Deployment of A Programmed Graphic User Interface (GUI) For Analysing Digital Filters Using Matlab-Algorithm

AKWUKWAEGBU ISDORE ONYEMA[1], OBICHERE JUDE-KENNEDY CHIBUZO[2], MFONOBONG ELEAZAR BENSON[3], PAULINUS-NWAMMUO CHIEDOZIE FRANCIS[4]
[1, 2, 3, 4] *Federal University of Technology Owerri, Nigeria*

Abstract- ***The time and frequency domains are confronted with unwanted noise, maximum ripple errors, voltage magnitude variations, slow data acquisition, poor distortions from frequency, phase, or delay, and harmonics. The work covers the development of the digital filter design program that helped to design low-pass, high-pass, band-pass, and band-reject filters in order to satisfy various constraints such as cutoff frequencies and maximum ripple errors. Matlab codes, commands, and syntax are used to program graphical user interface (GUI) filter design software for analyzing finite impulse response (FIR) and infinite impulse response (IIR) digital filters using the Hamming type of window and Parks-McClellan design methods. The corresponding magnitude response, phase response, impulse response, and pole-zero plot of the digital filters are displayed. A simulated real-time procedure and its varying responses due to varied input parameters are well highlighted or displayed in a graphic user interface (GUI) environment. The results revealed that the Hamming type of window deployed in the design of FIR digital filters recorded cutoff frequencies of 1000Hz, 1500Hz, 1500–2500Hz, and 1000–3000Hz for low-pass, high-pass, band-pass, and band-reject filters, respectively, with a sampling frequency of 8000Hz and an order of 20. Also, the magnitude-frequency response results of finite impulse response (IIR) digital filters designed using the Parks-McClellan design method in 9 iterations and the auto order of 20 recorded passband frequencies ranging from 1000–3000Hz and stopband frequencies ranging from 1000–4000Hz for these digital filters are presented.***

Indexed Terms- ***Graphic User Interface (GUI), Finite Impulse Response (FIR) filters, Infinite Impulse Response(IIR) digital filters, Hamming type of Window design method, Parks-McClellan design method, Magnitude-Frequency Responses, Maximum ripple errors.***

I. INTRODUCTION

Digital signal processing (DSP) refers to anything that can be done to a signal using code on a computer or DSP chip. To reduce certain sinusoidal frequency components in a signal's amplitude, digital filtering is done [1]. One may want to obtain the integral of a signal. If the signal comes from a tachometer, the integral gives the position. If the signal is noisy, then filtering the signal to reduce the amplitude of the noise frequencies improves the signal quality. For example, noise may occur from wind or rain at an outdoor music presentation. Filtering out sinusoidal components of the signal that occur at frequencies that cannot be produced by the music itself results in recording the music with little wind and rain noise. Sometimes the signal is corrupted not by noise, but by other signal frequencies that are of no present interest [2]. If the signal is an electronic measurement of a brain wave obtained by using probes applied externally to the head, other electronic signals are picked up by the probes, but the physician may be interested only in signals occurring at a particular frequency. By using digital filtering, the signals of interest can only be presented to the physician [3].

Originally, signal processing was done only on analog or continuous-time signals using analog signal processing (ASP) [4]. Until the late 1950s, digital computers were not commercially available. When they became commercially available, they were large and expensive, and they were used to simulate the performance of analog signal processing to judge its

effectiveness. These simulations, however, led to digital processor code that simulated or performed nearly the same task on samples of the signals that the analog systems did on the signals [5]. After a while, it was realized that the simulation coding of the analog system was actually a DSP system that worked on samples of the input and output at discrete time intervals. But implementing signal processing digitally, instead of using analog systems, was still out of the question [6]. The first problem was that an analog input signal had to be represented as a sequence of samples of the signal, which were then converted to the computer's numerical representation [7].

The same process would have to be applied in reverse to the output of the digitally processed signal. The second problem was that, because the processing was done on very large, slow, and expensive computers, practical real-time processing between samples of the signal was impossible [8].

The development of faster, cheaper, and smaller input signal samplers, Analogue to Digital Converters (ADCs) and output converters from digital data to analog data, Digital to Analogue Converters (DACs), began to make real-time digital signal processing (DSP) practical. Also, the processors were becoming smaller, faster, and cheaper and used more bits. Real-time replacements for analog systems may be just as small, cheap, and accurate and be able to process at a sample rate adequate for many analog signals [9]. However, testing and modification of the coding for DSP systems led to DSP systems that have no analog signal processing equivalents, yet sometimes perform the signal processing better than the DSP coding developed to replace analog systems.

The objective of this work is to design Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filters with their low-pass, high-pass, band-pass, and band-reject responses using a user-friendly Graphic User Interface (GUI) based working environment on the MATLAB platform. The codes are written in MATLAB syntax, as is the use of its GUI platform. The required system equations and expressions are incorporated into the program codes, and the operational flowcharts are designed for the sequential lines of code [10].

The significance of this work depends on its ability to perform mathematical operations on a sampled, discrete-time signal to reduce or enhance certain aspects of that signal in electronics, computer science, and mathematics. The program developed is useful in the effective teaching of filter design and analysis in electrical and electronic engineering. The graphic user interface (GUI) displays a simulated real-time procedure and its varying responses due to varied input parameters.

There are different methods of designing digital filters according to the types of filters involved. For finite impulse response (FIR) filters, the Remez exchange method or convolution method is used. In the Remez exchange algorithm, the user specifies a desired frequency response, a weighting function for errors from this response, and a filter order N. The algorithm then finds the set of N coefficients that minimizes the maximum deviation from the ideal response. Intuitively, this finds the filter that is as close to the desired response as possible given that you can use only N coefficients [11]. The convolution method involves mixing the input signals with the digital filter's impulse response (called the filter kernel). From this convolution, all possible linear filters' responses can be achieved.

Then, for infinite impulse response (IIR) filters, the recursion design method is deployed. When a filter is implemented by convolution, each sample in the output is calculated by weighting the samples in the input and adding them together. Recursion is an extension of the convolution method that uses previously calculated values from the output signal and then adds them to the input signal of the filter. Therefore, instead of using a filter kernel, recursive filters are defined by a set of recursion coefficients [12].

II. REVIEW OF LITERATURE

The tone control circuit in an ordinary car radio is a filter, as are the bass, midrange, and treble boosts in a stereo preamplifier [13]. Further examples of useful filters in audio are graphic equalizers, reverberators, echo devices, phase shifters, and speaker crossover networks. There are also examples of undesirable

filtering, such as the uneven reinforcement of certain frequencies in a room with bad acoustics.

A *digital* filter is just a filter that operates on digital signals, such as sound represented on a computer. It is a computation that takes one sequence of numbers (the input signal) and produces a new sequence of numbers (the filtered output signal) [14]. It is important to realize that a digital filter can do anything that a real-world filter can do [15].

Digital filters are used for two general purposes, namely signal separation and signal restoration. Analog (electronic) filters can be used for these same tasks. However, digital filters can achieve far superior results. Signal separation is needed when a signal has been contaminated with interference, noise, or other signals. For example, imagine a device for measuring the electrical activity of a baby's heart (EKG) while still in the womb. The raw signal will likely be corrupted by the breathing and heartbeat of the mother. A filter might be used to separate these signals so that they can be individually analyzed.

Signal restoration is used when a signal has been distorted in some way. For example, an audio recording made with poor equipment may be filtered to better represent the sound as it actually occurred. Another example is the deblurring of an image acquired with an improperly focused lens or a shaky camera [16].

It is common in digital signal processing (DSP) to say that a filter's input and output signals are in the time domain. This is because signals are usually created by sampling at regular intervals of time. But this is not the only way sampling can take place.

The second most common method of sampling is at equal intervals in space. For example, imagine taking simultaneous readings from an array of strain sensors mounted at one-centimeter increments along the length of an aircraft wing. Many other domains are possible, but time and space are by far the most common. When the term "time domain" is seen in DSP, it is remembered that it may actually refer to samples taken over time, or it may be a general reference to any domain that the samples are taken in [17].

Every linear filter has an *impulse response*, a *step response,* and a *frequency response*. Each of these responses contains complete information about the filter, but in a different form. If one of the three is specified, the other two are fixed and can be directly calculated. All three of these representations are important because they describe how the filter will react under different circumstances.

The most straightforward way to implement a digital filter is by convolution of the input signal with the digital filter's impulse response. All possible linear filters can be made in this manner. When the impulse response is used in this way, filter designers give it a special name called the *filter kernel*.

There is also another way to make digital filters called *recursion*. When a filter is implemented by convolution, each sample in the output is calculated by weighting the samples in the input and adding them together. Recursive filters are an extension of this, using previously calculated values from the output as well as points from the input. Instead of using a filter kernel, recursive filters are defined by a set of recursion coefficients [18].

To find the impulse response of a recursive filter, simply feed in an impulse and see what comes out. The impulse responses of recursive filters are composed of sinusoids that exponentially decay in amplitude. In principle, this makes their impulse responses infinitely long. However, the amplitude eventually drops below the round-off noise of the system, and the remaining samples can be ignored. Because of this characteristic, recursive filters are also called *Infinite Impulse Response or IIR filters*. In comparison, filters carried out by convolution are called *Finite Impulse Response (FIR) filters*.

It is known that the *impulse response* is the output of a system when the input is an *impulse*. In this same manner, the *step response* is the output when the input is a *step* or an *edge response*. Since the step is the integral of the impulse, the step response is the integral of the impulse response. This provides two ways to find the step response, namely, feeding a step waveform into the filter and seeing what comes out; or integrating the impulse response. Continous integration is used with continuous signals, while

discrete integration (a running sum) is used with discrete signals [19]. Frequency response can be found by taking the digital Fourier transform (DFT) using the fast Fourier transform (FFT) algorithm of the impulse response.

2.1 BRIEF THEORY OF FILTER

The most important part of any DSP task is to understand how information is contained in the signals you are working with. There are many ways that information can be contained in a signal. This is especially true if the signal is man-made. For instance, consider all of the modulation schemes that have been devised: AM, FM, single-sideband, pulse-code modulation, pulse-width modulation, etc. Fortunately, there are only two ways that are common for information to be represented in naturally occurring signals. These will be called "information represented in the time domain" and "information represented in the frequency domain."

Information represented in the time domain describes when something occurs and what the amplitude of the occurrence is. For example, imagine an experiment to study the light output of the sun. The light output is measured and recorded once every second. Each sample in the signal indicates what is happening at that instant and the level of the event. If a solar flare occurs, the signal directly provides information on the time it occurred, the duration, the development over time, etc. Each sample contains information that is interpretable without reference to any other sample. Even if only one sample from this signal is known, something about what is being measured is known. This is the simplest way for information to be contained in a signal.

In contrast, information represented in the frequency domain is more indirect. Many things in our universe show periodic motion. For example, a wine glass struck with a fingernail will vibrate, producing a ringing sound; the pendulum of a grandfather clock swings back and forth; stars and planets rotate on their axis and revolve around each other, and so forth. By measuring the frequency, phase, and amplitude of this periodic motion, information can often be obtained about the system producing the motion. The fundamental frequency and harmonics of the periodic vibration relate to the mass and elasticity of the material. A single sample, in itself, contains no information about the periodic motion and therefore no information about the wine glass. The information is contained in the relationship between many points in the signal.

The step response is useful in time domain analysis because it matches the way humans view the information contained in the signals. The step function is the purest way of representing a division between two dissimilar regions. It can mark the start of an event or the end of an event. It shows that whatever is on the left is somehow different from whatever is on the right. This is how the human mind views time-domain information: as a group of step functions dividing the information into regions of similar characteristics. The step response, in turn, is important because it describes how the dividing lines are being modified by the filter. Digital filters are less standardized, and it is common to see 99%, 90%, 70.7%, and 50% amplitude levels defined as the cutoff frequency [20].

Frequency domain filters are generally used to pass certain frequencies (the passband) while blocking others (the stopband). The four basic frequency responses are low-pass, high-pass, band-pass, and band-reject. High-pass, band-pass, and band-reject filters are designed by starting with a low-pass filter and then converting it into the desired response. For this reason, most discussions on filter design only give examples of low-pass filters. There are two methods for the low-pass to high-pass conversion: spectral inversion and spectral reversal. Both are equally useful [12, 21].

Digital filters can be implemented in two ways: by convolution (also called finite impulse response, or FIR) and by recursion (also called infinite impulse response, or IIR). Filters carried out by convolution can have far better performance than filters using recursion, but they execute much more slowly. The moving average is used in the time domain, the windowed-sinc is used in the frequency domain, and the finite impulse response (FIR) custom is used when something special is needed. Fast Fourier Transform (FFT) convolution is an algorithm for increasing the speed of convolution, allowing FIR filters to execute faster. The single-pole recursive

filter is used in the time domain, while the Chebyshev is used in the frequency domain. Recursive filters with a custom response are designed by iterative techniques [22].

The moving average filter is optimal for a common task: reducing random noise while retaining a sharp step response. This makes it the premier filter for time-domain encoded signals. However, the moving average is the worst filter for frequency domain encoded signals, with little ability to separate one band of frequencies from another. Relatives of the moving average filter include the Gaussian, Blackman, and multiple-pass moving averages. These have slightly better performance in the frequency domain, at the expense of increased computation time [23].

The moving average filter operates by averaging a number of points from the input signal to produce each point in the output signal. The equation for the moving average filter is written as:

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j] \quad (1)$$

Where x = input signal, y = output signal, and M = the number of points used in the moving average. This equation only uses points on one side of the output sample being calculated. Not only is the moving average filter very good for many applications, it is optimal for a common problem, reducing random white noise while keeping the sharpest step response [24].

2.0 DESIGN AND ANALYSIS OF FIR AND IIR DIGITAL FILTERS

The techniques used to generate the filter system's syntax representation, as well as the theoretical deductions for the areas under consideration, are highlighted. It is the first step towards translating the required formulas into executable MATLAB codes. The MATLAB compiler is made up of several phases, and these phases are all implemented as software modules that must be properly interfaced with the developed GUI's in order to produce the desired outputs. This involves translating the design specifications into lines of executable code while at the same time checking for possible errors and reporting them back to the programmer [25]. The following design procedures are considered in the program:

3.1 WINDOWED-SINC FILTERS

Windowed-sinc filters are used to separate one band of frequencies from another. They are very stable, produce few surprises, and can be pushed to incredible performance levels. These exceptional frequency domain characteristics are obtained at the expense of poor performance in the time domain, including excessive ripple and overshoot in the step response. When carried out by standard convolution, windowed-sinc filters are easy to program, but slow to execute. The Inverse Fourier The transform of this ideal frequency response produces the ideal filter kernel or impulse response. Equation (2) gives the sinc function, or sin(x)/x of the windowed – sinc.

$$h[i] = \frac{\sin(2\pi f_c i)}{i\pi} \quad (2)$$

Where: h = sinc function and f_c = cutoff frequency

Convolving an input signal with this filter kernel provides a perfect low-pass filter. The problem is, the *sinc function* continues to both negative and positive infinity without dropping to zero amplitude. While this infinite length is not a problem for mathematics, it is a show stopper for computers [26]. Several different windows are available, most of them named after their original developers in the 1950s. Only two are worth using, namely the Hamming window and the Blackman window. The Hamming window is given by equation (3).

$$w[i] = 0.54 - 0.46\cos(2\pi i/M) \quad (3)$$

Where: M = even numbers, W = Hamming window and i = Integer.

The Hamming windows run from i = 0 to M, for a total of M%1 points.

$$w[i] = 0.42 - 0.5\cos(2\pi i/M) + 0.08\cos(4\pi i/M) \quad (4)$$

The BlackMan window is given by the equation (4).

3.2 RECURSIVE FILTERS

Recursive filters are an efficient way of achieving a long impulse response without having to perform a long convolution. They execute very rapidly, but have less performance and flexibility than other digital filters. Recursive filters are also called Infinite Impulse Response (IIR) filters, since their impulse responses are composed of decaying exponentials. This distinguishes them from digital filters carried out by convolution, called Finite Impulse Response (FIR) filters. This is an introduction to how recursive filters operate and how simple members of the family can be designed [27].

Recursive filters are useful because they bypass long convolutions. For instance, consider what happens when a delta function is passed through a recursive filter. The output is the filter's impulse response and will typically be a sinusoidal oscillation that exponentially decays. Since this impulse response is infinitely long, recursive filters are often called infinite impulse response (IIR) filters.

3.3 PHASE RESPONSE

The zero phase filter is characterized by animpulse response that is symmetrical around sample zero. The actual shapedoesn't matter, only that the negative numbered samples are a mirror image ofthe positive numbered samples.

3.4 CHEBYSHEV FILTERS

Chebyshev filters are used to separate one band of frequencies from another. Although they cannot match the performance of the windowed-sinc filter, they are more than adequate for many applications. The primary attribute of Chebyshev filters is their speed, typically more than an order of magnitude faster than windowed-sinc. This is because they are carried out by recursion rather than convolution. The design of these filters is based on a mathematical technique called the z-transform [29].

The Chebyshev response is a mathematical strategy for achieving a faster roll-off by allowing ripples in the frequency response. Analog and digital filters that use this approach are called Chebyshev filters. These filters are named after their use of the Chebyshev polynomials, developed by the Russian mathematician Pafnuti Chebyshev (1821–1894) [30]. Chebyshev filters achieve a faster roll-off by allowing ripples in the passband. When the ripple is set to 0%, it is called a maximally flat or Butterworth filter. The Chebyshev filters here are called type 1 filters, meaning that the ripple is only allowed in the passband. In comparison, type2 Chebyshev filters have ripple only in the stopband. Type 2 filters are seldom used.

Type 1 Chebyshev filters are the most common Chebyshev filters. The gain (or amplitude) response as a function of angular frequency of the nth order low-pass filter is given by equation (5).

$$G_n(\omega) = |H_n(j\omega)| = \frac{1}{\sqrt{1+\varepsilon^2 T_n^2\left(\frac{\omega}{\omega_0}\right)}} \quad (5)$$

Where: ε is the ripple factor, ω_i is the cutoff frequency and Tn is a Chebyshev polynomial of the nth order.

The transfer function is then given by

$$H(s) = \frac{1}{2^{n-1}\varepsilon} \prod_{m=1}^{n} \frac{1}{(s - s_{pm}^{-})} \quad (6)$$

Where s_{pm}^{-} are only those poles with a negative sign in front of the real term in the above equation for the poles.

Type 2 Chebyshev filters are known as inverse Chebyshev. The type 2 is less common because it does not roll off as fast as type I, and requires more components. It has no ripple in the passband, but equiripple is present in the stopband. The gain is given by equation (7).

$$G_n(\omega, \omega_0) = \frac{1}{\sqrt{1 + \frac{1}{\varepsilon^2 T_n^2(\omega_0/\omega)}}}. \quad (7)$$

The poles of gain of the type II Chebyshev filter will be the inverse of the poles of the type I filter:

$$\frac{1}{s_{pm}^{\pm}} = \pm \sinh\left(\frac{1}{n}\operatorname{arsinh}\left(\frac{1}{\varepsilon}\right)\right)\sin(\theta_m) + j\cosh\left(\frac{1}{n}\operatorname{arsinh}\left(\frac{1}{\varepsilon}\right)\right)\cos(\theta_m) \quad (8)$$

The transfer function will be given by the poles in the left half plane of the gain function, and will have the same zeroes but these zeroes will be single rather than double zeroes.

Butterworth filter showed that low pass filters could be designed whose frequency response (gain) is given by equation (9).

$$G = \sqrt{\frac{1}{1+\omega^{2n}}} \quad (9)$$

Normalized Butterworth polynomials resulted to equation (10).

$B_n(s) = \prod_{k=1}^{\frac{n}{2}}\left[s^2 - 2s\cos\left(\frac{2k+n-1}{2n}\pi\right)+1\right]$ for n even

$$B_n(s) = (s+1)\prod_{k=1}^{\frac{n-1}{2}}\left[s^2 - 2s\cos\left(\frac{2k+n-1}{2n}\pi\right)+1\right]$$

for n odd (10)

III. IMPLEMENTATION OF THE DIGITAL FILTER PROGRAM

The digital filter design is a program that designs low-pass, high-pass, band-pass, and band-reject filters to satisfy various constraints such as cutoff frequencies and maximum error. In addition to the infinite impulse response (IIR) Kaiser filters, finite impulse response (FIR) filters are equally designed using the Windows and Parks-McClellan design methods, with their corresponding magnitude response, phase response, impulse response, and pole-zero plot displayed.

Procedures involved in an algorithm for the digital filter development program are presented as follows:

1. Users can specify filter constraints such as sampling frequency, pass band and stop band cutoff frequencies, as well as pass band and stop band errors.
2. Tolerance indicators, denoted by red lines, can be moved around by selecting them and pulling the mouse around, with an instant update of the filter's frequency response.
3. The filter's numerator and denominator coefficients can be either exported to an external file or exported to the workspace when variable names are specified by the user.
4. Various plots for phase, pole-zero, and impulse responses are available as context menus.
5. Various plot options enable the tool to be effectively used as a lecture aid in a classroom environment; that is, the width and color of the lines in the plots can be changed. The filter can be either of the options to 'zoom-in' or 'grid-on' are also provided [31].

The MATLAB-algorithm program for the implementation of this work is shown in figure 1.

The first four steps of the filter design process relate to the filter specifications object, while the last two steps involve the filter implementation object. Both of these objects are discussed in more detail in the following sections. Step 5: The Design of the Filter is the transition step from the filter specifications object to the implementation object. The analysis and verification step are completely optional. It provides a means for the filter designer to ensure that the filter complies with all design criteria. Depending on the results of this verification, steps 3 and 4 can be looped together, to either choose a different algorithm, or customize the current one. Figure 1 illustrates the help command for each step. Enter the help line at the MATLAB command prompt to receive instructions and further documentation links for the particular step.

Not all of the steps have to be executed explicitly. For example, you could move step 1 directly to step 5, and then the interim three steps are done by the Filter Design Toolbox. The details of each of the steps taken are discussed as follows[32, 33, 34, 35, 36]:

Step 1: Selecting a Response: If help fdesign/responses are typed at the MATLAB command prompt, a complete list of all possible filter responses available in the Filter Design Toolbox will be seen. After a response is chosen, say band-pass, the design of the Specifications Object is started by typing the following: d = fdesign.band-pass. This step cannot be skipped, nor is it automatically completed for the designer by the Filter Design Toolbox. A response is selected to initiate the filter design process.

Step 2: Decide on a SpecificationA specification is an array of design parameters for a given filter. The specification itself is a property of the specification object. It can be noted that a specification is not the same as the specifications object, but rather a specifications object contains a specification as one of its properties. When a filter response is selected, there are a number of different specifications available, each containing a different combination of design parameters. In the following example, first set the filter response and then ask for the specifications listing.

Step 3: Choose an AlgorithmThe availability of algorithms depends on both the chosen filter response and the design parameters. In other words, for the same low-pass filter, changing the specification string also changes the available algorithms.

Step 4: Customizing the Algorithm: The customization options available for any given algorithm depend not only on the algorithm itself, selected in Step 1, but also on the specification selected in Step 2. To explore all the available options, type the following at the MATLAB command prompt:

Step 5: Designing the Filter: This next task introduces a new object, the filter object, or dfilt.To create a filter, the design command is used.

Step 6: Design Analysis: The filter is designed and then analyzed to determine if all the design criteria are satisfied. In the filter design toolbox, analysis is broken into the following three main sections:

1. frequency domain analysis, which includes magnitude response, group delay, and poll zero.
2. Time domain analysis, which includes impulse response,
3. Step response implementation analysis, which includes quantization noise and cost.

Step 7: Realize or Apply the Filter to Input Data: After the filter is designed and optimized, it can be used to filter actual input data. The basic filter command takes input data x, filters it through the Filter Object, and produces output y: >> y = filter (FilterObj, x).

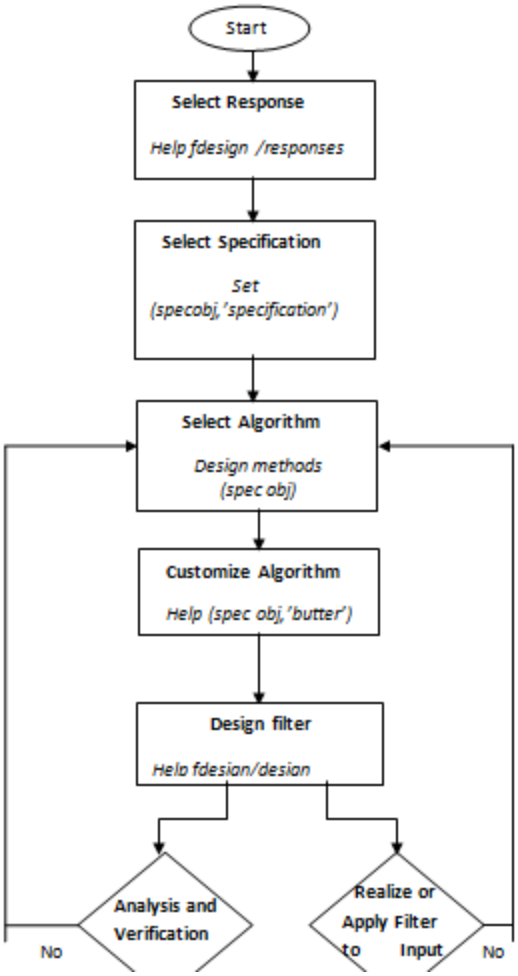

Figure 1: Program Flowchart for Implementation of Digital Filter Using MATLAB.

IV. RESULTS AND DISCUSSION

Some of the possible results of the filter design program are represented graphically and then discussed as follows:

1. Magnitude-Frequency Response of Finite Impulse Response (FIR) digital filters using the Hamming type of window design method

The results of magnitude-frequency responsesof Finite Impulse Response (FIR) digital low-pass, high-pass, band-pass, and band-reject filters generated using the Hamming type of window design method and an arbitrary chosen sampling frequency of 8000Hz and cut-off frequencies are presented graphically in figures 2 to 5 respectively.

a) Low-pass Filter Design using the Hamming Type of Window Design Method

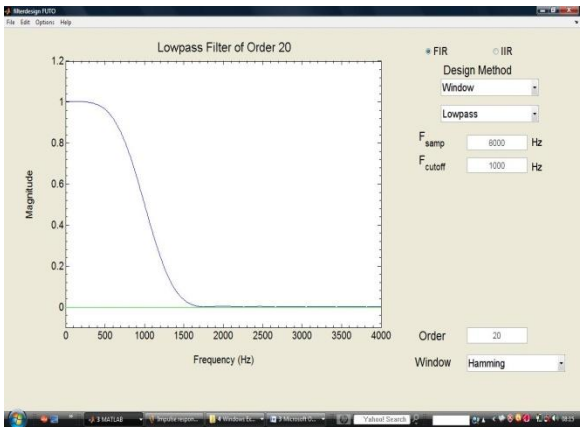

Figure 2: Magnitude-Frequency response of FIR (low-pass) filters using window design method.

The response displayed in figure 2 shows a Finite Impulse Response (FIR) filter of a low-pass characteristics that passes frequencies lower than the cut-off frequency (i.e. F_{cutoff} = 1000Hz) with a little guard band and attenuates frequencies higher than the cut-off frequency.

b) High-pass Filter Design using Hamming Type of Window Design Method:

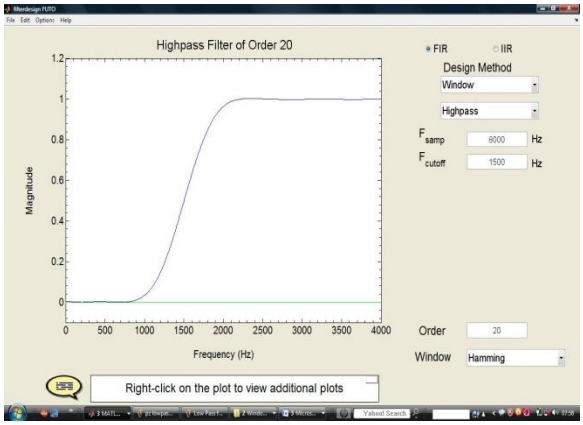

Figure 3: Magnitude-Frequency response of FIR (high-pass) filters using window design method.

The response displayed in figure 3 shows a Finite Impulse Response (FIR) filter of a high-pass characteristics that blocks frequencies lower than the cut-off frequency (i.e. F_{cutoff} = 1500Hz) with a little guard band and passes frequencies higher than the cut-off frequency. The cut-off frequency recorded is 1500Hz ($0< F_{cutoff}<0.5*F_{samp}$) with order of 20.

c) Band-pass Filter Design using Hamming Type of Window Design Method:

The response displayed in figure 4 shows a Finite Impulse Response (FIR) filter of band-pass characteristics that blocks frequencies within a specified frequency band (1500-2500Hz) and passes frequencies within the specified band. The result of magnitude-frequency response of FIR band-pass filter using hamming type of window design method maintained the cut-off frequencies 1 and 2 ranging from 1500Hz ($0< F_{cutoff}<0.5*F_{samp}$) to 2500Hz ($0< F_{cutoff}<0.5*F_{samp}$) with the order of 20.

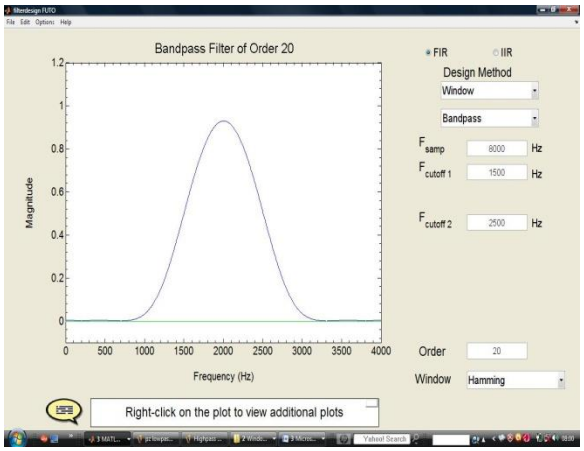

Figure 4: Magnitude-Frequency response of FIR (band-pass) filters using window design method.

d) Band-reject Filter Design using Hamming Type of Window Design Method:

The response displayed in figure 5 shows a Finite Impulse Response (FIR) filter of band-pass characteristics that blocks frequencies within a specified frequency band (1000-3000Hz) and passes frequencies outside the specified band. The cut-off frequencies 1 and 2 recorded in band-pass ranged from 1000Hz ($0< F_{cutoff}<0.5*F_{samp}$) to 3000Hz ($0< F_{cutoff}<0.5*F_{samp}$) with the order of 20.

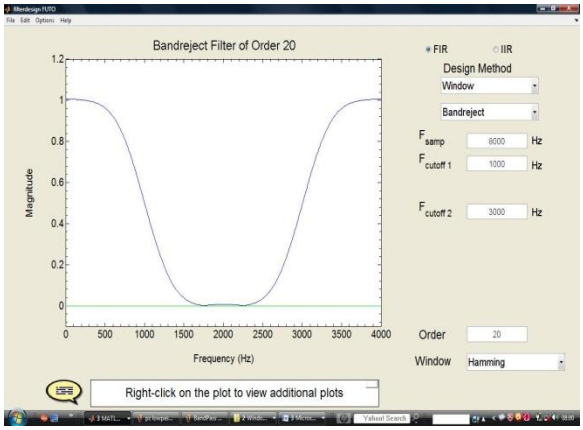

Figure 5: Magnitude-Frequency response of FIR (band-reject) filters using window design method.

2. Magnitude-Frequency Response of Finite Impulse Response (FIR) filters using Parks-McClellan design method:

The results of the magnitude-frequency response of the Finite Impulse Response (FIR) low-pass, high-pass, band-pass, and band-reject digital filters generated using the Parks-McClellan design method and an arbitrary chosen sampling frequency of 8000Hz, cut-off, passband, and stopband filter frequencies are shown graphically in figures 6 to 9 respectively.

(a) Low-pass Filter Design using the Parks-McClellan design method

The response displayed in Figure 6 shows a Finite Impulse Response (FIR) filter of low-pass characteristics that passes frequencies within the passband (1000Hz) and blocks frequencies above the stopband. A finite impulse response filter of low-pass digital filter designed in 9 iterations with the auto order of 20 recorded passband and stopband frequencies of 1000Hz and 1500Hz, respectively, with the same maximum ripple error value of 0.043545 in passband ($_{pass}$)and stopband ($_{stop}$) for an extremal frequency of 12.

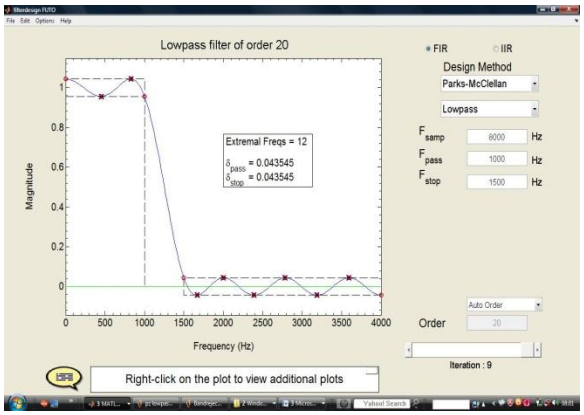

Figure 6: Magnitude-Frequency response of FIR (low-pass) filters using Parks-McClellan design method.

(b) High-pass Filter using Parks-McClellan Design Method:

The response displayed in Figure 7 shows a Finite Impulse Response (FIR) filter of high-pass characteristics that passes frequencies higher than the passband frequency (1500Hz) and blocks frequencies lower than the stopband frequency (1000Hz). A finite impulse response filter with a high-pass filter operation and design in 9 iterations with an auto order of 20 recorded passband and stopband frequencies of 1500Hz and 1000Hz, with the same maximum ripple error value of 0.0433545 in both the passband ($_{pass}$) and stopband ($_{stop}$) for an extremal frequency of 12.

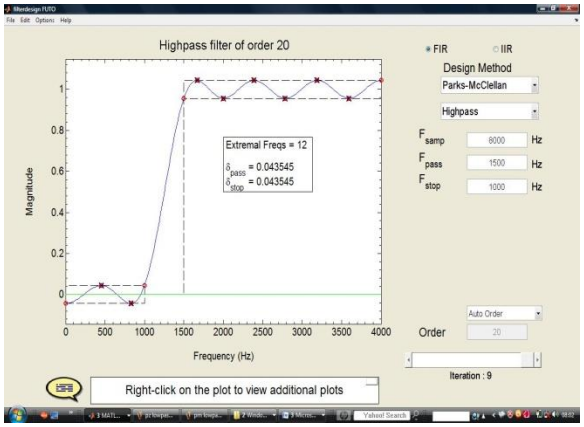

Figure 7: Magnitude-Frequency response of FIR (high-pass) filters using Parks-McClellan design method.

(c) Band-pass Filter using Parks-McClellan Design Method:

The response displayed in figure 8 shows a Finite Impulse Response (FIR) filter of band-pass characteristics that passes frequencies within a specified frequency band (1500-2500Hz) and blocks frequencies outside the specified band. A finite impulse response filter of high-pass filter operation and design in 9 iterations with auto order of 20 recorded passband frequencies ranging from 1500-2500Hz, Stopband Frequencies ranging from 0-1000Hz and 3000-4000Hz, the same maximum ripple error value of 0.045162 in both passband (δ_{pass}) and stopband (δ_{stop}) for extremal frequency of 12.

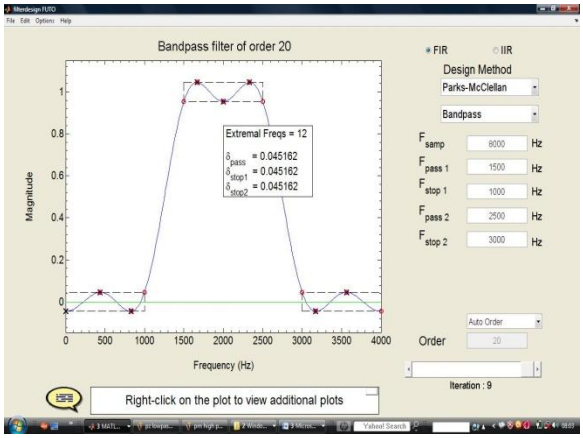

Figure 8: Magnitude-Frequency response of FIR (band-pass) filters using Parks-McClellan design method.

(d) Band-reject Filters using Parks-McClellan Design Method:

The response displayed in figure 9 shows a Finite Impulse Response (FIR) filter of band-reject characteristics that passes frequencies outside a specified frequency band (1000-3000Hz) and blocks frequencies within the specified band. A finite impulse response filter of band-reject filter operation and design in 9 iterations with auto order of 20 recorded passband frequencies ranging from 1000-3000Hz, Stopband Frequencies ranging from 0-1000Hz and 3000-4000Hz, the same maximum ripple error value of 0.045162 in both passband (δ_{pass}) and stopband (δ_{stop}) for extremal frequency of 6.

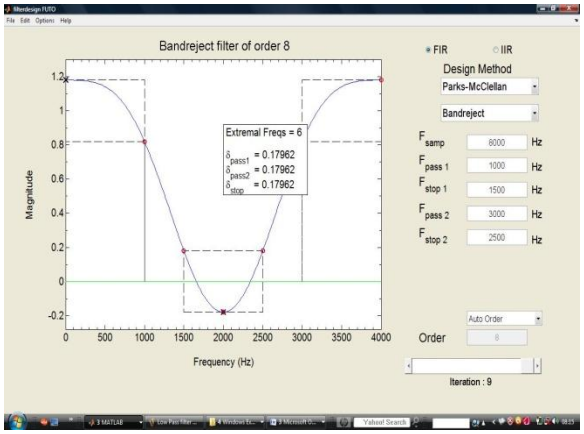

Figure 9: Magnitude-Frequency response of FIR (band-reject) filters using Parks-McClellan design method.

1. Magnitude-Frequency Response of Infinite Impulse Response (IIR) filters:

The results of magnitude-frequency Response of Infinite Impulse Response (IIR) filters of arbitrary chosen sampling frequency of 8000Hz, cut-off, passband, stopband, maximum ripple errors for both passband and stopband filter frequencies are shown graphically in figures 10 to 13 respectively.

(a) Butterworth Low-pass Filter:

The response displayed in figure10 shows an Infinite Impulse Response (IIR) filter of a Butterworth low-pass characteristics that passes frequencies lower than the passband frequency (i.e. F_{pass} = 1000Hz) with a little guard band up to stopband frequency (1500Hz) and attenuates frequencies higher than the stopband frequency. An infinite impulse response (IIR) filter of a Butterworth low-pass filter recorded the same maximum ripple error value of 0.04 in both passband (δ_{pass})and stopband (δ_{stop}) with auto order of 10.

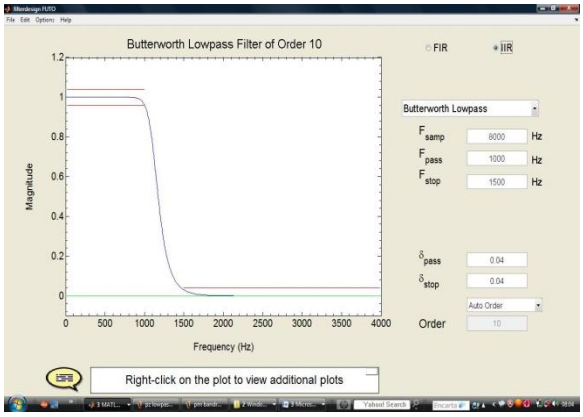

Figure 10: Magnitude-Frequency Response of IIR Butterworth (low-pass) filters.

(b) Butterworth High-pass Filter:

The response displayed in figure11 shows an Infinite Impulse Response (IIR) filter of a Butterworth high-pass characteristics that passes frequencies higher than the passband frequency (i.e. F_{pass} = 1500Hz) with a little guard band up to stopband frequency (1000Hz) and attenuates frequencies lower than the stopband frequency. An infinite impulse response (IIR) filter of a Butterworth high-pass filter recorded the same maximum ripple error value of 0.04 in both passband (δ_{pass})and stopband (δ_{stop}) with auto order of 10.

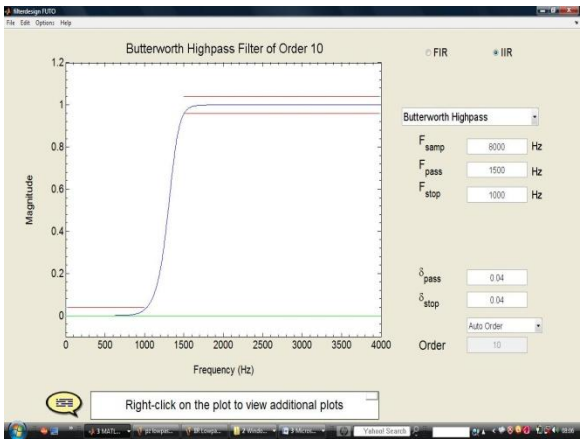

Figure 11: Magnitude-Frequency Response of IIR Butterworth (high-pass) filters.

(c) Butterworth Band-pass Filter:

The response displayed in figure 12 shows an Infinite Impulse Response (IIR) filter of a Butterworth band-pass characteristics that passes frequencies within a specified frequency band (1500-2500Hz) and blocks frequencies outside the specified band. An infinite impulse response (IIR) filter of a Butterworth band-pass filter recorded stopband frequencies ranging from 1000-3500Hz and the same maximum ripple error value of 0.04 in both passband (δ_{pass})and stopband (δ_{stop}) with auto order of 10.

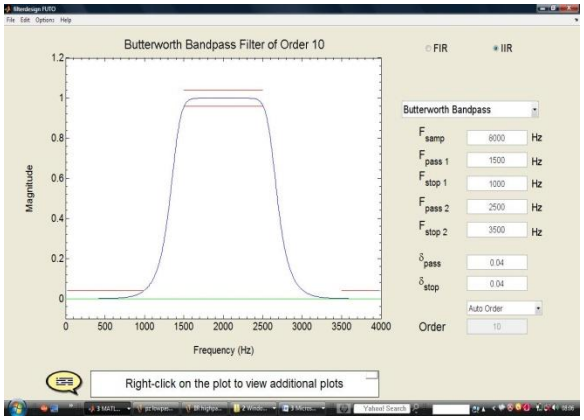

Figure 12: Magnitude-Frequency Response of IIR Butterworth (band-pass) filters.

(d) Butterworth Band-reject Filter:

The response displayed in figure 13 shows an Infinite Impulse Response (IIR) filter of a Butterworth band-reject characteristics that blocks frequencies within a specified frequency band (1500-2500Hz) and passes frequencies outside the specified band. An infinite impulse response (IIR) filter of a Butterworth band-reject filter recorded stopband frequencies ranging from 1000-3500Hz and the same maximum ripple error value of 0.04 in both passband (δ_{pass})and stopband (δ_{stop}) with auto order of 16.

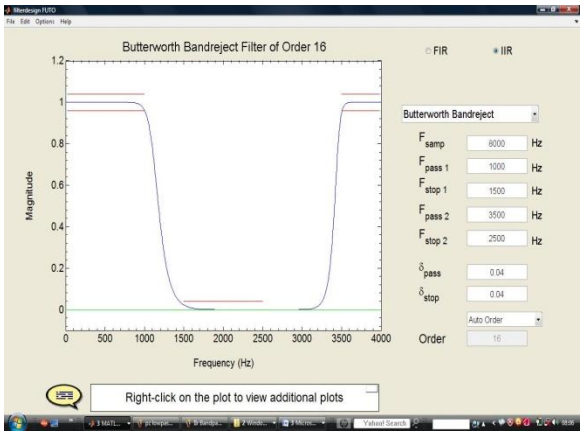

Figure 13: Magnitude-Frequency Response of IIR Butterworth (band-reject) filters.

The diagrams (filter responses) shown in figures 2 to 13 are only magnitude-frequency response of the different types of the digital filters (low-pass, high-pass, band-pass, band-reject, Butterworth low-pass, Butterworth high-pass, Butterworth band-pass, Butterworth band-reject). The filter design program is also deployed to analyze phase, impulse and pole-zero plot of the different types of the digital filters discussed in this paper, as illustrated in figures 14 to 16 respectively.

(a) Phase Response of Low-pass Filter:
The response displayed in figure 14 shows a phase-frequency response of a Finite Impulse Response (FIR) filter that passes frequencies lower than the cut-off frequency (i.e. F_{cutoff} = 1000Hz) with a little guard band and attenuates frequencies higher than the cut-off frequency. Hamming type of window design method is deployed to design and analyze the phase-frequency response of FIR (low-pass) filter with sampling frequency (F_{samp}) of 8000Hz and order of 10.

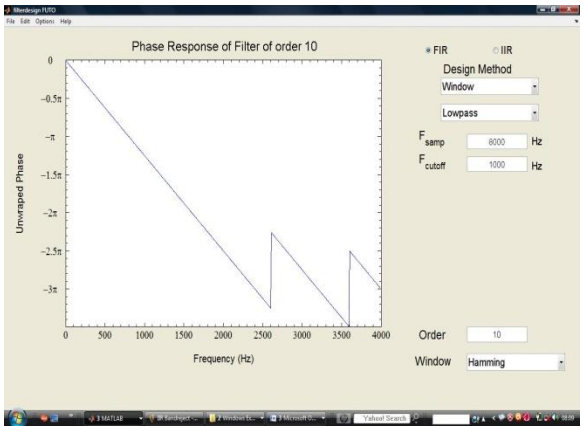

Figure 14: Phase-Frequency Response of FIR (low-pass) filters.

(b) Impulse Response of Low-pass Filter:
The response displayed in figure15 shows an Impulse Response of a Finite Impulse Response (FIR) filter that passes frequencies lower than the cut-off frequency (i.e. F_{cutoff} = 1000Hz) with a little guard band and attenuates frequencies higher than the cut-off frequency. Hamming type of window design method is deployed to design and analyze the phase-frequency response of FIR (low-pass) filter with sampling frequency (F_{samp}) of 8000Hz and order of 20.

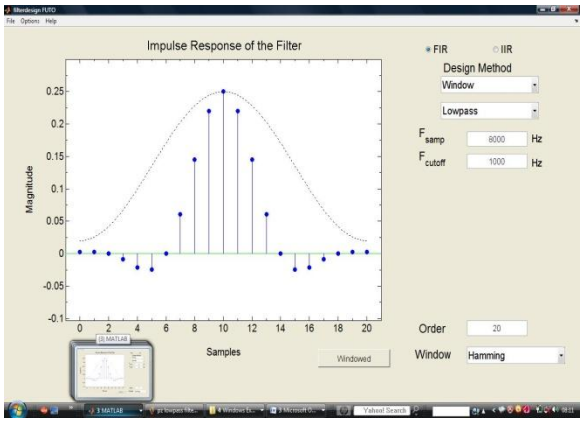

Figure 15: Impulse Response of FIR (low-pass) filters.

(c) Pole-Zero Plot of Low-pass Filter:
The response displayed in figure 16 shows a Pole-Zero Plot of a Finite Impulse Response (FIR) filter that passes frequencies lower than the cut-off frequency (i.e F_{cutoff} = 1000Hz) with a little guard band and attenuates frequencies higher than the cut-off frequency. Hamming type of window design method is deployed to design and analyze the pole-zero plot of FIR (low-pass) filters with the sampling frequency (F_{samp}) of 8000Hz and order of 20.

Figure 16: Pole-Zero Plot of FIR (low-pass) filters.

CONCLUSION

Matlab codes, commands, and syntax are deployed to program graphical user interface (GUI) filter design software for analyzing different types of digital filters, especially finite impulse response (FIR) and infinite impulse response (IIR) filters using the Hamming type of window and Parks-McClellan design methods. The work covered the graphical

programming and evaluation of the magnitude-frequency responses of the different types of digital filters (low-pass, high-pass, band-pass, band-reject, Butterworth low-pass, Butterworth high-pass, Butterworth band-pass, Butterworth band-reject). The phase, impulse, and pole-zero plot of the different types of digital filters are also analyzed and evaluated in this paper.

The programming results revealed that the low-pass filter and high-pass filter recorded the same maximum ripple error value of 0.043545, whereas the band-pass filter and band-reject filter equally recorded the same maximum ripple error value of 0.045162 in both the passband ($_{pass}$)and stopband ($_{stop}$) for the extremal frequency range of 6-12. The magnitude-frequency responses of the infinite impulse response (IIR) filters of the Butterworth low-pass, high-pass, band-pass, and band-reject filters at an arbitrary sampling frequency of 8000Hz recorded the same maximum ripple error value of 0.04 in both the passband ($_{pass}$)and the stopband ($_{stop),\ with}$ pass frequencies ranging from 1000-2500Hz and stopband frequencies ranging from 1000-3500Hz. The phase-frequency, impulse and pole-zero plot responses of finite impulse response (FIR) low-pass filters presented the cut-off frequency of 1000Hz with the sampling frequency of 8000Hz and the order ranging from 10–20.

REFERENCES

[1] J. O Smith III, Introduction to Digital Filters with Audio Applications, Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, September 2018 Edition.

[2] www.wikipedia.org/digital filter, 2021.

[3] A. Antoniou, Digital filters: Analysis, Design, and Applications, New York, NY McGraw-Hill, 1993.

[4] Sigmon, K., MATLAB Primer, Department of Mathematics, University of Florida, 2020.

[5] Phillips and Nagle, Digital Signal Processing Systems Analysis and Design, Prentice Hall, 2018.

[6] Brogan, William L, Modern Control Theory, 3rd Edition, ISBN 0135897637, 1999.

[7] Dorf and Bishop, Modern Signal Processing Systems, 10th Edition, Prentice Hall, 2005.

[8] Chen, Chi-Tsong, Linear System Theory and Design, 3rd Edition, ISBN 0195117778, 2000.

[9] The MathWorks, Getting Started with Signal Processing Systems Toolbox 8, The MathWorks, Natick, MA, 2013–2020.

[10] The Mathworks, Getting Started with MATLAB Version 7, The MathWorks, Natick, MA, 2015–2020.

[11] The Mathworks, Using Simulink Version 6, The MathWorks, Natick, MA, 2017–2020.

[12] The Mathworks, Getting Started with Simulink1 Control Design 2, The MathWorks, Natick, MA, 2016–2020.

[13] Getting Started with MATLAB, The MathWorks, Inc., 2018.

[14] D. Hanselman and B. Littlefield, Mastering MATLAB 5, A Comprehensive Tutorial and Reference, Prentice Hall, Upper Saddle River, NJ, 2018.

[15] K. Sigmon, MATLAB Primer, CRC Press, Boca Raton, 2018.

[16] Using MATLAB, the MathWorks, Inc., 2018.

[17] B.D. Hahn, Essential MATLAB for Scientists and Engineers, John Wiley & Sons, 2019.

[18] D.R. Hill and D.E. Zitarelli, Linear Algebra Labs with MATLAB, Prentice Hall, 2020.

[19] D. Hanselman and B. Littlefield, Mastering MATLAB 5, A Comprehensive Tutorial and Reference, Prentice Hall, 2020.

[20] P. Marchand, Graphics and GUIs with MATLAB, CRC Press, 2020.

[21] Using MATLAB, the MathWorks, Inc., 2020.

[22] Using MATLAB Graphics, the MathWorks, Inc., 2020.

[23] B.D. Hahn, Essential MATLAB for Scientists and Engineers, John Wiley & Sons, New York, NY, 1997.

[24] D.R. Hill and D.E. Zitarelli, Linear Algebra Labs with MATLAB, Prentice Hall, 2018.

[25] B. Kolman, Introductory Linear Algebra with Applications, Prentice Hall,1997.

[26] R.E. Larson and B.H. Edwards, Elementary Linear Algebra, Third edition, D.C. Heath and Company, Lexington, MA, 1996.

[27] S.J. Leon, Linear Algebra with Applications, Fifth edition, Prentice Hall, Upper Saddle River, NJ, 1998.

[28] G. Strang, Linear Algebra and Its Applications, Academic Press, FL, 2020.

[29] The MathWorks Inc. MATLAB 7.0 (R14SP2), The MathWorks Inc., 2020.
[30] S. J. Chapman, MATLAB Programming for Engineers, Thomson, 2019.
[31] C. B. Moler, Numerical Computing with MATLAB, Siam, 2019.
[32] C. F. Van Loan, Introduction to Scientic Computing, Prentice Hall, 2020.
[33] D. J. Higham and N. J. Higham, MATLAB Guide, Siam, 2020.
[34] SIMULINK User's Guide, the MathWorks Inc, 2020.
[35] Leonard, N. E. and W. S. Levine, Using MATLAB to Analyze and Design Signal Processing Systems, The Benjamin/Cummings Publishing Company, Inc, 2020.
[36] MATLAB User's Guide, the MathWorks Inc, 2020.