

# A Comparison of Audible Digital Artifacts Between Linear and Minimum Phase Equalization in Digital Audio Processing

By  
Mat Vallejo  
ID#: 261148410  
ECSE 512  
McGill University  
December 6, 2024

## Abstract

This study investigates the perceptible impact of audio equalizers (EQs) designed with infinite impulse response (IIR) minimum-phase filtering compared to those designed with infinite impulse response (IIR) linear-phase filtering. The purpose of the investigation is to determine the conditions under which linear-phase FIR filters introduce audible “pre-ringing” to a processed signal in comparison to minimum-phase IIR filters. In the process, it highlights the benefits and use-cases of each design type, as well as the limitations inherent to each. The analysis includes a brief literature review of relevant foundational research followed by a section on background theory. Subsequent sections outline the mathematical underpinnings and design of two digital EQs—one minimum-phase IIR and one linear-phase FIR—designed using minimum-phase IIR techniques and the Parks-McClellan algorithm respectively. Both designs are implemented and tested in MATLAB with various audio inputs.

## I. Introduction

Audio equalization (EQ) is an active and fascinating field of research built on a foundation of insight and understanding that stems from telegraph, telephone, and communications engineering. During the early days of the telephone, high frequency information would degrade when transmitted over long distances. This drove the need for corrective signal processing on the receiving end to balance out the information loss during transmission aimed towards “equalizing” the signal—i.e. creating as close to a flat frequency response as possible at the system output. It is from these humble origins that all modern advancements in audio equalization can be traced, including those in the discrete-time and digital domains [1].

Modern advancements in equalizer design have led to an ever-increasing amount of user control over a variety of parameters that are now commonplace in both analog and digital audio EQ units. An early analog example of this is the graphic equalizer [2], featuring a set number of pre-established frequency bands with independent magnitude adjustment (boost and cut). Going one step further, George Massenburg and his collaborators introduced the “parametric” equalizer in 1972 [3], which incorporated functions to select center frequency, “Q” or peak shape, and the magnitude of the selected band. This revolutionary design extended user control to what could be considered artistic or musical levels and set a new standard for all audio EQ’s that followed.

Equalization, at its core, falls under the signal processing umbrella of filter design. In the discrete-time domain, we can distinguish two general categories of such filters: infinite impulse response (IIR) filters and finite impulse response (FIR) filters. When designing IIR filter systems for audio equalization, the approach is largely built on taking advantage of existing analog domain techniques and mapping them to the discrete-time domain. In their seminal text on discrete-time signal processing, Oppenheim and Schafer [4] discuss the prevalence of this approach, attributing it to the substantial body of work on analog filter design that would be unreasonable to neglect. The design of FIR filters, on the other hand, has no such analogous work and requires unique consideration. These considerations have resulted in the development of techniques such as windowing for FIR filter design, as well as an iterative algorithmic approach for filter optimization known as the Parks-McClellan algorithm.

This study will examine and compare the use of discrete-time minimum-phase IIR filters (filters with all poles and zeroes inside the unit circle of the z-domain) against their linear-phase

FIR counterparts in audio equalization. It will focus specifically on considerations for how and why the latter introduces a sonic effect known colloquially as “pre-ringing” in response to highly transient inputs.

## II. Background

To effectively implement a system for digital audio equalization, it is important to understand the relationship between continuous-time, discrete-time, and digital signals, as well as the methodology for filter design inherent to each. Such terminology is not consistent across the literature, necessitating some preliminary clarification for the purposes of this study, for which a basic understanding of signals and systems is assumed.

The class of *continuous-time* signals refers to signals for which the independent variable time ‘ $t$ ’ can be described at any point in time. Such signals are typically referred to as *analog* signals. *Discrete-time* signals can be classified as signals whose independent variable consists of discrete values often represented as a sequence of values ‘ $n$ ’. Both classes of signals can register dependent variable output (amplitude) as either continuous or discrete with no breakdown in the terminology. Signals such as audio can be represented by either of these methods, and these representations are exactly equivalent under certain conditions (input must be bandlimited and sampling frequency must be at least twice the Nyquist frequency, or highest frequency that the system needs to represent). *Digital* signals, on the other hand, have the unique quality of requiring both independent and dependent variables to be discrete due to the inherent way in which computers process binary information [4].

For this study, we focus our attention on discrete-time signals of the form

$$x[n] = x_a(nT), -\infty < n < \infty$$

where  $x_a(t)$  denotes an analog signal and  $x[n]$  is its discrete-time representation sampled at some periodic interval  $T$  where the  $n^{th}$  number in the sequence is equal to the value of the analog symbol at time  $nT$ . This provides the most convenient representation for the mathematics of the filter design, though the signals will ultimately be digitized (amplitude quantization) in the practical implementation and experiments.

Building from this brief conversation of signal classifications, it is logical to next establish a framework of understanding for filters and their design. For our purposes, we can consider a *filter* to be a specific type of linear time-invariant (LTI) system that allows for the

manipulation of the frequency content that passes through it, more specifically a system that can *pass* certain frequencies while *blocking* others. Figure 1 (taken from [4]) shows the magnitude response  $|H(e^{j\omega})|$  of a typical lowpass filter where

$$H(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h[k]e^{-j\omega k}$$

$\delta_{p1}$  and  $\delta_{p2}$  denote the tolerance for the passband amplitude bounds,  $\delta_s$  denotes the tolerance for the stopband amplitude, and  $\omega_p$  and  $\omega_s$  denote the passband and stopband frequencies respectively for the frequency  $\omega$ .

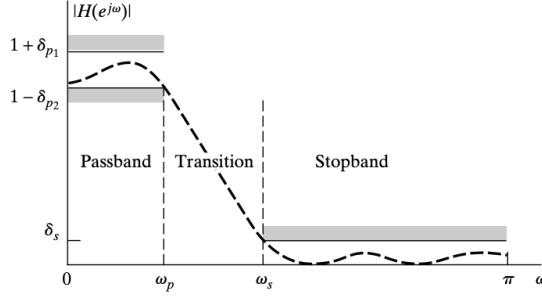


Figure 1

This representation is built from the definition of an ideal low-pass filter, who's frequency response is

$$H(\omega) = \begin{cases} 1 & |\omega| < \omega_c \\ 0 & \omega_c < \omega < \pi \end{cases}$$

and has an impulse response of the form

$$h[n] = \frac{\omega_c}{\pi} \operatorname{sinc}\left(\frac{\omega_c n}{\pi}\right)$$

Figure 1 is a helpful visualization of the (often competing) elements that need to be considered when designing a filter for which theoretically ideal circumstances are not possible. The goal of such a filter design is to find the system coefficients  $M, N, a_k, b_k$  for a rational system function

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^M b_k z^{-k}}{1 - \sum_{k=1}^N a_k z^{-k}}$$

corresponding to the system's poles and zeroes that appropriately approximate some desired frequency response  $H_d(\omega)$  where

$$H(\omega) = H(z)|_{z=e^{j\omega}}$$

and

$$H(\omega) \sim H_d(\omega).$$

Typically, decreasing the values of  $\delta_{p1}$ ,  $\delta_{p2}$ , and  $\Delta\omega = \omega_s - \omega_p$  will result in a closer fit to the desired filter response but necessitate a “higher order” filter that can only be realized by increasing the values of N and M and, therefore, the computational complexity of the filter. It should be noted that practical filter design is a process of compromise, where the interaction of limitations in these different design parameters may have less than ideal effects. To avoid this, careful consideration must be given to these parameters and implementation strategy to minimize wayward results. Other elements that affect the computational complexity of digital filter design (e.g., the form in which the filter is realized) must also be considered for practical implementation, though they are largely outside of the scope of this study.

At this stage, recall from the introduction that discrete-time filters can be broken down into two categories: IIR and FIR. The parameters in Figure 1 show us which filter elements need to be taken into consideration for designing an effective filter of either type.

Discrete-time IIR filter design relies heavily on transforming well-known continuous-time filter design methods into a discrete-time form that satisfies a given set of design parameters. An example of this can be seen in the impulse invariance method, which involves representing a continuous-time impulse response as a sampled version in the discrete-time domain [4]. Another use-case is the bilinear transform, which utilizes algebraic manipulation to map analog content in the  $s$ -domain to a discrete-time representation in the  $z$ -domain. This method results in frequency warping as the cutoff frequency approaches the Nyquist frequency, though methods have been proposed to reduce these effects [5]. The practical use of IIR filters has certain benefits such as convenient closed-form representations for many continuous-time IIR design methods which simplify their discrete-time realization, and the fact that many design considerations can be achieved using IIR filters of a relatively low filter order minimizing computational complexity. The latter of these benefits ensures a fast computation time, making these types of IIR filters ideal for audio environments where latency is undesirable (e.g., real time or live performance applications).

By contrast, discrete-time FIR filters are a special case of the rational system function  $H(z)$  where  $A(z) = 1$ . The underlying goal of the FIR filter design is not necessarily any different from its IIR counterpart, though it should be noted that FIR filters themselves differ in two major

ways. First, FIR filters are inherently stable by design. Second, they are typically only utilized when a linear phase shift is desired due to the higher filter order required to implement them accurately. However, this linear phase shift can be highly desirable in audio applications like mastering, where preserving the phase relationships between frequency bands is of critical importance and real time processing (latency) is not an issue.

The conditions for a general linear phase (GLP) system  $H(\omega)$  can be described as

$$H(\omega) = A(\omega)e^{-j(\alpha\omega - \beta)}$$

where  $A(\omega)$ ,  $\alpha$ , and  $\beta$  are real values. An important property of a real FIR filter  $H(z)$  is that its classification as a GLP system is dependent on its impulse response satisfying the following symmetry requirement where

$$h[k] = \varepsilon h[M - k] \quad k = 0, 1, \dots, M$$

and  $\varepsilon = \pm 1$ . This results in four categories of GLP FIR filters depending on the sign of  $\varepsilon$  and whether the filter order  $M$  is even or odd.

### III. Description of Algorithms

This study aims to develop two parametric audio peak equalizers (sharp increase/decrease in gain around a controllable center frequency) using two different classes of filters: one minimum-phase IIR and one linear-phase FIR.

#### a.) Minimum-Phase IIR Filter

The minimum-phase IIR filter for the first equalizer is implemented as a bi-quadratic or “biquad” parametric filter, which derives its name from its transfer function in the  $z$ -domain taking the form of a ratio of two quadratic functions. The term parametric is used to describe the included functionality of bandwidth adjustment “Q” and target frequency or “frequency sweep” of the equalizer (described in the analog domain in [3]). The transfer function of the biquad filter is shown as

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} + a_2 z^{-2}}$$

where it can be seen as a second-order filter. Second-order filters are commonly used due to the sensitive nature of higher-order IIR filters to quantization noise, which causes them to become unstable. To avoid this when increased filter order is desired or necessary to meet design

requirements, higher-order filters can be broken down and implemented as cascaded series of biquad filters.

For this filter, emphasis is placed on the design requirements that result in a minimum-phase filter. A given transfer function  $H(z)$  has poles and zeros corresponding to the coefficients  $a_k$  and  $b_k$  respectively. For such a filter to be considered minimum-phase, it must satisfy the requirement that all its poles and zeros lie within the unit circle in the  $z$ -domain, implying causality and stability for both  $H(z)$  and its inverse  $H_I(z) = 1/H(z)$ . The case of poles and zeros lying directly on the unit circle is not directly considered for this study, but generally speaking poles on the unit circle are avoided, and zeros on the unit circle, though not strictly minimum-phase, imply behavior similar to that of minimum-phase designs.

The parametric component of the filter design stems from a controllable quality factor or “Q” is defined here as

$$Q = \frac{f_0}{bw}$$

where  $f_0$  = center frequency in Hz, which is transformed into normalized angular frequency by

$$\omega_c = 2\pi \frac{f_0}{f_s}$$

and bw refers to the bandwidth of the filter. The calculation of optimal filter coefficients can be seen in [5] and [6] and are implemented in this design based on the architecture provided in [7]. They are as follows

$$b_0 = 1 + \frac{VK}{Q} + K^2, b_1 = 2(K^2 - 1), b_2 = 1 - \left(\frac{VK}{Q}\right) + K^2$$

$$a_0 = 1 + \frac{K}{Q} + K^2, a_1 = 2(K^2 - 1), a_2 = 1 - \frac{K}{Q} + K^2$$

where V is linear gain defined as  $V = 10^{\frac{G_{dB}}{20}}$ . The numerator and denominator coefficients are all normalized by  $\frac{b_k}{a_0}$  and  $\frac{a_k}{a_0}$  respectively for  $k = 0, 1, 2$ .

In audio equalization, minimum-phase systems like this biquad filter have the advantage of fast implementation time due to their low filter order and minimal computational complexity. This is useful in a number of scenarios where low-latency playback is desired. The drawbacks of minimum-phase IIR filters in audio equalization are few, but it is important to note that this design does cause phase-distortion near the passband. However, in practice, this distortion does

not sound audibly unnatural, unless exaggerated to an extreme level. In fact, many audio aficionados describe these phase non-linearities as audibly pleasing.

### b.) Linear-Phase FIR Filter

FIR filters present a different set of design challenges. To begin with, FIR filters are inherently non-recursive, meaning they introduce no feedback into the system. These filters can also easily be designed to have perfectly linear phase, which is advantageous in certain scenarios such as audio mastering where any frequency warping through EQ could be considered undesirable. The transfer function for the FIR filter can be represented as

$$H(z) = \sum_{n=-\infty}^{\infty} h_n z^{-n} = \sum_{n=0}^M b_n z^{-n}$$

where M represents the filter order.

One of the main challenges of implementing FIR filters in audio is that they require a much higher filter order to achieve similar results to an IIR design with relatively low filter order, resulting in significant latency in processing. Though the phase relationship is perfectly maintained during this processing, the computational complexity becomes problematic in scenarios where near-zero latency is desired (communications, live music). It is not, however, particularly problematic in situations like audio post-production where a playback delay is entirely tolerable.

There are numerous methods for designing an FIR filter to meet certain design specifications, namely the widow technique, frequency sampling, and Chebyshev-type approximations [4]. For this study, the implementation uses the powerful optimization method for FIR filter design originally introduced in [8]. Now known as the Parks-McClellan method, this optimization has become commonplace in FIR filter design. Effectively, this method seeks to minimize the maximum error between a Chebyshev polynomial and the desired filter frequency response. The error function can be defined as

$$E(\omega) = W(\omega)|H(\omega) - H_d(\omega)|$$

where  $W(\omega)$  is a weight function,  $H(\omega)$  is the filter to be designed, and  $H_d(\omega)$  is the ideal desired filter

$$H_d(\omega) = \begin{cases} 1 & |\omega| < \omega_c \\ 0 & \omega_c < \omega < \pi \end{cases}$$

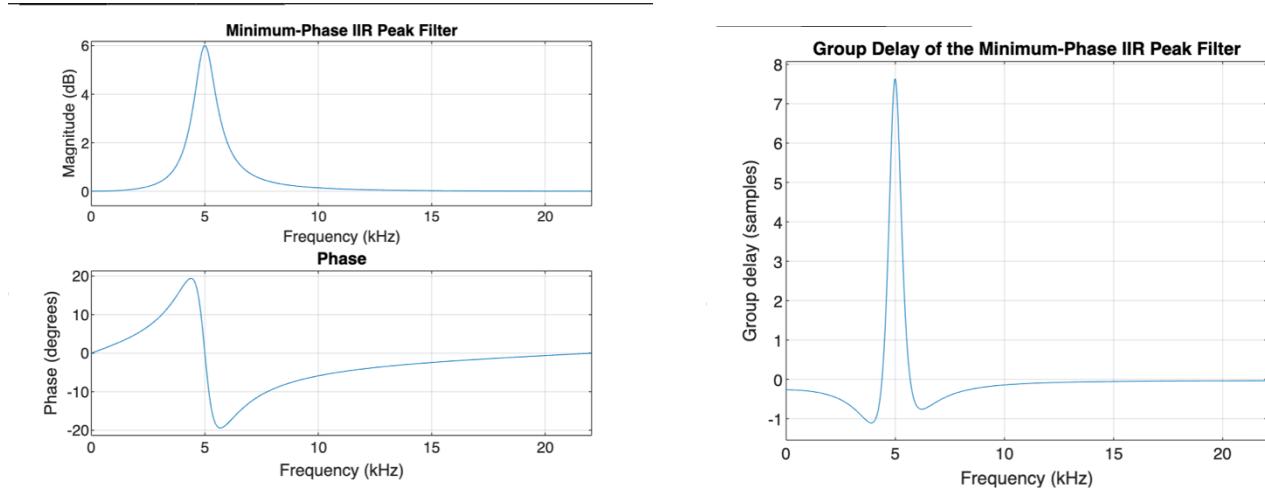
The algorithm then minimizes error over the coefficients by efficiently solving

$$\min (\max |E(\omega)|).$$

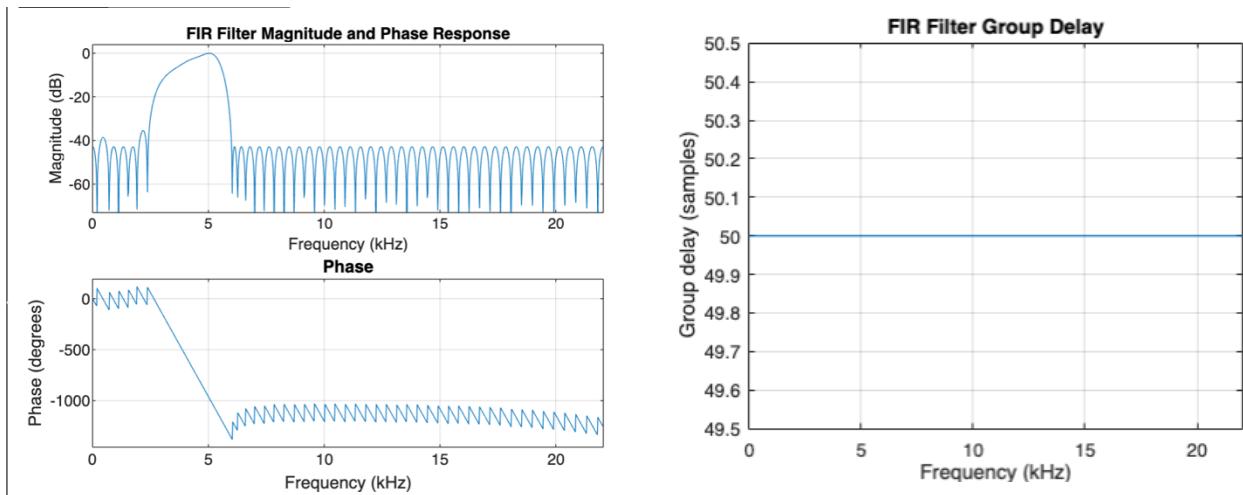
For the purposes of this study, the Parks-McClellan algorithm is implemented with the built-in MATLAB function **firpm**. Specific details on the design can be found in [8]. Using this method, a linear phase FIR filter is constructed of the order M=100, resulting in an effective filter with linear phase response.

#### IV. Results and Discussion

Both the minimum-phase IIR and linear-phase FIR filters were designed and implemented in MATLAB. The design consists of a 6dB boost at the target frequency of 5kHz with a relatively narrow bandwidth. The frequency and group delay for both filters are shown in Figure 1.



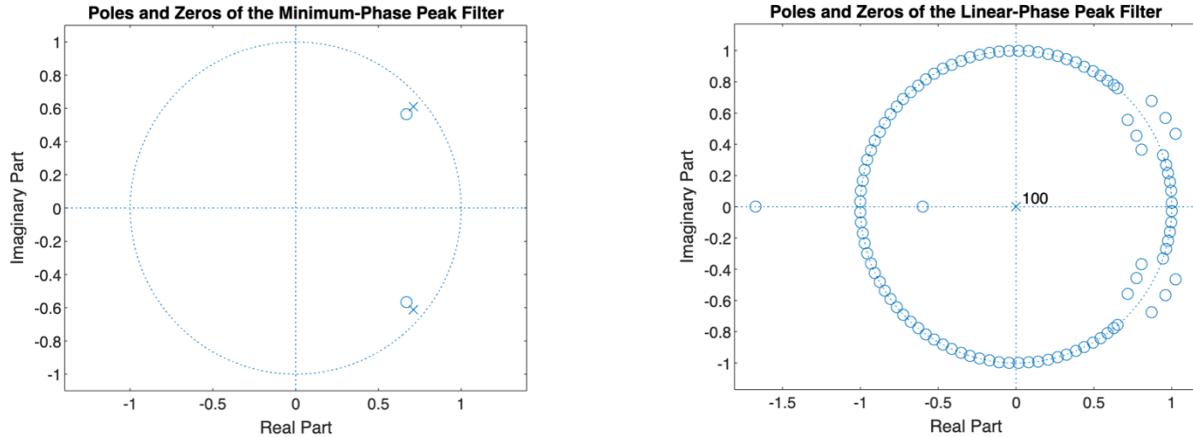
**Figure 1.** Frequency response and group delay of IIR minimum-phase peak filter



**Figure 2.** Frequency response and group delay of FIR linear-phase filter

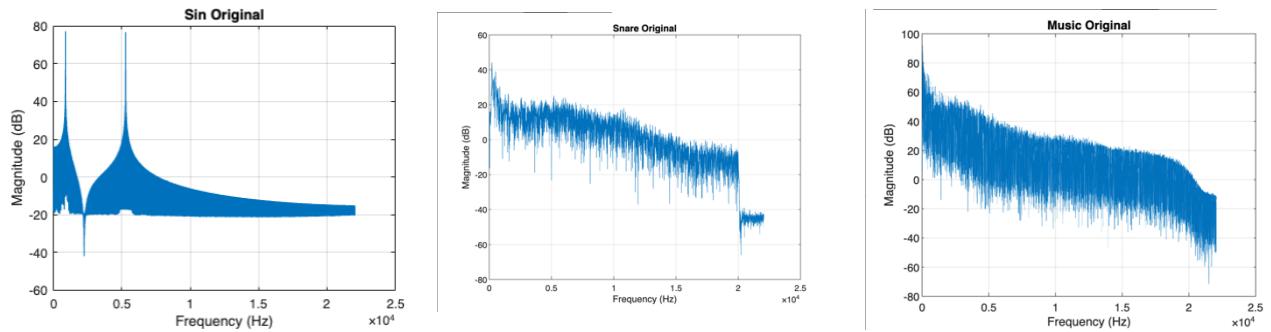
It can be seen clearly by inspection that the minimum-phase IIR filter has a non-linear phase response at the target frequency, as well as frequency warping shown by the spike in group delay. The FIR filter, on the other hand, has a completely linear phase response across the target band with a constant group delay of 50 samples across all frequencies. There are inherent differences in the respective designs due to their algorithmic implementation, namely that the IIR peak filter has no attenuation around the center frequency, i.e. it boosts at 5kHz without removing frequency content outside of its range of effect. The FIR filter by contrast has sharp attenuation outside of its passband, an adjustable feature of the Parks-McClellan algorithm.

Plots of each filters pole-zero diagram were also included for analysis to verify the design and shed light on each filter's intended effect (Figure 3). In the case of the IIR filter, it can be seen to have two poles and two zeros occurring in conjugate pairs. This is in line with the design of the second-order biquad transfer function. Notice that all poles and zeros lie inside of the unit circle, satisfying the requirements of the minimum-phase design. The FIR filter showcases the higher-order implementation required for the Parks-McClellan algorithm, where filter order  $M=100$ . The plot shows the passband where the zeros deviate from the unit circle, and the ripple bands where the zeros are located on the unit circle.



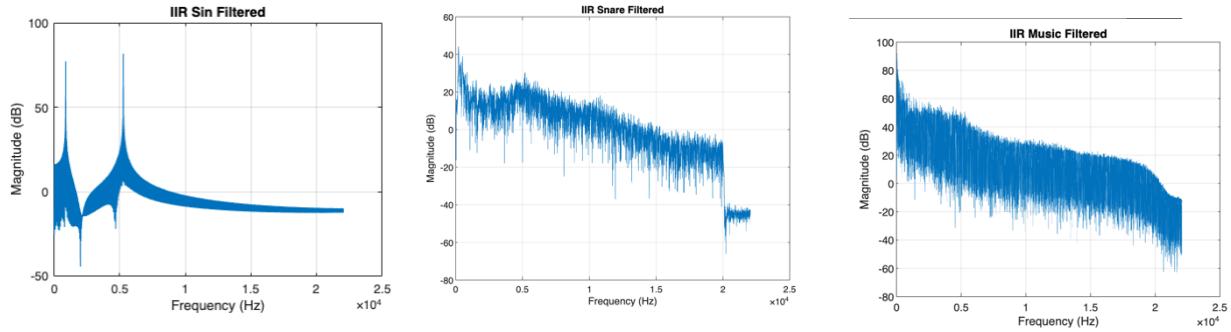
**Figure 3.** Pole-zero diagrams of the IIR minimum-phase filter and FIR linear-phase filter

To gauge the audible effects of each design, three separate audio inputs were passed through each filter. The first of these inputs is a sinusoidal input generated with a digital synthesizer comprised of a 440Hz component and a 5.1kHz component. The second is a short excerpt of musical content with full-frequency energy. The third is a sharp transient in the form of a snare hit. Plots of the original and filtered frequency responses of each input can be seen below.



**Figure 4.** Original (unfiltered) frequency response of three audio inputs: sine wave (440Hz, 5kHz), snare, and music excerpt

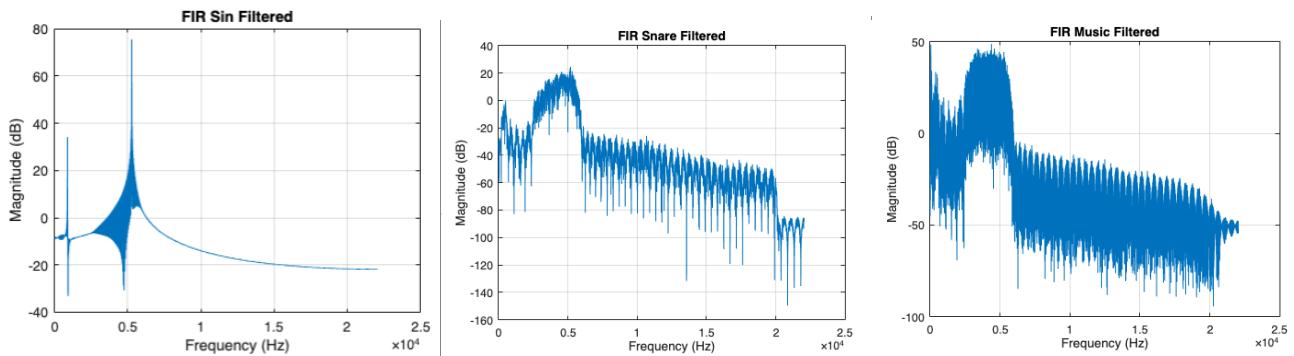
The analysis was first done using the IIR filter. Results show the expected bump in frequency content at 5kHz that was in line with reports of having a smooth, unobtrusive auditory effect on the signal content, and can be seen in Figure 5.



**Figure 5.** Filtered content of audio inputs with minimum-phase IIR filter (sine wave, snare, music excerpt)

that the IIR filter has subtle, though clearly noticeable effects. The filtered audio content had audible differences from the original, but even with 6dB (perceived as a doubling of loudness) of change they were not disruptive. The program for implementing the filter executes almost instantaneously, making it clear that this type of filter would be an obvious choice for real-time audio equalization, due to the subtle nature of its effect on the audio itself as well as the low computational complexity. Simply cascading more biquads with different coefficients would allow for further control, such as implementing a stopband on either side of the passband, without introducing a significant computational load.

The analysis on the FIR filter had, expectedly, very different results. This is partially due to the stopband included in this design that minimized content outside of the selected passband range, but also the nature of the linear-phase component. The frequency response for each audio input is shown in Figure 6.



**Figure 6.** Filtered content of audio inputs with linear-phase FIR filter (sine wave, snare, music excerpt)

In investigating the effects of “pre-ringing” in the linear-phase filter, it was extremely difficult to discern. This was in part due to difficulties encountered in designing a perfect peak around 5kHz without significantly altering stopband characteristics, but also due to the fact that, upon inspection with professional tools, the pre-ringing effect really only occurs when manipulating the fundamental frequency of a given piece of audio. In other words, pre-ringing was expected and somewhat noticeable with the sine wave input that aligns almost perfectly with the filter’s target frequency, but was not noticeable with the snare hit as the snare contained full-frequency content despite being highly transient in the time-domain.

Ultimately, this investigation showcases the benefits and use-cases of two differently designed filters. Minimum-phase IIR filters are a strong choice for real-time applications due to their low computational complexity and minimal audible artifacts when deployed on audio content, whereas linear-phase FIR filters are useful for audio post-production and scenarios where maintain the phase relationship equally across all frequency bands is desired.

## Appendix A.

Minimum-phase IIR filter MATLAB code:

```
% Min-phase IIR Peak Filter
fs = 44100; % Hz

% Filter parameters
gain_db = 6;
f0 = 5000;
bw = 1000;

% Quality factor
Q = f0 / bw;

% Calculate intermediate values
wcT = 2 * pi * f0 / fs;
K = tan(wcT / 2);
V = 10^(gain_db / 20);

% Compute filter coefficients
b0 = 1 + V * K / Q + K^2;
b1 = 2 * (K^2 - 1);
b2 = 1 - V * K / Q + K^2;
a0 = 1 + K / Q + K^2;
a1 = 2 * (K^2 - 1);
a2 = 1 - K / Q + K^2;

% Normalize coefficients
B = [b0, b1, b2] / a0;
A = [a0, a1, a2] / a0;

% Plot frequency response
figure;
freqz(B, A, 1024, fs);
title('Minimum-Phase IIR Peak Filter');

figure;
grpdelay(B, A, 1024, fs);
title('Group Delay of the Minimum-Phase IIR Peak Filter');

% Pole-zero diagram
figure;
zplane(B, A);
title('Poles and Zeros of the Minimum-Phase Peak Filter');

%% Process and Analyze Audio
% Load the audio file
audioFileName = 'Sin440_5k.wav';
[audioIn, audioFs] = audioread(audioFileName);

% Ensure the sampling rates match
if audioFs ~= fs
    error('Audio file sampling rate must match filter sampling rate.');
end

% Frequency response of the original audio signal
figure;
plot_frequency_spectrum(audioIn, fs, 'Sin Original');

% Apply the IIR filter to the audio
audioOut = filter(B, A, audioIn);

% Frequency response of the filtered audio signal
figure;
plot_frequency_spectrum(audioOut, fs, 'IIR Sin Filtered');

% Play the original audio file
disp('Playing original audio');
sound(audioIn, fs);
pause(length(audioIn) / fs + 1);
```

```
% Play the filtered audio file
disp('Playing filtered audio');
sound(audioOut, fs);
```

Linear-Phase FIR filter MATLAB code:

```
%% Lin-phase FIR filter

% Filter parameters
fs = 44100;
numTaps = 101;

% FIR Equalizer Design
bands = [0 300 600 4800 4900 5000 5300 fs/2] / (fs/2);
desired = [0 0 0 0 1 0 0];
weights = [1 1 1 1];

% Build the filter
firCoeffs = firpm(numTaps - 1, bands, desired, weights);

% Plot frequency response
figure;
freqz(firCoeffs, 1, 1024, fs);
title('FIR Filter Magnitude and Phase Response');

figure;
grpdelay(firCoeffs, 1, 1024, fs);
title('FIR Filter Group Delay');

% Pole-zero diagram
figure;
zplane(firCoeffs, 1);
title('Poles and Zeros of the Linear-Phase Peak Filter');

%% Process and Analyze Audio
% Load the audio file
audioFileName = 'Sin440_5k.wav';
[audioIn, audioFs] = audioread(audioFileName);

% Ensure the sampling rates match
if audioFs ~= fs
    error('Audio file sampling rate must match FIR filter sampling rate.');
end

% 2.) Frequency response of the original audio signal
figure;
plot_frequency_spectrum(audioIn, fs, 'Sin Original');

% Apply the FIR filter to the audio
audioOut = filter(firCoeffs, 1, audioIn);

% Frequency response of the filtered audio signal
figure;
plot_frequency_spectrum(audioOut, fs, 'FIR Sin Filtered');
%%

% Play the original audio file
disp('Playing original audio');
sound(audioIn, fs);
pause(length(audioIn) / fs + 1);

% Play the filtered audio file
disp('Playing filtered audio');
sound(audioOut, fs);
```

## Bibliography

- [1] V. Välimäki and J. Reiss, "All About Audio Equalization: Solutions and Frontiers," *Applied Sciences*, vol. 6, no. 5, p. 129, May 2016, doi: [10.3390/app6050129](https://doi.org/10.3390/app6050129).
- [2] Bohn, Dennis A.; Operator Adjustable Equalizers: An Overview [PDF]; Rane Corporation, Everett, WA; Paper 6-025; 1988 Available: <https://aes2.org/publications/elibrary-page/?id=4628>
- [3] G. Massenburg, "Parametric Equalization," *J. Audio Eng. Soc.*, vol. 20, no. 5, pp. 14-19, 1972.
- [4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2010.
- [5] S. J. Orfanidis, "Digital Parametric Equalizer Design with Prescribed Nyquist-Frequency Gain".
- [6] R. Bristow-Johnson, "The Equivalence of Various Methods of Computing Biquad Coefficients for Audio Parametric Equalizers".
- [7] "Peaking Equalizers," DSPRelated, [Online]. Available: [https://www.dsprelated.com/freebooks/filters/Peaking\\_Equalizers.html](https://www.dsprelated.com/freebooks/filters/Peaking_Equalizers.html). [Accessed: Dec. 2, 2024].
- [8] T. Parks and J. McClellan, "Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase," *IEEE Trans. Circuit Theory*, vol. 19, no. 2, pp. 189–194, 1972, doi: [10.1109/TCT.1972.1083419](https://doi.org/10.1109/TCT.1972.1083419).
- [9] C. R. Guarino, "Audio equalization using digital signal processing," in *Proceedings of the 63rd Convention of the Audio Engineering Society*, Los Angeles, CA, USA, May 15–18, 1979.
- [10] R. Berkovitz, "Digital equalization of audio signals," in *Proceedings of the Audio Engineering Society 1st International Conference on Digital Audio*, Rye, NY, USA, June 3–6, 1982.
- [11] C. H. Slump *et al.*, "Design And Implementation Of A Linear-phase Equalizer In Digital Audio Signal Processing," in *Workshop on VLSI Signal Processing*, Los Angeles, CA: IEEE, 1992, pp. 297–306. doi: [10.1109/VLSISP.1992.641062](https://doi.org/10.1109/VLSISP.1992.641062).
- [12] V. Bruschi, V. Välimäki, J. Liski, and S. Cecchi, "Linear-Phase Octave Graphic Equalizer," *J. Audio Eng. Soc.*, vol. 70, no. 6, pp. 435–445, Jul. 2022, doi: [10.17743/jaes.2022.0014](https://doi.org/10.17743/jaes.2022.0014).
- [13] M. A. Clements and J. W. Pease, "On causal linear phase IIR digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 4, pp. 479–484, Apr. 1989, doi: [10.1109/29.17528](https://doi.org/10.1109/29.17528).
- [14] C. R. Guarino, "Audio Equalization Using Digital Signal Processing," in *Proc. 63rd Audio Engineering Society (AES) Convention*, Los Angeles, CA, USA, May 1979.
- [15] S. Cecchi, L. Palestini, E. Moretti, and F. Piazza, "A New Approach to Digital Audio Equalization," in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA: IEEE, Oct. 2007, pp. 62–65. doi: [10.1109/ASPAA.2007.4393011](https://doi.org/10.1109/ASPAA.2007.4393011).