Interpolation Filtering

David Perez
11/13/2023
College of Engineering
University of Nebraska-Lincoln
Lincoln, Nebraska, United States
dperez11@huskers.unl.edu

***Interpolation is a mathematical concept in which a system has points inserted to adjust the nature of the system.* In this report, the interpolation technique will be used to distort signals from a given audio file. Additionally, low pass filters will be investigated as another form of signal processing.**

I. Introduction

This report seeks to investigate the 4 different scenarios in which interpolation filtering can be used to adjust audio signals. The first approach will be to simply up sample our audio signal and insert zeros into our original signal. Next, hold-interpolation will be used to move those up sampled zeros to the previous value that each audio sample had. Followed by that will be the linear interpolation which places the new up sampled zeros to linear lines between the original audios datapoints. Lastly, low pass filtering will be used to remove undesirable frequencies.

II. Filters

A. Upsampling

The first filter of interest is a simple up sampling of the original signal. The input signal was arbitrary speech audio file that was sampled at 8000 samples per second. Below in Figure 1 shows a partition (samples 1501 – 1600) of this original signal.

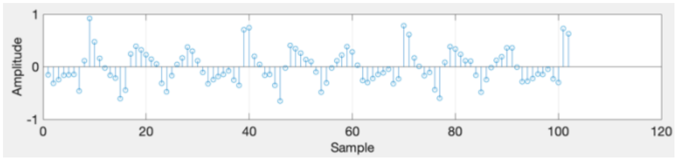

Fig. 1. Original Signal Partition

Given these samples, we now want to perform and up sampling of this signal by a factor of 4. This can be achieved with the kron() Matlab function by passing in partitioned samples along with the vector [1;0;0;0]. Doing so, produces the new samples shown in **Figure 2 and 3**. Something worth noting is that when performing up sampling or interpolation of any sorts adds a delay to the input signal. To remove this delay the signal would have to be shifted back to the zero location, but this has been omitted for clarity.

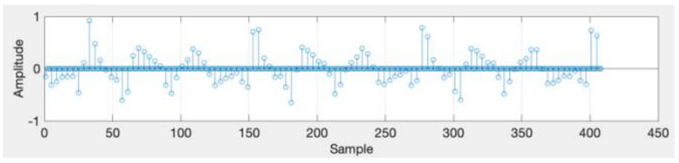

Fig. 2. Upsample by Factor of 4

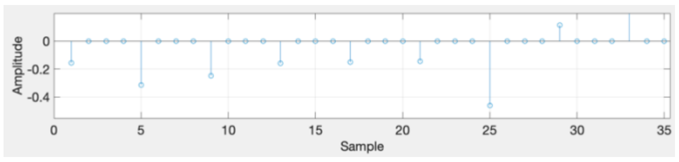

Fig. 3. Upsample by Factor of 4 Zoomed

B. Hold Interpolation

Using this new up sampled audio our signal hold interpolation can then be performed. Hold interpolation sets those new zeros to be equal to the previous sample of the original signal. This interpolation can be seen in **Figure 4 and 5** below.

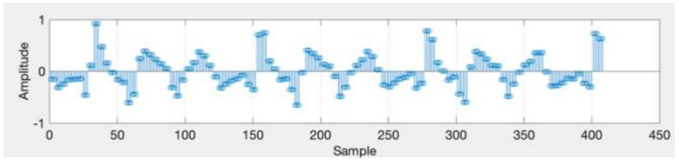

Fig. 4. Hold-Interpolation

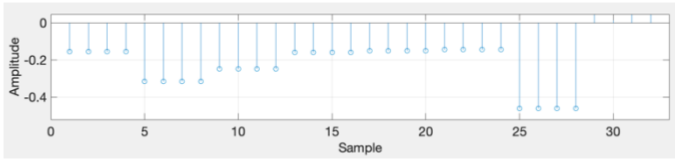

Fig. 5. Hold Interpolation Zoomed

C. Linear Interpolation

Like hold interpolation, linear interpolation adds extra information into the signal. Instead of adding additional values that are the same as the most recent sample, linear interpolation places a linear line of samples between two samples in the original audio. It achieves this by taking the new up sampled samples and puts them on the linear line connecting two original data samples. This type of interpolation can be achieved using Matlabs intfilt() function and passing that information through the filter() function to apply it to our up sampled audio. This linear interpolation is shown in **Figures 6 and 7.**

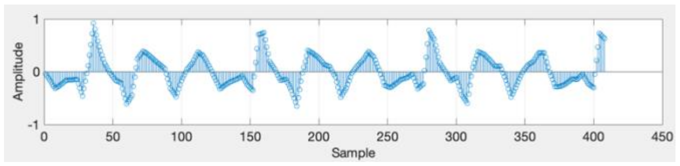

Fig. 6. Linear Interpolation

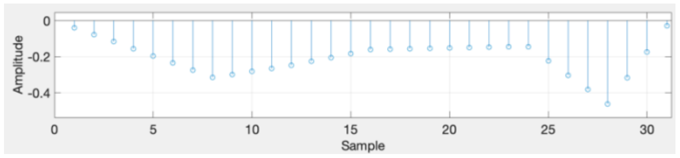

Fig. 7. Linear Interpolation Zoomed

D. Lowpass FIR Filter

The next filter of interest is the FIR low pass filter. This filter has a pass band edge at 929 Hz with the designed cut off frequency being 1.2 * 929 Hz. We then take that new value and divide it by half the sampling frequency of the original audio signal to get our cutoff frequency to it into Matlabs normalized frequency. For the order of this filter, a value of 128 was arbitrarily chosen. Additionally, we create a pass band ripple at 1 dB and a stop band attenuation at 40 dB. Using Matlabs Freqz() function we can then generate the plot show in **Figure 8**.

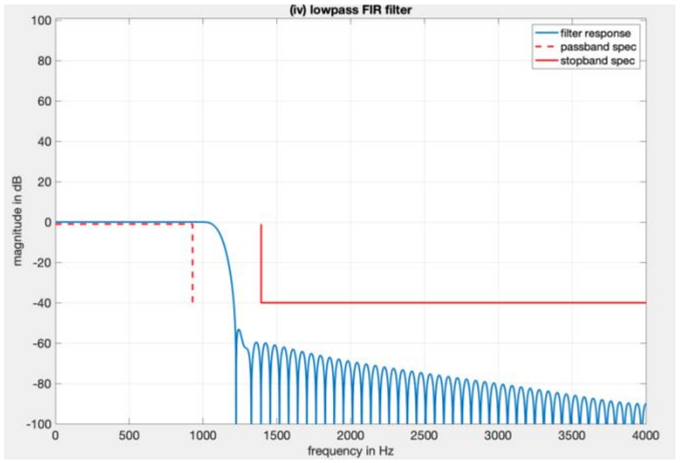

Fig. 8. Part (iv) Lowpass Filter

To observe the effect of this filter, the same samples from the previous interpolation filtering was used and this low pass filter was applied to that data using Matlabs filter() function.

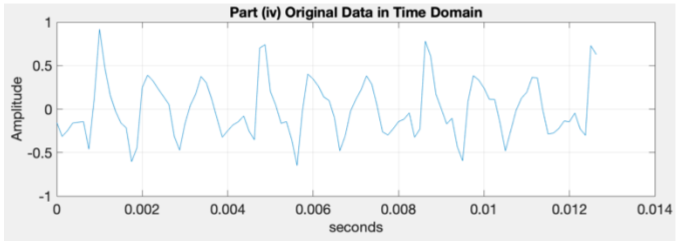

Fig. 9. Part (iv) Original Data In Time Domain

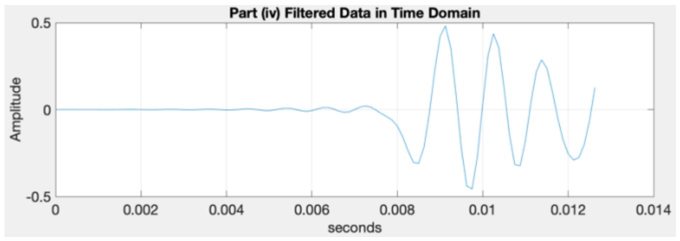

Fig. 10. Part (iv) Filtered Data In Time Domain

To get a better understanding of how this data effected our signal is often more insightful to observe the signals changes in the frequency domain. To see how this filter effects the frequency of our signals, the original and new signal can be seen in the frequency domain in **Figures 11.**

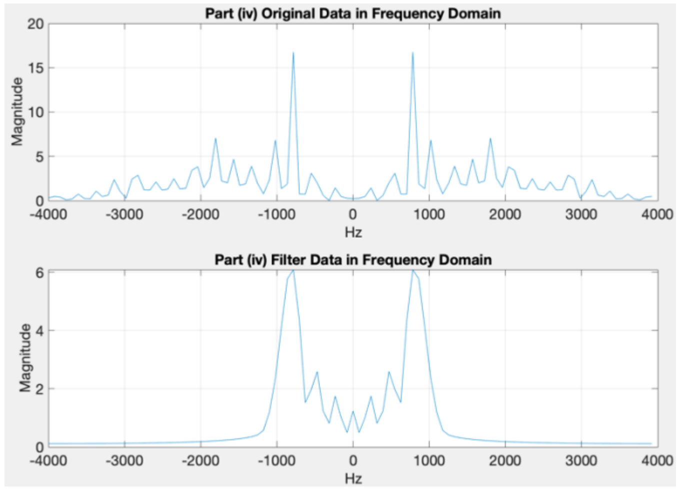

Fig. 11. Part (iv) Frequency Domain Plots

When observing the frequency domain plots of the before and after filtering we can see how the low pass filter blocks out the higher frequencies from the input audio signal. Since we typically don't experience ideal filters, additional 1.2 times the sampling frequency is more realistic, therefore making the cutoff frequency of our filter at about 1114.8 Hz. In **Figure 11** plots this can be seen by the drop off in magnitude on the bottom plot at that frequency.

E. Second Lowpass FIR Filter

For our second low pass filter 200 samples were used to run the low pass filter through. This new low pass filter was chosen to have an order of 200. For the cutoff frequency, a value of 1680 was used (1.2 * 1400). The resulting filters impulse response can be seen in **Figure 12.**

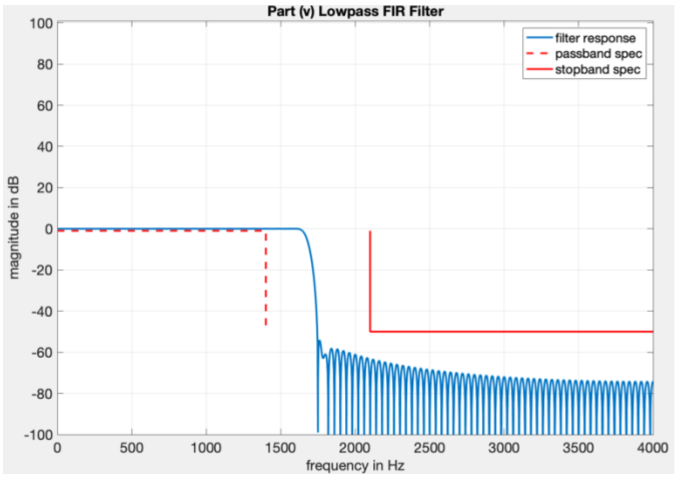

Fig. 12. Part (v) Second Low Pass Filter

When observing this new filter we can see how the higher the order increased the speed that the signal magnitude dropped on of the filters impulse response shown in **Figure 12**. Again, Its also worth noting how this rapid drop off occurs at the cutoff frequency of 1680.

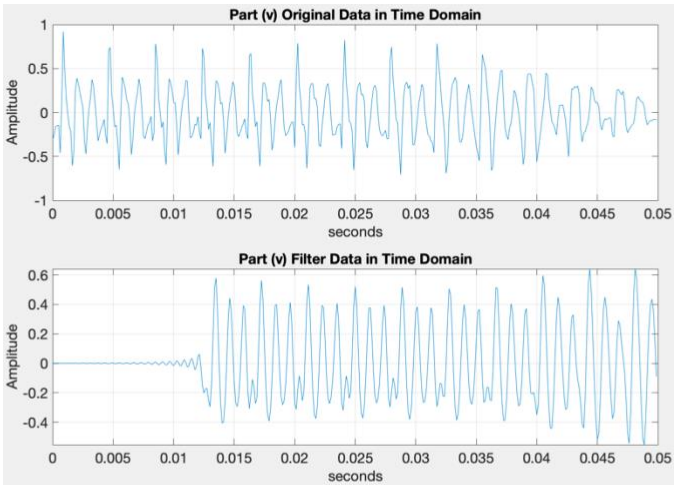

Fig. 13. Part (v) Second Low Pass Filter Time Domain

Analysizing the **Figure 13** it is apparent that there was a lot of higher frequencies at the start of the original data since the filtered data has little to zero magnitude at the start of its time domain plot. For further analysis, a plot of the original datas frequency domain vs the filtered frequency domain can be seen in **Figure 14.**

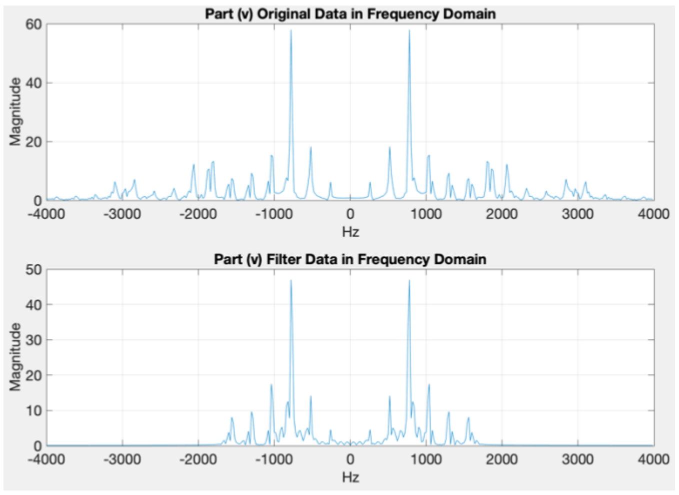

Fig. 14. Part (v) Second Low Pass Filter Frequency Domain

Observing the Freuency domain plot it can be seen that original data signal did in fact contain higher frequencies (though with low magnitude). When looking at the filtered data, we see that the filter performed as expected and removed the higher frequencies from the input signal. Since the cutoff frequency was set to 1680, from this point on the filtered firequency domain plot shows values of zeros.

III. CONCLUSION

Through the analysis and experimentation of various filtering techniques we observed how these techniques allow for varied distortions in the audio signals. We started off with basic up sampling which adds zeros to our input signals. This was followed up by hold interpolation which enabled us to add additional samples that contained the same value as the most recent sample in the original signal. This was then followed by linear interpolation which connected a linear line of samples between each original data sample. Lastly, two low pass filters were implemented to cut out undesirable frequencies from our input signal. All in all, this experimentation emphasized some basic filtering techniques and how these various filters can be used to adjust audio signals with Matlab.