5-BAND AUDIO EQUALIZER USING THE DUAL RECURSIVE RUNNING SUM FILTER

Digital Signal Processing, ECSE-512
Rayan Isran, McGill ID: 260950238
rayan.isran@mail.mcgill.ca
10th December, 2020

ABSTRACT

This paper outlines the design of a 5-band linear phase audio equalizer comprised of a parallel network of dual recursive running sum (DRRS) filters, along with the MATLAB code and GUI implementation. Implementation of the algorithm was tested by applying gains of +5 dB and -5 dB to each frequency band while keeping others unchanged at zero gain. Graphs are shown that compare the frequency spectrum of the signal before and after equalization. The implementation allows the user to choose music presets or their own values for the gain of each band.

TABLE OF CONTENTS

1. Introduction

1.1 Audio Equalization

Any system that changes the audio signal passed through it to improve transmission or use of the signal can be considered an audio effects system. These effects can be applied to an analog sound signal through electronic circuits or instruments, such as RC-networks, or simulated through computer tools in what is commonly known as digital signal processing. Digital audio effects are algorithms which take in a digital input audio signal, modify them according to the desired parameters, and deliver the desired output digital audio signal.

One of the most commonly used effects in digital signal processing is audio equalization, in which the relative strength of different frequency bands within a signal is adjusted, changing the way the original audio sounds. The term has its origins in early telegraph engineering, when high frequency losses over long distances had to be corrected or "equalized" at the receiver side to match the transmitter side spectrum, due to the inherent capacitances of long wires [1].

Tone control is a class of audio equalization effects. The most basic tone control system contain two knobs labeled '*bass'* and '*treble*' which control the level, or more commonly known as the gain, of the low and high frequencies within a signal, typically from 100-300 Hz and 3-10 kHz respectively. Tone control circuits were first introduced for use with gramophones in 1949 [2]. In 1952, Baxandall [3] devised a tone control system using potentiometers, giving the user full control over the applied gain.

1.2 Digital Approaches to Tone Control

Digital audio equalizers emerged in the late 1970s and early 1980s [4]-[6]. They can be classified into four categories: graphic, console, parametric, and tone-control. However, this paper will focus on graphic equalizers, a generalized form of tone-control algorithms [7].

While the simplest tone-control system uses knobs to control the low and high frequencies, a graphic equalizer divides the audible spectrum logarithmically into five or more frequency bands that together cover the human audible spectrum (20-20000 Hz). The gain of each band, G, can be adjusted independently using a *boost* or *cut* control. The gains are often controlled using sliders and can be varied within the range of ± 12 dB, corresponding to approximately $0.25 < G < 4$ for each filter. The equalizer is named so because the position of sliders can be understood as a graph of the equalizer’s magnitude response against the frequency, which makes the tool intuitive to use despite the number of controls. Some digital music players have presets, such as rock, classical, pop, and jazz, where particular gains are set for each frequency band in the signal to match the corresponding music style.

Graphic equalizers can either be implemented by cascading equalizing filters [8], creating a parallel bank of band-pass filters [9], or using a hybrid of the two networks [10]. Infinite impulse response (IIR) filters are used throughout, thus, previous samples of the input and output are required. IIR filters are used partly to emulate the analog models, but also to minimize the processing delay; finite impulse response (FIR) filters typically exhibit higher latency for the same performance and require more computational power due to the high filter order [7]. FIR graphic equalizers have been developed to realize a linear phase response, which may be preferred to minimize the phase distortion, but this topic is largely debated among audio engineers due to an audible delay with the FIR filter approach [11].

1.2.1 Cascaded Equalizers

In the cascaded form, the output of one filter becomes the input of the next. Each filter determines the magnitude response of the system around its centered frequency according to the gain chosen. The overall frequency response is the product of the individual filter responses (Eq. 1). Ideally, there is unity gain at all other band center frequencies to avoid interaction between the filters, but in practice, each gain covers a wide frequency band, therefore, the total gain differs from the calculated value.

$$H(z) = G_0 \prod_{n=1}^{N} H_n(z) \qquad (Eq. 1)$$

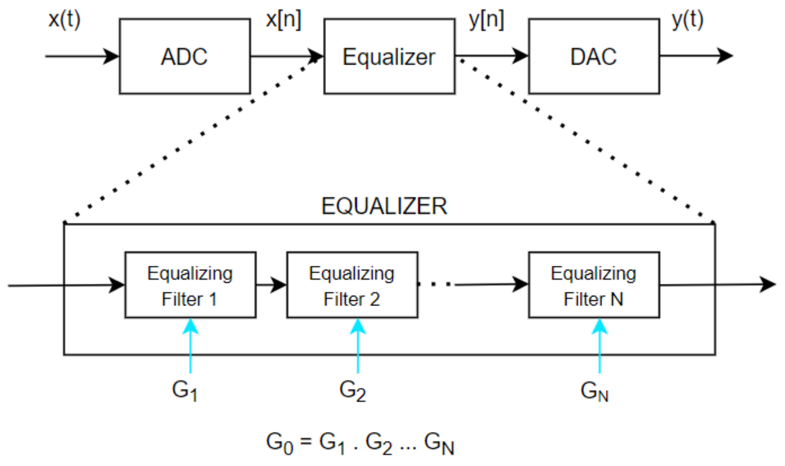


Figure 1.1: A block diagram showing the process of cascade equalization using N equalizing filters. x(t) is the analog signal, x[n] is the digitized input signal, y[n] is the equalized output digital signal, and y(t) is the converted analog output signal. G_N represents the gain of filter F_N.

1.2.2 Parallel Equalizers

In this implementation, the filters are arranged in parallel. The input signal is split and sent to the input of each band-pass filter, as shown in Figure 1.2. Each filter only allows a narrow frequency band to pass through, with a control that allows the user to set its gain. Ideally, the center frequencies are configured so that the original signal is reconstructed when the output of each band-pass filter is added together. The overall transfer function of the parallel equalizer is obtained as the sum of all band filters:

$$H(z) = \sum_{n=1}^{N} G_n H_n(z) \qquad (Eq. 2)$$

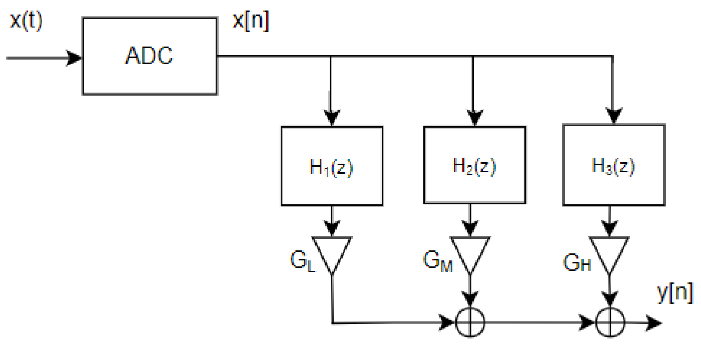


Figure 1.2: Parallel implementation of the graphic equalizer in which three bands are shown.

In theory, the parallel and cascaded implementations are mathematically equivalent but their designs are treated differently. The parallel implementation avoids accumulating phase errors and potentially, quantization noise present in the cascade. However, Eq. 2 implies that the phase of each band-pass filter affects the total frequency response. For good accuracy it is necessary to optimize the gain and phase of each filter; this is inevitably more complicated than the design of a cascade structure.

Graphic equalization is commonly used in live performances and recording studios. A common use is to tune a room, adjusting the frequencies to compensate for the frequency response distortion or resonance in the room [12]. The goal is to obtain a desired frequency response, flattening extreme peaks, and achieving greater consistency among performance venues, public address systems, and mobile microspeakers [13]. Other uses may include increasing the drum and bass frequencies at a dance party, or reducing bass sounds when listening to a person speaking. Equalizers are also found in digital music software, such as *Audacity*, and used by audio artists to adjust the frequency response to accommodate the listener's preference.

2. The Dual Recursive Running Sum (DRRS) Filter

FIR filters maintain a linear phase response and consequently minimize any phase distortion in the signal, which may be preferred in some audio applications. However, FIR filters are computationally expensive because of the high filter order. This makes them unsuitable for tone control, especially during live performances where minimal delay is required. In 1986, Lim [14] proposed a technique that significantly reduced the complexity of sharp FIR filters in what is called the Frequency Response Masking (FRM) technique. Lim [15] and Lian [16] proposed clever methods to design linear phase FIR filters using long delay lines and relatively few operations per sample. In 1989, Yang [17] proposed a tone control approach using the running recursive sum (RRS) filter.

The RRS filter is an FIR filter structure with an impulse response given by:

$$h(n) \triangleq \begin{cases} 1, & n = 0, 1, 2, \dots, L-1 \\ 0, & otherwise \end{cases}$$

The output signal, $y(n)$, is given by the convolution of the signal with the impulse response of the filter:

$$y(n) = h(n) * x(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-m) = \sum_{k=0}^{L-1} x(n-k) \qquad (Eq. 3)$$

$$y(n) = \ x(n) + x(n-1) + \ x(n-2) + \cdots + x(x-L+1) \qquad (Eq. 4)$$

Thus, we can see why this is called a "running sum". If divided by the length of the impulse response, L, it becomes a moving average filter. The z-transform of running sum filter is given by:

$$H_{RRS}(z) = 1 + z^{-1} + z^{-2} + \cdots + z^{-L+1} = \frac{1-z^{-L}}{1-z^{-1}} \qquad (Eq. 5)$$

With a time domain difference equation of:

$$y(n) = \ [x(n) - x(n-L)] + y(n-1) \qquad (Eq. 6)$$

And a cascade structure as:

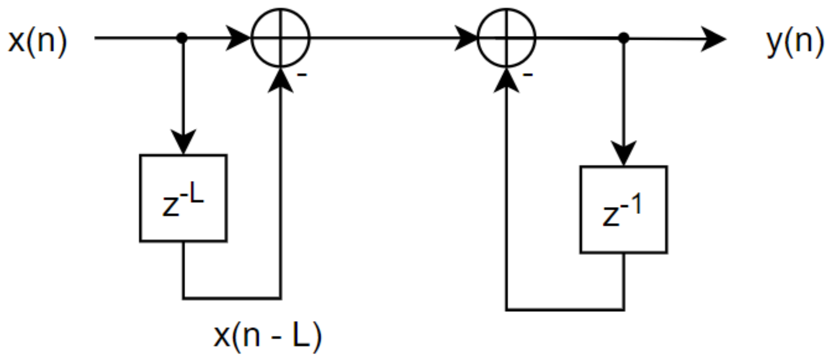


Figure 2.1: The cascaded Direct-Form I structure of the RRS.

And we can see that this is an efficient version of the standard moving average filter, as only two additions are required per output sample.

The frequency response of the filter is given by the DFT of the impulse response:

$$H_{RRS}(e^{-jw}) = \frac{1 - e^{-j\omega L}}{1 - e^{-j\omega}} = \frac{e^{-\frac{jwL}{2}}\sin(\frac{\omega L}{2})}{e^{-\frac{jw}{2}}\sin(\frac{\omega}{2})} \qquad (Eq.7)$$

$$\triangleq Ne^{-\frac{j\omega(L-1)}{2}} asinc\, N(\omega) \qquad (Eq.8)$$

Where $\frac{L-1}{2}$ is the delay introduced by the linear phase term. As only two additions are required per sample, we can experiment with cascading two filters, in what is called the Dual Recursive Running Sum (DRRS) filter. By cascading two RRS filters, the stopband attenuation can be increased, decreasing the stopband attenuation requirements on the equalizer [18]. This can be shown by writing a few lines of MATLAB code and plotting the results, as shown in Figure 2.2. It can also be seen that multiplying the response by the selected coefficients will result in unity DC gain, thereby keeping the amplitude of the signal approximately unchanged.

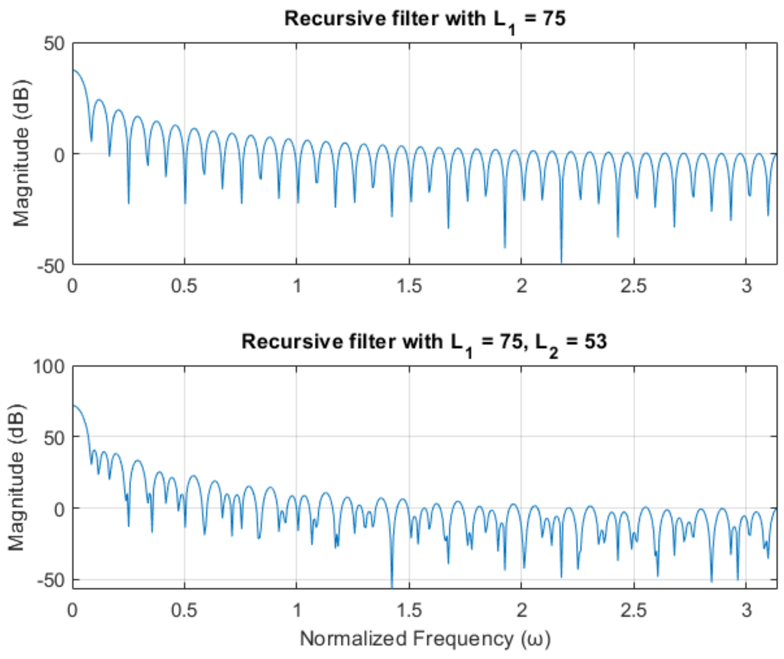


Figure 2.2: The magnitude response of the RRS (top) and the DRRS (bottom) with values of L_1 = 75, L_2 = 53. The stopband attenuation is greater with the DRRS by approximately 10 dB.

With a cascade of two filters, the frequency response of the DRRS is given by:

$$H_{DRRS}\left(e^{-jw}\right) = e^{-\frac{jw}{F_s}(L_1+L_2-2)}\frac{L_1 L_2 \sin\left(\frac{\omega L_1}{2F_s}\right)\sin(\frac{\omega L_2}{2F_s})}{\left[\sin\left(\frac{\omega}{2F_s}\right)\right]^2} \qquad (Eq. 9)$$

And the coefficients for the DRRS can approximately be chosen as follows:

$$L_1 = \frac{1}{2}\frac{Fs}{Fc} \qquad (Eq. 10)$$

$$L_2 = int(\frac{L_1}{\sqrt{2}}) \qquad (Eq. 11)$$

Where *Fs* represents the sampling rate of the signal, *Fc* represents the cut-off frequency, and *int()* represents the closest odd integer to, in Eq. 11.

3. Proposed Approach and Algorithm for Equalization

The 5-band audio equalizer created in this paper uses the DRRS approach, i.e., a parallel implementation, using five frequency bands. For each band, the input signal is passed through a DRRS filter with a user-adjusted gain (if desired). The output digital audio signal is then obtained by summing the frequency-adjusted signal of each band. Four DRRS pairs are required for a five-band equalizer. The cut-off frequency values chosen follow a logarithmic scale to match human audio perception. Coefficient values were initially calculated using *Eq. 10* and *Eq. 11*, but were adjusted slightly based on the output after testing the algorithm with white noise. The sampling rate of the input is assumed to be 44.1 kHz (standard CD audio sampling rate), and the cut-off frequencies that were chosen are listed below:

Band	Frequency range	L_1	L_2
Low-pass	0 – 500 Hz	39	27
Band-pass 1	500 – 1000 Hz	21	15
Band-pass 2	1000 – 2000 Hz	7	5
Band-pass 3	2000 – 4000 Hz	5	3
High-pass	4000 – 10000 Hz	-	

Table 3.1: The DRRS coefficients used to compute the cutoff frequencies for each band

The network diagram is shown in Figure 3.1. For clarity, (L_1, L_2) represents the low-pass DRRS pair; (B_1, B_2) represents the band-pass 1 pair; (C_1, C_2) represents the band-pass 2 pair, and (H_1, H_2) represents the high-pass pair.

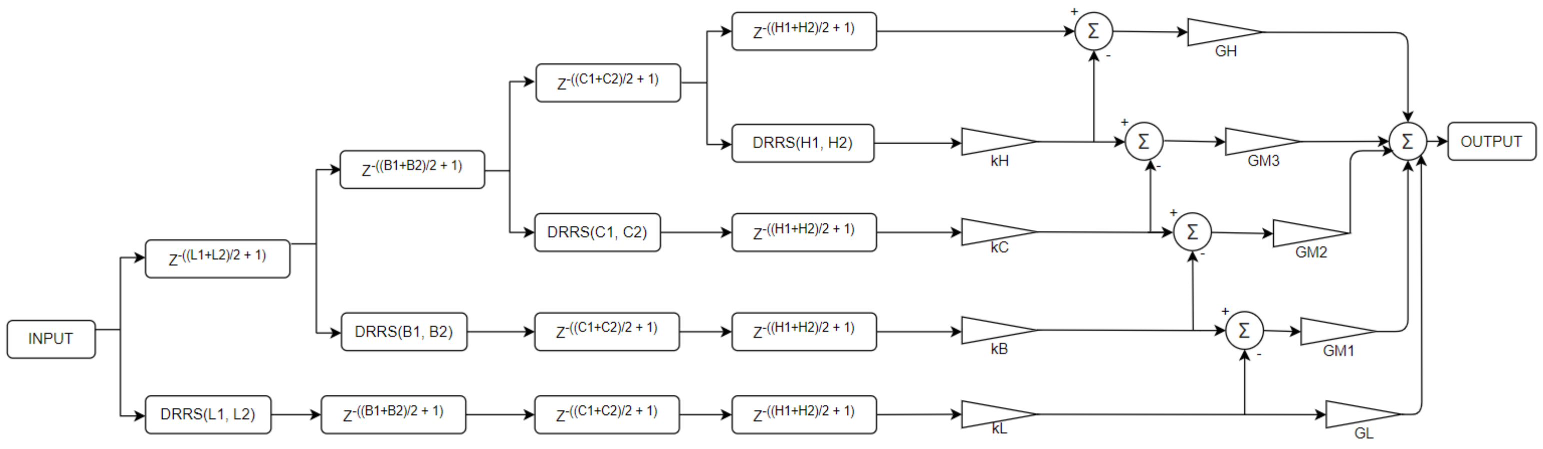


Figure 3.1: The network diagram of the 5-band tone control equalizer.

The transfer function of the network is given by:

$$F_{net}(z) = F_H(z) + FM_3(z) + FM_2(z) + FM_1(z) + F_L(z)$$

$$F_{net}(z) = G_H z^{-\frac{L1+L2+B1+B2+C1+C2}{2}+3}\,[z^{-\frac{H1+H2}{2}+1} - kH\; DRRS(H_1, H_2)] +$$

$$G_{M3} z^{-\frac{L1+L2+B1+B2}{2}+2}\left[k_H z^{-\frac{C1+C2}{2}+1}\, DRRS(H_1, H_2) - k_C\, z^{-\frac{H1+H2}{2}+1}\, DRRS(C_1, C_2)\right] +$$

$$G_{M2} z^{-\frac{L1+L2}{2}+1}\left[k_C z^{-\frac{B1+B2+H1+H2}{2}+2}\, DRRS(C_1, C_2) - k_B\, z^{-\frac{C1+C2+H1+H2}{2}+2}\, DRRS(B_1, B_2)\right] +$$

$$G_{M1} z^{-\frac{L1+L2}{2}+1}\left[k_B z^{-\frac{C1+C2+H1+H2}{2}+2}\, DRRS(B_1, B_2) - k_L\, z^{-\frac{B1+B2+C1+C2+H1+H2}{2}+3}\, DRRS(L_1, L_2)\right] +$$

$$G_L K_L z^{-\left[\frac{B1+B2+C1+C2+H1+H2}{2}+3\right]}\, DRRS(L_1, L_2) \qquad (Eq. 12)$$

Ten delay blocks are required to ensure the adders have the same phase. Half-delay units are avoided as all DRRS coefficients are odd numbers. To offset the amplification effect of the filters, attenuators *kH*, *kC*, *kB*, and *kL* are included, with values of $\frac{1}{H_1H_2}$, $\frac{1}{C_1C_2}$, $\frac{1}{B_1B_2}$, and $\frac{1}{L_1L_2}$ respectively. The band pass and high pass filters are obtained by subtracting the output of the DRRS from the delayed versions of the input.

G_H, G_{B3}, G_{B2}, G_{B1}, and G_L represent the individual gains for each frequency band. The UI has options to choose presets for specific music styles, but the user can control the sliders of each band for full control.

The MATLAB *audioread* and *audiowrite* functions are used to load and export audio files respectively. When a file is loaded, the user can view the input frequency spectrum. The output spectrum of the signal, based on the gain changes of each band, is shown if the user clicks on the 'Generate' button. The MATLAB implementation contains the following files:

Function	**Parameters**	**Description**
DRRS.m	L_1, L_2, signal	Computes the DRRS using the coefficients L_1 and L_2. Returns the filtered signal.
delay.m	amount, signal	Delays a signal by *amount* number of samples. Returns the delayed signal.
equalizer_preset.m	Preset	Returns the gains for all bands for a chosen musical preset.
spectrum.m	signal, sampling rate	Returns the FFT of the signal from its time-domain equivalent.
audio_eq.m	G_L, G_{M1}, G_{M2}, G_{M3}, G_H	Computes the transfer function of the DRRS network using the signal with the gains of each frequency band. Returns the equalized signal.

Table 3.2: Listing of MATLAB functions used for the implementation

The architecture of the program is summarized as follows:

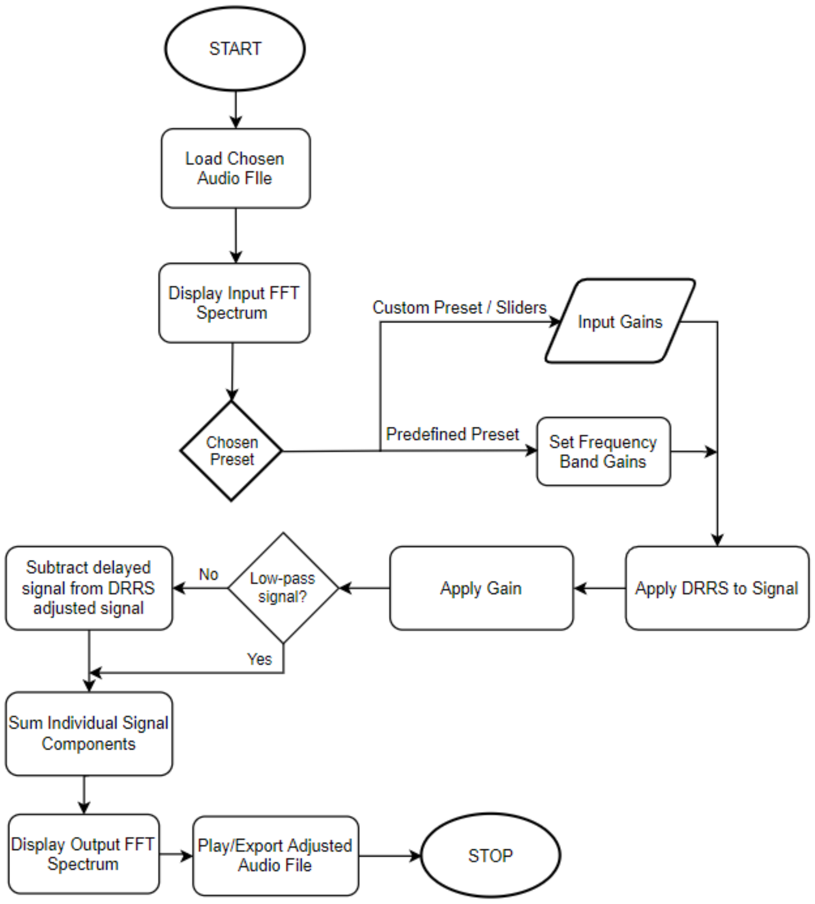
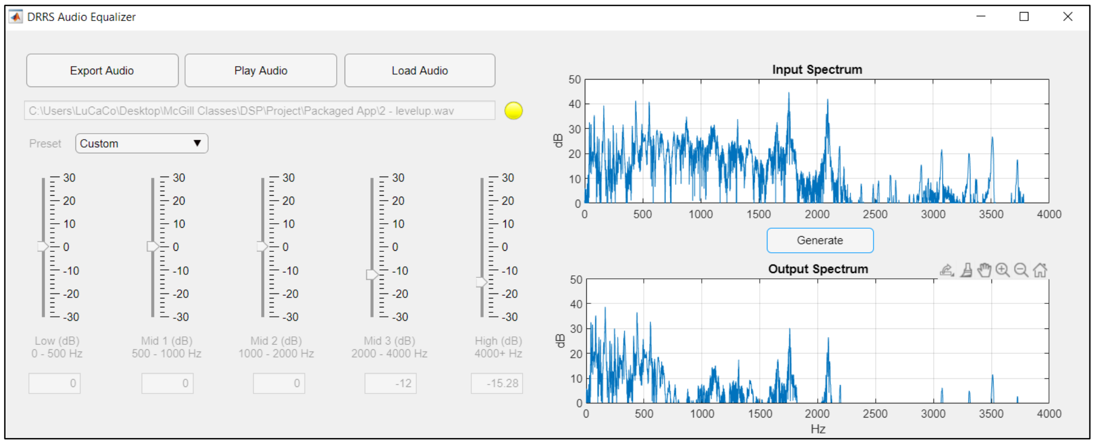


Figure 3.2: Flowchart depicting the equalization algorithm as implemented in MATLAB.

The layout of the program is shown below:



C:\Users\LuCaCo\Desktop\McGill Classes\DSP\Project\Packaged App\2 - levelup.wav

Preset: Custom

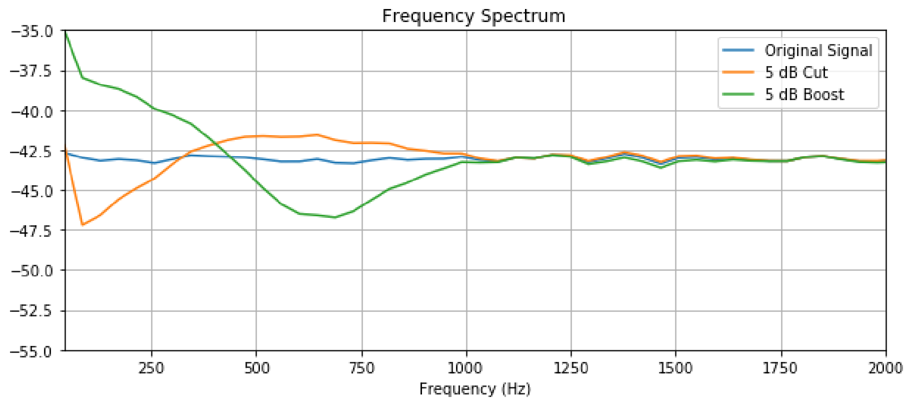
Low (dB) 0 - 500 Hz	Mid 1 (dB) 500 - 1000 Hz	Mid 2 (dB) 1000 - 2000 Hz	Mid 3 (dB) 2000 - 4000 Hz	High (dB) 4000+ Hz
0	0	0	-12	-15.28

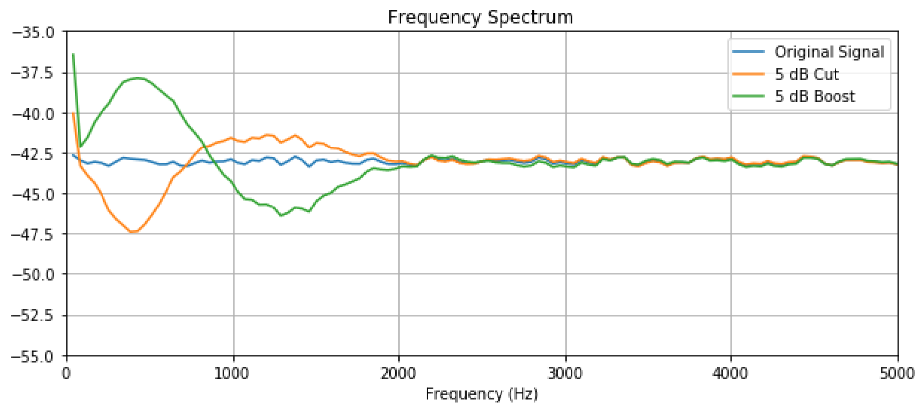
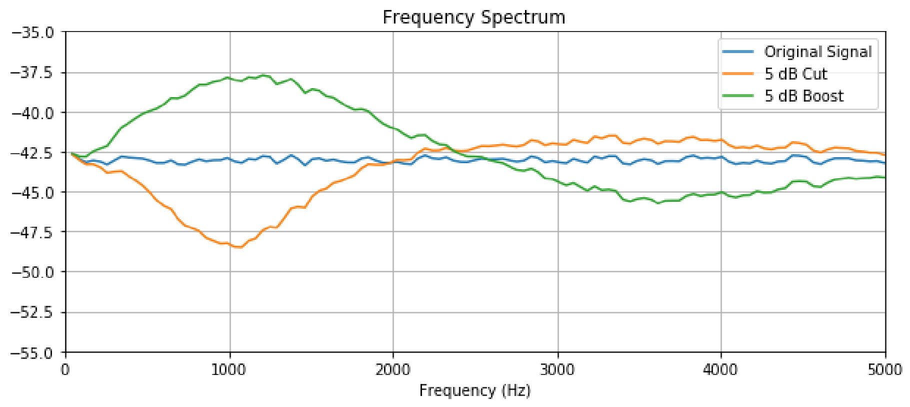
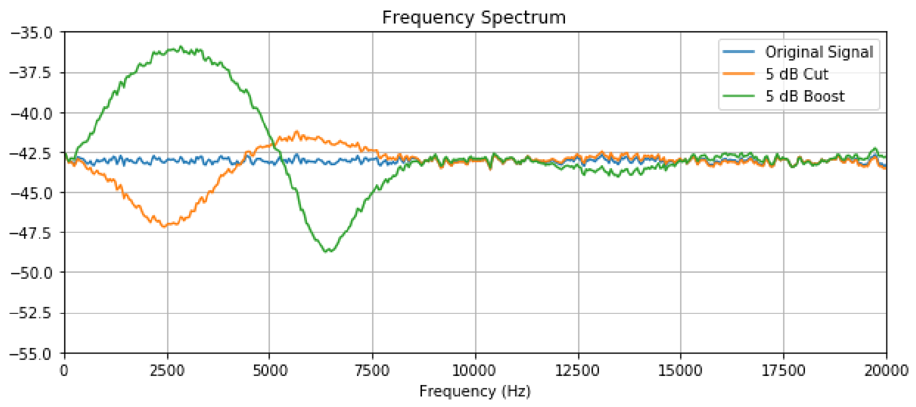
Figure 3.3: The DRRS Audio Equalizer interface with the 'Custom' Preset chosen and user chosen gains for each band.

4. Results & Analysis

Each frequency band was individually tested using white noise with frequencies evenly distributed across the human perceptible spectrum (20–20000 Hz) and a cut of approximately 5 dB, followed by a boost of approximately 5 dB, while leaving all other band gains unchanged at 1. The sampling rate of the white noise was 44.1 kHz, standard to that used in CD formats. We noted the change first by listening to the altered audio signal for a ‘feel’ in change of the frequencies, and then by comparing the frequency spectrum of the original signal with its altered counterpart. The interface displays the input and output spectrum graphs, both of which convert the signal into the frequency domain via a Fast Fourier Transform (FFT). To corroborate the results, the altered .wav files were exported and loaded in the *Audacity* software tool, where data of the frequency spectrums were saved. The data from the original signal and altered signals were then plotted using Python and are shown in Table 4.1.

Secondly, the results of some mixed frequencies are tested followed by discussion on the effectiveness of the filter by comparing the interaction between neighbouring filters.

Frequency Band	
Low-pass	

Band-pass 1	
Band-pass 2	
Band-pass 3	

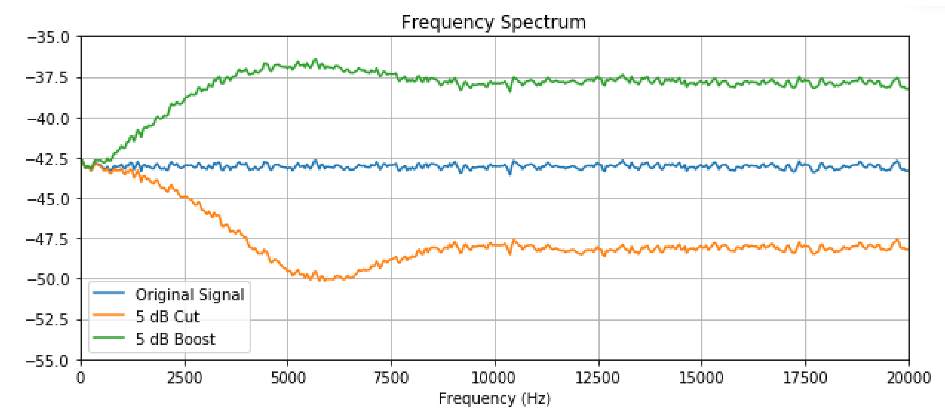
High-pass	

Table 4.1: Graphs depicting the frequency spectrum of the input signal with the equalized signals

As can be seen from the graphs, the filtering effects are far from ideal; the frequency response from each band interacts over a wide range of frequencies, as expected from the background discussion earlier. In particular, for the band-pass filters, boosting the signal by 5 dB tends to cause some attenuation at subsequent frequencies. To mitigate some of these “unwanted” frequency interaction effects, the user can adjust the gain of neighboring frequency bands. The general trend from the graphs in the table above as well as testing by listening to the equalized audio showed that the highest frequency bands contains the highest sensitivity to changes in gain.

The below figures show the spectrum of the included *'levelup.wav'* file, first without any change to the signal (left), and secondly, its adjusted spectrum (right) with multiple gains applied:

	Low (0-500 Hz)	Mid (500 – 1000 Hz)	Mid (1000 – 2000 Hz)	Mid (2000 – 4000) Hz	High (4000+) Hz
Gain (dB)	10	0	10	-10	-10

Table 4.2: Chosen gains for the for the signal whose frequency spectrum is shown in Figure 4.1

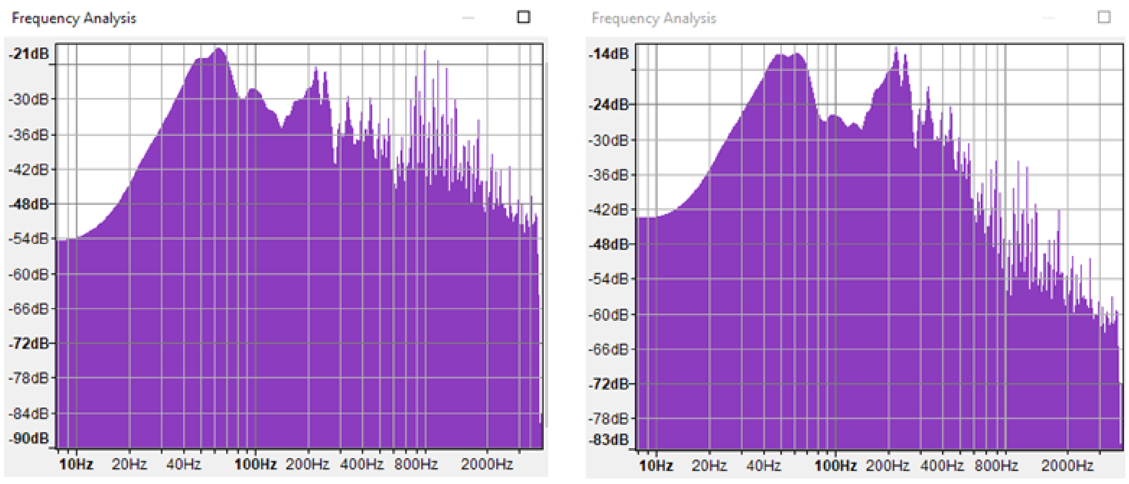


Figure 4.1 Spectrum of the signal in Audacity before equalization (left) and after equalization (right)

5. Conclusion

The designed five-band audio equalizer works as intended, as listeners can distinguish between changes in each frequency band when the gains are altered. With all gains set to 1, i.e., 0 dB, no audible changes can be heard in the equalized audio. Although an FIR filter, the filter is implemented in an IIR form and is simple yet very efficient as only 4 additions are computed per DRRS. The algorithm works reasonably fast for audio that is less than a minute long, but its efficiency could be improved by buffering and sequentially processing the signal in frames. With the DRRS approach, the range of the frequency bands and therefore shape of the tone control can be adjusted by changing the DRRS coefficients.

Appendix A contains instructions on how to use the program.

6. References

[1] V. Välimäki and J.D. Reiss, "All About Audio Equalization: Solutions," *Applied Sciences*, vol. 6, no. 5, p. 129, May 2016.

[2] D.T.N. Williamson, "The Williamson Amplifier," *Wireless World*, vol. 55, 1949.

[3] P.J. Baxandall, "Negative-feedback tone control," *Wireless World*, vol. 58, no. 10, pp. 402-405, 1952.

[4] G.W. McNally, "Microprocessor mixing and processing of digital audio signals," *Journal of the Audio Engineering Society*, vol. 27, no. 10, pp. 793-803, 1979.

[5] C.R. Guarino, "Audio equalization using digital signal processing," in *In Proceedings of the 63rd Convention of*, Los Angeles, CA, USA, May 1979.

[6] Y. Hirata, "Digitalization of conventional analog filters for recording use," *Journal of the Audio Engineering Society*, vol. 29, no. 5, pp. 333-337, 1981.

[7] Joshua D. Reiss and Andrew P. McPherson, *Audio Effects: Theory, Implementation and Application*.: CRC Press, 2014.

[8] V. Välimäki and J. Rämö, "Optimizing a high-order graphic equalizer for audio processing," *IEEE Signal Processing Letters*, vol. 21, no. 3, pp. 301-305, 2014.

[9] R.A. Greiner and M. Schoessow, "Design aspects of graphic equalizers," *Journal of the Audio Engineering Society*, vol. 31, no. 6, pp. 394-407, 1983.

[10] C. Heidelberger and M. Erne, "Design of a DSP-based 27 band digital equalizer," in *Audio Engineering Society Convention 90*, Paris, France, France, 1991.

[11] C.H. Slump et al., "Design and implementation of a linear-phase equalizer in digital audio signal processing," in *IEEE Workshop on VLSI Signal Processing*, Napa Valley, CA, USA, 1992.

[12] L. Bregitzer, *Secrets of Recording: Professional Tips, Tools & Techniques*. New York: Focal Press, 2009.

[13] S. Cecchi, M. Virgulti, P. Andrea, B. Ferruccio P. Francesco, and L. Junfeng, "Investigation on audio algorithms architecture for stereo portable devices," *Journal of the Audio*

Engineering Society, vol. 64, no. 1/2, pp. 75-88, 2016.

[14] Y. Lim, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters," *IEEE Transactions on Circuits and Systems*, vol. 33, no. 4, pp. 357-364, 1986.

[15] Y.C. Lim, "Linear-phase digital audio tone control," *Journal of the Audio Engineering Society*, vol. 35, no. 1/2, pp. 38-40, 1987.

[16] Y.C. Lim and Y. Lian, "Linear-phase digital audio tone control using multiplication-free FIR filter," *Journal of the Audio Engineering Society*, vol. 41, no. 10, pp. 791-794, 1993.

[17] R.H. Yang, "Linear-phase digital audio tone control using dual RRS structure," *Electronics Letters*, vol. 25, no. 5, pp. 360-362, 1989.

[18] Adams. J. and A. Wilson, "Some efficient digital prefilter structures," *IEEE transactions on circuits and systems*, vol. 31, no. 3, pp. 260-266, 1984.

Appendix A: Program Documentation

Figure A.1 shows the interface of the program when the *DRRS Audio Equalizer.mlapp* file is opened:

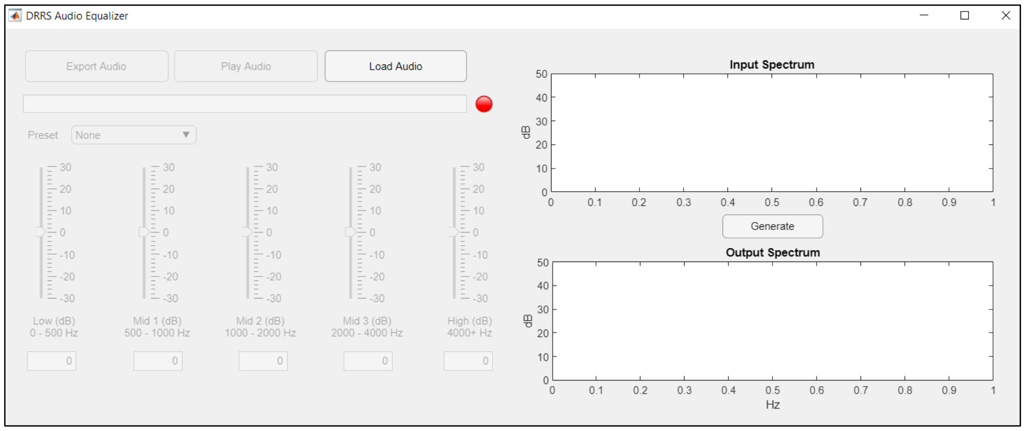


Figure A.1: The MATLAB GUI with no file loaded

The red light indicates that no file has been loaded; therefore, most functionality in the interface is locked. To enable the sliders, presets, and file manipulation features, the user can load any **two** channel **.wav** file that has a **44.1 kHz** sampling rate. Five test sample files have been provided.

When a file is loaded, the light turns yellow. The user can select a preset or move the sliders. For precision, the user can manually select the 'Custom Preset', where they will be prompted to input the gain values of each band. To reset the gain values, the user can select the 'None' preset.

The user can update the spectrum of the altered signal by clicking the 'Generate' button. The graphs follow standard MATLAB functionality, so it is possible to click on points, zoom in or out, or highlight a particular area of the graph. The equalized signal's spectrum does not update in real-time; this was done to minimize the processing power of the machine being used to run it.

To play the audio, click the 'Play Audio' button. The user can toggle the button to Play/Stop the audio. To save the equalized audio, the user can click the 'Export Audio' button.

Since the algorithm is only capable of non-real time processing, all features are locked in the program during playback, until playback ends completely or it is stopped using the 'Stop Audio' (play) button. The responsivity of the program is dictated by the processing power of the machine being used to run it.