EEE 304 – Lab 3

Sampling, Aliasing and Equalization using LabVIEW

Introduction

This lab introduces some fundamental concepts in sampling theory and reconstruction through intuitive examples. This lab exercise three major sections

- Study of aliasing effects while sampling a given signal.
- Analysis of Upsampling (i.e.) increasing the sampling rate of a signal.
- Design of Music Equalizers.

A basic introduction is provided to each of these tasks in addition to guidelines for implementing them using National Instruments LabVIEW.

Exercise 1 – Aliasing

Sampling is the very first step towards converting the analog signal into a digital signal. But adequate number of samples must be taken from a given analog signal in order to effectively reconstruct it back from its samples. The 'adequate' number of samples needed is determined by the Nyquist-Shannon theorem. The theorem states that the perfect reconstruction of an analog signal is possible when the sampling frequency is greater than twice the maximum frequency of the signal being sampled.

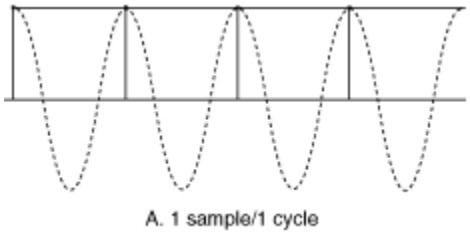


Figure 1. Sampling a sinusoid at Nyquist frequency (Aliased)

For a given input signal of bandwidth f_0, the sampling frequency f_s should be strictly greater than $2f_0$, to ensure perfect reconstruction of the signal from the samples. If $f_s = 2f_0$, then f_s is said to be the Nyquist frequency. It is important to note that information may be lost if a signal is sampled exactly at the Nyquist frequency. For example, the sine wave in Figure 1 has a frequency of 1/2 Hz. The Nyquist frequency is therefore 1 Hz. If we sample the sine wave at a rate of 1 Hz, say at t=0, t=1, t=2, and so on, all the sample values selected will be 0. The signal will look as if it were identically 0, and no reconstruction method will be able to recreate the 1/2 Hz sine wave. This indicates that $f_s > 2f_0$ is a strict condition to be met.

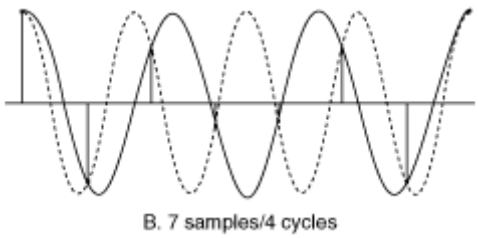


Figure 2(a). Sampling at fs < 2f

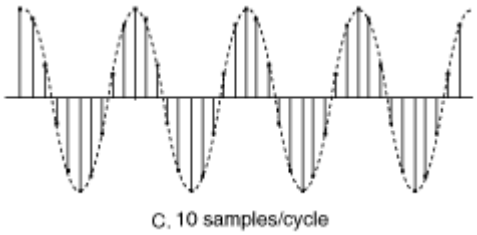


Figure 2(b). Sampling at fs > 2f

As seen in Figure 2(a), when the sampling frequency is less than the Nyquist rate, the signal is aliased to a frequency less than the original frequency. Perfect reconstruction is observed in Figure 2(b) where the Nyquist criterion is met. But when sampling frequency is exactly equal to the Nyquist rate the reconstructed waveform appears as an alias at DC. This LabVIEW exercise illustrates the aliasing effect in both the time and frequency domains. Figure 3 illustrates the basic block diagram to be designed for this exercise.

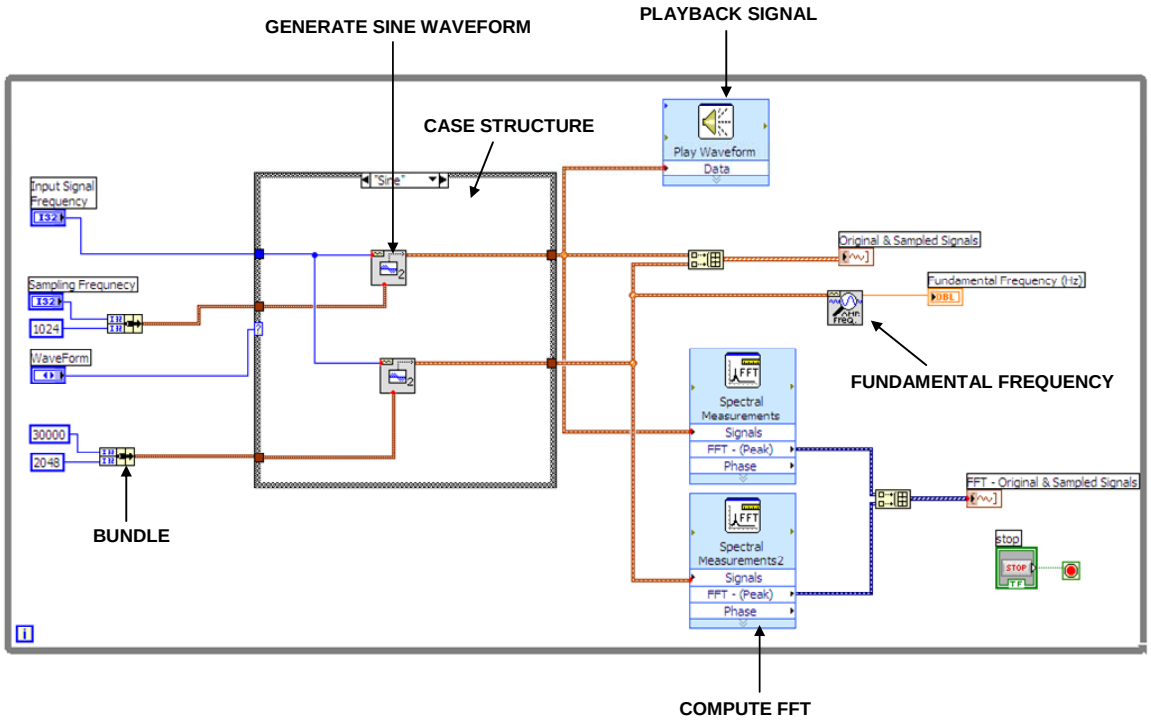
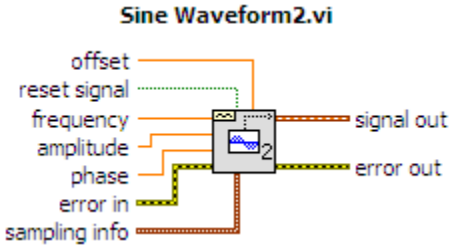


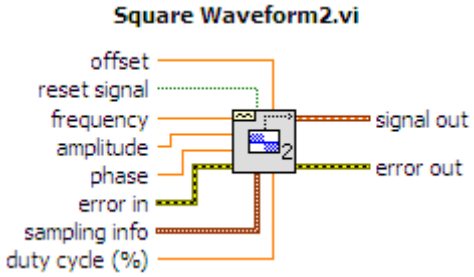
Figure 3. Block diagram for aliasing

1.1 Generating sine and square waveforms

The first step in this exercise is to generate input waveforms with a desired input frequency and a specific sampling rate. The block diagram needs to be built with a provision to change both these parameters dynamically and observe the aliasing effect. In this exercise we will use *sine* and *square* waveforms as the input signals. To generate a sine waveform with a desired frequency, we make use of the function S***ine Waveform2***



This function generates a sine waveform for the given *frequency*, *amplitude* and *phase*. The parameter *sampling info* contains the input sampling frequency and the number of samples in the input signal generated. Similarly the square waveform can be generated using the function ***Square Waveform2***

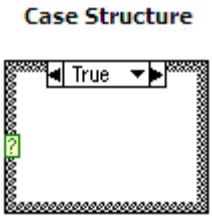


These two waveform generation functions are not available with the basic LabVIEW package and hence provided to you as a part of this exercise.

To place external VI's in your block diagram, right-click on the block diagram workspace and choose the option ***Select a VI*** and select the desired LabVIEW function.

1.2 Choosing the input signal

To allow the user to choose between the two input signals explained in the previous section, a ***Case*** structure is used. A Case structure has one or more sub-diagrams or cases, exactly one of which executes when the structure is executed. The value wired to the ***selector*** terminal determines which case to execute and can be *Boolean*, *string*, *integer* or *enumerated type*. You can right-click on the border of the structure to add or delete cases.



In this exercise, you need a Case structure with two cases, one for *sine waveform* and the other for *square waveform*. The ***selector*** value is of type ***Enum***. Place a Case structure in your block diagram and switch to the Front panel. Now, right-click on the Front Panel workspace and choose the control ***Enum*** from the *Controls Palette* from ***Modern>>Ring & Enum***. Right-click on the control and choose ***Properties*** from the shortcut menu. Select the ***Edit Items*** tab from the ***Properties*** window and insert two values, *Sine* and *Square*. Save the changes, switch back to the block diagram and connect the newly created control to the ***selector*** terminal of the Case structure. You will observe that the case labels change to the newly created enumeration items, namely *Sine* and *Square* respectively. Select *Sine* from the drop down case label. Place the ***Sine Waveform2 VI*** inside the case structure as shown in Fig. 3. Now select Square from the drop down case label, and place ***Square Waveform2 VI*** inside the case structure and follow same steps as before.

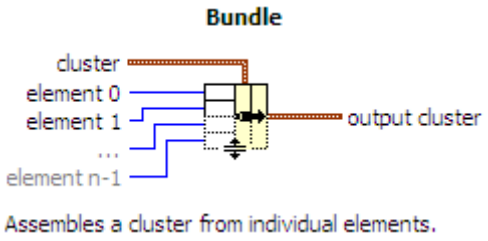
Please note that Figure 3 shows the block diagram only for the case of Sine waveform.

1.3 Illustrating Aliasing Effect

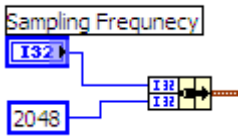
As seen in Figure 3, there are two ***Sine Waveform2*** functions under the case *Sine*. The first waveform will be used to study the effect of aliasing while the second waveform generator has a constant but very high sampling frequency in order to display the original (unaliased) analog signal. As explained in Section 1.1, the input to the Sine Waveform2 function includes the input frequency and the sampling information and we will leave the other parameters to hold their default values. Build two ***Slider*** controls for the input and sampling frequencies respectively.

- Slider Control 1 (Input Frequency) : 15 Hz to 2000 Hz.
- Slider Control 1 (Sampling Frequency) : 300 Hz to 6000 Hz.

For both the controls, right click on them from Front Panel and choose the ***Properties*** option. Change the ***Data Entry*** and ***Scale*** values to have the values specified here. Now, as you will observe from the block diagram, the input parameter ***Sampling Info*** is a cluster of two variables, namely the *Sampling Frequency* and the *number of samples*.



Hence use the function ***Bundle*** to create the cluster. For the first Sine wave generator, the cluster is built with the ***Sampling Frequency*** and a constant value '2048' (samples). For the second generator the ***Sampling Info*** cluster contains two constants '30000' (Sampling frequency) and '2048' (samples). (In case you have difficulty creating a ***constant***, then create a ***numeric control***, then right click on it, and select ***change to constant***). The ***Input Frequency*** control is connected to both the generators as shown in Figure 3.



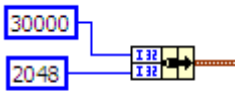
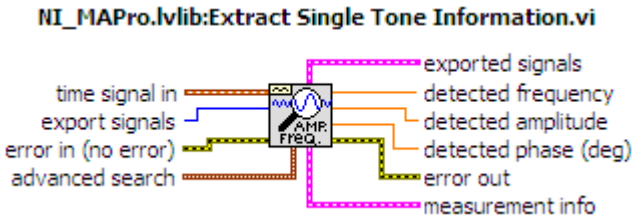


Figure 4(a). Sampling info for generator 1 **4(b). Sampling info for generator 2**

Further, to illustrate the aliasing effect in the frequency domain, use the *Express VI* ***Spectral Measurements*** and apply FFT to the waveforms from both the generators. To plot the waveforms from both the generators in a single graph, combine both the waveforms into an array and plot the array output using a graph in the Front Panel. To do this, select ***Build Array*** from ***Programming >> Array.*** Connect the output from the first sine waveform generator. Now, right click on the build array and select ***Add Input*** and connect the output of the second sine waveform generator. Create a new waveform graph in the Front Panel and right-click to choose the ***Properties***. Select the ***Scales tab*** and uncheck ***Auto Scale*** option. For the parameter ***Maximum*** and ***Minimum***, enter the values 0.005 and 0 respectively. Similarly plot both the FFT magnitude responses in a single waveform graph. In this case you need not make changes to the ***Scales tab***. Further to quantify the effect of aliasing, play the waveform obtained from the first generator as shown in Figure 3. When the Nyquist criterion is not met, the aliasing will be obvious when heard.

Repeat all the steps specifying for the case of "Square" with the ***Square Waveform2*** function. Please note that the same controls and indicators created can be used for the other case also. Since, only one case is executed every time a structure is executed, the graphs will be suitably updated.

The block diagram contains another function to compute the fundamental frequency of the input signal. In this example, since the inputs are of single tone, the fundamental frequency is the tone itself. But if there are number of components, the fundamental frequency will be different.



This function takes a signal in and finds the single tone with the highest amplitude or searches a specified frequency range and the single tone frequency, amplitude and phase.

This function ***Extract Single Tone Information*** can be found under the section ***Programming>>Waveform>>Analog Waveform>>Waveform Measurements***. The final Front Panel layout can be found in Figure 5.

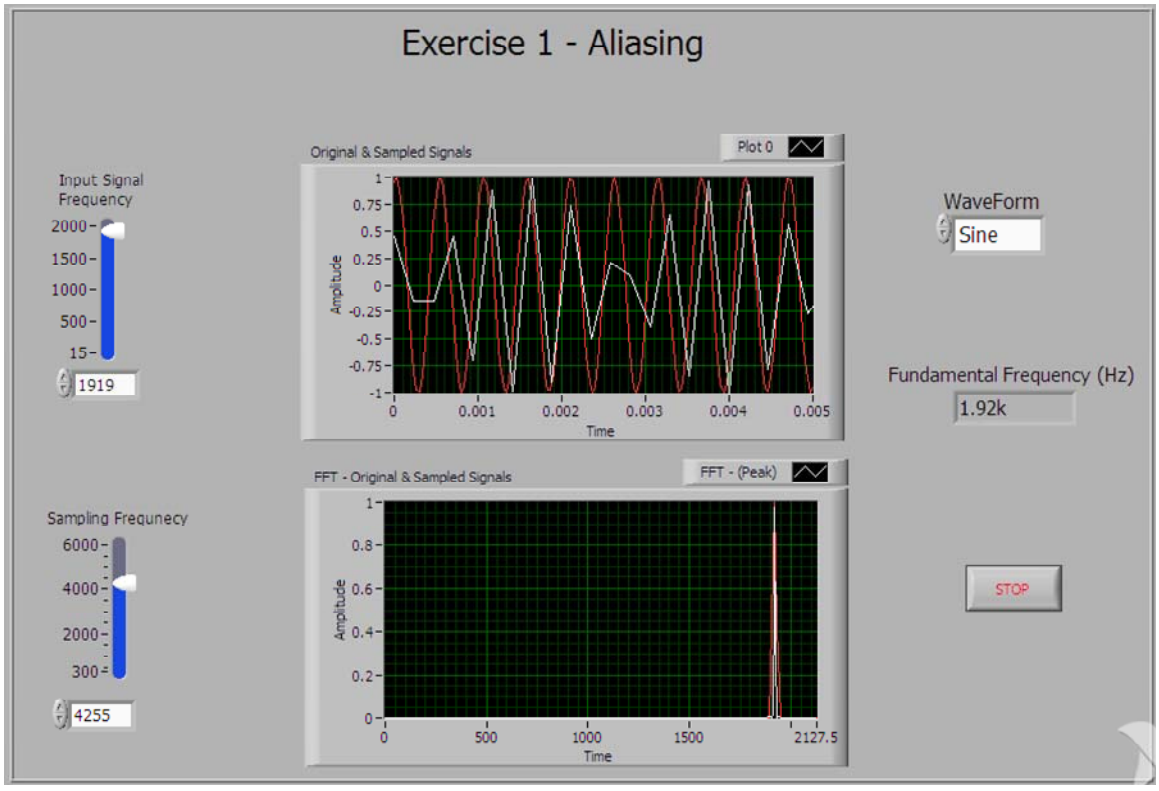


Figure 5. Front Panel Layout for Exercise 1

EXERCISE 1

To check the aliasing effects, fix the sampling frequency and change the input frequency from minimum to maximum. Observe the aliased and original signals in time and frequency domains. Fix the sampling frequency Fs = 2000 Hz. For which input frequencies do you observe aliasing?

Are you able to justify this by listening to the tones? Set Fs = 2000 Hz and listen to the tones at f1 = 500 Hz and f2 = 1500 Hz. Also listen to the tones at f1 and f2 for Fs = 4000 Hz. Give your comments on this.

Now change the waveform to square. Here you can observe that even if the sampling frequency is around 4 times the input frequency (say Fs = 4000 Hz and fo = 1000 Hz), the output is not aligned perfectly in the frequency domain. Comment on this. You can change the FFT response display from linear scale to dB scale by double clicking on the Spectral Measurements block to see this effect more clearly.

For the case of sine wave, set sampling frequency from 1000-4000 Hz in steps of 500Hz and note the frequency at which you observe aliasing. Make a table of this.

Provide the screenshots of the block diagram and the front panel.

Exercise 2 – Analysis of Upsampling

Upsampling is used to increase the sampling rate of a given input signal. The procedure below defines the upsampling by an integer factor.

Consider a discrete signal $f(k)$ on a digital frequency range. Let L denote the upsampling factor.

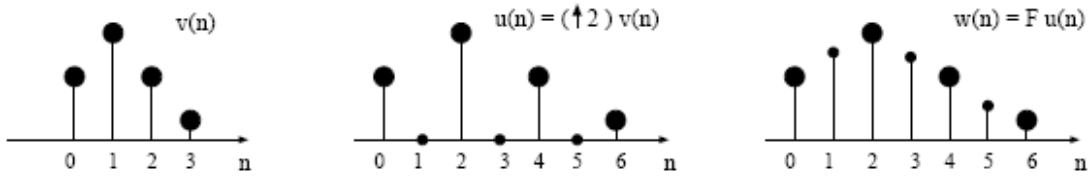
- Insert $L-1$ zeros between each sample in $f(k)$. This can be equivalently expressed as

$$g(k)=\begin{cases} f\left(\frac{k}{l}\right), & \text{if } \frac{k}{l} \text{ is an integer} \\ 0, & \text{otherwise} \end{cases} \qquad (1)$$

- Filter with a low-pass filter which, theoretically, should be the sinc filter with cut-off frequency at $\frac{\pi}{L}$.

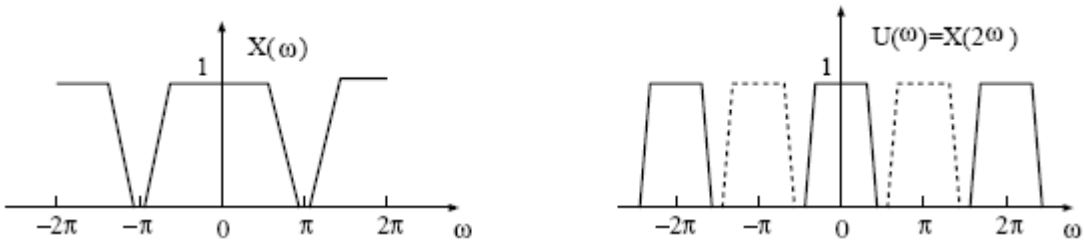
So upsampling first inserts $L-1$ zeros between any two samples of the input signal and then the low-pass filter acts as an interpolator and obtains the appropriate values for the zeros inserted between the samples.

The figure below describes the process of upsampling by a factor of 2.



Here, $v(n)$ is the input sequence, $u(n)$ is upsampled (zero inserted) sequence and $w(n)$ is the output of interpolation filter (Low Pass Filter).

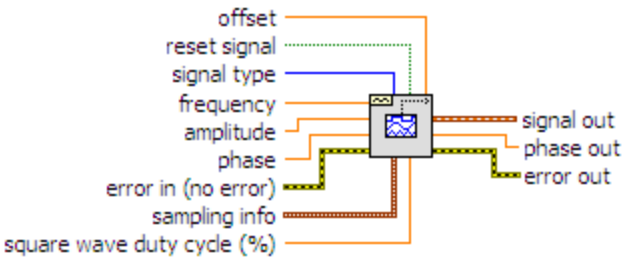
The effect of upsampling can also be studied in the frequency domain. When a input signal $v(n)$ is upsampled by a factor of 2 (i.e.) $u = v\left(n/2\right)$, then in the frequency domain $U(\omega) = V(2\omega)$, where $U(\omega)$ and $V(\omega)$ denote the signals in the frequency domain.



As shown in the figure above, the Fourier transform of the output of the expander is a frequency-scaled version of the Fourier transform of the input (i.e.) ω is replaced by ωL, where L is the upsampling factor (in our case L=2). We can use a low pass interpolation filter with appropriate cutoff to remove all the frequency-scaled images of the signal. An interpolator fills in the missing samples and hence the upsampling operation is also referred as interpolation.

2.1 Generating the input signal

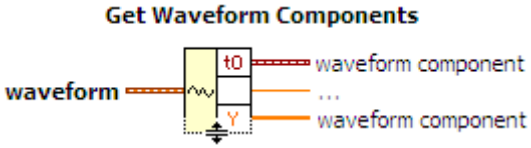
The first step in this exercise is to generate a discrete time input signal which will be upsampled by a factor of 2. This exercise will demonstrate the effects of upsampling in both time and frequency domains. A triangular waveform is used as the input and can be generated using the LabVIEW function ***Basic Function Generator***.



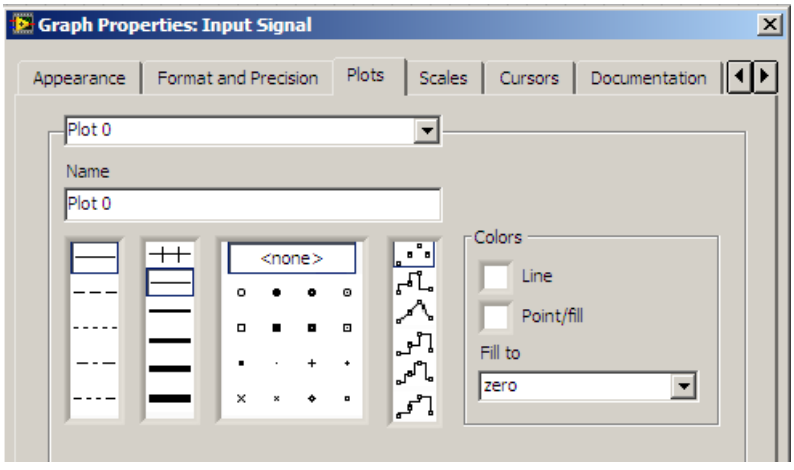
The following *constant* values are chosen for the input parameters to this function

- Frequency – 10 Hz
- Signal type – Triangular wave
- Sampling info
    - Fs – 100 Hz
    - # samples - 32

The output of this function is of data type *waveform* and will be converted to an array for further processing. This is achieved by using the function ***Get Waveform Components*** (You can search for functions from the functions palette as explained in the previous labs).



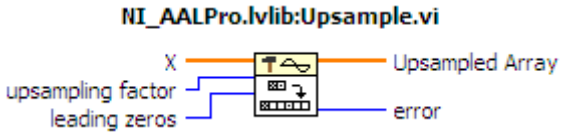
The original signal needs to be plotted as discrete samples in this exercise. For this, place a waveform graph in the Front Panel, right-click on this and choose ***Properties*** from the shortcut menu. Select the ***Plots*** tab and make the changes shown below.



Also, compute the FFT magnitude response of the input signal and plot it. The FFT magnitude response can be computed as done in Lab 2 (Use 256 point FFT). A subVI can be placed in your block diagram by right-clicking on the block diagram workspace and choosing ***Select a VI*** from the menu. The subVI (**Custom_FFT.vi**) for computing this is provided for your reference. Do not make the changes to the waveform graph options shown above for the frequency domain plots. The block diagram for this exercise is shown in Figure 6.

2.2 Upsampling the signal

The next step is to upsample the generated signal by a factor of 2. The upsampling operation expressed in Equation 1 can be implemented in LabVIEW using the function ***Upsample***. This can be found under ***Signal Processing>>Signal Operation***



This function inserts zeros between the samples of the input sequence ***X***, where the ***upsampling factor*** determines the number of zeros. Connect a constant value 2 to the ***upsampling factor***.

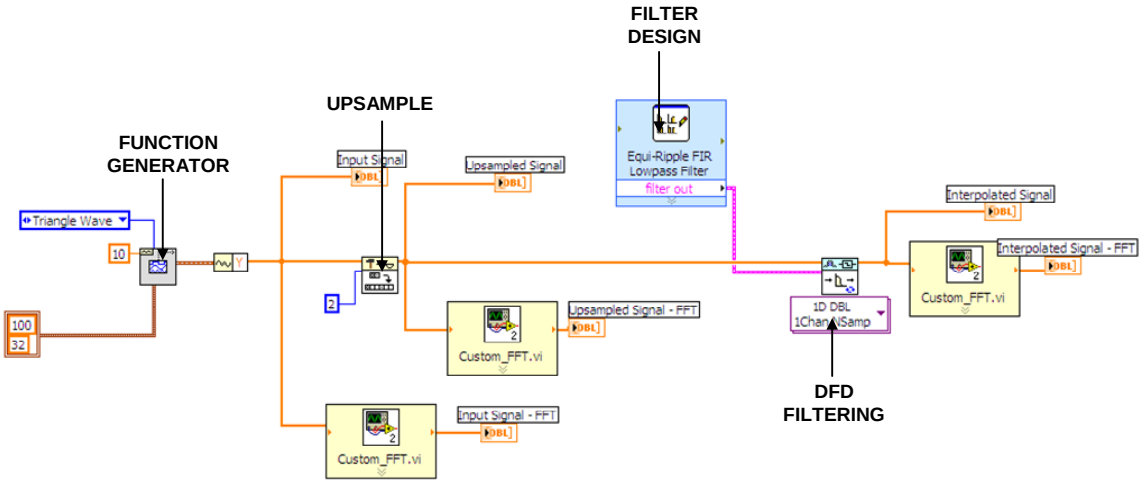


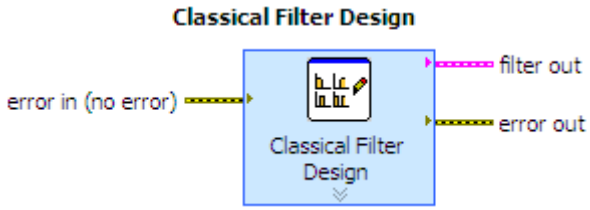
Figure 6. LabVIEW Block Diagram for Exercise 2

As in the previous section, plot the upsampled signal (discrete) and the FFT magnitude response of the same.

➡ **What do you observe from the FFT response of the upsampled signal?**

2.3 Interpolation Filter

To perform interpolation to the upsampled signal, you need to generate a low pass filter, whose cut-off frequency is theoretically at $\frac{\pi}{L}$. Here, π denotes the Nyquist rate and it is equal to 50 Hz in our example. But, designing a perfect filter is a critical design issue and in most cases we end up designing approximate filters. Further, the order of the filter becomes really high, when we design filters with a very sharp cut-off. This motivates the use of high upsampling factors in real time applications, which allows the filter design to be crude. Following this practice, we will also design a fairly crude low pass filter to perform interpolation. We use the LabVIEW function ***Classical Filter Design*** to create the FIR filter (low pass)



Once the function is placed in the block diagram, a configuration dialog window opens, in which enter the following specifications

- Sampling Frequency – 100 Hz
- Passband Edge Frequency – 10 Hz
- Stopband Edge Frequecy – 35 Hz
- Design Method – Equi-ripple FIR

Notice that the filter designed will not have a very sharp cut-off. Make a note of the order of this filter designed. The upsampled signal can be filtered with the designed filter using the toolkit function ***DFD Filtering***.

As earlier, plot the filtered time-domain signal (discrete) and the FFT magnitude response. The layout of the Front Panel of this exercise is shown in Figure 7.

EXERCISE 2

Observe that the signal output (in time domain) from the interpolation filter is delayed. Find out the delay (in samples) and explain the reason for this.

Compare the amplitudes of the original and interpolated signals in time domain and make suitable comments.

Provide the screenshots of the block diagram and the front panel.

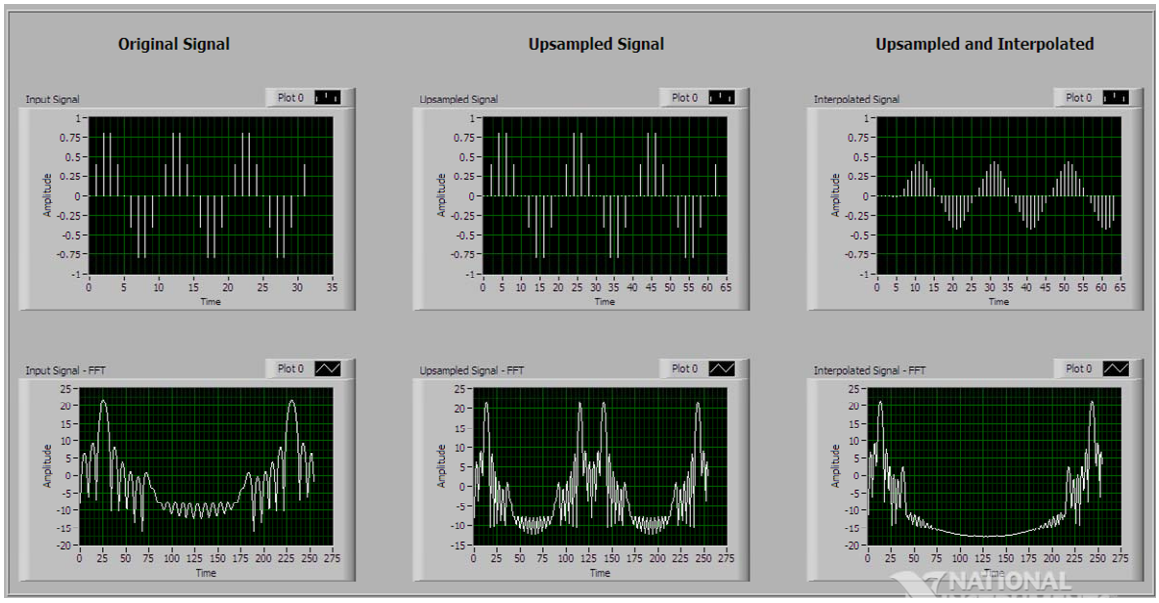


Figure 7. Front Panel Layout for Exercise 2

Exercise 3 – Music Equalizer

In this exercise you will need to implement a simple music equalizer. Equalizer is used to amplify or attenuate a particular band of frequencies of a given signal in order to get better sound effects. Figure 8 shows the design of a basic equalizer. As shown in the figure, the input signal is divided into different frequency bands by a series of bandpass filters (BPF) and then each band is attenuated or boosted by different Gain factors (G) and all the bands are added finally to generate a composite signal. The frequency bands to be used in your design are provided in Table 1. You can use the sample speech file (*cleanspeech.wav*) provided in Lab 1 as the input signal.

Hint : Use the Express VI ***Filter*** to split the signal into bands and multiply the filtered signal with a Gain factor. For this experiment allow the Gain factors to vary in the range 0 to 10. You can use the function ***Compound Arithmetic*** to combine the filter outputs. You can use the subVI ***Playsound.vi*** in your block diagram to playback the speech. Figure 9 and Figure 10 illustrate the sample block diagram and Front Panel Layouts for your reference.

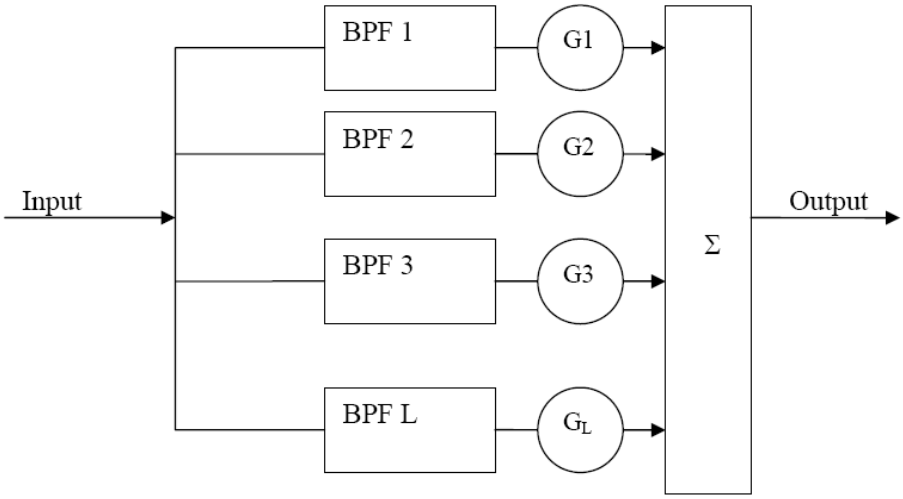


Figure 8. Design of a simple equalizer

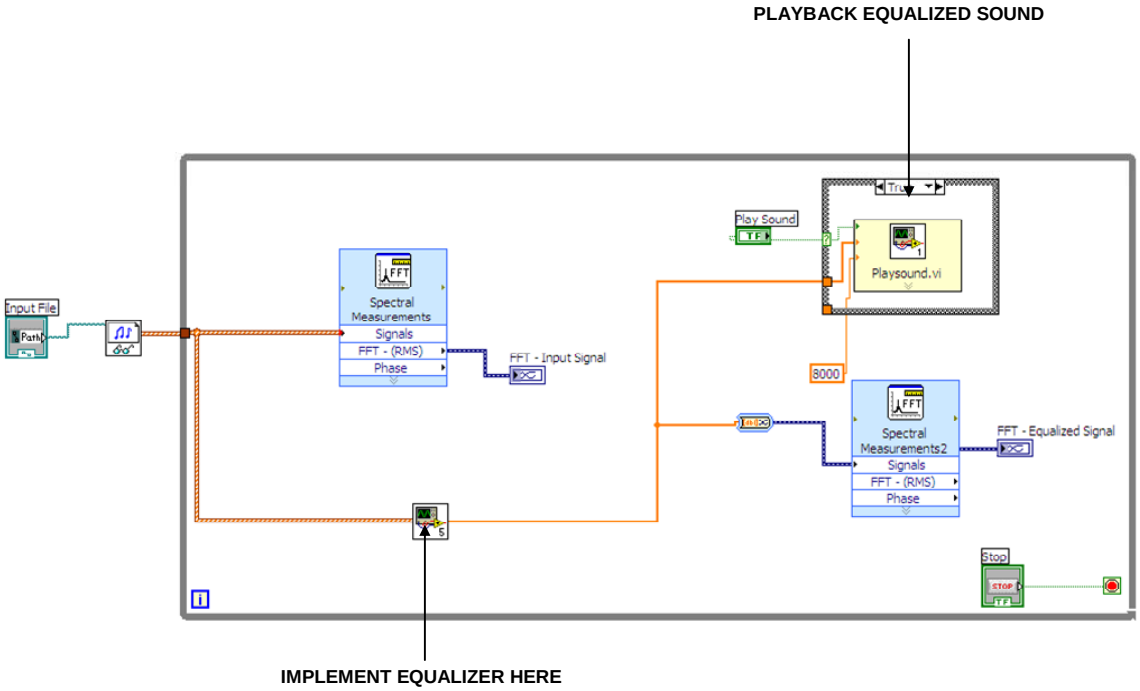


Figure 9. Sample Block Diagram Layout for Exercise 3

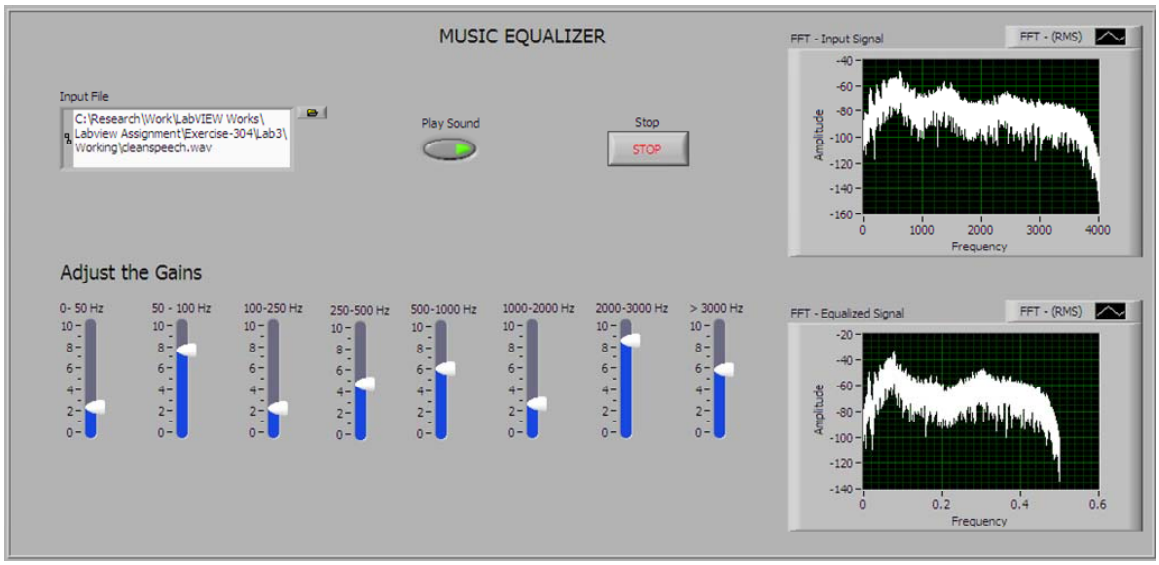


Figure 10. Front Panel Layout for Exercise 3

Band	Lower Cutoff (Hz)	Higher Cutoff (Hz)
1	0	50
2	50	100
3	100	250
4	250	500
5	500	1000
6	1000	2000
7	2000	3000
8	3000 or higher	

Table 1. Frequency ranges for Exercise 3

EXERCISE 3

Provide the screenshot of your block diagram and front panel for Exercise 3 and the sub-VI of the Equalizer.

Vary the gains of the frequency ranges and comment on the changes in the output sound that you observe.