K.L.E Society's

K.L.E. INSTITUTE OF TECHNOLOGY

HUBBALLI-580027

(AICTE approved, affiliated to VTU and ISO 9001:2015 Certified Institute)

Department of Electronics and Communication Engineering.

(NBA Accredited Program)

LABVIEW MANUAL ON DIGITAL FILTERS

2020-2021

Under the Guidance of:	**Team B8**	**USN**
Mr. Santhosh Hosamane	1. Praveen Kadappanavar	2KE17EC055
	2. Prashanth Joshi	2KE17EC052
	3. Prathik Bannimath	2KE17EC054
	4. Sandeep Goudar	2KE17EC066

CONTENT

1. INTRODUCTION TO LABVIEW

1.1 HISTORY OF LABVIEW:

- LabVIEW is a visual programming language developed by National Instruments.
- It was launched for various operating systems like Microsoft Windows, Mac OS, Linux, Unix.
- It was Launched in the Year 1986. It began as a commercial software and requires License to run.
- National Instruments has released a non-commercial use LabVIEW called LabVIEW NXG and LabVIEW Community Editions on April 28th, 2020.
- The non-commercial version was released to encourage students to learn and develop their skills in LabVIEW.

1.2 ABOUT LABVIEW:

- LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is a graphical programming language that uses icons instead of lines of text to create applications.
- In contrast to text-based programming languages that use instructions to determine the order of program execution, LabVIEW uses dataflow programming.
- In data flow programming, the flow of data through the nodes on the block diagram determines the execution order of the VIs and functions.
- VIs, or virtual instruments, are LabVIEW programs that imitate physical instruments.
- In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel.
- After you build the front panel, you add code using graphical representations of functions to control the front panel objects.
- You add this graphical code, also known as G code or block diagram code, to the block diagram. The block diagram somewhat resembles a flowchart.
- The block diagram, front panel, and graphical representations of code compose a VI.
- LabVIEW is unique for its capabilities to interface with and control different types of hardware modules and instruments from NI (National Instruments).
- LabVIEW is a software that does not require much of coding proficiency from the user.

2.INTRODUCTION TO LABVIEW ADD-ONS

2.1 LABVIEW ADD-ONS:

- Since LabVIEW is a graphical Programming Language it is difficult to include all the facilities under a single bundle of software.
- For that reason, NI(national Instruments) have introduced a concept called Add-Ons.
- Add-Ons cover graphical instruments regarding a particular field of Application. This makes it easier for user to install only those options which are necessary for them and reduces the difficulty of looking throughout the software for required functionalities.
- Out of many Add-Ons or Toolkits available in the LabVIEW Software, the ones required for our work are, DFD (Digital Filter Design) Toolkit, VAS (Vision Acquisition Software) and Vision Development Module (VDM) Toolkit.

I. DFD (Digital Filter Design) Toolkit:

- As the name implies this software is mainly focused on helping users to design and develop Digital Filters of different parameters. The toolkit also supports user with options required to develop in different ways also.
- Digital Filter Design Toolkit also includes tools for single-rate and multi-rate digital filter design, floating-point to fixed-point conversion, filter analysis, simulation on a desktop computer, and filter bank design and analysis.
- The **Main Features** of DFD Toolkit Involves:
  a) **Comprehensive Analysis Tools :** You can use the Filter Analysis VIs to evaluate the characteristics of digital filters. You can examine the frequency response, group delay, phase delay, impulse response, step response, and pole-zero placement of a digital filter.
  b) **Large Selection of Filter Structures :** When you design digital filters with the Digital Filter Design Toolkit, you can select from one of 23 possible filter structures based on design methods such as Elliptic, Butterworth, Chebyshev, Kaiser etc the filter structures involve FIR Direct form, IIR direct form and filters also based on cascaded 2^{nd} order.

c) **Special Digital Filter Design:** The Special Filter Design VIs help you design IIR notch/peak filters, IIR comb filters, maximally flat filters, narrowband filters, and group delay compensators.

d) **Adaptive Filter Design:** The Adaptive Filters VIs such as AFT Create FIR LMS which creates a FIR adaptive filter with standard least mean square algorithm, help you create adaptive filters with different algorithm such as LMS (Least Mean Square) and RLS (Recursive Least Square). Adaptive filters extract meaningful signal features by adjusting filter coefficients automatically according to the dynamic conditions of the input signals and the environment.

e) **Filter Bank Design and Analysis:** The Digital Filter Design Toolkit includes VIs that you can use to design and analyse filter banks.

II. Vision Acquisition Software (VAS):

- **Description:**

  The Vision Acquisition Software (VAS) is a set of drivers and utilities used to acquire, display, and save images from a wide range of camera types, including cameras using GigE Vision, IEEE 1394 (FireWire), USB 2.0, USB 3 Vision, or the Camera Link standard.

- **Components:** VAS is made up of the following three drivers.

  a. NI-IMAQ - Acquisition from National Instruments frame grabbers, as well as general display, file saving and acquisition functions
  b. NI-IMAQdx - Acquisition from GigE Vision, IEEE-1394 cameras, USB cameras that are Direct Show compliant (VAS 2009 onwards), and some IP cameras.
  c. NI-IMAQ I/O - Configuration of FPGA included in the PCI-8254R, PCI-8255R, Compact Vision System.

- **Usage:** The drivers included with NI Vision Acquisition Software (VAS) allow you to acquire, display, and save images from a wide range of cameras. However, only the NI VDM and NI VBAI software packages include built-in support for advanced image processing and analysis. Note that a license for VAS is included with all NI Frame Grabbers .

III. Vision Development Module (VDM):

- **Description:** Vision Development Module (VDM) provides machine vision and image processing functions for LabVIEW, C/C++, Visual Basic, and .NET environments.
- **Included In:** VDM is a separately licensed module installed into LabVIEW.
- **Components:** VDM includes the Vision Assistant tool, a prototyping and code generation tool like NI Vision Builder in its menu-driven interface. Vision Assistant is a useful tool for developing and testing a series of image analysis and processing steps from which code can be generated.
- **Usage:** In general, VDM functions are used for image processing and analysis. Some example uses include: pattern matching, particle analysis, edge detection, thresholding, histograms, and optical character recognition.
- The Vision Development Module is supported on Windows and LabVIEW Real-Time. This means you can use VDM on remote targets such as the NI CVS (Compact Vision System) or an NI Smart Camera.

3. INTRODUCTION TO VI'S IN LABVIEW

3.1 LABVIEW VIRTUAL INSTRUMENTS (VIs):

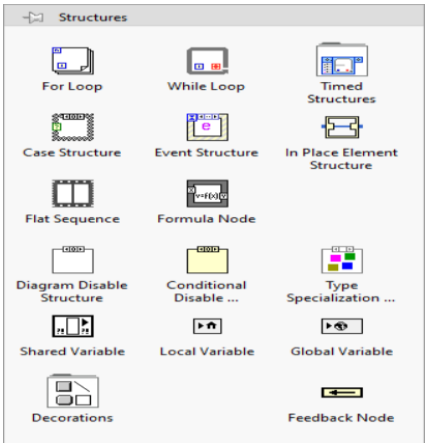
There are many Virtual Instruments (VI's) in the LabVIEW and some of the important ones are mentioned below, these can be accessed by Function's palette and Controls palette by right clicking on Block diagram and Front panel. The Functions chosen on block diagram appear as their respective controls on front panel, they are:

I. Structures:

Owning Palette- Programming VIs and Functions.

Requires- Base Development System.

It consists of For loop, while loop, case structures. These structures are used to enclose the VIs in a loop or implement different cases for a same basic circuit.

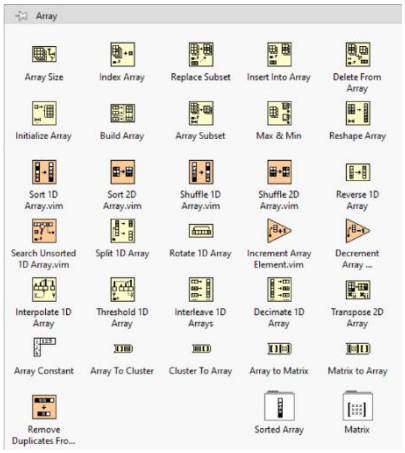


II. Array:

Owning Palette- Programming VIs and Functions.

Requires- Base Development System.

It consists of different types of arrays as shown in the figure. Used to store multiple dimensions of array.

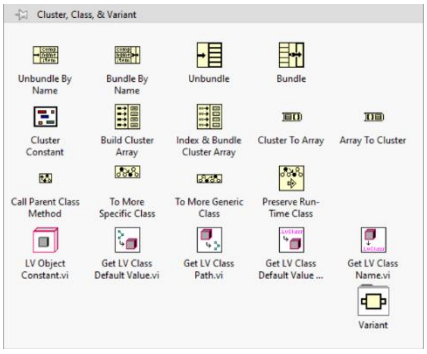


III. Clusters, class, and variant:

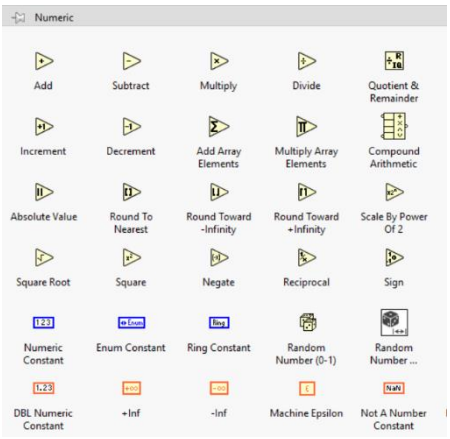
Owning Palette- Programming VIs and Functions.

Requires- Base Development System.

It consists of VIs used to form a cluster of elements from selection.



IV. Numeric:

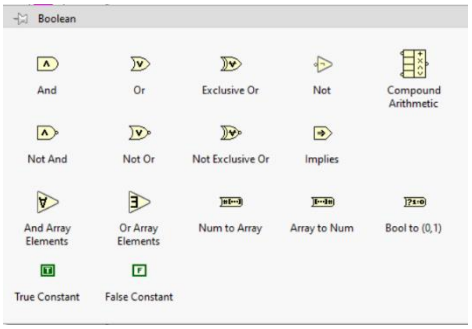


Owning Palette- Programming VIs and Functions.

Requires- Base Development System.

It consists of Numeric type VIs which are used to perform mathematical operations such as addition, subtraction, multiplication etc.

V. Boolean:

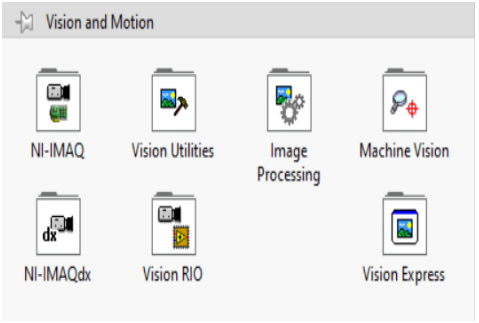


Owning Palette- Programming VIs and Functions.

Requires- Base Development System.

It consists of VIs used to perform logical operations like and, not, or, and true or false operations.

VI. Vision and Motion:

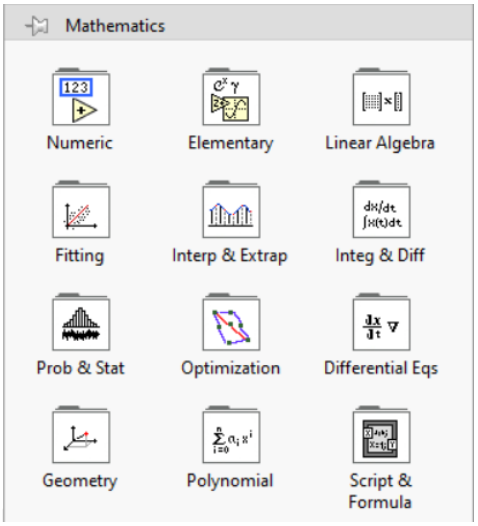


Owning Palette- Functions.

Requires- Vision Acquisition Software and Vision Development Module.

The respective folders contain many other VIs which are mentioned in the upcoming experiments.

VII. Mathematics:



Owning Palette: Functions.

Requires- Base Development system.

It consists of mathematical VIs that can be used to perform mathematical operations.

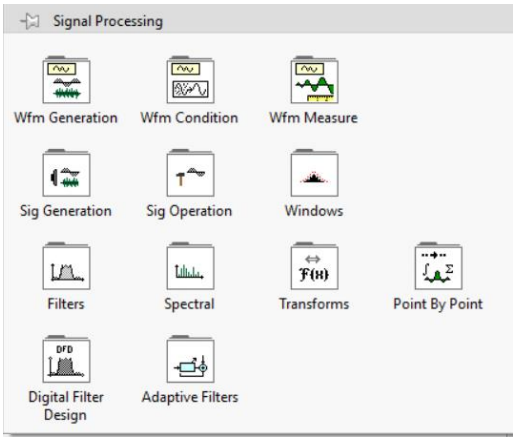
on data as well as digital signals.

VIII. Signal processing:

Owning Palette: Functions.

Requires: DFD Toolkit.

It consists of VIs responsible for signal generation, waveform visualization, signal operations and Filters both digital and analog.

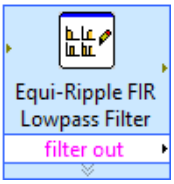


IX. Miscellaneous: These are some of the VIs belonging to one of the folders classified above and are often used in the experiments appearing in this manual and below is the detailed overview of those VIs.

- **Classical Filter Design Express VI:**

Owning Palette: Filter Design VIs.

Requires: Digital Filter Design Toolkit.



Contains the following options:

Filter types: Specifies the type of filter that this VI creates. The valid values include Lowpass, High pass, Band Pass, and Band Stop. The default is Lowpass.

Filter Specifications: Contains the following options.

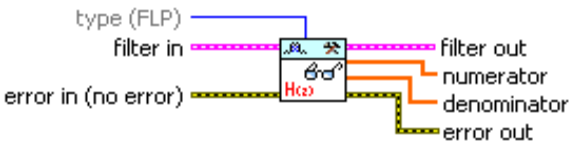
Sampling frequency, Passband edge frequency 1&2, Passband ripple, Stopband edge frequency 1&2, Stopband attenuation.

Design method: Specifies the method that this Express VI uses to design the filter. Options include Butterworth, Chebyshev, Inverse Chebyshev, Elliptic, Kaiser Window, Dolph-Chebyshev Window, and Equi-Ripple FIR. The default is Elliptic.

- **DFD Get Transfer Function VI:**

Owning Palette: Utilities VIs.

Requires: Digital Filter Design Toolkit.



Type specifies the type of transfer function to retrieve.

Filter In specifies the input filter

Error In describes error conditions that occur before this node runs. This input provides standard error in functionality.

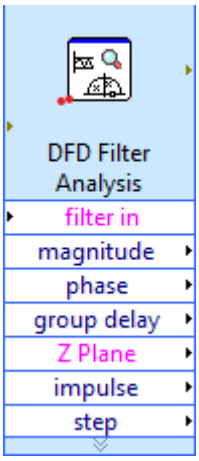
Filter Out returns the filter in unchanged.

Numerator returns the numerator polynomial of the transfer function in ascending order of z^–1.

Denominator returns the denominator polynomial of the transfer function in ascending order of z^–1.

Error Out contains error information. This output provides standard error out functionality.

- **Digital Filter Analysis Express VI:**



Owning Palette: Filter Analysis VIs.

Requires: Digital Filter Design Toolkit.

This Analyses the specified characteristics of a filter.

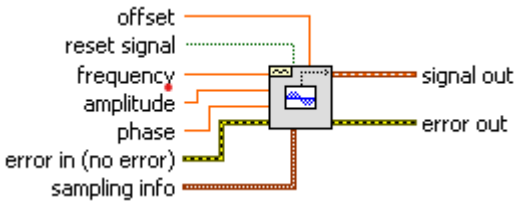
Parameter: show impulse response, show step response, show pole-zero plot, show magnitude response, show phase response, Show group delay, Magnitude, Phase.

- **Sine Waveform VI :**

Owning Palette**:** Waveform Generation VIs.

Requires: Full Development System.

This type Generates a waveform containing a sine wave.



Offset is the DC offset of the signal. The default is 0.0.

Reset Signal, if TRUE, resets the phase to the **phase** control value and the time stamp to zero. The default is FALSE.

Frequency is the frequency of the waveform in units of hertz. The default is 10.

Amplitude is the amplitude of the waveform. The amplitude is also the peak voltage. The default is 1.0.

Phase is the initial phase, in degrees, of the waveform. The default is 0. The VI ignores **phase** if **reset signal** is FALSE.

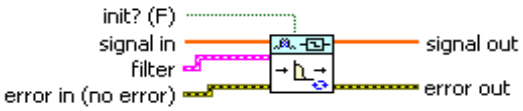
Error In describes error conditions that occur before this node runs. This input provides standard error in functionality.

Sampling Info contains sampling information.

Signal Out is the generated waveform.

Error Out contains error information. This output provides standard error out functionality.

- **DFD Filtering Array:**



Owning Palette: Processing VIs

Requires: Digital Filter Design Toolkit

This Filters an input signal continuously.

Wire data to the **signal in** input to determine the polymorphic instance to use or manually select the instance.

Init? controls the initialization of internal states. The default is FALSE, in which this VI initializes internal states from the final states of the previous call to the current VI instance. If you select TRUE, this VI initializes internal states to zero.

Signal In specifies the input array of single-channel samples you want to process.

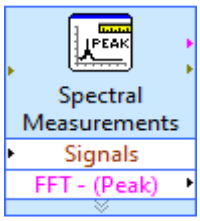
Filter specifies the input filter.

Error In describes error conditions that occur before this node runs. This input provides standard error in functionality.

Signal Out returns an array of filtered single-channel samples.

Error Out contains error information. This output provides standard error out functionality.

- **Spectral Measurements:**



Owning Palette: Signal Analysis Express VIs.

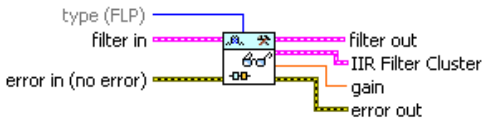
Requires: Full Development System.

This Performs FFT-based spectral measurements, such as the averaged magnitude spectrum, power spectrum, and phase spectrum on a signal.

Contains the following options:

Magnitude (Rms), Magnitude (Peak), Power Spectrum, Power Spectral Density.

- **DFD Get Cascaded Coefficient VI:**



Owning Palette: Utilities VIs.

Requires: Digital Filter Design Toolkit.

It Converts a filter to an infinite impulse response (IIR) filter cluster that is compatible with the IIR Filter Cluster output in the Advanced IIR Filtering VIs.

- **Simulate Signal Express VI:**

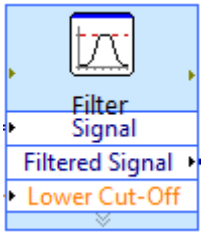
Owning Palette: Input Express VIs.

Requires: Base Development System.

It Simulates a sine wave, square wave, triangle wave, sawtooth wave, or noise signal.

Parameter: Signal, Timing, Time Stamps, Reset Signal.

- **Filter Express VI:**

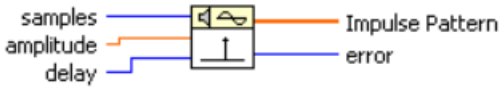


Owning Palette: Signal Analysis Express VIs.

Requires: Full Development System.

It Processes signals through filters and windows.

- **Impulse Pattern:**



Owning Palette: Signal Generation VIs.

Requires: Full Development System.

It Generates an array containing an impulse pattern.

If the Impulse Pattern is represented by the sequence *X*, the Impulse Pattern VI generates the pattern according to the following equation.

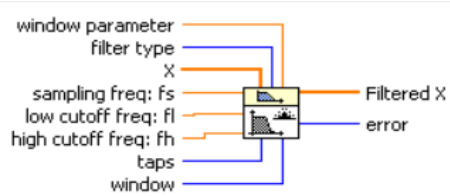
X_i={A if i = d.

X_i ={0 elsewhere.

for $i = 0, 1, 2, \ldots, n - 1$,

where *A* is the amplitude, *d* is the delay, and *n* is the number of samples.

- **FIR Windowed Filter:**



Owning Palette: Filters VIs.

Requires: Full Development System.

It Filters the input data sequence, 'X'**,** using the set of windowed FIR filter coefficients specified by the sampling frequency: fs**,** low cut-off frequency: fl**,** high cut-off frequency: fh, and number of taps**.** Wire data to the **X** input to determine the polymorphic instance to use or manually select the instance.

- **Zero Padder Array:**

Owning Palette: signal Operation VIs.

Requires:Full Development System.

It Resizes the input sequence Input Array to the next higher valid power of 2, sets the new trailing elements of the sequence to zero, and leaves the first n elements unchanged, where n is the number of samples in the input sequence. Wire data to the Input Array input to determine the polymorphic instance to use or manually select the instance.

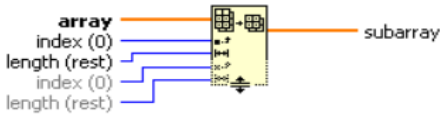
- **FFT (Fast Fourier Transform) VI:**

Owning Palette: Transforms VIs.

Requires: Full Development system.

It Computes FFT of the input sequence X. Wire data to the X input to determine the polymorphic instance to use or manually select the instance.

- **Array Subset Function:**



Owning Palette: Array VIs and Functions.

Requires: Base Development system.

It returns a portion of array starting at index and containing length elements.

- **Complex to Polar Function:**

Owning Palette: Complex Functions.

Requires: Base Development System.

It Breaks a complex number into its polar components.

- **Unwrap Phase:**

Owning Palette: Signal Operation VIs.

Requires: Full Development System.

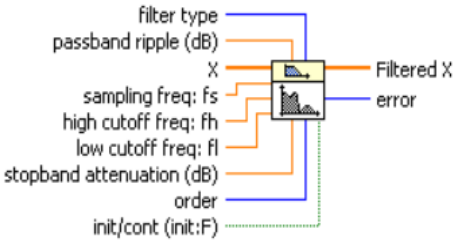
It Unwraps the phase array by eliminating discontinuities whose absolute values exceed either pi or 180^0 , depending on the units you specify in phase unit.

- **Important Filter VIs:**

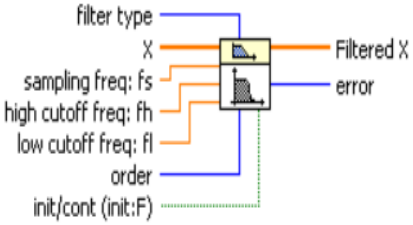
  Owning palette: Filter VIs

  Requires: Full Development System.

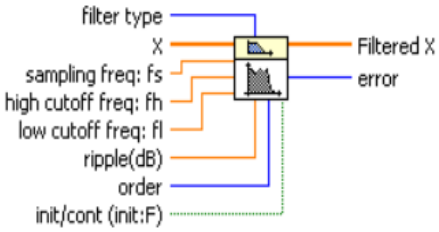
  All the Filter VIs shown below generate their respective filters by calling their type of Coefficients VI. For example, Elliptic filter VI generates a digital Elliptic filter by calling the Elliptic Coefficients VI.



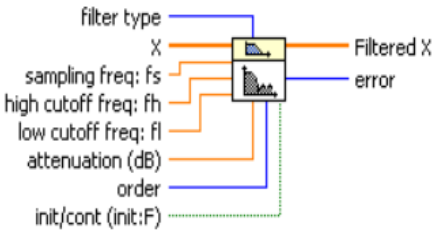
a.Elliptic Filter



b. Butterworth Filter



c.Chebyshev Filter



d.Inverse Chebyshev Filter

EXPERIMENT 1 a.

DESIGN OF FIR LPF AND FIR HPF USING RECTANGULAR WINDOW

Aim: To design FIR lowpass and high pass filter using rectangle window for the lower cutoff frequency= 100Hz for LPF, whereas for HPF lower cutoff frequency= 300Hz and sampling frequency=1000Hz.

Theory:

FIR Filter:

- Finite impulse response (FIR) filter is a filter whose impulse response is of finite duration, because it settles to zero in finite time which may have internal feedback and may continue to respond indefinitely .
- The impulse response of an N^{th} order discrete-time FIR filter lasts exactly N+1 samples (from first nonzero element through last nonzero element) before it then settles to zero.
- FIR filters can be discrete-time or continuous-time, and digital or analog.

For a causal discrete-time FIR filter of order *N*, each value of the output sequence is a weighted sum of the most recent input values**:**

$$Y(n) = \sum_{K=0}^{N} b_K x(n-k)$$

where**:**

- x(n) is the input signal.
- y(n) is the output signal.
- N is the filter order.
- An N^{th} order filter has N+1 term on the right-hand side.

Windowing:

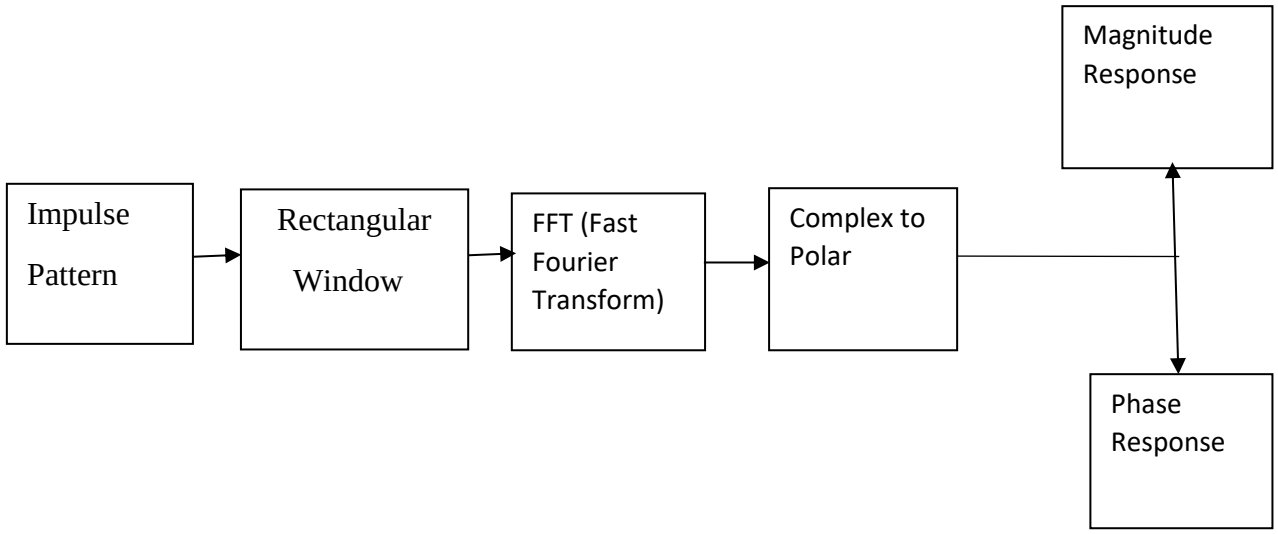
windowing means that we multiply the desired (usually ideal) infinite duration impulse response $h_d(n)$ by a finite duration window (or window function) w(n) to get a soft truncation.

Rectangular window:

$$w[n] = \begin{cases} 1, & 0 \le n \le M, \\ 0, & otherwise\ . \end{cases}$$

Where, M is the window length in samples.

Block diagram:



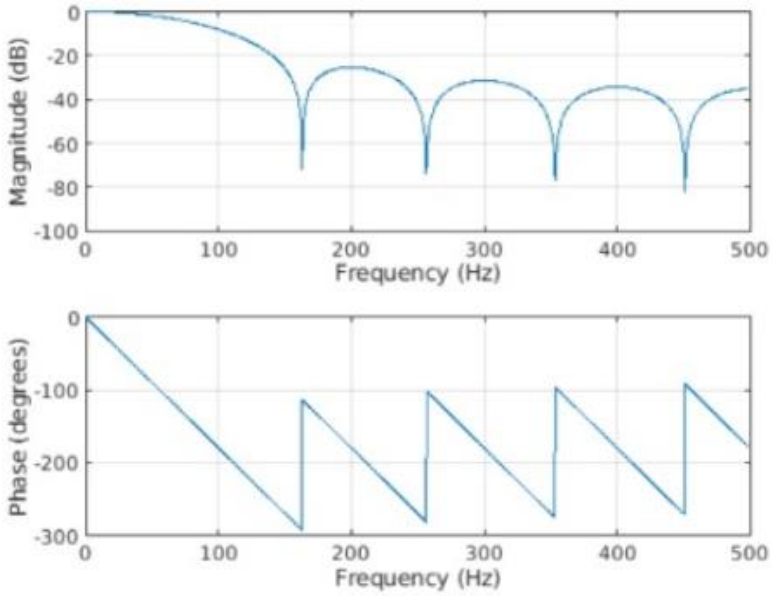
Procedure:

1. Import the desired impulse response block.
2. Import the FIR windowed filter.vi block
3. Specify the cutoff frequency and sampling frequency according to the problem statement. Also specify the low pass and high pass filter type and rectangular windowing function.
4. Import zero padder.vi and connect its input port to the output port of the FIR windowed filter.vi block.
5. Import FFT and connect its input port to the output port of the zero padder.vi block.
6. The output of FFT is connected to 'complex to polar' block to convert complex output into polar form to obtain the graphs.
7. Then the magnitude component is given to a case structure to opt between logarithmic or linear magnitude response.
8. Then both magnitude and phase responses are observed in the Front panel graphs.
9. Note: The Filter type can be changed from the front panel to choose between different types of filters and same goes for Windowing technique also.

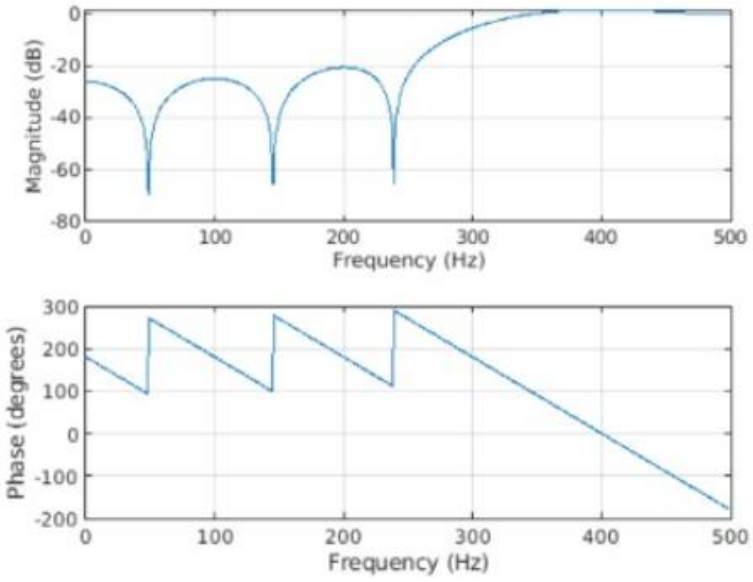
Waveforms:

Following are the expected Magnitude and Phase responses of the respective filters.

a) FIR Low Pass Filter using Rectangular Window:

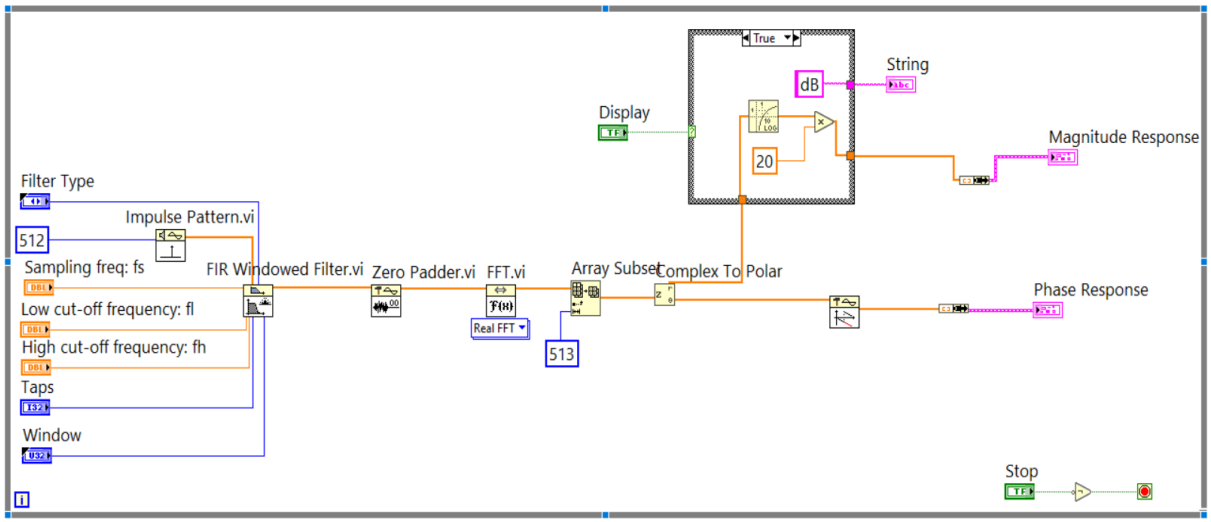


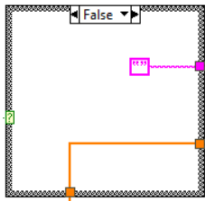
b) FIR High Pass Filter using Rectangular Window:



Block Diagram created in LabVIEW for FIR Filters:

a). FIR Low Pass Filter and High Pass Filter using Rectangular Window:

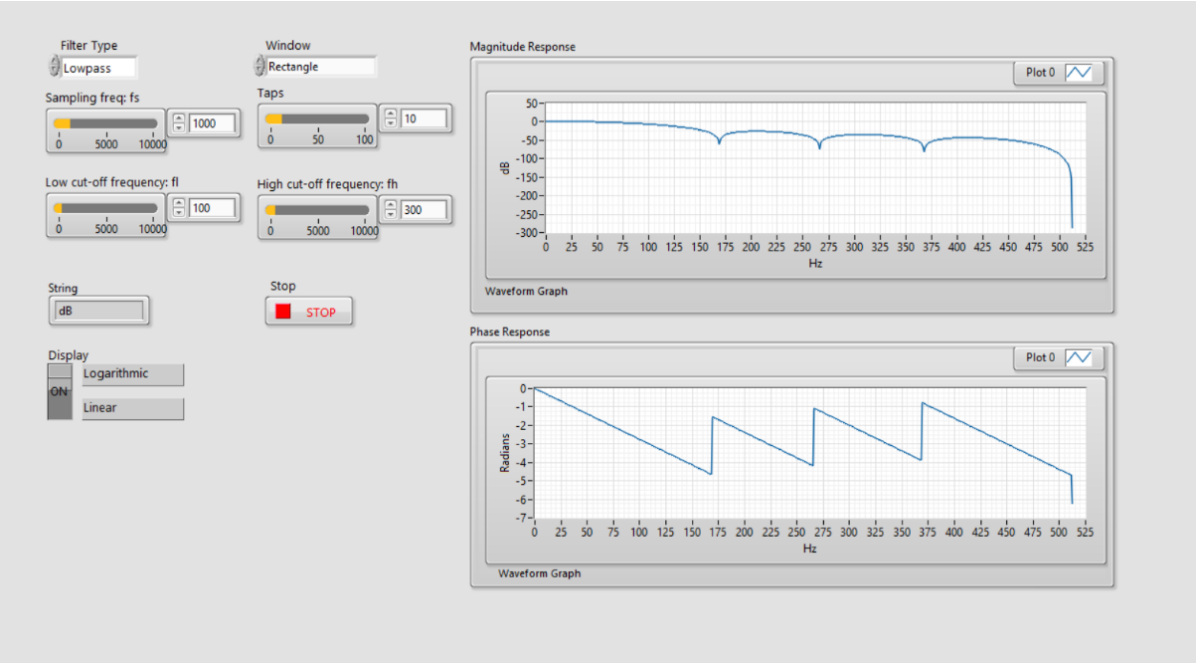




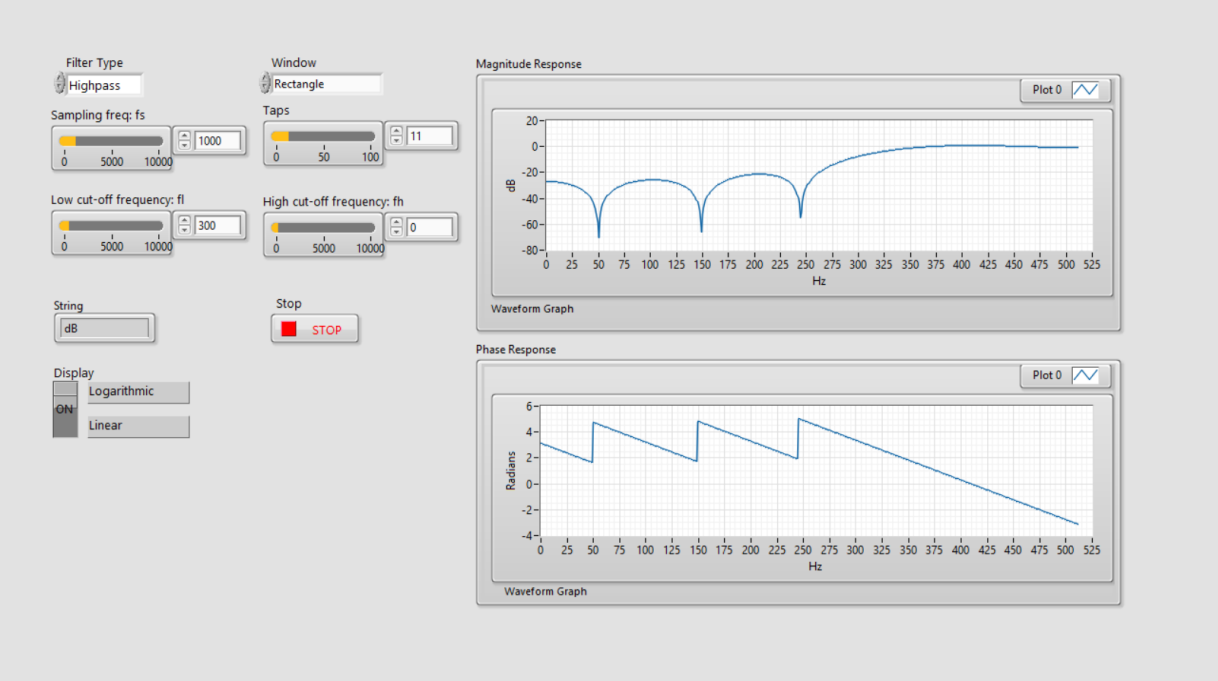
In above figure the "True Case" can be replaced by the "False Case" to obtain the output in linear form instead of Logarithmic form and the filter type can be changed between LPF and HPF.

Front Panels Created in LabVIEW for FIR Filters:

a) Front panel of FIR LPF Using Rectangular Window:



b) Front panel of FIR HPF Using Rectangle Window:



Conclusion: The FIR Low pass Filter and FIR High pass Filter are designed according to the given specification and the magnitude and phase response of both the filters are verified.

EXPERIMENT 1 b.

DESIGN OF FIR BAND STOP FILTER USING HAMMING WINDOW

Aim: To design FIR band stop filter using hamming window for the Lower cutoff frequency= 2000Hz, Higher cutoff frequency= 3500Hz and sampling frequency=8000Hz.

Theory:

FIR Filter:

- Finite impulse response (FIR) filter is a filter whose impulse response is of finite duration, because it settles to zero in finite time which may have internal feedback and may continue to respond indefinitely .
- The impulse response of an N^{th}-order discrete-time FIR filter lasts exactly N+1 samples (from first nonzero element through last nonzero element) before it then settles to zero.
- FIR filters can be discrete-time or continuous-time, and digital or analog.

For a causal discrete-time FIR filter of order *N*, each value of the output sequence is a weighted sum of the most recent input values**:**

$$Y(n) = \sum_{K=0}^{N} b_K x(n-k)$$

where**:**

- x(n) is the input signal.
- y(n) is the output signal.
- N is the filter order.
- An N^{th} order filter has N+1 terms on the right-hand side.

Band Stop:

- The band stop filter, also known as a band reject filter, passes all frequencies except for those within a specified stop band which are greatly attenuated.
- Also, just like the band pass filter, the band stop (band reject or notch) filter is a second order (two-pole) filter having two cut-off frequencies, commonly known as

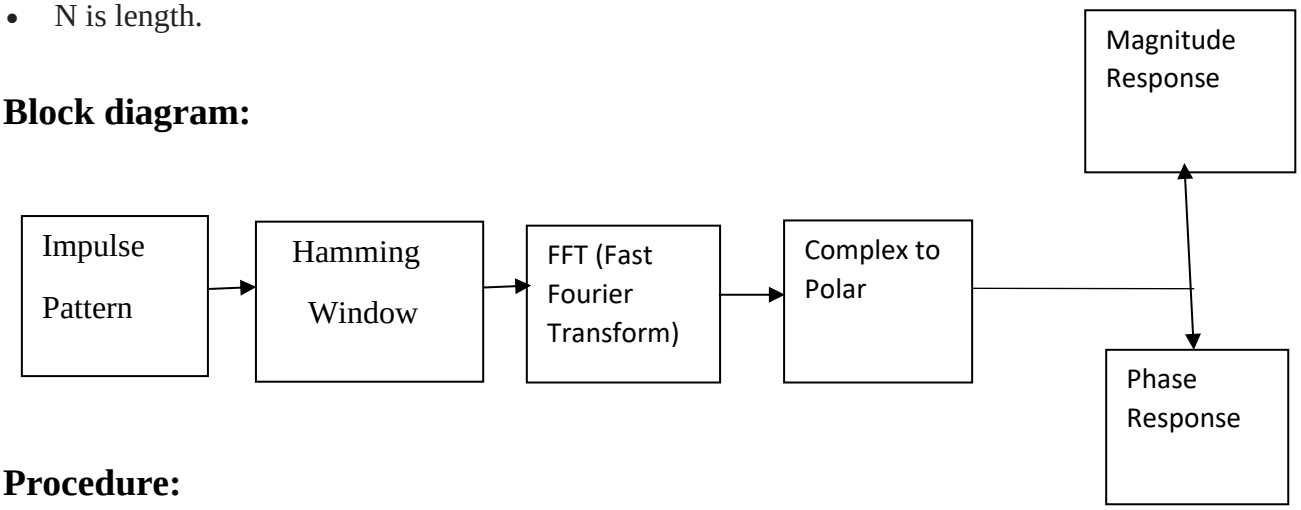
- The -3dB or half-power points producing a wide stop band bandwidth between these two -3dB points.

Hamming Window:

$$w(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \le n \le M-1$$

- M is the filter order, an M^{th} order filter has M+1 terms on the right-hand side.
- w(n) is the windowing function.
- N is length.

Block diagram:



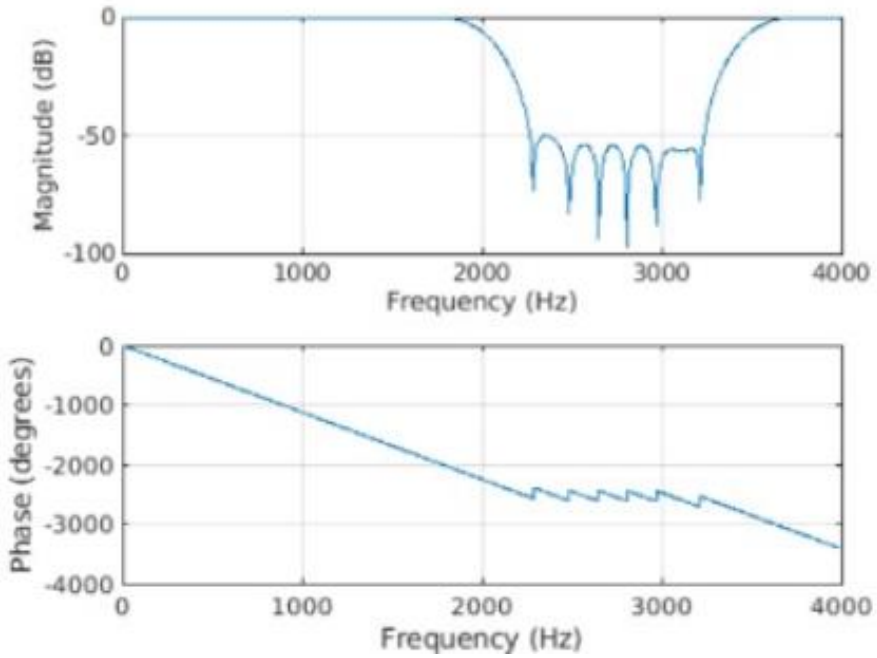
Procedure:

1. Import the desired impulse response block.
2. Import the FIR windowed filter.vi block
3. Specify the cutoff frequency and sampling frequency according to the problem statement. Also specify the low pass and high pass filter type and rectangular windowing function.
4. Import zero padder.vi and connect its input port to the output port of the FIR windowed filter.vi block.
5. Import FFT and connect its input port to the output port of the zero padder.vi block.
6. The output of FFT is connected to 'complex to polar' block to convert complex output into polar form to obtain the graphs.
7. Then the magnitude component is given to a case structure to opt between logarithmic or linear magnitude response.
8. Then both magnitude and phase responses are observed in the Front panel graphs.

9. Note: The Filter type can be changed from the front panel to choose between different types of filters and same goes for Windowing technique also.

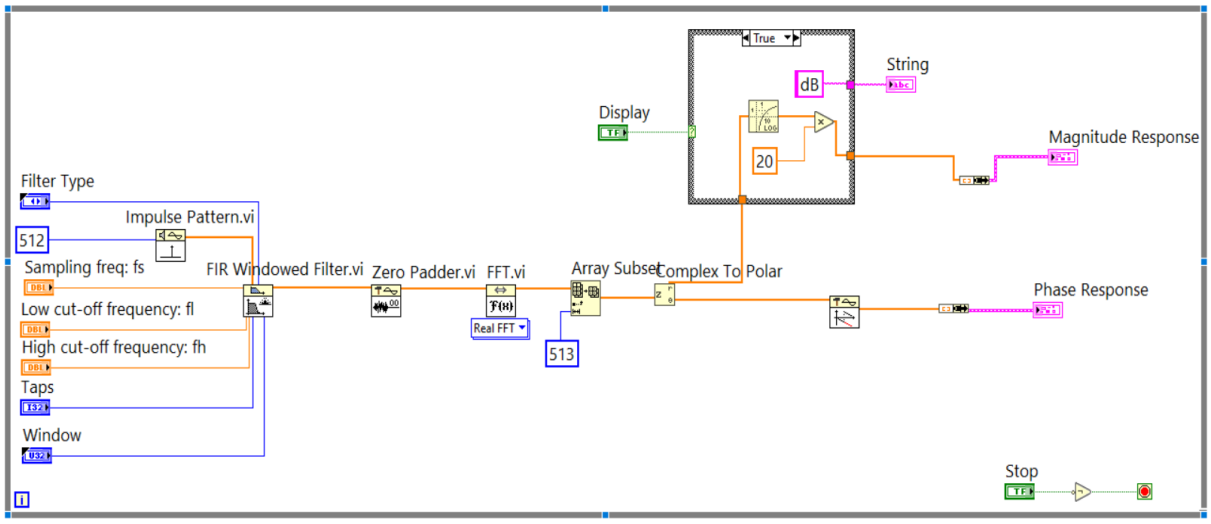
Waveforms:

Following are the expected Magnitude and Phase response of the FIR Band Stop filter using Hamming Window:

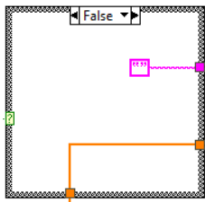


Block Diagram created in LabVIEW for FIR Filters:

a). FIR Band Stop Filter using Hamming Window:

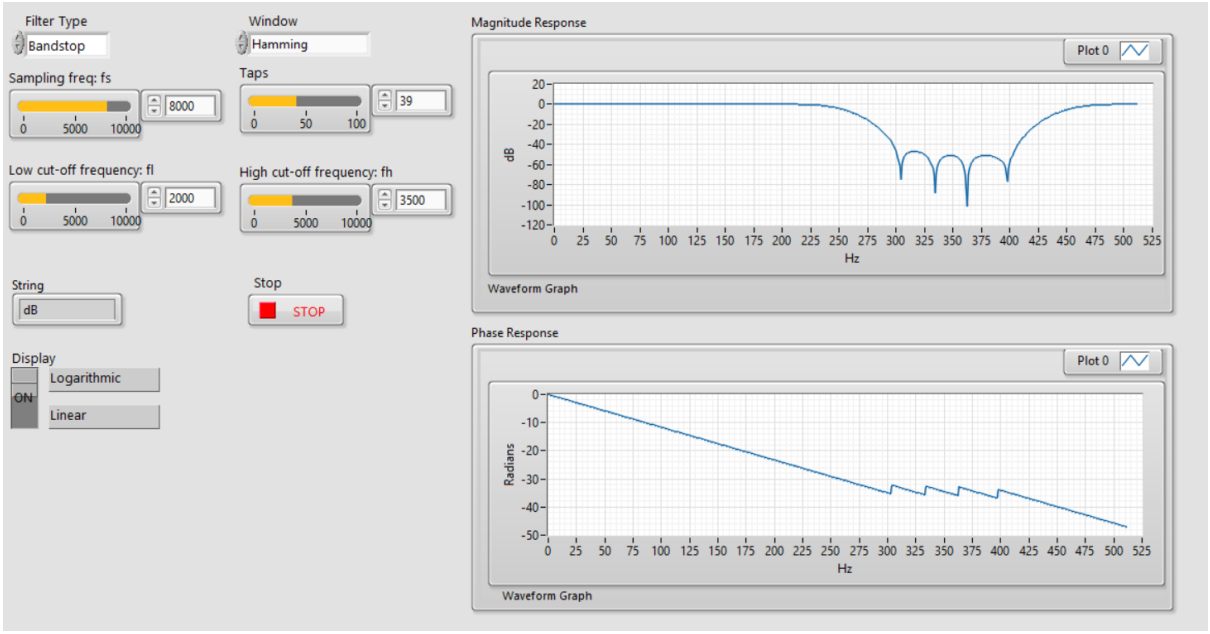


In above figure the “True Case” can be replaced by the “False Case” to obtain the output in linear form instead of Logarithmic form and the filter type can be changed to Band Stop.



Front Panels Created in LabVIEW for FIR Filters:

a) Front panel of FIR Band Stop Filter using Hamming window:



Conclusion: The FIR Band Stop Filter is designed according to the given specification and the magnitude and phase response of the filter is verified.

EXPERIMENT 1 c.

DESIGN OF FIR BAND PASS FILTER USING HANNING WINDOW

Aim: To design Chebyshev Bandpass IIR filter for the Passband frequency=250Hz, Stopband frequency= 400Hz and sampling frequency=1500Hz.

Theory:

FIR Filter:

- Finite impulse response (FIR) filter is a filter whose impulse response is of finite duration, because it settles to zero in finite time which may have internal feedback and may continue to respond indefinitely .
- The impulse response of an N^{th}-order discrete-time FIR filter lasts exactly N+1 samples (from first nonzero element through last nonzero element) before it then settles to zero.
- FIR filters can be discrete-time or continuous-time, and digital or analog.

For a causal discrete-time FIR filter of order *N*, each value of the output sequence is a weighted sum of the most recent input values**:**

$$Y(n) = \sum_{K=0}^{N} b_K x(n-k)$$

where**:**

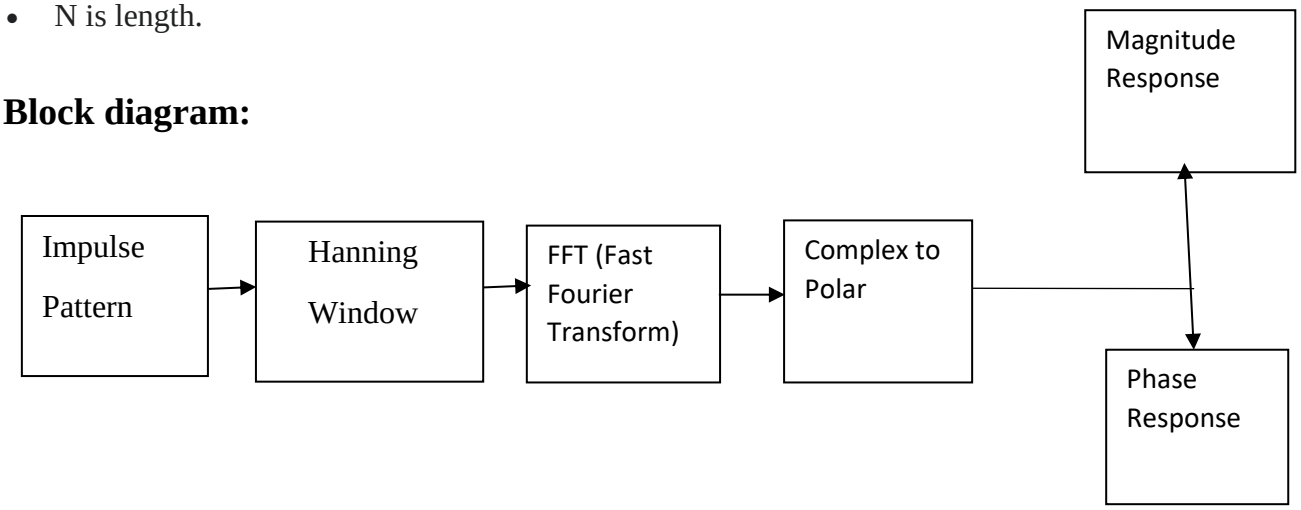
- x(n) is the input signal.
- y(n) is the output signal.
- N is the filter order.
- An N^{th} order filter has N+1 terms on the right-hand side.

Hanning Window:

$$w(n) = 0.5 - 0.5cos\left(\frac{2\pi n}{M-1}\right) \qquad 0 \le n \le M-1$$

- M is the filter order, an M[th] order filter has M+1 terms on the right-hand side.
- w(n) is the windowing function.
- N is length.

Block diagram:

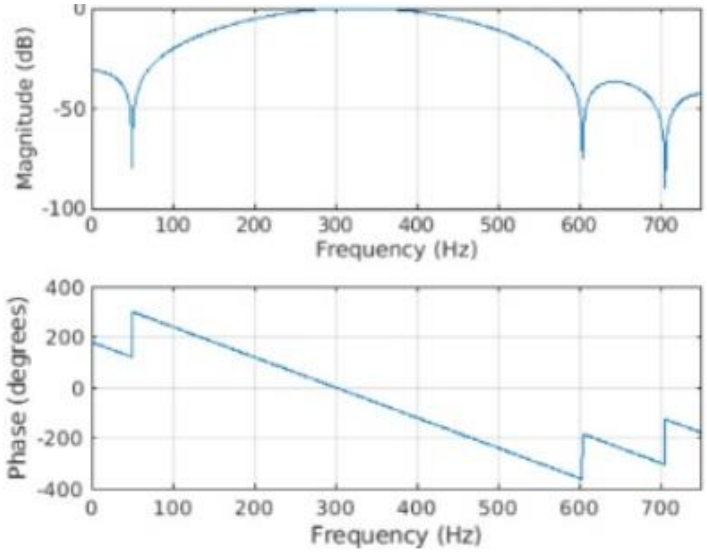


Procedure:

1. Import the desired impulse response block.
2. Import the FIR windowed filter.vi block
3. Specify the cutoff frequency and sampling frequency according to the problem statement. Also specify the low pass and high pass filter type and rectangular windowing function.
4. Import zero padder.vi and connect its input port to the output port of the FIR windowed filter.vi block.
5. Import FFT and connect its input port to the output port of the zero padder.vi block.
6. The output of FFT is connected to 'complex to polar' block to convert complex output into polar form to obtain the graphs.
7. Then the magnitude component is given to a case structure to opt between logarithmic or linear magnitude response.
8. Then both magnitude and phase responses are observed in the Front panel graphs.
9. Note: The Filter type can be changed from the front panel to choose between different types of filters and same goes for Windowing technique also.

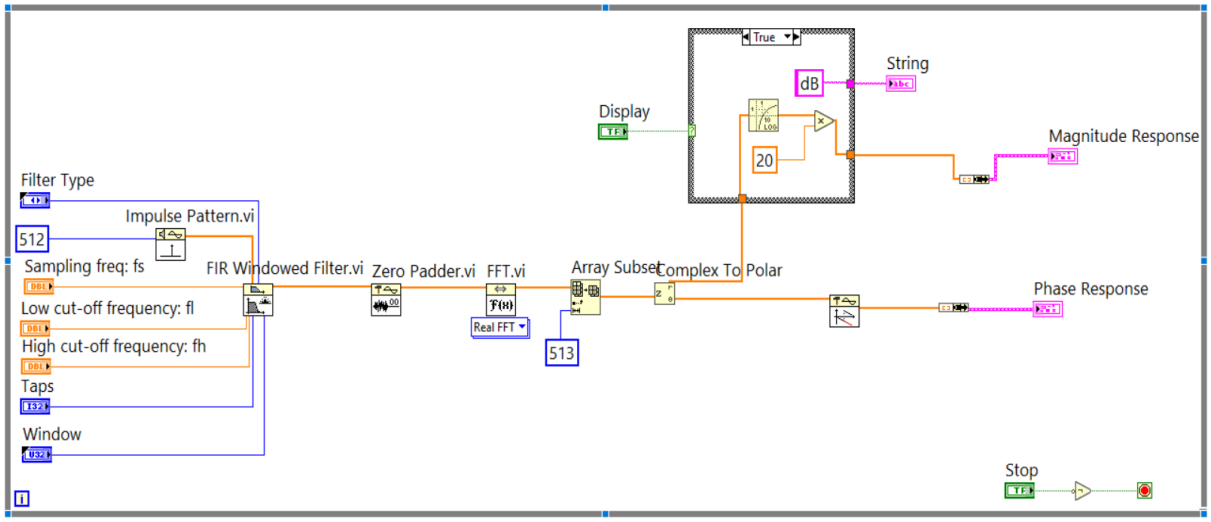
Waveforms:

Following are the expected Magnitude and Phase response of the FIR Bandpass filter using Hanning Window:

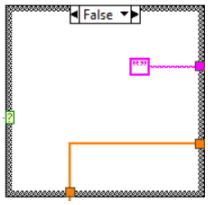


Block Diagram created in LabVIEW for FIR Filters:

a). FIR Band Pass Filter Using Hanning Window:

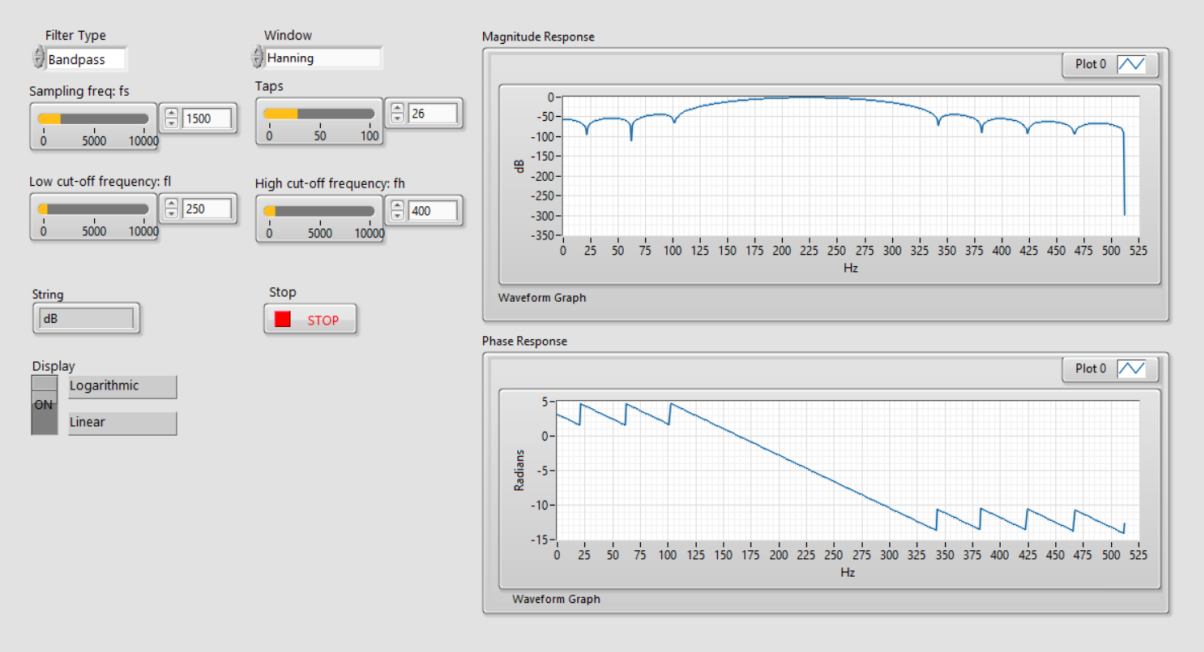


In above figure the “True Case” can be replaced by the “False Case” to obtain the output in linear form instead of Logarithmic form and the filter type can be changed and windowing type can also be changed.



Front Panels Created in LabVIEW for FIR Filters:

a) Front panel of FIR Band Pass Filter Using Hanning Window:



Conclusion: The FIR Band Pass Filter is designed according to the given specification and the magnitude and phase response of the filter is verified.

EXPERIMENT 2 a.

DESIGN OF BUTTERWORTH LOW PASS AND HIGH PASS FILTERS

Aim: To design Butterworth lowpass and high pass IIR filters for the Lower cutoff frequency= 100Hz, Higher cutoff frequency= 300Hz and sampling frequency=1000Hz

Theory:

- IIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications (the other type being FIR). "IIR" means "Infinite Impulse Response."
- The impulse response is "infinite" because there is feedback in the filter
- IIR filters can achieve a given filtering characteristic using less memory and calculations than a similar FIR filter.

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

where:

- b_j the set of forward coefficients.
- N_b is the number of forward coefficients.
- a_k is the set of reverse coefficients
- N_a is the number of reverse coefficients.
- X_i is the current input.
- X_{i-j} is the past inputs.
- Y_{i-k} is the past outputs.

Butterworth Filter:

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.

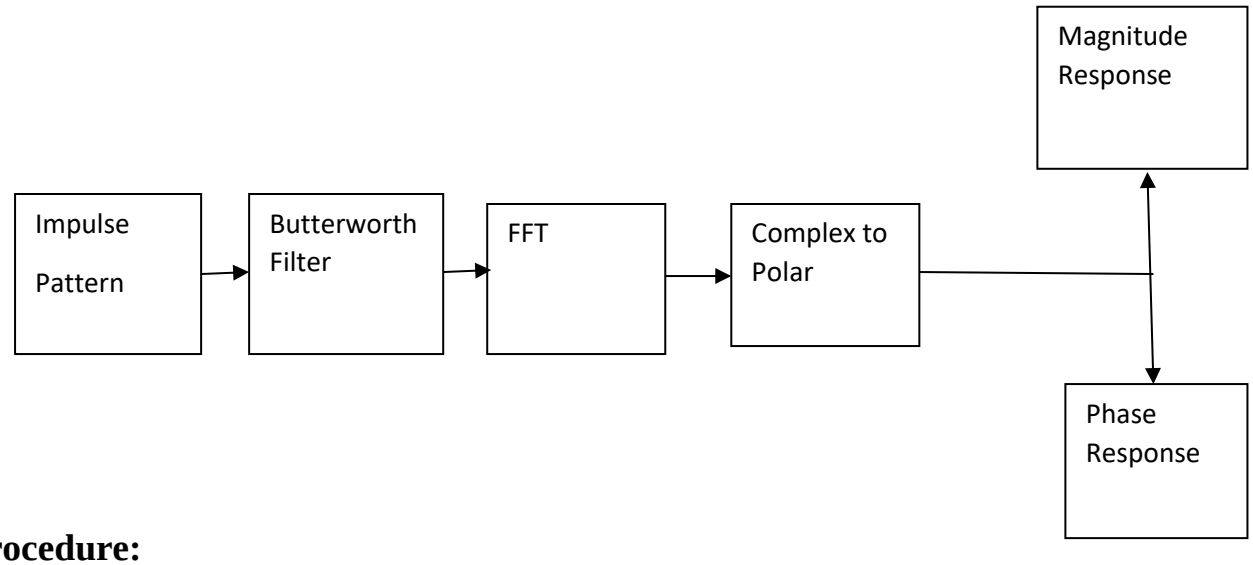
The transfer function for Butterworth filter is given by:

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

Where, n is the order of filter.

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

Block diagram:



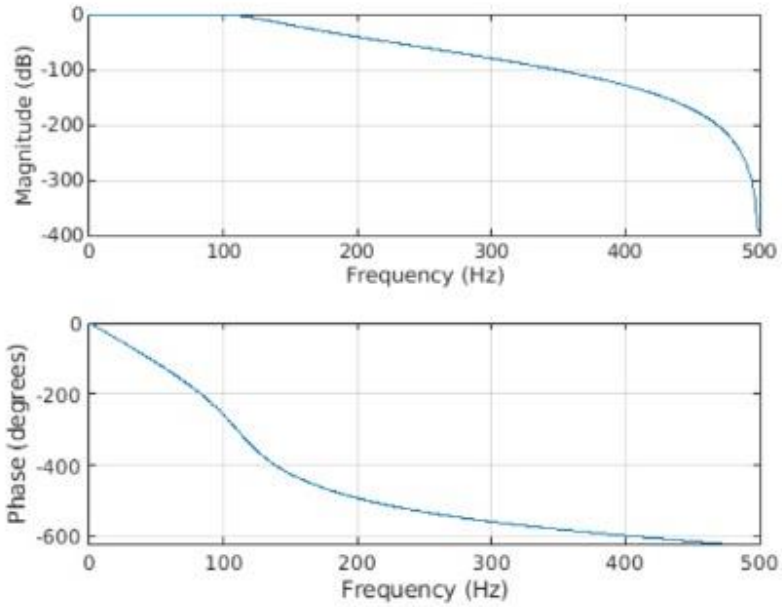
Procedure:

1. Import the desired impulse response block.
2. Import the Butterworth Filter block.
3. Specify the cutoff frequency and sampling frequency and other parameters according to the problem statement.
4. Also specify the low pass and high pass filter type.
5. The output of FFT is connected to 'complex to polar' block to get the required magnitude and phase response.
6. Then the magnitude component is given to a case structure to opt between logarithmic or linear magnitude response.
7. Then both magnitude and phase responses are observed in the Front panel graphs
8. Note: The Filter type can be changed from the front panel to choose between different types of filters and same thing goes for filter design type also.

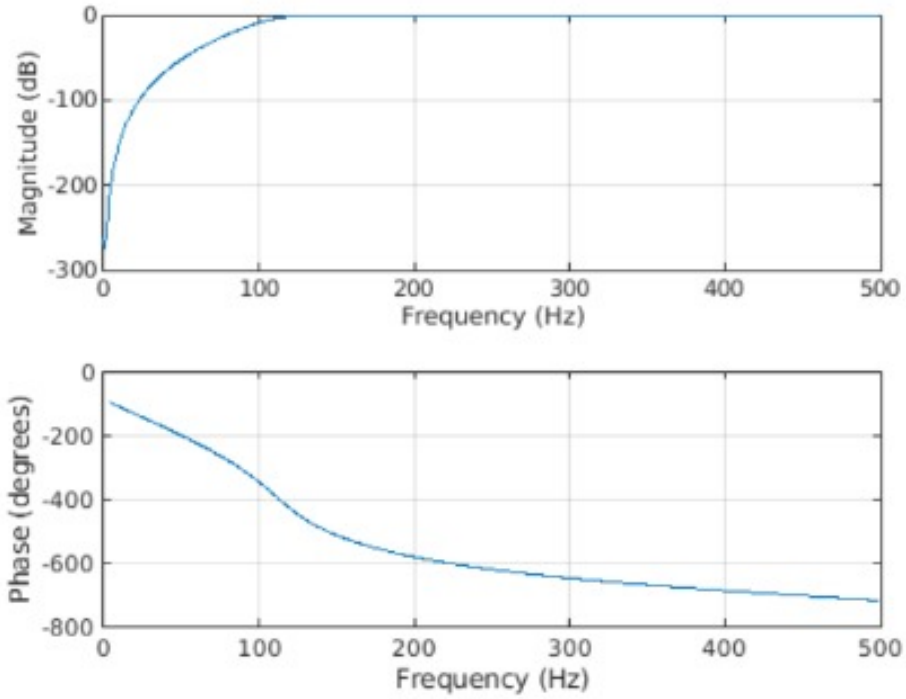
Waveforms:

Following are the expected waveforms for Butterworth Low pass Filter and High pass Filter.

a) Butterworth Low pass Filter:

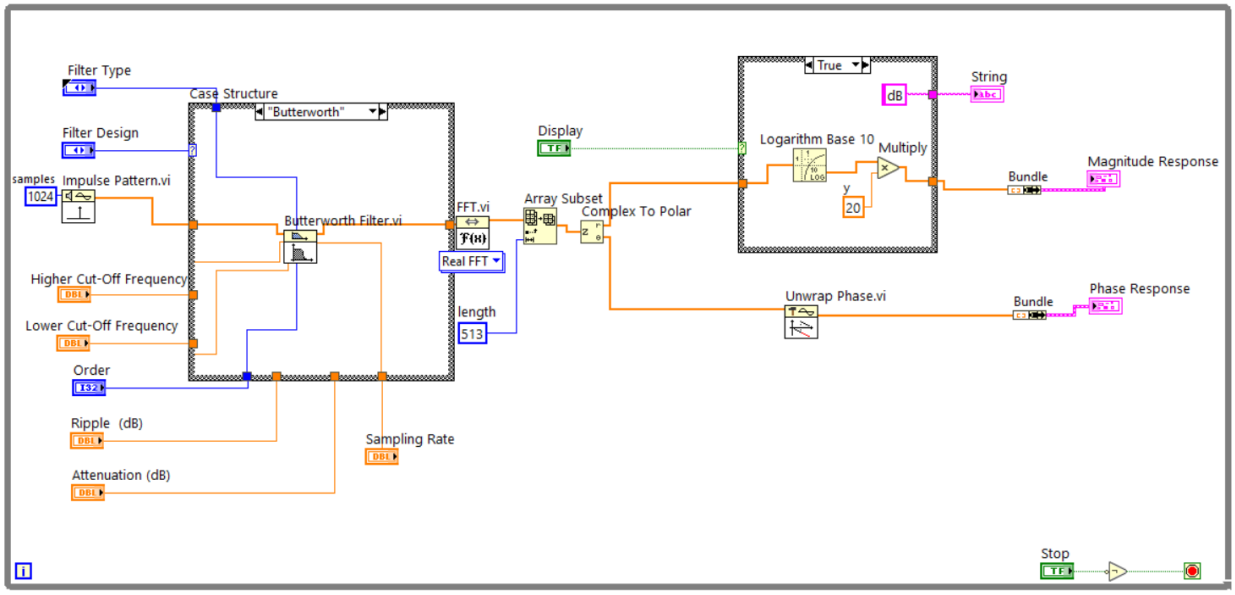


b) Butterworth High Pass Filter:

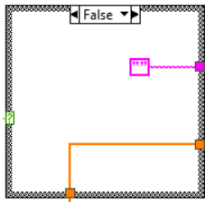


Block Diagram Created in LabVIEW for IIR Filters:

a) Butterworth IIR Low Pass Filter and High Pass Filter:

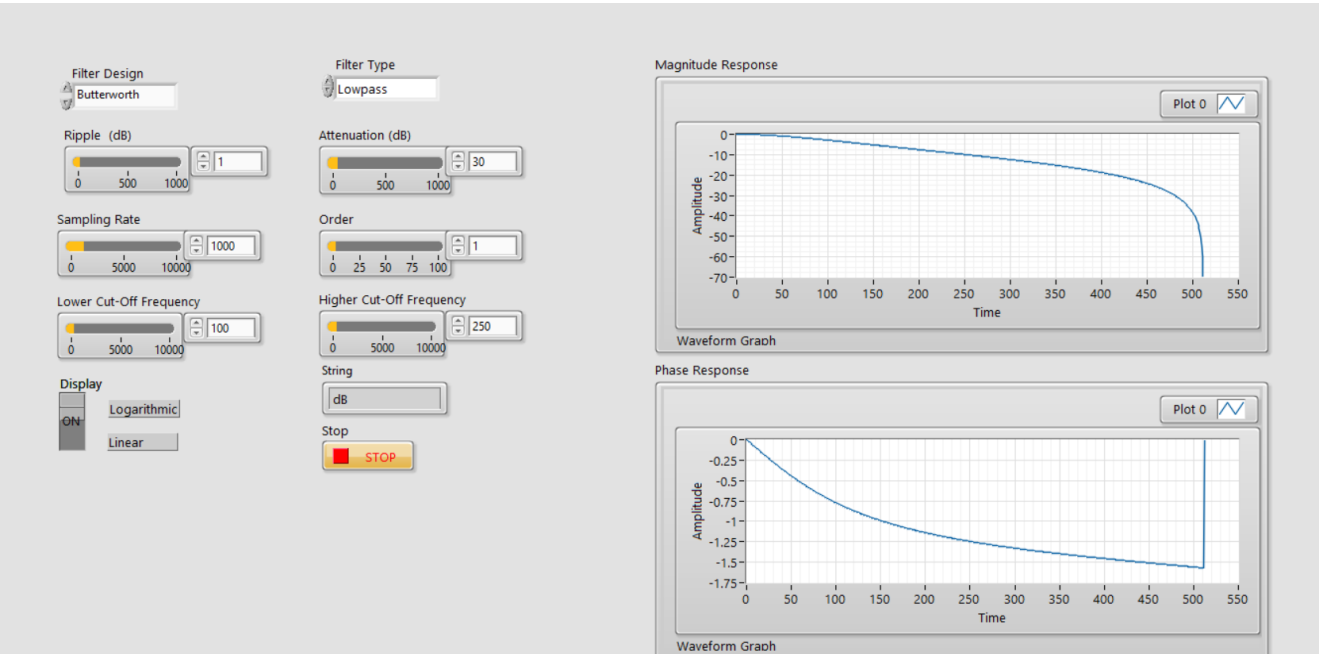


b) In above figure the "True Case" can be replaced by the "False Case" to obtain the output in linear form instead of Logarithmic form and the filter type can be changed between LPF and HPF.

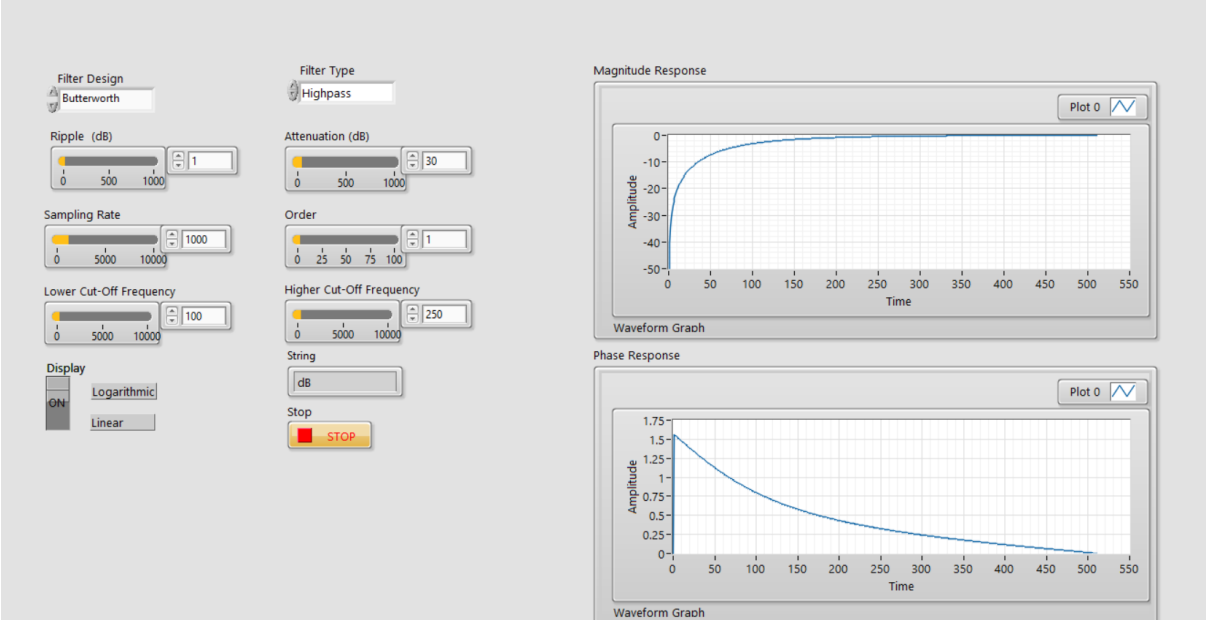


Front Panels created in LabVIEW for IIR Filters:

a) Front panel of Butterworth LPF:



b) Front panel of Butterworth HPF:



Conclusion: The Butterworth IIR Low pass Filter and High pass Filter are designed according to the given specification and the magnitude and phase response of both the filters are verified.

EXPERIMENT 2 b.

DESIGN OF ELLIPTIC BAND STOP FILTER

Aim: To design Elliptic Band stop IIR filter for the Passband frequency= 200Hz, Stopband frequency= 350Hz and sampling frequency=1000Hz.

Theory:

- IIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications (the other type being FIR). "IIR" means "Infinite Impulse Response."
- The impulse response is "infinite" because there is feedback in the filter
- IIR filters can achieve a given filtering characteristic using less memory and calculations than a similar FIR filter.

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

where:

- b_j the set of forward coefficients.
- N_b is the number of forward coefficients.
- a_k is the set of reverse coefficients
- N_a is the number of reverse coefficients.
- X_i is the current input.
- X_{i-j} is the past inputs.
- Y_{i-k} is the past outputs.

Elliptic Filter:

Elliptic filters have the following characteristics:

• Minimization of peak error in the passband and the stopband

• Equiripple 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.

The transfer function is given by:

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

Where,

- $U_N(X)$ the Jacobian is elliptic function of order N.
- ε is a constant related to passband ripple.

They provide a realization with the lowest order for a particular set of conditions.

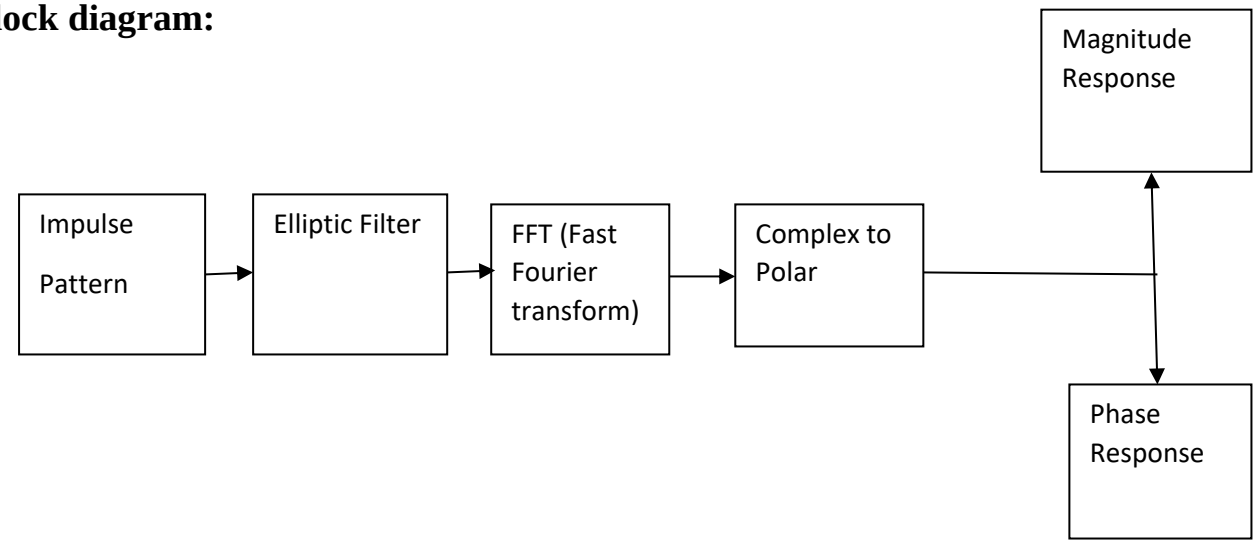
Selection of a Digital filters for any application among all can be done by considering following points:

• Does the analysis require a linear-phase response?

• Can the analysis tolerate ripples?

• Does the analysis require a narrow transition band?

Block diagram:

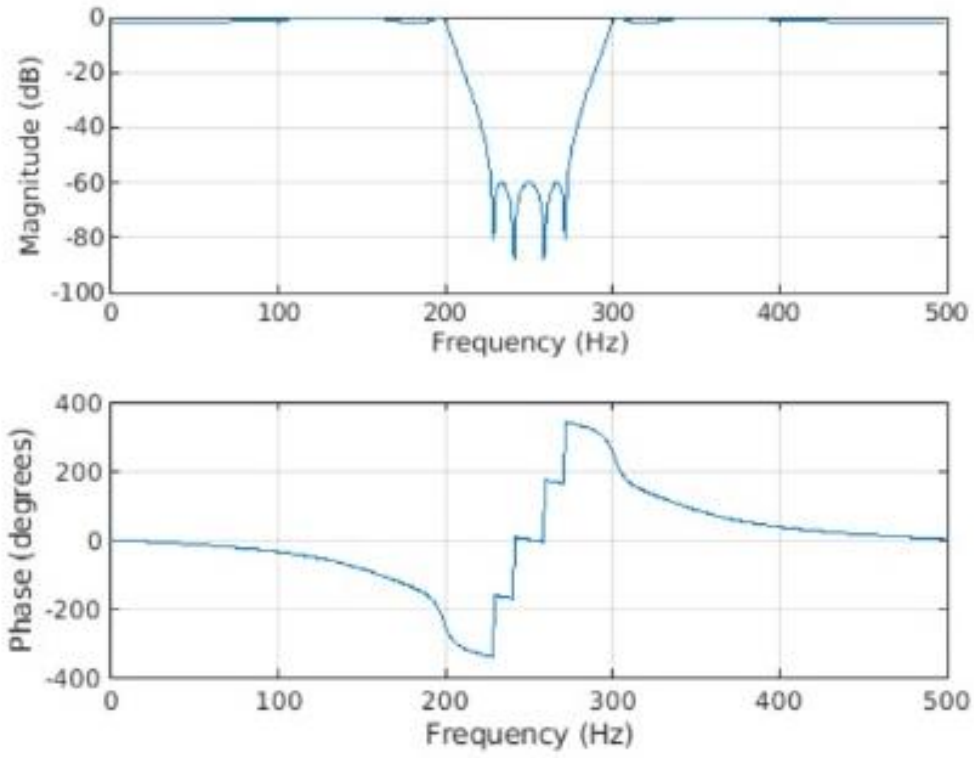


Procedure:

1. Import the desired impulse response block.
2. Import the Butterworth Filter block.
3. Specify the cutoff frequency and sampling frequency and other parameters according to the problem statement.
4. Also specify the low pass and high pass filter type.
5. The output of FFT is connected to 'complex to polar' block to get the required magnitude and phase response.
6. Then the magnitude component is given to a case structure to opt between logarithmic or linear magnitude response.
7. Then both magnitude and phase responses are observed in the Front panel graphs
8. Note: The Filter type can be changed from the front panel to choose between different types of filters and same thing goes for filter design type also.

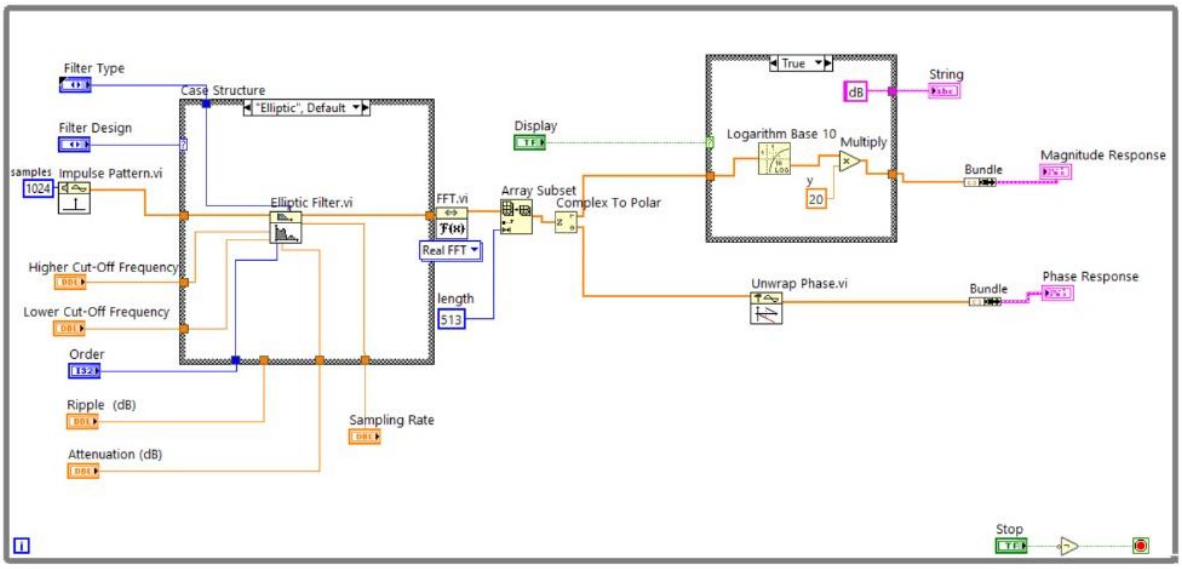
Waveforms:

Expected waveform of Elliptic Band Stop Filter:



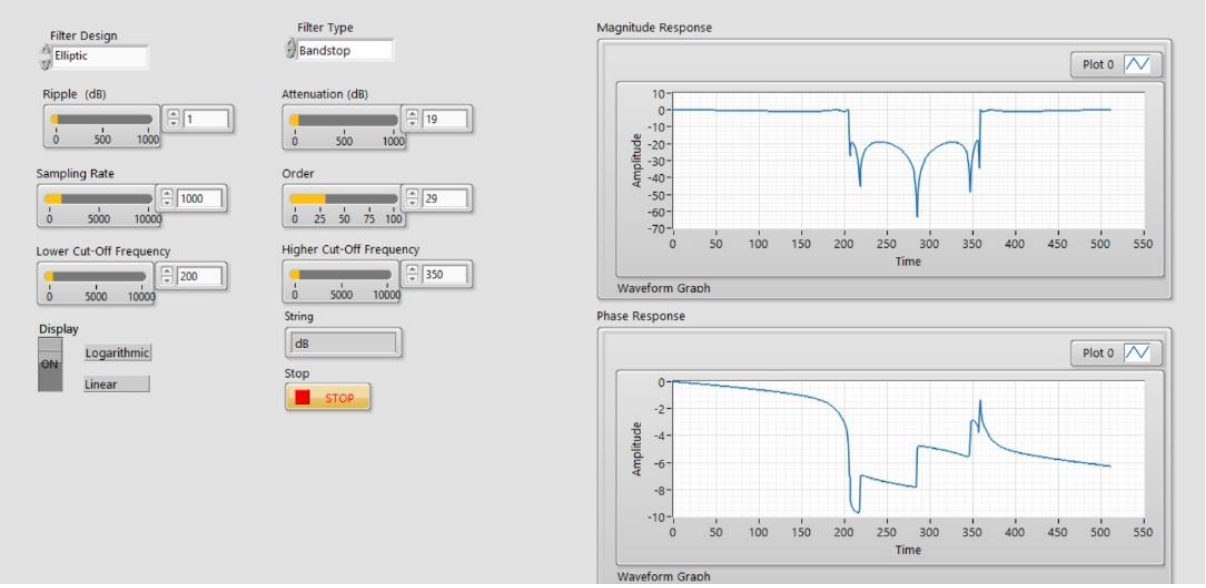
Block Diagram Created in LabVIEW for IIR Filters:

Elliptic IIR Band Stop Filter:



Front Panel Created in LabVIEW for IIR Filters:

Front panel of Elliptic Band stop Filter:



Conclusion: Elliptic Band Stop Filter is designed according to the given specification and the magnitude and phase response is also verified.

EXPERIMENT 2 c.

DESIGN OF CHEBYSHEV BAND PASS FILTER

Aim: To design Chebyshev Bandpass IIR filter for the Passband frequency= 200Hz, Stopband frequency= 350Hz and sampling frequency=1000Hz.

Theory:

- IIR filters are one of two primary types of digital filters used in Digital Signal Processing (DSP) applications (the other type being FIR). "IIR" means "Infinite Impulse Response."
- The impulse response is "infinite" because there is feedback in the filter
- IIR filters can achieve a given filtering characteristic using less memory and calculations than a similar FIR filter.

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

where:

- b_j the set of forward coefficients.
- N_b is the number of forward coefficients.
- a_k is the set of reverse coefficients
- N_a is the number of reverse coefficients.
- X_i is the current input.
- X_{i-j} is the past inputs.
- Y_{i-k} is the past outputs.

Chebyshev Filter:

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 roll off 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.

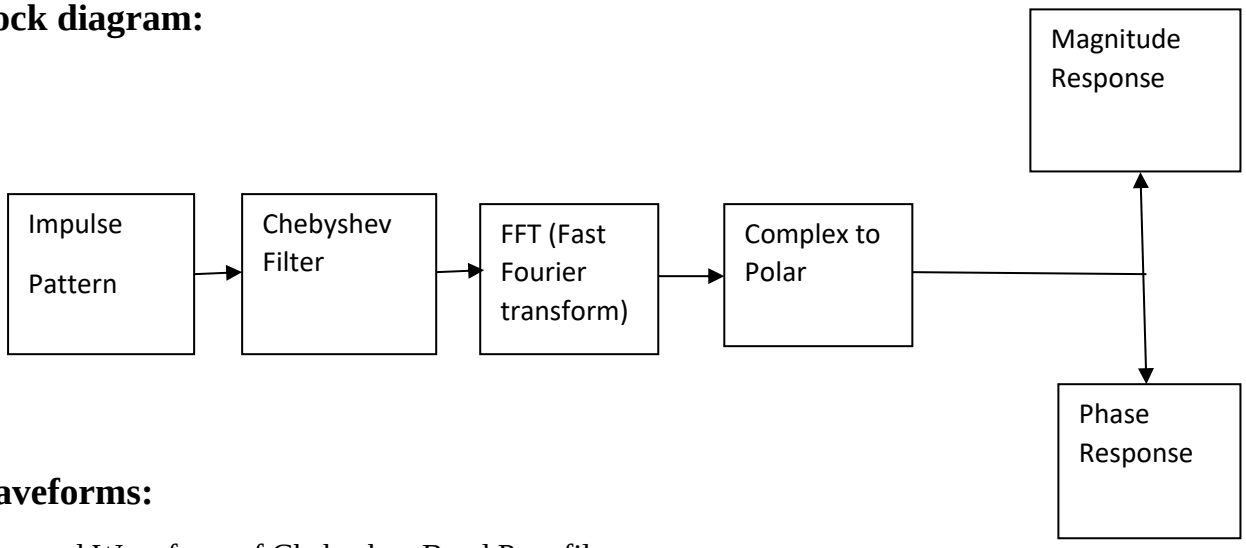
The sharp transition between the passband and the stopband of a Chebyshev filter produces smaller absolute errors and faster execution speeds than a Butterworth filter. The frequency response of the filter is given by:

$$|H(\Omega)|^2 = \left(1 + \varepsilon^2 T_N^2\left(\Omega/\Omega_P\right)\right)^{-1} \qquad T_N = \cos(N\cos^{-1}x) \quad |x| \le 1$$
$$\cos(N\cosh^{-1}x) \quad |x| \ge 1$$

where:

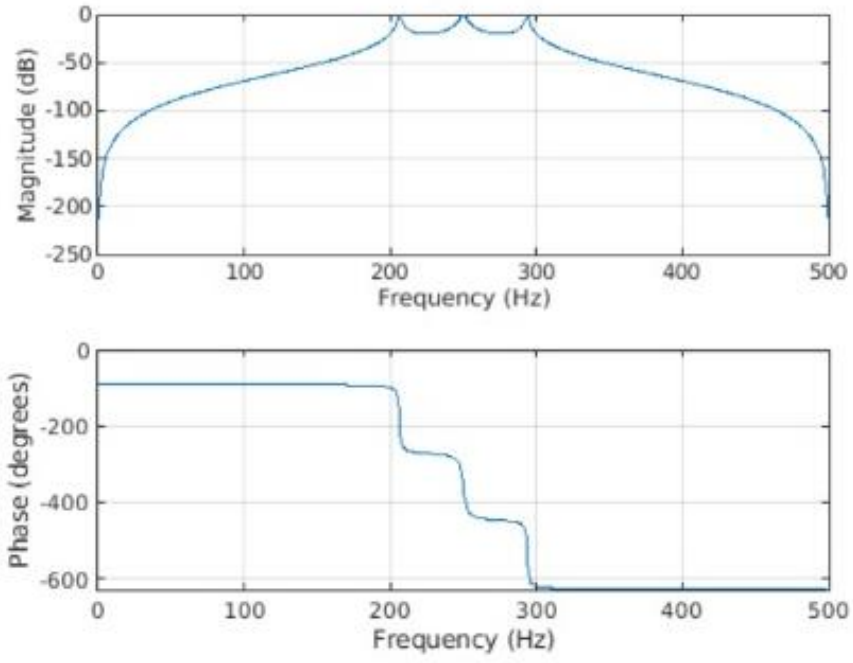
- ε is a parameter of the filter is related to ripple present in the passband.
- $T_N(x)$ is the Nth order Chebyshev polynomial defined as shown above.

Block diagram:



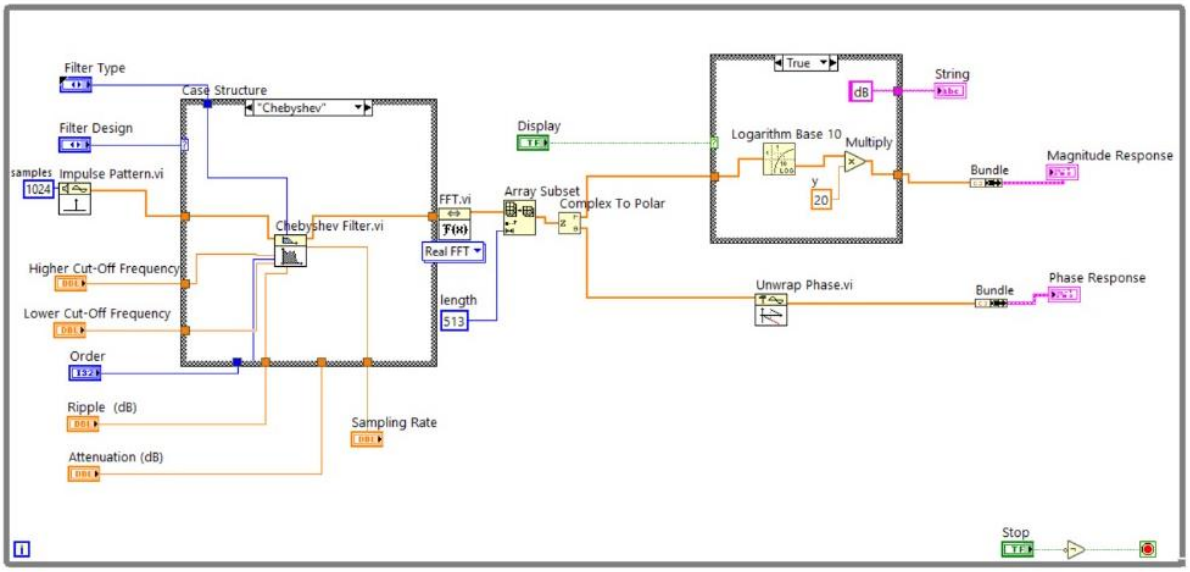
Waveforms:

Expected Waveform of Chebyshev Band Pass filter:



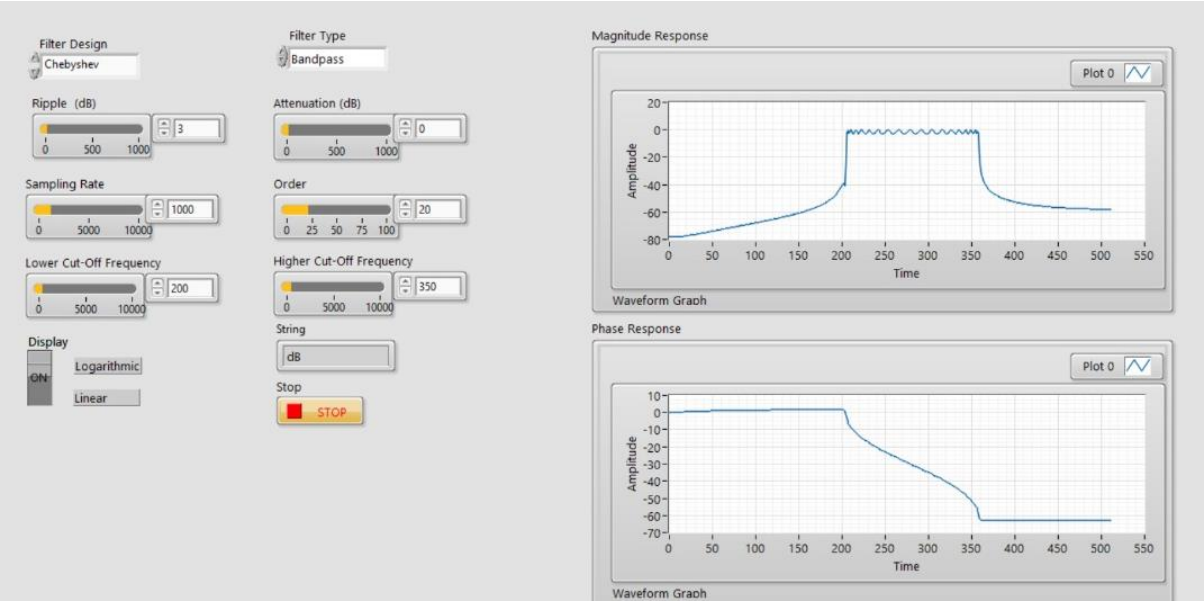
Block Diagram Created in LabVIEW for IIR Filters:

Block diagram of Chebyshev Band Pass Filter:



Front Panels created in LabVIEW for IIR Filters:

Front panel of Chebyshev Band pass Filter:



Conclusion: Chebyshev Band Pass Filter is designed according to the given specification and the magnitude and phase response is also verified.