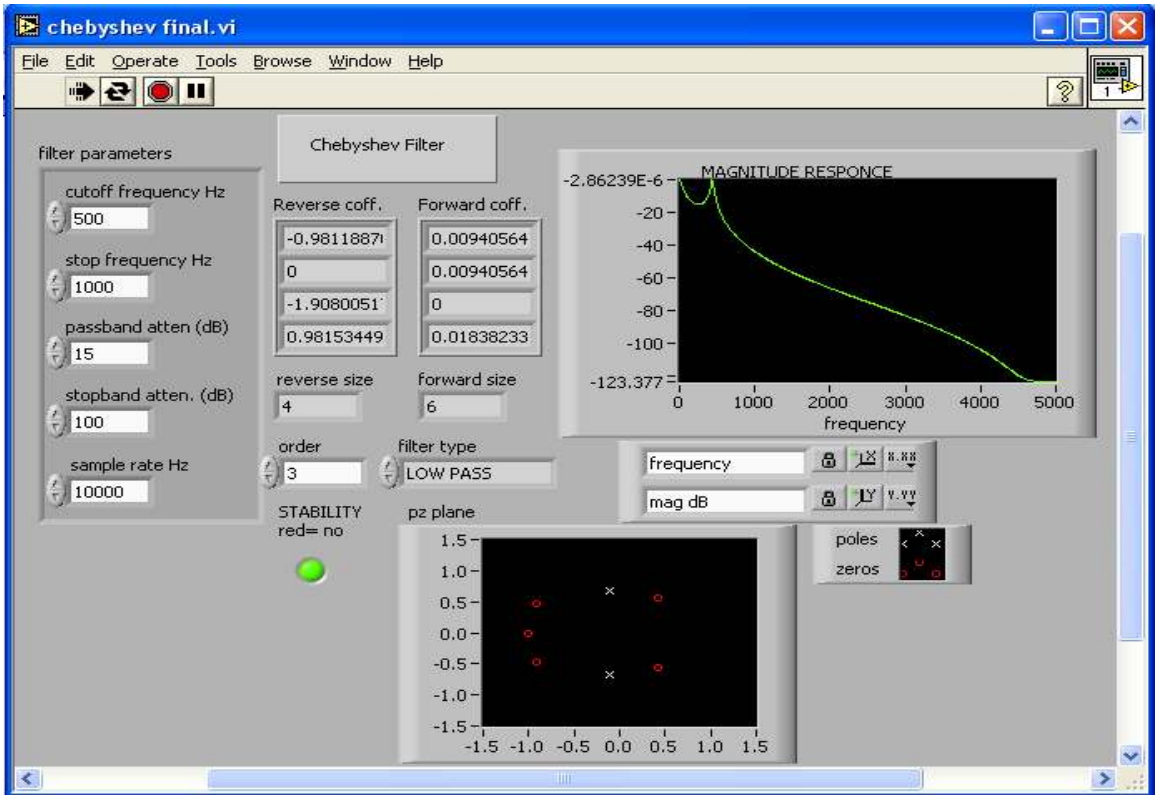


Figure 7.2 Response of Chebyshev Lowpass Filter for 3rd order

Figure 7.2 is response of Chebyshev lowpass filter. The specifications for this filter are given below:

cutoff frequency – 500 Hz,

stop frequency– 1000Hz,

passband attenuation- 15 dB,

stopband attenuation- 100dB,

sample rate- 10000 Hz,

order -3

Chebyshev filters are analog or digital filters having a steeper roll-off and more passband ripple (type I) or stopband ripple (type II) than Butterworth filters. Chebyshev filters have the property that they minimize the error between the idealized and the actual filter characteristic over the range of the filter, but with ripples in the passband.The pole-zero plot the designed filter for above specification is also shown in figure 7.1. If we increase the order of the filter keeping the same specifications, the magnitude response plot slope becomes steep.

7.3 Inverse Chebyshev Filter

Lowpass Filter

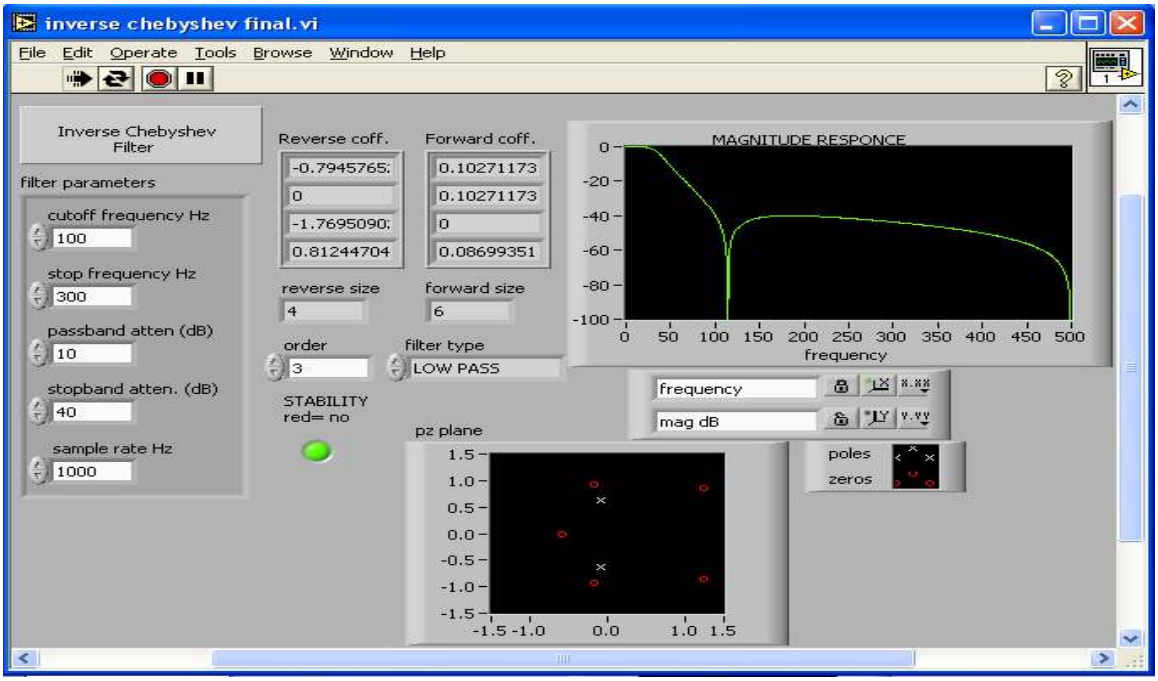


Figure 7.3 Response of Inverse Chebyshev Lowpass Filter for 3rd order

Figure 7.3 is response of Inverse Chebyshev lowpass filter. The specifications for this filter are given below:

cutoff frequency – 500 Hz,

stop frequency– 1000Hz,

passband attenuation- 15 dB,

stopband attenuation- 100dB,

sample rate- 10000 Hz,

The pole-zero plot the designed filter for above specification is also shown in figure 7.1. If we increase the order of the filter keeping the same specifications, the magnitude response plot slope becomes steep.

7.4 Elliptic Filter

Lowpass Filter

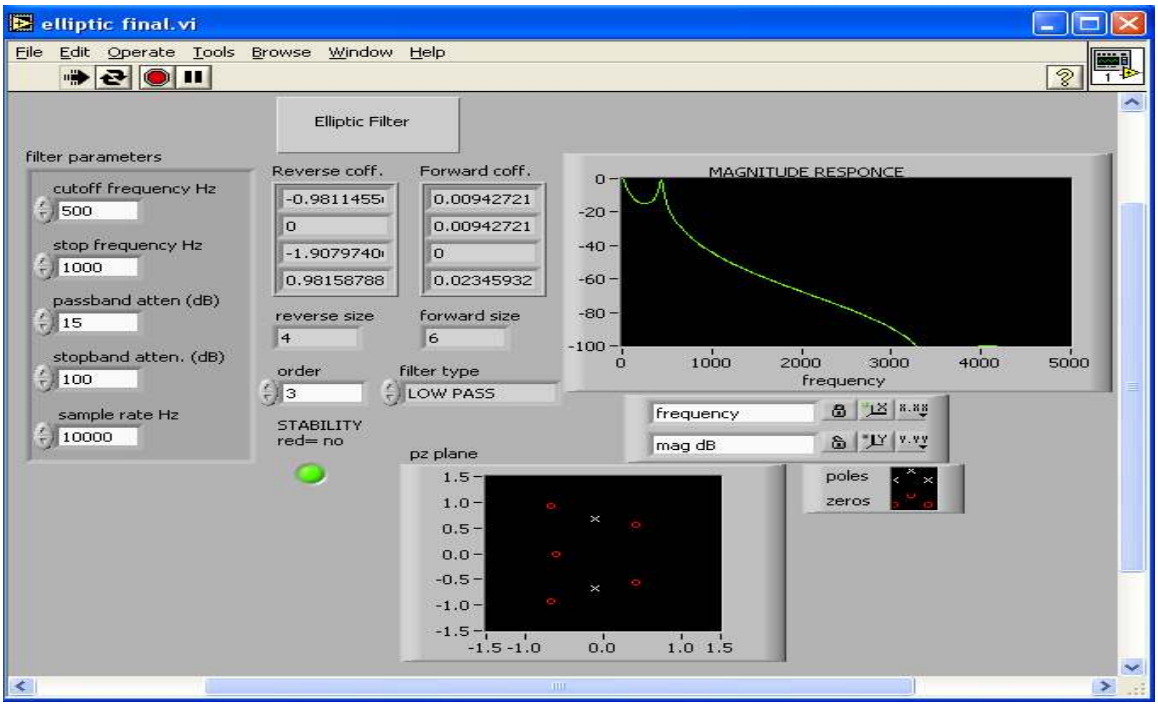


Figure 7.4 Response of Elliptic Lowpass Filter for 3rd order

Figure 7.4 is response of Elliptic lowpass filter. The specifications for this filter are given below:

cutoff frequency – 500 Hz,
stop frequency– 1000Hz,
passband attenuation- 15 dB,
stopband attenuation- 100dB,
sample rate- 10000 Hz,
order -3

The pole-zero plot the designed filter for above specification is also shown in figure 7.1. If we increase the order of the filter keeping the same specifications, the magnitude response plot slope becomes steep.

From the above results we analyze that the transition band become sharper between passband and stopband on increasing the order of filter with same other parameters.

Compared to a Butterworth filter as shown in figure 7.1 Chebyshev filter figure 7.2, and Inverse Chebyshev figure 7.3, can achieve a sharper transition between the passband and the stopband with a lower order filter. The sharp transition between the passband and the stopband of these filter produces smaller absolute errors and faster execution speeds than a Butterworth filter**.**

Compared with the same order Butterworth or Chebyshev filters, the elliptic filters figure 7.4 provide the sharpest transition between the passband and the stopband, which accounts for their widespread use.

7.5 Comparison of results of IIR filter implemented in MATLAB and LabVIEW

7.5.1 Butterworth Filter

Specifications: cutoff frequency – 500 Hz,

stop frequency–1000Hz,

passband attenuation-15,

stopband attenuation-40,

sample rate- 10000 Hz,

Order-7

MATLAB Result

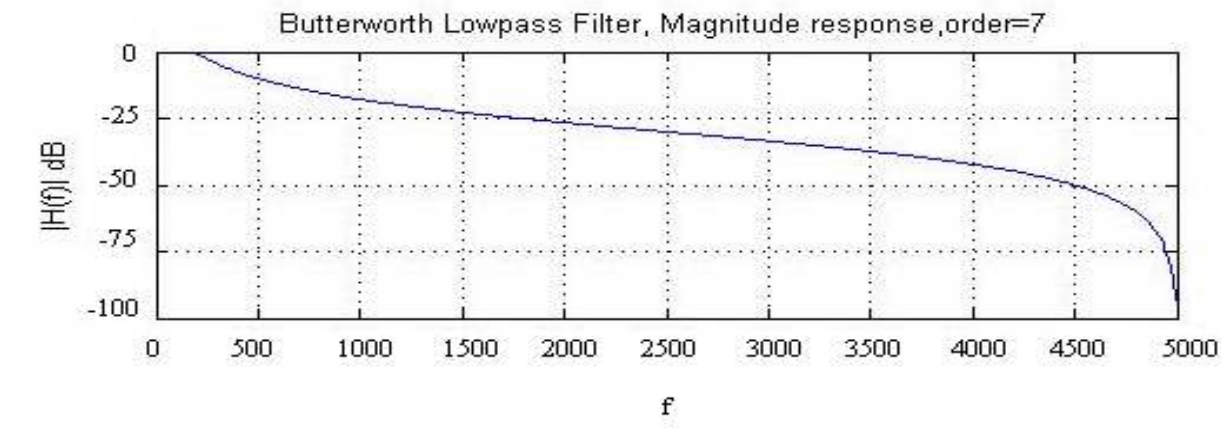


Figure 7.17(a) MATLAB Result

LabVIEW Result

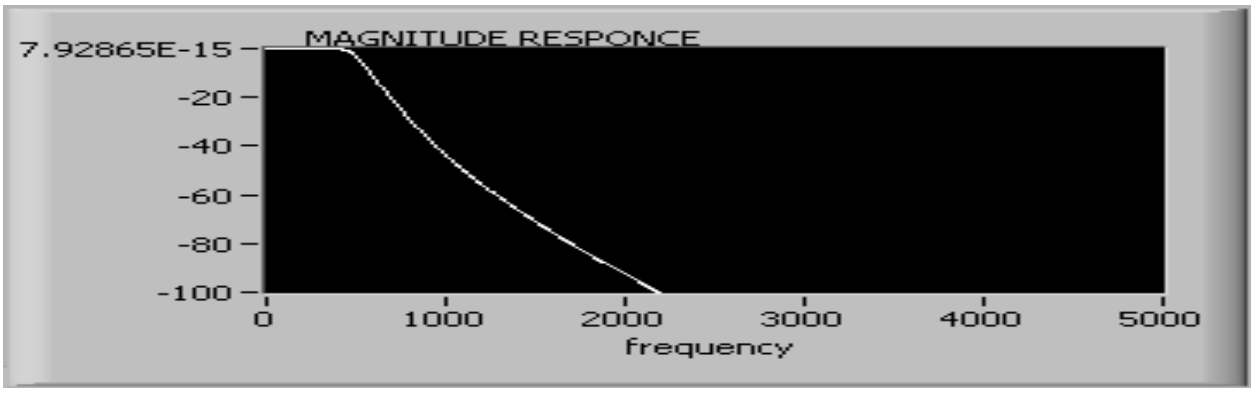


Figure 7.17(b) LabVIEW Result

7.5.2 Chebyshev Filter

Specifications: cutoff frequency – 500 Hz,
stop frequency–1000Hz,
passband attenuation-15,
stopband attenuation-40,
sample rate- 10000 Hz,
Order-3

MATLAB Result

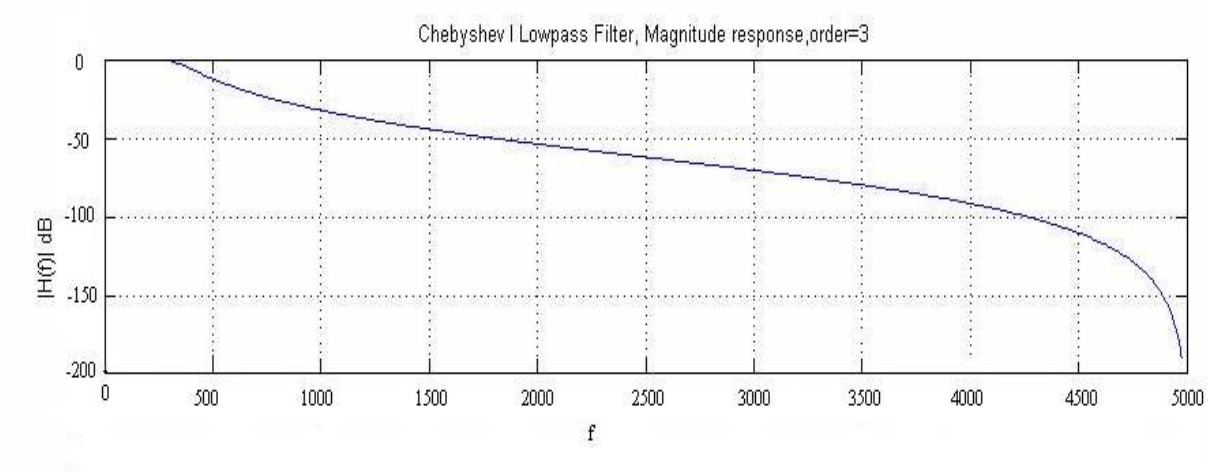


Figure 7.18(a) MATLAB Result

LabVIEW Result

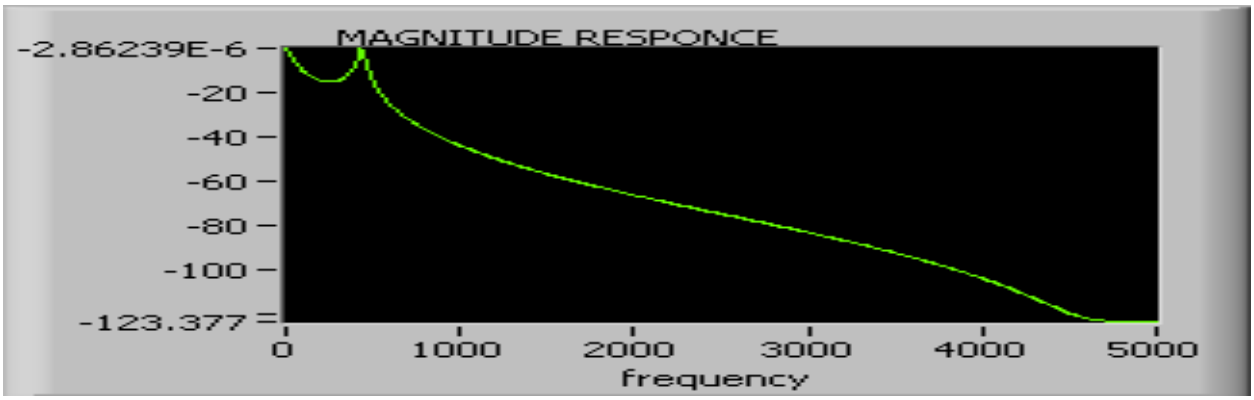


Figure 7.18(b) LabVIEW Result

7.5.3 Elliptic Filter

Specifications: cutoff frequency – 500 Hz,
stop frequency–1000Hz,
passband attenuation-15,
stopband attenuation-40,
sample rate- 10000 Hz,
Order-4

MATLAB Result

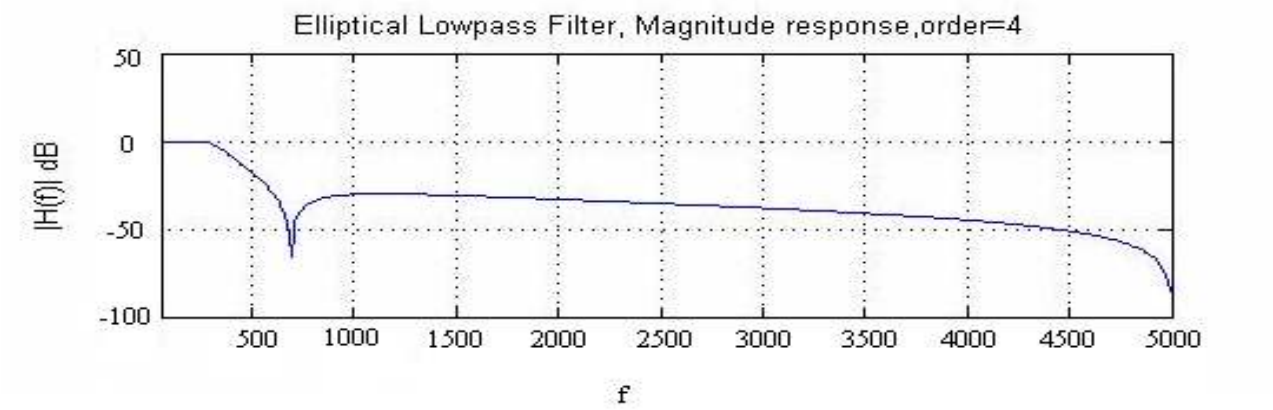


Figure 7.19(a) MATLAB Result

LabVIEW Result

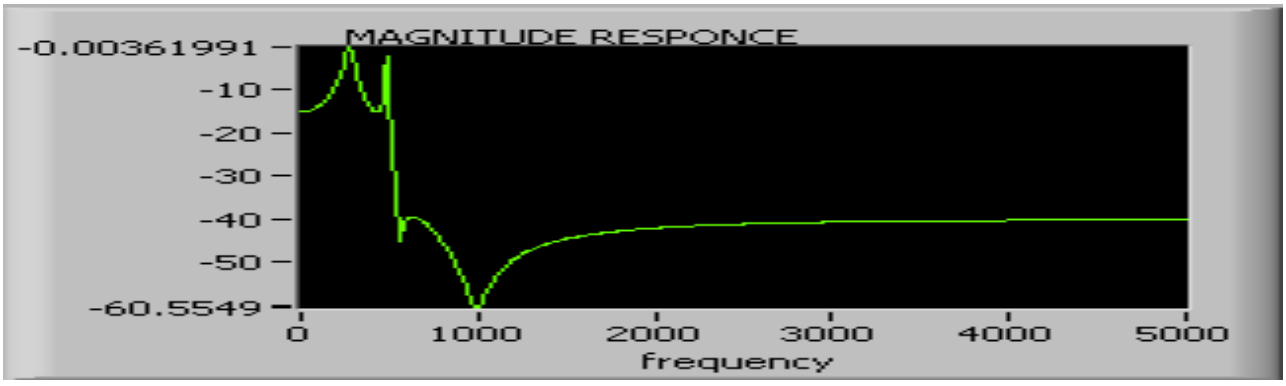


Figure 7.19(b) LabVIEW Result

Comparative Discussion on results:

In the comparison with the same specification, like cutoff frequency, stop frequency, passband attenuation, stopband attenuation, sample rate and order of the filter, transition band is improved for Butterworth, Chebyshev and Elliptic Filter as compared to MATLAB results.

It is also analyzed that the transition band becomes sharper between passband and stopband on increasing the order of filter keeping same the other parameters.
Compared to a Butterworth filter as shown in figure 7.1 Chebyshev filter figure 7.2, and Inverse Chebyshev figure 7.3, can achieve a sharper transition between the passband and the stopband with a lower order filter. The sharp transition between the passband and the stopband of these filter produces smaller absolute errors and faster execution speeds than a Butterworth filter. Compared with the same order Butterworth or Chebyshev filters, the elliptic filters figure 7.4 provide the sharpest transition between the passband and the stopband, which accounts for their widespread use.

When the LABVIEW results of every filter ,that is, Butterworth, Chebyshev and Elliptic Filter, etc were compared with the MATLAB results it is found that LABVIEW simulated filters produce improved performance over MATLAB simulated filters , when the allocation of the kind of filters like cutoff frequency, stop frequency, pass band attenuation ,stop band attenuation , sample rate ,order of filter ,etc, was kept same .

Thus it is better to simulate the filters in LABVIEW as it gives improved performance when compared with MATLAB

CHAPTER 8

CONCLUSION & FUTURE SCOPE OF THE WORK

A Digital IIR Filter system is developed which can be used for remote operation via Internet using National Instrument's LabVIEW software and Digital Signal Processing Tool kit. In this project, the design of IIR filters was considered. Several results from theory were verified in the design. The characteristics of a number of important approximations Butterworth, Chebyshev, and Elliptic were affirmed from the results obtained. Experimental results are very enthusiastic.

In LabVIEW the parameters can be changed at the time of execution of the program but in case of MATLAB it is not possible. There is smooth Transition Band in LabVIEW Design and Least Square error in case of LabVIEW is also less than MATLAB Design. LabVIEW Design is based on G-Programming so that the analysis of the performance can be done very easily. In LabVIEW analysis of all types of Filters (LP, HP, BP, and BS) is possible in single program. In MATLAB all have separate programs. Design Constraints are more accurate in case of LabVIEW because we can take good approximation on LabVIEW but in MATLAB all real things are implemented there is no approximation in programming if requiring than added on program (There is no Need of user Definition on approximation). It is also verified that on increasing the order of any filter the transition band decreases for the same parameters.

LabVIEW Design is much closed to real world implementation with DSP Kit and Filter Design kit and as per requirement we can change the all parameter at the time of implementation also. MATLAB Design implementation is fast in real world application, there is no unwanted system delay, than LabVIEW because that will be directly loaded to hardware with out large process there is no need of any interfacing. In LabVIEW the Output Window (Front panel) just like your Instrument (Virtual) so there is feeling of working on Instrument. Processing time of program execution is some nanoseconds in

LabVIEW 7.1 version but in MATLAB 7.0.4 version it is little bit high. Memory Requirement in LabVIEW design is also less than MATLAB design.

LabVIEW has been used in industrial as well as in college purposes. It is a simulation based tool. Simulation can probably be done more efficiently using specialized tools. VI has obvious advantage in integration of simulation and actual implementation as well as visualization of the results. Thus these are major savings in application development effort. Very complex instruments and control systems can be developed efficiently using LabVIEW tools. VI is used for every thing from small laboratory experiments to plant level and wide area automation. It is more user-friendly then most of the software use, like MATLAB and it has graphical codes which is easier to understand comparing it to other numerical codes.

The constraint can be implemented (Limitation of Response, just like threshold) on Filter design performance (Magnitude, Phase and Group delay) to make the performance as per our requirement .Which can help the some Specific Application just like Bar code Reader Filter, Sonar and Radar System and Edge detection in Image processing. We can check the stability of Filter with respect to pole radius and pole location also. The IIR Differentiator (This is Just like the combination of some specific Filter to predict the next response or detect the Change of out put) can be designed in LabVIEW which is most excited research topic in DSP.

REFERENCES

[1] FDS - Filter Design System for SPW, Product Data Sheet, Comdisco, Inc., 1990. A.H Gray and J. D Markel, "A Computer Program for Designing Digital Elliptic Filters. IEEE Trans. Acous., Speech, Signal Processing, vol. ASSP-24, pp.529-538, 1973

[2] J. H. McClellan. T. W. Parks and L. R. Rabiner, "A Computer Program for Designing Optimum FIR Linear Phase Digital Filters", IEEE Trans. Audio Electroacoustic, AU-2 1, pp.506-526, 1973

[3] L. R. Rabiner, N. Y. Graham, and H. D. Helms, "Linear programming design of IIR digital filters with arbitrary magnitude function," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-22, pp. 117–123, 1974

[4] A. T. Chottera and G. A. Jullien, "A linear programming approach to recursive filter design with linear phase," IEEE Trans. Circuits Syst.,vol. CAS-29, pp. 139–149, 1982

[5] M.F. Fahmy. M. Abo-Zahhad and M.1. Shoby, "Design of Selective Linear Phase Switched-Capacitor Filters with Equiripple Passband Amplitude Responses", IEEE Trans. on Circuits and Systems, CAS-35, no. 10, pp. 1220-1229, 1988

[6] M.F. Fahmy, M.Abo-Zahhad and M.I. Shoby "Design of Odd Degree Linear Phase Sampled-Data Bandpass Filters with Equiripple Amplitude Response" Inter. J. of Circuit Theory and Applications, vol. 17. pp. 87- 10 1, 1989

[7] S. Chen, "Minimum sensitivity IIR filter design using principal component approach", vol. 8, pp. 342-347, 1991

[8] V. Sreeram and P. Agathoklis, "Design of linear phase IIR filters via impulse response gramians," IEEE Trans. Signal Processing, vol. 40,pp. 389–394, 1992

[9] M. Abo-Zahhad and T. Henk. "Design of Selective Low-pass Sampled- Data and Digital Filters Exhibiting Equiripple Amplitude and Phase Error Characteristics" Inter. J. of Circuit Theory and 1 Applications, vol. 23. pp. 59-74, 1995

[10] M. Abo-Zahhad, Sabah M.A. and M. Yaseen, "Interactive Software Development for the Design and Synthesis of Lattice and Bireciprocal Wave-Digital Filters", Proc. of the 2nd Inter. Conf. on Engineering Research, ICER-95, Port-Said, Egypt, pp. 199-2 16, 1995

[11] R. C. Eberhart, Kennedy J. A new optimizer using particle swarm theory,Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995

[12] J. Kennedy, R. C. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995

[13] Proakis, J.G., and D.G. Manolakis, "Digital Signal Processing, Principles Algorithms, and Applications", 3d ed., Pearson Education, Inc., 1996

[14] Later Mohammed Abo-Zahhad et.al. , Filter designer: A complete design and synthesis program for lumped, wave-digital, FIR and IIR filters , 1996

[15] Thede Les, "Analog and Digital Filter Design using C ", Prentice Hall, Inc., Upper Saddle River, NJ, 1996

[16] Jackson L. B., "Digital Filters and Signal Processing", 3d ed., Kluwer Academic Publishers, 1996

[17] Wells, L. K. and Travis, J., "LabVIEW for Everyone Graphical Programming Made Even Easier", Prentice Hall, Inc., Upper Saddle River, New Jersey, 1997

[18] W. S. Lu, S. C. Pei, and C. C. Tseng, "A weighted least-squares method for the design of stable 1-D and 2-D IIR digital filters," IEEE Trans. Signal Processing, vol. 48, pp. 1–10, 1998

[19] Chugani, M. L., "LabVIEW Signal Processing", Prentice Hall, Inc., Upper Saddle River, New Jersey, 1998

[20] W. S. Lu, "Design of stable IIR digital filters with equiripple passbands and peak-constrained least-squares stopbands," IEEETrans. Circuits Syst. II, vol. 46, pp. 1421–1426, 1999

[21] Essik, J., “Advanced LabVIEW Labs”, Prentice Hall, Inc., Upper Saddle River, New Jersey, 1999

[22] Xi Zhang and Hiroshi Iwakura , “Design of IIR Digital Allpass Filters Based on Eigenvalue Problem” vol. 13, pp. 1333-1338, 1999

[23] Balbir Kumar and Ashwani Kumar, “Design of Efficient FIR Filters for the Amplitude Response”, vol. 16, pp. 122-125, 1999

[24] Chia-Nan Chang, Hui-Kang Teng, Jun-Yuan Chen, and Huang-Jen Chiu, Department of Electronic Engineering, National Taiwan University of Science and Technology, “Computerized Conducted EMI Filter Design System Using LabVIEW and Its Application”, vol. 15, pp. 144-149, 2000

[25] W. S. Lu, “Design of stable minimax IIR digital filters using semi definite programming,” Proc. Int. Sym. Circuits and Systems, vol. 1, pp. 355–358, 2000

[26] M. C. Lang, “Least-squares design of IIR filters with prescribed magnitude and phase responses and a pole radius constraint,” IEEE Trans. Signal Process., vol. 48, pp. 3109–3121, 2000

[27] S. Chen, R. H. Istepanian and B. L. Luk, Digital IIR filter design using adaptive simulated annealing, Digital Signal Processing, Vol.11, No.3,pp. 241-251, 2001

[28] Beyon, J. Y., “Hands-On Exercise Manual for LabVIEW Programming, Data Acquisition and Analysis”, Prentice Hall, Inc., New Jersey, 2001

[29] C. C. Tseng and S. L. Lee, “Minimax design of stable IIR digital filter with prescribed magnitude and phase responses,” IEEE Trans. Circuits Syst. I, vol. 49, pp. 547–551, 2002

[30] W. S. Lu and T. Hinamoto, “Optimal design of IIR digital filters with robust stability using conic-quadratic-programming updates,” IEEE Trans. Signal Process., vol. 51, pp. 1581–1592, 2003

[31] A. Slowik, M. Bialko, Evolutionary design of IIR digital filters with non-standard amplitude characetristics, 3-rd National Conference on Electronics, Kolobrzeg, , pp. 345-350, 2004

[32] C. C. Tseng, “Design of stable IIR digital filter based on least power error criterion,” IEEE Trans. Circuits Syst.□, vol. 51, pp. 1879-1888, 2004

[33] D. J. Krusienski, W. K. Jenkins, Particle swarm optimization for adaptive IIR filter structures, Evolutionary Computation, 2004, CEC 2004, Congress on Vol. 1, pp. 965 - 970, 2004

[34] Namjin Kim, "Digital Signal Processing System-Level Design Using LabVIEW", Elsevier Inc.,vol. 1, 122-127, 2005

[35] Oppenheim, A.V., R.W. Schafer, "Discrete Time Signal Processing", 2d ed., Pearson Education, 2005

[36] Clark C. L., "LabVIEW Digital Signal Processing and Digital Communications", Tata McGRAW-HILL, 2005

[37] N. Kehtarnavaz and C. Gope , "DSP system design using LABVIEW and SIMULINK: A comparative evaluation, vol. 4, pp. 165-169, 2006

[38] Aimin Jiang and Hon Keung Kwan,"IIR Digital Filter Design with Novel Stability Criterion Based on Argument Principle" , Department of Electrical and Computer Engineering,University of Windsor , vol. 1, pp. 126-131, 2007

[39] Adam Slowik and Michal Bialko , " Design and Optimization of IIR Digital Filters with Non-Standard Characteristics Using Particle Swarm Optimization Algorithm",Department of Electronics and Computer Science, Technical University of Koszalin, vol. 1, pp.276-282, 2007