Optimization of Feedback Delay Networks for Room Acoustics Reproduction

Alexandre St-Onge

Music Technology Area
Schulich School of Music
McGill University
Montreal, Canada

July 2026

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts.

2026/07/14

Abstract

This thesis explores the optimization of feedback delay networks for reducing coloration and matching the characteristics of a target room impulse response in real-time applications. The proposed methods are designed to be applied directly on a real-time implementation without the need for implementing a differentiable feedback delay network. The sfFDN library, written in C++, provides a modular architecture supporting various FDN topologies and features, including filter feedback matrices, time-varying delays, velvet-noise, and graphic equalizer-based attenuation filters. Performance measurements demonstrate that the implementation is suitable for real-time applications and compares favorably with existing implementations. For parameter optimization, an approach using gradient-based optimizers with finite difference gradient approximation is developed. Colorless reverberation is achieved by optimizing the feedback matrix and input/output gains using a combined spectral flatness and temporal sparsity loss function. Room impulse response matching is accomplished by optimizing the attenuation and tone correction filter parameters using a mel-scale energy decay relief loss. Results show that the proposed optimization achieves comparable quality to state-of-the-art differentiable approaches while being significantly faster. A companion graphical application, the FDN Sandbox, is also presented, enabling real-time experimentation with FDN configurations and visualization of relevant acoustic metrics.

Résumé

Cette thèse explore l'optimisation de réverbérateur artificiel (FDN) pour réduire la coloration et faire correspondre les caractéristiques d'une réponse impulsionnelle cible dans des applications en temps réel. Les méthodes proposées sont conçues pour être appliquées directement sur une implémentation en temps réel sans nécessiter l'implémentation d'un FDN différentiable. La librairie sfFDN, écrite en C++, offre une architecture modulaire supportant diverses topologies de FDN et fonctionnalités, y compris des matrices de mixage, des délais variables dans le temps, des filtres "velvet-noise" et des filtres d'atténuation fondée sur des égaliseurs graphiques. Des mesures de performance démontrent que l'implémentation est adaptée pour les applications en temps réel et se compare favorablement avec des implémentations existantes. Pour l'optimisation des paramètres, une approche utilisant des optimisateurs fondée sur le gradient avec une approximation du gradient par différence finie est développée. La réverbération sans coloration est obtenue en optimisant la matrice de mixage et les gains d'entrée/sortie en utilisant une fonction de perte combinée de planéité spectrale et de dispersion temporelle. La correspondance avec une réponse impulsionnelle est accomplie en optimisant les paramètres d'atténuation et de correction tonale à l'aide d'une fonction de perte basée sur la décroissance énergétique en échelle Mel. Les résultats montrent que l'optimisation proposée atteint une qualité comparable aux approches différentielles à la pointe de la technologie tout en étant significativement plus rapide. Une application graphique associée, FDN Sandbox, est également présentée, permettant l'expérimentation en temps réel avec diverses configurations de réverbérateurs, ainsi que la visualisation de métriques acoustiques pertinentes.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Gary Scavone, for his support and invaluable guidance throughout the completion of this thesis. I am also grateful to professors Philippe Depalle and Marcelo Wanderley, whose courses made my time at McGill truly enjoyable and inspiring. I would also like to thank my friends and colleagues in the Music Technology Department, and especially my fellow CAML colleagues, for their support during these past two years. Special thanks to Andrew Allen for introducing me to the wonderful world of DSP and music technology, without whom I probably would never have had the courage to leave my career to pursue this Master's degree. Thanks to my parents and family for their support and encouragment during my time back in Montréal. Finally, I would like to thank my partner, Molly, for her love and support during this time.

Contents

List of Tables

List of Acronyms

FDN feedback delay network . . . 1

RIR room impulse response . . . 3

IR impulse response . . . 28

GA genetic algorithm . . . 26

GEQ graphic equalizer . . . 18

MFCC Mel-frequency cepstral coefficients . . . 27

EDT_{10} early decay time . . . 27

EDC energy decay curve . . . 4

EDR energy decay relief . . . 4

SPSA simultaneous perturbation stochastic approximation . . . 27

DDSP differentiable digital signal processing . . . 1

FDNTB feedback delay network toolbox . . . 30

IIR infinite impulse response . . . 33

FIR finite impulse response . . . 33

Chapter 1

Introduction

1.1 Motivation

Artificial reverberation has a rich history dating back to the 1960s, beginning with the work of Manfred Schroeder (Schroeder & Logan, 1961). Since then, many methods have been proposed to simulate acoustic spaces, each with its own trade-offs. One of the most popular approaches is the feedback delay network (FDN), favored for its high flexibility and relatively low computational cost.

However, the large number of parameters that must be chosen for an FDN can make it difficult to achieve perceptually convincing results. Poor parameter choices quickly lead to unpleasant metallic artifacts or unnatural decay profiles in the resulting sound. This tuning task becomes even more challenging when targeting specific reverberation characteristics, such as replicating the acoustic characteristics of a real-world room. In recent years, differentiable digital signal processing (DDSP) techniques have emerged as a powerful approach to tackle this problem. They have proven highly effective for reducing unwanted coloration and automatically matching synthetic reverberators to measured impulse responses.

Despite these advances, bridging the gap between research and practical use remains a challenge. Implementing a differentiable FDN for optimization purposes often results in code that is tightly coupled with machine learning frameworks and too computationally heavy for real-time audio appli-

cations like video games, virtual reality, or live music production. The main motivation of this thesis is to investigate whether advanced optimization techniques can be successfully applied directly to a highly efficient, non-differentiable FDN implementation. By bringing these tuning workflows to real-time audio environments, this work aims to demonstrate that high-quality, automatically tuned artificial reverberation can be achieved without sacrificing runtime performance.

1.2 Structure of the Thesis

This thesis is structured as follows: Chapter 2 will review some of the background concepts that are relevant to this work, including room impulse responses and a brief history of artificial reverberation methods. Chapter 3 will introduce the FDN reverberator structure and its properties, as well as some of the prior methods that have been proposed to tune its parameters. Chapter 4 will describe the main work performed for this thesis, including the presentation of a real-time-capable C++ implementation of the FDN structure and the framework used to optimize its parameters. Finally, Chapter 5 will summarize the main contributions of this thesis and suggest some directions for future work.

Chapter 2

Background

2.1 Room Impulse Responses

This section introduces the concept of room impulse responses (RIRs) and some of their properties that will be of importance when evaluating different artificial reverberation methods.

2.1.1 Stages

RIRs are often described as having three stages: direct sound, early reflections, and late reverberation. The direct sound refers to the sound wave that travels directly from the source to the receiver, unimpeded. Early reflections are the first few echoes that arrive at the receiver after the direct sound as the sound waves bounce off nearby surfaces. Finally, late reverberation refers to the later part of the RIR where individual echoes are no longer distinguishable, and the response resembles exponentially decaying Gaussian noise (Välimäki et al., 2012). Figure 2.1 shows an example of an RIR with the different stages indicated.

2.1.2 Reverberation Time

Reverberation time (RT_{60}) is the time it takes for sound to decay by 60 dB (Kuttruff & Vorländer, 2024). It is a commonly used metric to characterize the reverberation of a room and is usually

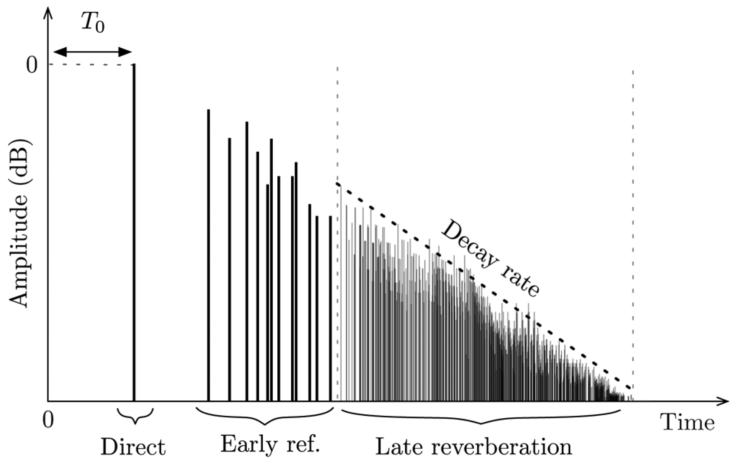


Figure 2.1 Example of an RIR with the direct sound, early reflections, and late reverberation stages indicated. Reprinted from Välimäki et al. (2012).

measured by analyzing the energy decay curve (EDC). The EDC is a measure introduced by Schroeder (1966), that is obtained by reverse integrating the squared impulse response.

$$EDC(t) = \sum_{\tau=t}^{\infty} h^2(\tau), \tag{2.1}$$

where $h(t)$ is the impulse response. The resulting curve is then usually converted to a dB scale and the RT_{60} can be estimated through simple linear regression. Figure 2.2 shows an example of an RIR and its corresponding EDC.

While this gives us a general measure of the reverberation time, it does not provide information about how the reverberation time varies with frequency. To obtain the frequency-dependent reverberation time, the RIR can be processed through an octave, or third-octave, band filter bank and the EDC can be calculated for each band separately.

Jot (1992) proposed a time-frequency representation of the EDC, called the energy decay relief (EDR). The EDR is obtained by first applying a short-time Fourier transform (STFT) to the RIR and then reverse integrating the squared magnitude of the resulting spectrogram along the time

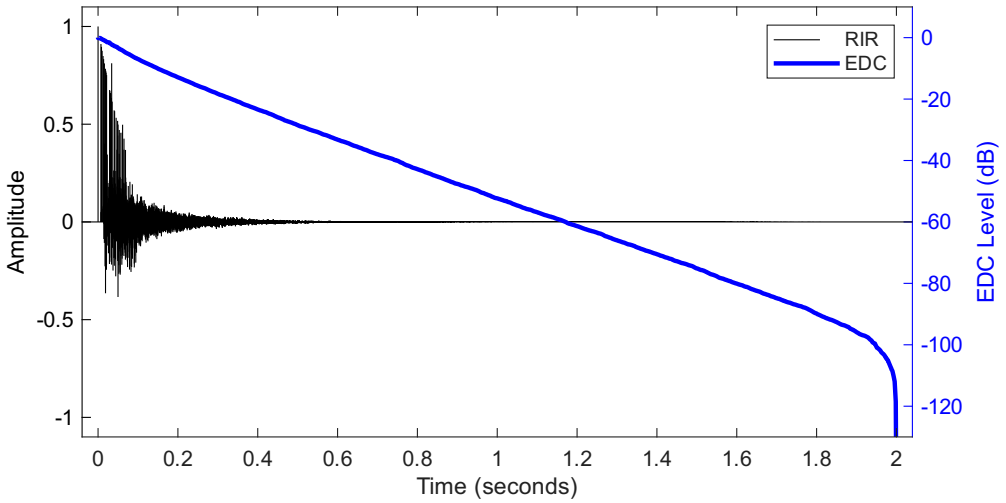


Figure 2.2 Example of an RIR and its corresponding EDC.

axis:

$$EDR(t, f) = \sum_{\tau=t}^{\infty} |H(\tau, f)|^2, \tag{2.2}$$

where $H(\tau, f)$ is the Short-time Fourier transform at time τ and frequency f. Figure 2.3 shows an example of an EDR obtained from the same RIR as in Fig. 2.2.

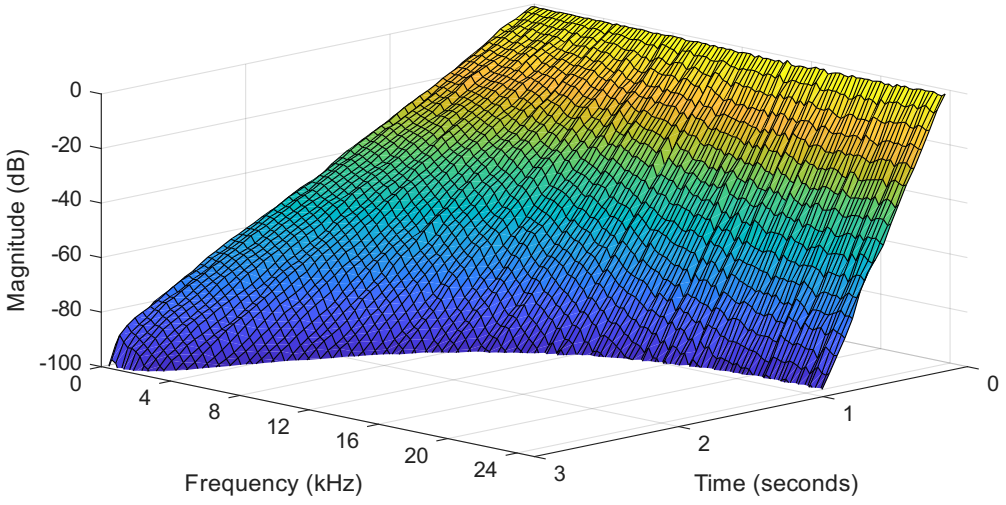


Figure 2.3 Example of an RIR and its corresponding energy decay relief.

2.1.3 Echo Density

Echo density is a measure of the number of echoes, or reflections, that arrive at the receiver over time. According to Schroeder (1962), around 1000 echoes per second are required for reverberation to be perceived as natural, but more recent work places this number even higher, at 4000 echoes per second (Rubak & Johansen, 2003). This can be challenging to achieve with artificial reverberation methods, and too low an echo density can lead to unwanted "temporal coloration" (Rubak, 2005) (i.e., flutter echoes, "whooshing" sounds, etc.). The echo density grows over time until it reaches the late reverberation stage, at which point the individual echoes are no longer distinguishable. This time t_{mix} is referred to as the mixing time.

A common method to measure the echo density of an RIR was proposed by Abel and Huang (2006) and is based on counting the number of samples in the RIR that are outside the standard deviation of a chosen window. This so-called "echo density profile" is given by the equation:

$$\eta(t) = \frac{1/\mathrm{erfc}(1/\sqrt{2})}{2\delta + 1} \sum_{\tau=t-\delta}^{t+\delta} w(\tau)\mathbf{1}\left\{|h(\tau)| > \sigma\right\}, \tag{2.3}$$

with

$$\sigma = \left[\sum_{\tau=t-\delta}^{t+\delta} w(\tau)h^2(\tau)\right]^{1/2}, \tag{2.4}$$

where $2\delta + 1$ is the window length, $w(\tau)$ is the window function, $h(t)$ is the impulse response, and $\mathbf{1}\{\cdot\}$ is the indicator function which is equal to 1 if the condition is true and 0 otherwise. The mixing time is defined as the time at which the echo density profile $\eta(t) \geq 1$. Figure 2.4 shows an example of an echo density profile and mixing time obtained with Equation (2.3).

2.1.4 Modal and Frequency Density

Modal density is a measure of the average number of modes per Hertz (Jot & Chaigne, 1991). In musical acoustics, a mode is a natural vibration pattern, or resonance, present in a system, like a vibrating string or a room. While in real RIRs the modal density increases with frequency, artificial

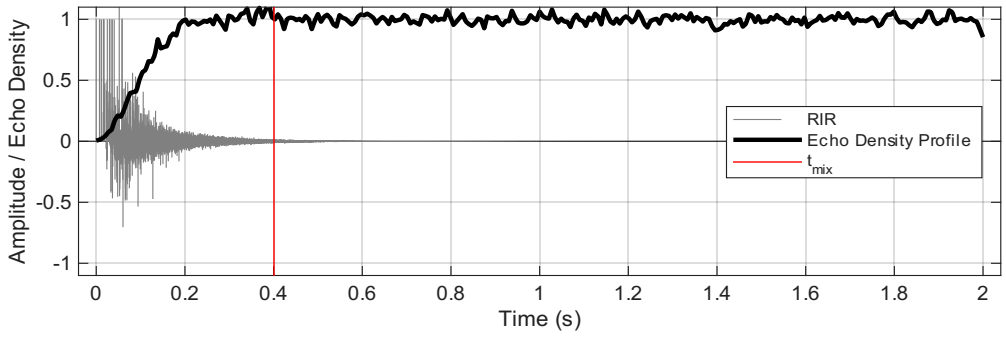


Figure 2.4 Example of an RIR and its corresponding echo density profile.

reverberation methods based on delay lines and comb filters exhibit a constant modal density across frequencies (Jot & Chaigne, 1991; Schlecht & Habets, 2019). If the modal density is too low, the resulting reverberation can be perceived as metallic and ringing tones can be heard. In a way, modal density can be thought of as the frequency-domain equivalent of echo density. The magnitude distribution of the modes also plays an important role in the perception of reverberation. If a relatively small number of modes have a much higher magnitude than the rest, they can dominate over the others, resulting in a lower audible modal density than the theoretical one (Schlecht & Habets, 2019). Jot and Chaigne (1991) use the term "frequency density" to refer to this audible modal density.

Having introduced the properties used to characterize and evaluate reverberation, namely the reverberation time, echo density, and modal density, the next section reviews the methods that have been developed to reproduce these properties artificially.

2.2 Artificial Reverberation

This chapter introduces different methods used for artificial reverberation. Many different techniques have been developed since the 1960s to simulate the sound propagation that occurs in a real room (Schroeder & Logan, 1961). These techniques have been used in various applications, ranging from music production, where the artistic qualities of the reverberation are more important, to architectural acoustics, where the most accurate simulation of sound propagation is desired.

Reverberation techniques can be broadly classified into three categories: computational acoustics, convolution-based methods, and delay networks (Välimäki et al., 2012).

2.2.1 Computational Acoustics

Reverberation methods based on computational acoustics aim to simulate the sound propagation in a real room using methods based on physical equations, mainly the wave equation. These methods can be further classified into wave-based methods and geometric methods (Savioja & Svensson, 2015; Siltanen et al., 2010).

Wave-based methods

Wave-based methods are techniques that directly solve the wave equation to simulate sound propagation. While it is possible to solve the equation analytically for a simple shoebox room, more complex room geometries require the use of numerical methods that operate either in the time domain or the frequency domain (Välimäki et al., 2012). The lossless wave equation is given by

$$c^2 \Delta p = \frac{\partial^2 p}{\partial t^2}, \tag{2.5}$$

where p is the acoustic pressure, c is the speed of sound, and Δ is the Laplacian operator (Kuttruff & Vorländer, 2024). This equation can be solved using various methods, including digital waveguides (Smith, 1985), finite difference methods (Savioja et al., 1994), as well as boundary element and finite element methods (Siltanen et al., 2010). The advantage of such methods is that they can accurately model wave phenomena such as diffraction and interference, as long as the space is modeled with sufficient detail. However, the high computational cost of these methods makes them impractical for real-time applications; they are instead often used to generate impulse responses offline that can then be auralized through convolution.

Geometric methods

Geometric methods provide another way to simulate sound propagation by assuming that sound behaves like rays that travel in straight lines. This approximation means that wave phenomena such as diffraction and scattering are not modeled automatically, as in wave-based methods, and the discrepancy becomes more apparent at lower frequencies where the wavelength of sound becomes comparable to the size of the modeled space (Savioja & Svensson, 2015). Geometric methods are generally implemented by finding specular reflection paths between a sound source and a receiver. Techniques like the image source method (Allen & Berkley, 1979) are able to find all the possible paths in a given geometry up to a certain reflection order, while other methods like ray tracing (Krokstad et al., 1968, 1983) use a stochastic approach to find a subset of the possible paths. The image source method is especially well suited for finding the early reflection paths, but higher-order reflections can quickly become expensive to calculate as their number increases exponentially with reflection order. On the other hand, ray tracing methods usually limit themselves to a constant number of rays, allowing them to simulate late reflections more efficiently (Siltanen et al., 2010).

2.2.2 Convolution

Perhaps the most straightforward way to implement artificial reverberation is to convolve an audio signal with an impulse response measured from a real room or generated using computational acoustics methods. When implemented naively, convolution can be computationally expensive for real-time applications, especially since room impulse responses can be several seconds long. For an impulse response of length N, convolution in the time domain using an FIR filter would require N multiplications and N additions per output sample. For this reason, convolution reverberation is usually implemented in the frequency domain using the fast Fourier transform (FFT), as it significantly reduces the computational cost. Furthermore, to avoid the inherent latency of FFT-based convolution, partitioned convolution methods have been developed (Gardner, 1995; Wefers, 2015). One downside of convolution using a measured RIR is that the resulting reverberation

is fixed to the specific source/receiver position pair captured in the measurement, and changing any of the reverb characteristics in real time is much more difficult than with other reverberation methods. Wefers (2015) provide a good overview of different methods of filter exchanges, enabling time-varying implementations of FIR filters with partitioned convolution.

2.2.3 Delay Networks

The oldest method of artificial reverberation is based on the use of delay lines. Schroeder and Logan (1961) are often credited with being the first to introduce the idea of artificial reverberation in a digital system. The first reverberator they proposed consisted of a single delay line. Without feedback, the system produces a single echo and has a completely flat frequency response. By adding feedback and a corresponding feedback gain g, it is possible to obtain multiple echoes. As the length of the delay line gets shorter, the frequency response of this comb filter becomes more pronounced and can be heard as a metallic sound that they refer to as "coloration."

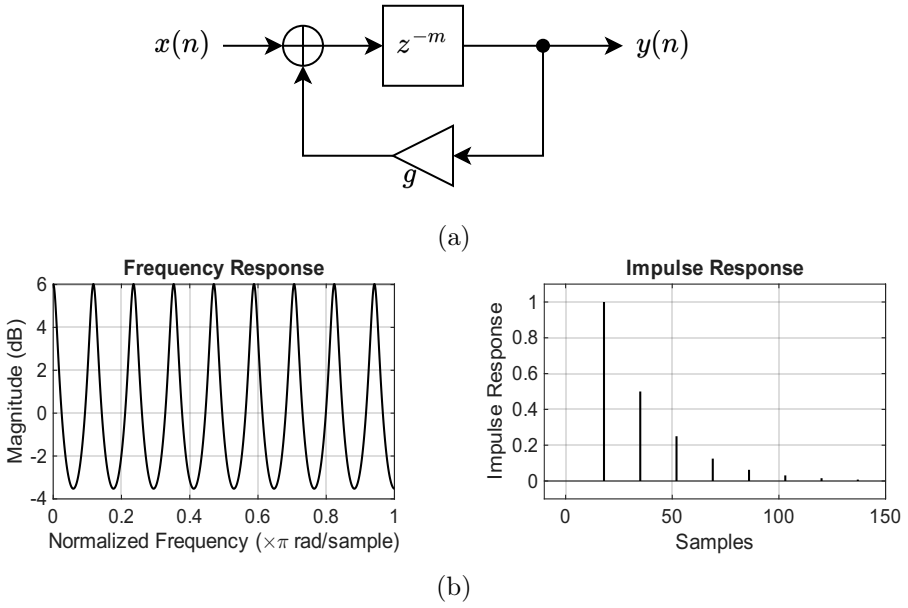


Figure 2.5 (a) Comb filter structure. (b) Frequency and impulse response of a comb filter with delay $m = 17$ and feedback gain $g = 0.5$.

To mitigate this issue, the authors found that by choosing the right mixing ratio between the

delayed and undelayed sound, the comb filter structure can be made into an allpass filter. The transfer function of this so-called "Schroeder allpass" is given by

$$H(z) = \frac{-g + z^{-M}}{1 - gz^{-M}}. \tag{2.6}$$

A single allpass filter does not produce enough echoes by itself to be perceived as reverberation and instead sounds more like a flutter echo. To increase the echo density, multiple allpass structures are chained together in series. They found that using five allpass filters gave good results, as too many filters in series would cause the resulting impulse response to build up its maximum intensity too slowly before decaying, which can be perceived as unnatural.

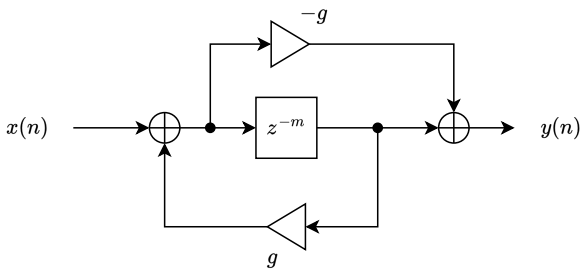


Figure 2.6 Schroeder allpass filter structure.

In a subsequent paper, Schroeder (1962) introduced another configuration of the reverberator, where a parallel bank of four feedback comb filters is followed by two allpass filters in series.

By using enough comb filters in parallel, it is possible to obtain a higher echo density relatively quickly as the echo density of a comb filter is constant in time and only depends on the delay length. Even with four such filters in parallel, the resulting density still falls short of the density expected of a real room. In contrast, comb (or allpass) filters in series produce an echo density that starts low and increases over time. By combining parallel and series configurations, the filters in series act as a multiplier on the echo density of the parallel section, allowing the echo density to build up much more quickly.

Moorer (1979) later extended Schroeder's reverberator structure by inserting filters inside the

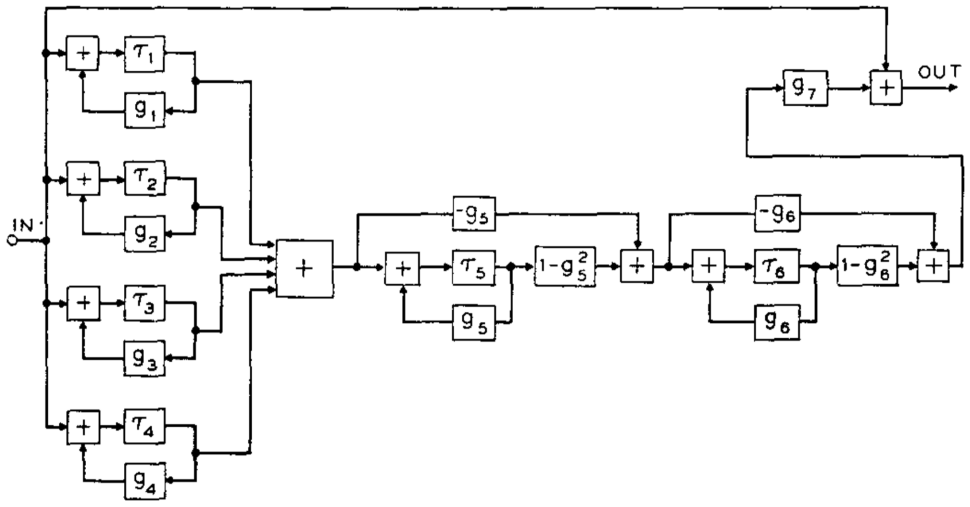


Figure 2.7 Schroeder reverberator. Four feedback comb filters in parallel are followed by two allpass filters in series. Reprinted from (Schroeder, 1962).

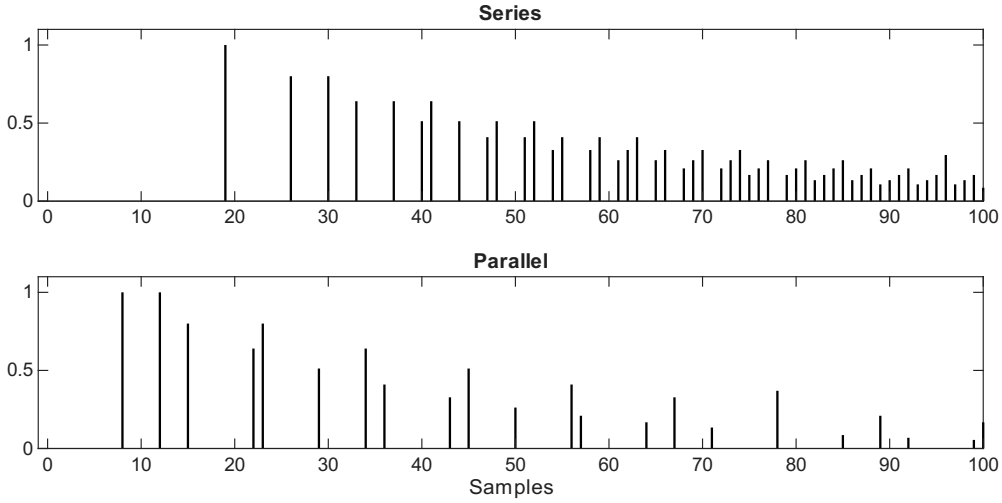


Figure 2.8 Impulse response of two comb filters with delays $m_1 = 7$ and $m_2 = 11$ and feedback gain $g = 0.8$. The two filters are placed in series (top) and in parallel (bottom).

loop of the comb filters to better simulate frequency-dependent absorption. One-pole lowpass filters were used for their low computational cost, but only a crude approximation of the frequency-dependent decay present in real rooms can be achieved with this structure.

By combining allpass, comb, and one-pole filters, it is possible to create a number of different reverberator structures. An example of such a structure can be found in Dattorro (1997), as shown in Fig. 2.9. "Freeverb" is another popular reverberator implementation with a structure composed of eight parallel lowpass-filtered comb filters followed by four Schroeder allpass filters in series[1].

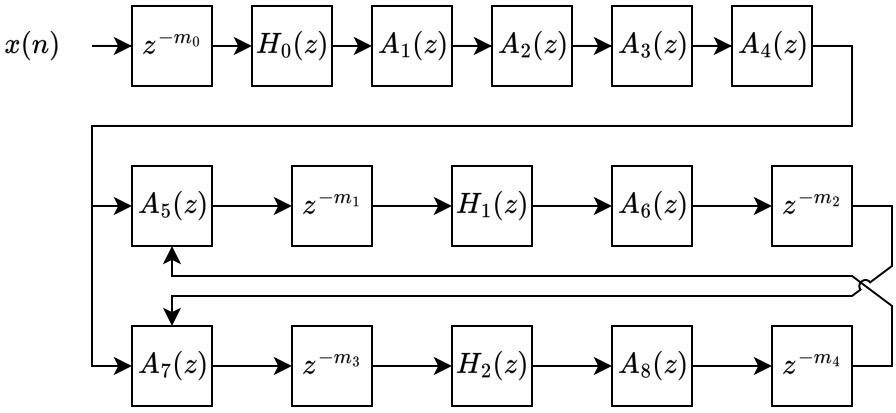


Figure 2.9 Dattorro reverberator. z^{-m} are delay lines, $A_i(z)$ are Schroeder allpass filters, and $H_i(z)$ are one-pole filters. The output samples are taken from multiple points in the structure and mixed together to produce the final output.

This chapter introduced three broad families of artificial reverberation: computational acoustics methods, which physically model sound propagation at a high computational cost; convolution, which reproduces a measured RIR exactly but offers little flexibility and no parametric control; and delay networks, which trade physical accuracy for computational efficiency and parametric control. Among the delay-network methods, the comb and allpass structures discussed above remain difficult to control, as their coloration and echo density depend on a large set of interacting parameters. The following chapter focuses on the FDN, a generalization of these structures in which the delay lines are coupled through a feedback matrix, providing a more flexible and controllable framework for artificial reverberation.

[1] https://ccrma.stanford.edu/~jos/pasp/Freeverb.html

Chapter 3

Feedback Delay Networks

This chapter introduces FDNs, which are a type of reverberator model that builds on the ideas introduced in Section 2.2.3 but with the addition of a feedback matrix that allows complex feedback loops to be created between the delay lines.

3.1 Background

In the January 1972 issue of *Studio Sound* magazine, Gerzon (1972) presented a reverberation unit where two or more delay lines are interconnected through the use of an orthogonal feedback matrix. While the results were promising, especially considering the limitations at the time (the delay lines needed to be implemented with tape delays), Gerzon noted the presence of unwanted coloration in the output similar to what could be heard in other comb-filter-based reverberators. The original diagram of the system is shown in Fig. 3.1.

Stautner and Puckette (1982) also proposed a similar structure that they referred to as a "recursive delay network." Their design was meant to simulate the way sound bounces across a room by using a network of four delay lines, each assigned to a physical speaker placed around the listener. Their delay network is based on a comb filter structure as shown in Fig. 2.5, with the difference equation

$$y(t) = x(t - m) + gy(t - m), \tag{3.1}$$

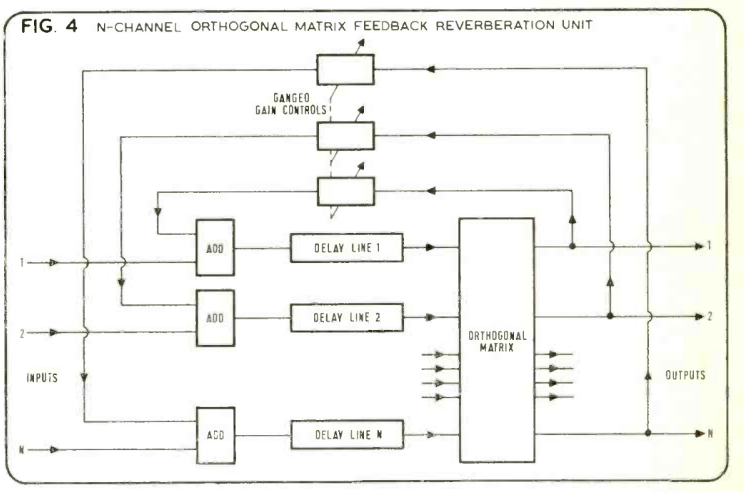


Figure 3.1 Block diagram of the reverberation unit proposed by Gerzon. Reprinted from (Gerzon, 1972).

where $y(t)$ is the output signal at time t, $x(t)$ is the input signal, m is the delay length, and g is the feedback gain. Expressed in the z-domain, the system equation is given by

$$Y(z) = z^{-m}(X(z) + gY(z)). \tag{3.2}$$

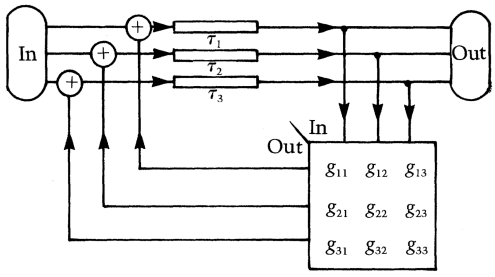
The delay element of the comb filter is then replaced by N delays in parallel and the feedback gain g is replaced by a feedback gain matrix A, where each element a_{ij} represents the gain from the output of delay line j to the input of delay line i. The system equation of this new structure is then given by

$$Y(z) = D_m(z)\big(X(z) + AY(z)\big), \tag{3.3}$$

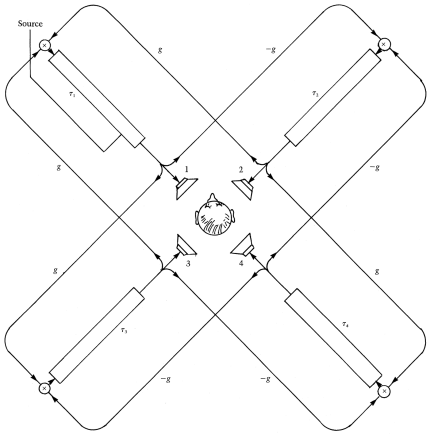
$$D_m(z) = \begin{bmatrix} z^{-m_1} & 0 & \cdots & 0 \\ 0 & z^{-m_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & z^{-m_n} \end{bmatrix}, \tag{3.4}$$

where Y and X are now vectors of length N containing the output and input, respectively, and D_m is a diagonal matrix containing the delay elements z^{-m_i}, where m_i is the delay length of the i-th

delay line. The block diagram of the system is shown in Fig. 3.2.



(a) Block diagram of the recursive delay network.



(b) 4-channel representation of the delay network with physical speaker placement.

Figure 3.2 Reprinted from (Stautner & Puckette, 1982).

Jot and Chaigne (1991) later helped formalize the feedback delay network structure into its current form by adding a vector b of input gains and a vector c of output gains to the system. The network is then defined by the system of equations

$$y(t) = \sum_{i=1}^{N} c_i q_i(t) + dx(t), \tag{3.5}$$

$$q_j(t + m_j) = \sum_{i=1}^{N} a_{ij} q_i(t) + b_j x(t), \tag{3.6}$$

$$\tag{3.7}$$

where $q_i(t)$ is the output of the i-th delay line at time t, $y(t)$ is the output of the system, $x(t)$ is the input signal, a_{ij} is a matrix coefficient, b_j is the input gain for the j-th channel, c_i is the output gain for the i-th channel, and d is the direct gain term. Using the z-transform, the system becomes

$$Y(z) = c^{\mathsf{T}} q(z) + dX(z), \tag{3.8}$$

$$q(z) = D(z)\big(Aq(z) + bX(z)\big), \tag{3.9}$$

By removing $q(z)$, the transfer function of the system can be expressed as

$$\frac{Y(z)}{X(z)} = H(z) = \mathbf{c}^\intercal \left[\mathbf{D_m}(z)^{-1} - \mathbf{A} \right]^{-1} \mathbf{b} + d. \tag{3.10}$$

3.2 Attenuation Filter Design

The FDNs presented so far are assumed to be lossless and stable, but it is desirable to be able to control the decay time of the reverberation. This is achieved by adding attenuation filters inside the feedback loop. Each filter gain is dependent on the delay length of the corresponding delay line and the desired decay time. If a constant decay time across all frequencies is desired, a simple gain g_i can be applied to each delay line, with

$$g_i = 10^{\frac{-3m_i}{T_{60} f_s}}, \tag{3.11}$$

where m_i is the delay length of the i-th delay line in samples, T_{60} is the desired decay time in seconds, and f_s is the sampling frequency (Jot & Chaigne, 1991). This kind of attenuation is often implemented by including the gain directly in the feedback matrix A by multiplying the matrix by a diagonal matrix $G = \mathrm{diag}(g_1, g_2, \ldots, g_n)$ and using the new feedback matrix AG instead. The block diagram of the system is shown in Fig. 3.3.

To better simulate the frequency-dependent decay of real rooms, more sophisticated filtering strategies should be used. Jot and Chaigne (1991) proposed a design method using a first-order lowpass filter to achieve a specific decay time at frequency zero t_0 and at the Nyquist frequency t_{ny}. The filter coefficients are given by

$$\tau_0 = 10^{\frac{-3m_i}{t_0 f_s}}, \quad \tau_{ny} = 10^{\frac{-3m_i}{t_{ny} f_s}}, \quad \alpha = \frac{\tau_0}{\tau_{ny}} \tag{3.12}$$

$$a_1 = \frac{1-\alpha}{1+\alpha}, \tag{3.13}$$

$$H(z) = \tau_{ny} \frac{1 - a_1}{1 - a_1 z^{-1}}. \tag{3.14}$$

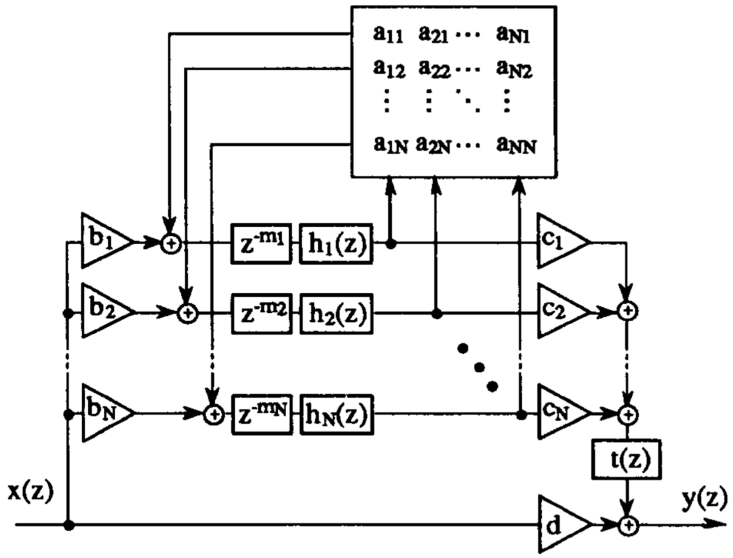


Figure 3.3 Block diagram of the feedback delay network proposed by Jot. Reprinted from (Jot, 1992).

For greater control over the decay time across frequencies, it might be desirable to use higher-order filters such as graphic equalizers (GEQs) (Jot, 2015; Välimäki et al., 2024). In their 2024 paper, Välimäki et al. (2024) propose a two-stage design method for GEQs. In the first stage, a first-order low-shelf filter is used to approximate the overall desired frequency response. For the second stage, a GEQ is used to further refine the frequency response of the attenuation filter. It was shown that this method offers more accurate control over the decay time across frequencies compared to using a graphic equalizer (GEQ) alone.

3.3 Feedback Matrices

The feedback matrix is a crucial component of the FDN and determines how energy is distributed across the delay lines. The main goal when choosing the feedback matrix is to ensure that the resulting FDN is lossless. An FDN is said to be lossless when the roots of the characteristic polynomial of the system are all located on the unit circle (Jot & Chaigne, 1991; Schlecht & Habets,

2017b). The characteristic polynomial of the system is given by

$$p_{A,m}(z) = \det\big[D(z^{-1}) - A\big], \tag{3.15}$$

which suggests that the losslessness of the system is dependent on both the feedback matrix A and the delay lengths m. Unitary matrices are a common choice, as Jot and Chaigne (1991) showed that if the feedback matrix is unitary, the resulting FDN is guaranteed to be lossless regardless of the choice of delay lengths. A matrix A is said to be unitary if it satisfies the equation $AA^H = I$, where A^H is the conjugate transpose of A and I is the identity matrix. If the matrix is real-valued, the condition can be simplified to $AA^T = I$ and the matrix is said to be orthogonal. Below are some common choices of feedback matrices that have been used in the literature.

Hadamard Matrix

The Hadamard matrix is constructed recursively as follows:

$$H_1 = [1], \quad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{2^k} = \begin{bmatrix} H_{2^{k-1}} & H_{2^{k-1}} \\ H_{2^{k-1}} & -H_{2^{k-1}} \end{bmatrix}. \tag{3.16}$$

The Hadamard matrix only exists for size 1, 2 and multiples of 4. Every coefficient of the matrix is either 1 or -1, ensuring maximum energy diffusion between the delay lines. For the matrix to be orthogonal, the coefficients need to be normalized by $\frac{1}{\sqrt{N}}$, where N is the size of the matrix. This matrix becomes especially interesting when the size is a power of two, as the matrix-vector multiplication can then be implemented efficiently using the fast Hadamard transform in $O(n \log n)$ time instead of $O(n^2)$ time for a general matrix-vector multiplication.

Householder Matrix

The Householder matrix is defined as

$$H = I - 2v_n v_n^T, \tag{3.17}$$

where v_n is a unit vector of length N. When $v_n = \frac{1}{\sqrt{N}}[1, 1, \ldots, 1]^T$, the resulting Householder matrix is of the form:

$$H_N = \begin{bmatrix} \frac{N-2}{N} & -\frac{2}{N} & \cdots & -\frac{2}{N} \\ -\frac{2}{N} & \frac{N-2}{N} & \cdots & -\frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{2}{N} & -\frac{2}{N} & \cdots & \frac{N-2}{N} \end{bmatrix},$$

which allows efficient implementation of the matrix multiplication with only $2N$ operations. The downside of this matrix is that as the matrix size increases, the coefficients along the diagonal approach 1 while the off-diagonal coefficients approach 0, causing the system to behave more like a group of parallel comb filters.

Random Orthogonal Matrix

A random orthogonal matrix can be generated by first creating a random matrix A' with entries drawn from a normal distribution and then applying QR decomposition to it. The QR decomposition is the decomposition of a matrix into an orthogonal matrix Q and an upper triangular matrix R. The orthogonal matrix can then be constructed with the equation:

$$A = Q\,\text{sign}(R_{diag}), \tag{3.18}$$

where Q is the orthogonal matrix obtained from the QR decomposition, R_{diag} is the diagonal matrix formed from the diagonal entries of R, and $sign(\cdot)$ is the sign function (Stewart, 1980). While these matrices do not benefit from any special structure that allows for efficient implementation, they

have been shown to produce good results in terms of the perceptual quality of the reverberation (Heldmann & Schlecht, 2021). Furthermore, it is possible to tune the matrix A' to generate a random orthogonal matrix with specific properties, which will be useful for the optimization of the feedback matrix described in Section 4.2.2.

Circulant Matrix

Rocchesso and Smith (1997) proposed the use of circulant matrices inside FDNs. Circulant matrices are of the form

$$A = \begin{bmatrix} a_0 & a_1 & \cdots & a_{N-1} \\ a_{N-1} & a_0 & \cdots & a_{N-2} \\ \vdots & \vdots & \ddots & \vdots \\ a_1 & \cdots & a_{N-1} & a_0 \end{bmatrix}. \tag{3.19}$$

This type of matrix is of interest for two reasons. First, the matrix-vector multiplication can be implemented in $O(n \log n)$ time using the FFT. Second, the matrix A can be constructed from a single vector $a = [a_0, a_1, \ldots, a_{N-1}]$ of length N using the FFT. As will be discussed later in Section 4.2.2, this can allow for a more efficient optimization of the feedback matrix, as only N parameters need to be optimized instead of the N^2 parameters needed for a random orthogonal matrix.

Filter Feedback Matrix

More recently, Schlecht and Habets (2020) introduced a new kind of feedback matrix where the coefficients of the matrix are replaced by FIR filters. These matrices have been shown to greatly increase the echo density of the resulting FDN and can be implemented efficiently by cascading multiple feedback matrices separated by banks of delay lines.

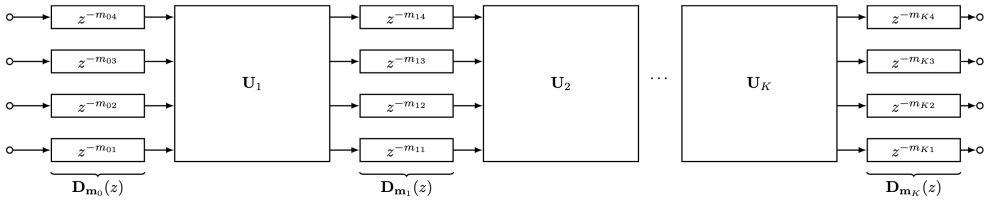


Figure 3.4 Block diagram of a feedback delay network with filter feedback matrix. Each block U_i is a feedback matrix. Reprinted from (Schlecht & Habets, 2020).

3.4 Delay Lengths

The choice of delay lengths is also an important factor in the design of an FDN, as it has an impact on the echo density, the mixing time, and the modal distribution of the resulting reverberation. As pointed out by Schroeder and Logan (1961), the delay lengths should be chosen to be mutually prime to avoid the superposition of echoes in the impulse response as well as to avoid the peaks of the comb filter responses lining up, which can cause unwanted coloration. Rocchesso and Smith (1997) showed that the order of the FDN, i.e., the number of modes, is equal to the sum of the delay lengths:

$$\mathfrak{N} = \sum_{i=1}^{n} m_i, \tag{3.20}$$

which suggests that longer delay lengths will lead to a higher modal density. However, longer delay lengths also lead to a slower echo density buildup, so a balance needs to be found to achieve good reverberation quality. Schlecht and Habets (2017a) present certain patterns to avoid when choosing delay lengths:

Clustering Clustering happens when the delay lengths are close to each other or are integer multiples of each other. Such a configuration will lead to a sparse impulse response with distinct clusters of echoes, resulting in a fluctuating echo density.

Common Prime Factor If all delays share a common prime factor q, then the system can only ever generate echoes at times that are integer multiples of q, preventing the echo density from ever reaching the expected maximum.

Low-Order Dependencies Low-order dependencies happen when the delay lengths can be expressed as linear combinations of each other with small integer coefficients. From (Schlecht & Habets, 2017a), an example of such a pattern would be the delay lengths $m = [49, 51, 100]$. With this configuration, every echo coming out of the third delay line ($m_3 = 100$) will be superimposed with echoes coming out of the first two delay lines. This results in an echo density profile that is equal to the echo density of an FDN with only the first two delay lines.

A good starting point for choosing delay lengths is to use mutually prime numbers uniformly distributed between a minimum and maximum delay length chosen based on the desired reverberation time and mixing time (Prawda et al., 2020; Schlecht & Habets, 2017a).

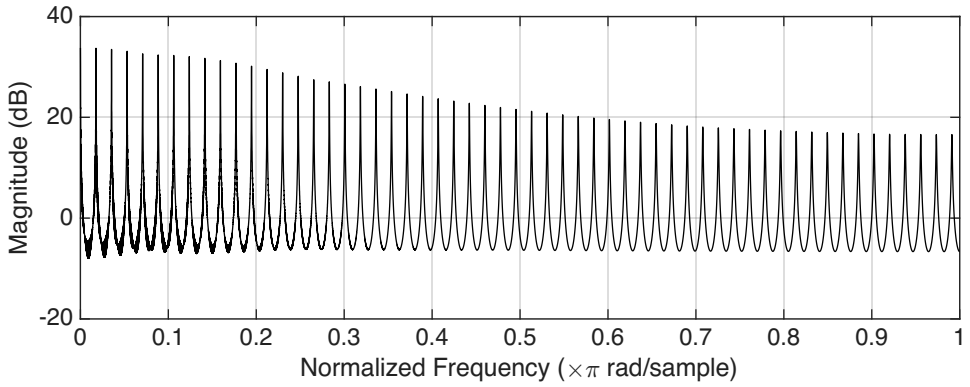
3.5 Tone Correction Filter

While the introduction of a frequency-dependent attenuation filter is necessary to control the reverberation time of the reverberator, it also has the effect of modifying the envelope of the frequency response. Jot (1992) and Jot and Chaigne (1991) proposed the addition of a "tone correction" filter $t(z)$, placed outside the feedback loop, allowing independent control of the frequency response and decay characteristics of the system. Figure 3.5 shows the effect of the tone correction filter on the frequency response of a comb filter with lowpass attenuation.

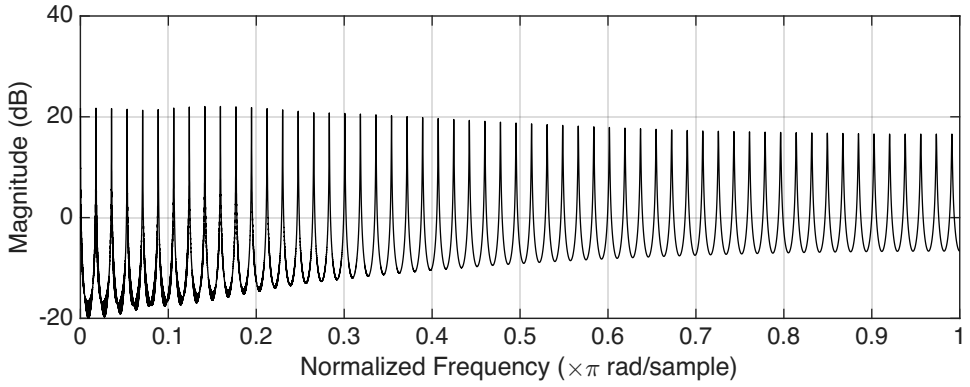
3.6 Modal Distribution

Schlecht and Habets (2019) presented a numerical method to compute the modal decomposition of the FDN transfer function using the Ehrlich-Aberth Iteration method. Using the transfer function defined in Equation (3.10), and the system order $\mathfrak{N}$ defined in Equation (3.20), the modal decomposition of the FDN can be expressed as

$$H(z) = d + \sum_{i=1}^{\mathfrak{N}} \frac{\rho_i}{1 - \lambda_i z^{-1}}, \tag{3.21}$$



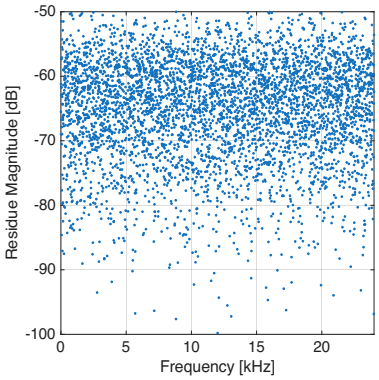
(a)



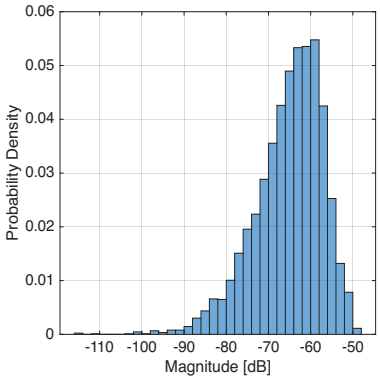
(b)

Figure 3.5 Frequency response of a comb filter + lowpass without tone correction (top). Frequency response of a comb filter + lowpass with tone correction (bottom). The tone correction filter is a first order IIR filter designed according to the method proposed in (Jot & Chaigne, 1991).

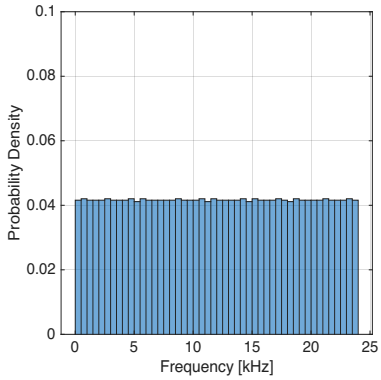
where λ_i are the poles of the system and ρ_i are the corresponding residues. An implementation of this method is available in MATLAB through the FDN Toolbox (Schlecht, 2020). This modal decomposition allows for statistical analysis of the modal distribution of the system, which can be used to analyze the coloration of an FDN, as will be discussed in Section 4.2.2. Figure 3.6 shows the modal decomposition and excitation distribution of an FDN of size 8 with $\mathfrak{N} = 8944$. The mode frequencies histogram shows how the modes of an FDN are equally distributed across the frequency spectrum.



(a) Modal decomposition of an FDN.



(b) Modal excitation distribution of an FDN.



(c) Modal frequency distribution of an FDN.

Figure 3.6 The histogram of the modal excitation distributions in (b) is derived from the residue magnitudes of the modal decomposition shown in (a). The histogram (c) shows an almost equally spaced distribution of mode frequencies.

Heldmann and Schlecht (2021) conducted a listening test to evaluate the correlation between the modal excitation distribution and the perceived coloration. They found that the Rayleigh distribution could be used to model the modal excitation distribution of FDNs. The Rayleigh distribution is given by

$$f(x;\sigma;\mu) = \frac{x-\mu}{\sigma^2} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \quad x \geq 0, \tag{3.22}$$

where σ is the scaling factor that determines how narrow or wide the distribution is, and μ is the gain shift factor. A correlation was found between the scaling factor σ and the perceived coloration, where narrower distributions (smaller σ) were perceived as less colored than wider distributions (larger σ).

The preceding sections showed that designing an FDN involves choosing a large number of interdependent parameters, including the feedback matrix, the delay lengths, the attenuation and tone correction filters, and the input and output gains. Each of these parameters affects the decay, coloration, and modal behavior of the output, and choosing them by hand is difficult. This motivates the automatic optimization methods reviewed in the following section.

3.7 Automatic Optimization

While FDNs are a powerful and flexible method to generate artificial reverberation, the high number of parameters that need to be chosen (i.e., delay lengths, feedback matrix, filter gains, input and output gains, etc.) can make it difficult to achieve good results. For some scenarios, it might be desirable to match the characteristics of a specific RIR, introducing more constraints on the choice of FDN parameters. This chapter will review some of the methods that have been proposed to automatically choose the parameters of an FDN.

3.7.1 Gradient-Free Optimization

The first attempts at automatically optimizing the parameters of an FDN were done through the use of genetic algorithms (GAs). GAs are a class of optimization algorithms inspired by the process of natural selection where a population of N candidate solutions evolves over a number of generations. Each candidate solution is evaluated through a fitness function, and the best candidates are selected to produce offspring for the next generation using different flavors of crossover and mutation operations (Mitchell, 1996).

Chemistruck et al. (2012) used a GA to generate the matrix coefficients of a 4-channel FDN with a fitness function based on the mean squared error between the power envelope of the generated RIR and a target RIR. Through listening tests, they found that their optimized FDNs struggled to convincingly match the target RIRs. Coggin and Pirkle (2016) used a similar fitness function to optimize the delay lengths m_i, input gains b_i and output gains c_i of a 16-channel FDN. The Hadamard matrix was used as the feedback matrix and an FIR filter was added at the input of

the FDN to simulate early reflections. Test subjects were asked to rate the difference between the generated and target RIRs through an ABC/HR test. The results showed that the optimized FDNs still struggled to accurately match the target RIRs, especially for RIRs of larger rooms.

More recently, Ibnyahya and Reiss (2022) proposed a method to optimize the matrix coefficients, input and output gains, direct gain, and delay lengths of a 16-channel FDN using a GA with a fitness function based on the Mel-frequency cepstral coefficients (MFCC). They combined their approach with the analysis method proposed by Jot (1992) to design the attenuation and tone correction filters of the FDN using 10-band GEQ filters. Again, an FIR filter was added at the input of the FDN to simulate early reflections. This FIR filter was designed by truncating the target RIR at the early decay time (EDT_{10})[1]. Study participants were still able to distinguish the generated RIRs from the target RIRs, but objective evaluation of the generated spectrograms, EDT_{10}, and RT_{60} showed that the MFCC-based fitness function was successful at reducing the errors between the generated and target RIRs.

Other gradient-free optimization methods, such as Bayesian optimization (Bona et al., 2022) or simultaneous perturbation stochastic approximation (SPSA) (Martínez Ramírez et al., 2021), have also been used to optimize the parameters of reverberators and other audio plugins. The advantage of these methods is that the target reverberator does not need to be differentiable, and the optimization can even be performed on reverberators for which the implementation is not publicly available (*i.e.*, black-box optimization). The main disadvantage of these methods is that they can be much slower than gradient-based optimization, especially when the number of parameters to optimize is large.

3.7.2 Differentiable Digital Signal Processing

DDSP is a family of techniques that requires the implementation of digital signal processing operations in a way that allows the use of automatic differentiation software such as PyTorch or TensorFlow (Hayes et al., 2024). The ability to compute the gradients of a loss function via backprop-

[1]The EDT_{10} is defined as the time it takes for the sound to decay by 10 dB (Kuttruff & Vorländer, 2024).

agation allows for the use of powerful gradient-based optimizers like the Adam optimizer (Kingma & Ba, 2015). The main challenge of using DDSP for FDN optimization is the need to implement a differentiable version of the FDN structure, including differentiable delay lines, filters, and feedback matrix. This implementation is usually tightly coupled with the optimization framework used, making it difficult or impossible to use the same implementation for both optimization and real-time audio processing.

Dal Santo et al. (2023) proposed a method to reduce the coloration of an FDN by optimizing the matrix coefficients and gains b_i, c_i and d. The FDN implementation (DiffFDN) was made differentiable by using frequency sampling. Two loss functions were used during optimization: a frequency-domain loss function based on spectral flatness, and a time-domain loss function based on the temporal sparsity of the generated impulse response (IR). Objective evaluation and listening tests showed that the method was able to achieve a significant reduction in coloration compared to non-optimized FDNs. This approach was later extended to match a target RIR by combining the colorless optimization with an analysis-synthesis approach to design the filters of the FDN (Dal Santo et al., 2024).

Mezza et al. (2024) also presented an optimization method based on a differentiable FDN. As opposed to the previous work, every component of this differentiable FDN is implemented in the time-domain, with the exception of the bank of delay lines which uses the FFT to implement fractional length delays. In addition to optimizing the matrix coefficients and gains, they were also able to optimize the delay lengths using differentiable delay lines implemented in the frequency-domain. The filter coefficients for the attenuation and tone correction filters were also optimized. Their loss function used the mel-scale EDR to measure the error between the generated and target RIRs. Objective evaluation showed that their method was able to accurately match the characteristics of the target RIRs, but no listening tests were performed. Table 3.1 summarizes the parameters optimized by the different methods described in this section.

As summarized in Table 3.1, the surveyed methods reflect a shift away from gradient-free GAs, which optimize only a subset of parameters on small FDNs, toward gradient-based DDSP approaches

Authors	Parameters Optimized							N	Algorithm
	A	b_i	c_i	d	m_i	$H(z)$	$T(z)$		
Chemistruck et al., 2012	✔					✔		4	GA
Coggin and Pirkle, 2016		✔	✔		✔	⚙	⚙	16	GA
Ibnyahya and Reiss, 2022	✔	✔	✔	✔	✔	⚙	⚙	16	GA
Bona et al., 2022				✔	✔	✔	✔	64	Bayesian
Dal Santo et al., 2023	✔	✔	✔	✔				*	Adam
Dal Santo et al., 2024	✔	✔	✔	✔		⚙	⚙	*	Adam
Mezza et al., 2024	✔	✔	✔	✔	✔	✔	✔	*	Adam

Table 3.1 Summary of the parameters optimized by different methods. A refers to the feedback matrix, b_i and c_i refer to the input and output gains, respectively, d refers to the direct gain, m_i refers to the delay lengths, $H(z)$ refers to the attenuation filters, and $T(z)$ refers to the tone correction filters. The number of channels for the FDN is indicated in the second to last column, and the optimization algorithm used is indicated in the last column. * indicates that the method supports an arbitrary number of channels. ✔ indicates that the parameter was optimized, while ⚙ indicates that the parameter was chosen using an analysis-synthesis approach.

that jointly optimize the feedback matrix, gains, and filters for FDNs of arbitrary size. The feedback matrix and the input, output, and direct gains are tuned by nearly all recent methods, whereas the delay lengths and the attenuation and tone correction filters are more often fixed or designed through an analysis-synthesis approach. A recurring limitation of the most accurate methods is that the differentiable implementations required by DDSP are tightly coupled to their optimization frameworks and are not designed for real-time audio processing. The remainder of this thesis addresses this gap: the next chapter introduces a real-time-capable FDN implementation together with an optimization framework built around it.

Chapter 4

Methodology

4.1 Real-time FDN Implementation

This chapter introduces the sfFDN library, a C++library designed to bridge the gap between research and real-time audio processing. sfFDN is easily extensible such that new FDN extensions can be easily integrated and is also performant enough to be used in production environments.

When it comes to FDN research, the current state-of-the-art is the feedback delay network toolbox (FDNTB), a MATLAB library developed by Schlecht (2020). While the library is extremely flexible and provides useful functions, such as the modal decomposition of an FDN, it was never designed for efficiency and its slow speed can become a bottleneck during optimization procedures where a great number of impulse responses need to be generated and analyzed. It can also be difficult to judge the quality of a particular FDN topology without being able to listen to it. For this reason, an application was also developed in parallel to sfFDN that allows users to experiment with different FDN topologies and parameters in real-time. The FDN Sandbox application provides a graphical user interface where users can listen to the generated impulse responses and look at relevant plots (e.g., impulse response, spectrogram, energy decay curve, etc.). It also supports the playback of audio files through the FDN.

4.1.1 Performant and Efficient Implementation of an FDN

The FDN topology implemented in sfFDN is based on the canonical structure shown in Fig. 4.1. This topology can be separated into seven building blocks: the input gains (green), the delay lines (yellow), the feedback matrix (orange), the loop filters (red), the output gains (blue), the tone correction filter (purple), and the direct gain (gray). At the heart of the library is the `AudioProcessor` interface. In the context of sfFDN, an audio processor is defined as a class that can take N_{in} channels of audio, apply a transformation (e.g., filtering, delay, mixing matrix), and output N_{out} channels of audio. Complex audio processing tasks can be achieved by chaining multiple audio processors together in series using the `AudioProcessorChain` class. The `FilterBank` class can similarly be used to group multiple single-channel audio processors into a bank of parallel processors for multi-channel processing. The rest of this section will go over these different building blocks and some of the `AudioProcessor` implementations that are provided in the library. The letter N will be used to refer to the number of channels of the FDN.

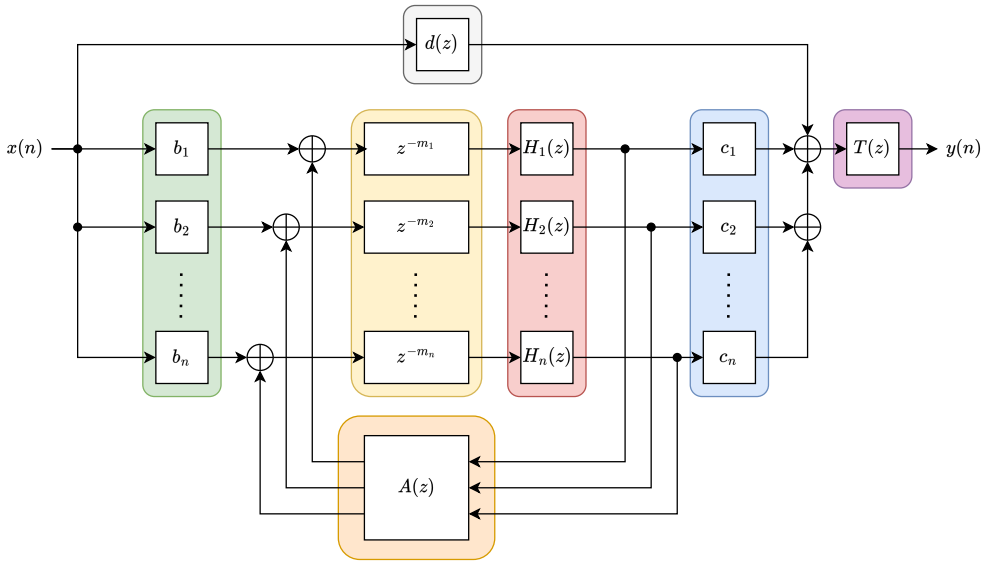


Figure 4.1 Architecture of the FDN implementation.

Input/Output Gains

The input gains block supports any processor that takes a single channel of audio as input and outputs N channels of audio. Conversely, the output gains block consists of any processor that takes N channels of audio as input and outputs a single channel of audio. The simplest and most common implementation of these blocks is a simple gain processor that applies a scalar gain (b_i, c_i) to each channel. This functionality is provided by the `ParallelGains` class which can either split a single input channel into N output channels (input gains) or sum N input channels into a single output channel (output gains). FIR filters are commonly added at the input and/or output of the FDN to simulate early reflections and increase echo density (Jot, 1997). This effect can be achieved by chaining an `FIR` processor with the `ParallelGains` processor. For longer FIR filters, the `PartitionedConvolver` processor can be used to greatly reduce the computational cost of the convolution by performing the operation in the frequency domain using a non-uniform partition convolution algorithm (Wefers, 2015). Fagerström et al. (2020) proposed a novel FDN structure where the input and output gains are replaced by velvet noise filters, resulting in an increase in echo density. This so-called velvet-noise FDN can be implemented easily in sfFDN by using the `SparseFIR` class, which provides an efficient implementation of sparse FIR filters, especially suited for velvet-noise sequences. The `FilterBank` processor can also be used to create a bank of parallel filters, allowing for each channel to have its own unique FIR or velvet-noise filter.

Delay Lines

For efficiency reasons, sfFDN restricts the main delay lengths to integer values, but fractional and time-varying delay lines can easily be integrated by including a `DelayInterp` or `DelayTimeVarying` processor inside the loop filter block. The `GetDelayLengths()` function provides a convenient way to generate delay lengths based on several heuristics:

Random Randomly generate delay lengths pulled from a uniform distribution.

Gaussian Randomly generate delay lengths pulled from a Gaussian distribution.

Prime Randomly generate delay lengths pulled from a uniform distribution. Delays are guaranteed to be prime numbers.

Uniform Delays are uniformly spaced between a minimum and maximum value.

Prime Power Delays are integer powers of prime numbers. Based on the implementation found in the Faust library[1].

Steam Audio Re-implementation of the delay length generation method used in the reverberator of the Steam Audio Library[2]. This heuristic is based on the Prime Power method but with some amount of randomization added.

Mean delay Delays are generated based on Eq. (40) from (Schlecht & Habets, 2017a). The resulting delays are logarithmically spaced to obtain a desired mean delay length and standard deviation.

Loop Filters

The loop filters block is an optional block that supports any processor that takes N channels of audio as input and outputs N channels of audio. Most commonly, a bank of N parallel filters is used to control the decay time of the FDN. The three filtering methods covered in Sec. 3.2 can be implemented in sfFDN through the `CreateAttenuationFilterBank()` function. If one RT_{60} value is provided, the resulting filter bank will implement the simple proportional gain method from Equation (3.11). If two RT_{60} values are provided, the resulting filter bank implements the method proposed by Jot and Chaigne (1991). Finally, if 10 RT_{60} values are provided, the resulting filter bank implements a 10-band GEQ using the two-stage design approach proposed by Välimäki et al. (2024). Classes for arbitrary infinite impulse response (IIR) and finite impulse response (FIR) filters are also provided for more flexibility.

[1] https://github.com/grame-cncm/faustlibraries/blob/master/delays.lib#L229
[2] https://github.com/ValveSoftware/steam-audio/blob/master/core/src/core/reverb_effect.cpp#L302

As mentioned in Sec. 4.1.1, fractional and time-varying delays can be implemented by including a `DelayInterp` or `DelayTimeVarying` processor inside the attenuation filter block. Both classes support linear and allpass interpolation methods. The linear interpolation method, while slightly more computationally efficient than allpass interpolation, introduces frequency-dependent attenuation which will limit the maximum achievable RT_{60}, especially at higher frequencies. Figure 4.2 shows the effect of linear interpolation on the frequency response of an FDN. The top spectrogram was computed using the response from an FDN with fractional delay lengths of $m_{i\prime} = m_i + 0.5$ samples, as this is where linear interpolation shows the most significant attenuation. The bottom spectrogram was computed using the response from an FDN with time-varying delay lengths. The delay lengths were modulated by a sine wave with frequency randomly chosen between 0.1 and 5 Hz and a modulation depth randomly chosen between 1.5 and 19 samples. Higher modulation depths and frequencies started to introduce audible pitch-shifting effects, which may or may not be desirable depending on the use case.

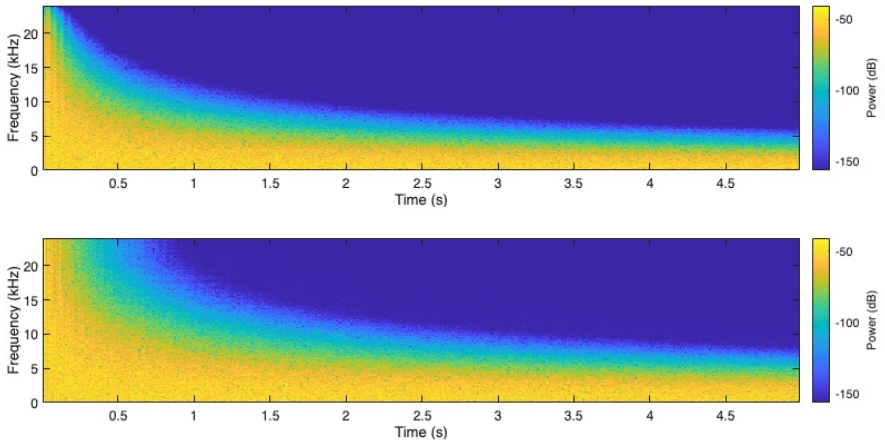


Figure 4.2 Top: Spectrogram of an FDN with fractional delay lengths of $m_{i\prime} = m_i + 0.5$ samples. Bottom: Spectrogram of an FDN with time-varying delay length. Linear interpolation was used in both cases.

Allpass filters can also be included in the loop filter block and have been shown to increase the echo density of the FDN (Väänänen et al., 1997; Välimäki et al., 2012). The

`SchroederAllpassSection` class provides an implementation of the allpass structure shown in Fig. 2.6. The *zita-rev1*[3] reverberator is an example of an FDN that includes a parallel bank of Schroeder allpass filters in series with the delay lines.

Feedback Matrix

The feedback matrix block supports any processor that takes N channels of audio as input and outputs N channels of audio. Common feedback matrices implemented in sfFDN by the `ScalarFeedbackMatrix` class include the Hadamard, Householder, random orthogonal, circulant, the allpass and nested allpass feedback matrices from (Schlecht, 2021), and the identity matrix. Arbitrary matrices are also supported by specifying the matrix coefficients directly. The `FilterFeedbackMatrix` class is also provided to implement the filter feedback matrix structure proposed by Schlecht and Habets (2020) and shown in Fig. 3.4. The matrix multiplications are performed using `Eigen`[4] for fast performance.

Tone Correction Filter

The tone correction filter block is an optional block that supports any processor that takes a single channel of audio as input and outputs a single channel of audio. This block is mainly used to apply a final filter to the output of the FDN.

Direct Gain

The direct gain block is a simple scalar gain applied to the direct sound path of the FDN. This gain can be used to control the balance between the direct sound and the reverberated sound.

4.1.2 Performance measurements

Performance measurements of the sfFDN library were taken to ensure that the implementation is suitable for real-time applications as well as to provide real-world numbers for some of the recent

[3] https://ccrma.stanford.edu/~jos/pasp/Zita_Rev1.html
[4] https://libeigen.gitlab.io/

enhancements that have been proposed in the last few years, such as the Velvet-noise FDN or the filter feedback matrix structure. Authors will sometimes provide theoretical computational costs for their proposed methods in terms of the number of floating point operations, but these numbers often fail to reflect the actual performance of the proposed method in a real implementation. Table 4.1 shows the time taken to process 128 samples of audio for different FDN configurations implemented with the sfFDN library. A block of 128 samples represents approximately 2.66 ms of audio at a sample rate of 48 kHz. To avoid dropouts in the audio stream, the FDN needs to be able to do all of its processing inside that deadline of 2.66 ms. The measurements were taken on a computer with an Intel Core i9-12900K CPU. The baseline configuration consists of an FDN with a random orthogonal feedback matrix, scalar input and output gains, and no loop or tone correction filters. The "Filter Feedback Matrix" (FFM) configuration uses a filter feedback matrix composed of four mixing stages. The "Schroeder Allpass" configuration adds a parallel bank of four Schroeder allpass filters in series to the loop filter block. The "Velvet-noise" FDN configuration implements the velvet-noise FDN structure proposed by Fagerström et al. (2020), where each input and output gain is replaced by a sparse velvet-noise filter with 15 non-zero coefficients. The "Time Varying" configuration implements time-varying delay lengths with first-order allpass interpolation in each feedback path of the FDN.

FDN Size	4	6	8	10	16	24	32
Baseline	0.351	0.535	0.651	0.876	1.529	2.648	4.167
FFM	1.405	2.170	2.718	3.567	6.302	12.616	19.792
FDN + Schroeder	2.356	3.437	4.456	5.807	9.337	13.752	19.710
Velvet FDN	1.887	2.968	3.586	4.520	7.050	11.424	16.027
Time Varying FDN	3.401	4.964	6.646	8.334	13.389	20.434	28.761

Table 4.1 Time (in µs) taken to process 128 samples of audio for different FDN configurations

Most FDNs include some kind of attenuation filter in the feedback loop in order to control the decay time of the reverberation. As these filters usually do not impact the echo or modal density of the FDN, the choice of which filter to use depends on the desired control over the RT_{60} across frequencies and the computational budget available. To better illustrate the cost of each

filter type, the performance of three different parallel banks of filters was measured and is shown in Table 4.2. The "One Pole" configuration implements the simple proportional gain method from Equation (3.11) using a first-order low-shelf filter. The "3 Band" configuration implements a simple 3-band equalizer composed of second-order low-shelf, peaking, and high-shelf IIR filters in series. Finally, the "Two Stage GEQ" configuration implements the two-stage design approach for graphic equalizers proposed by Välimäki et al. (2024) using a 10-band graphic equalizer. These results show that the most computationally expensive part of the FDN is the attenuation filters.

FDN Size	4	6	8	10	16	24	32
One Pole	0.721	1.085	1.435	1.786	2.878	4.316	5.732
3 Band	5.829	5.835	3.052	6.265	3.938	4.805	5.386
Two Stage GEQ	3.388	5.055	6.623	8.399	13.307	20.176	27.057

Table 4.2 Time (in µs) taken to process 128 samples of audio for different filter types and FDN sizes.

To put these numbers into perspective, a publicly available FDN implementation was also benchmarked. Prawda et al. (2020) developed a flexible FDN implementation using the JUCE framework[5]. The implementation supports multiple FDN sizes and incorporates a 10-band GEQ as the attenuation filter. The topology was replicated in sfFDN and Table 4.3 shows the measurements for both implementations for multiple FDN sizes.

FDN Size	4	6	8	10	16	24	32
sfFDN	3.8	5.6	7.3	9.2	15.0	22.5	31.2
Prawda et al., 2020	6.02	9.55	15.0	18.3	39.2	75.4	112.0

Table 4.3 Time (in µs) taken to process 128 samples of audio for different FDN implementations.

Finally, measurements of the reverberator included in the Steam Audio[6] library were also taken. Steam Audio is a library developed by Valve Software designed for spatial audio processing in video games and virtual reality applications. The library implements a 16-channel FDN with a

[5] https://juce.com/
[6] https://valvesoftware.github.io/steam-audio/

Hadamard feedback matrix, 3-band attenuation filters, and a 3-band tone correction filter followed by four Schroeder allpass filters in series. Again, the topology was replicated in sfFDN and the measurements for both implementations are shown in Table 4.4. These measurements show that the performance of the FDN implementation presented in this thesis is comparable to or better than the performance of other publicly available implementations, making it suitable for real-time applications.

FDN Size	16
sfFDN	6.33
Steam Audio	9.62

Table 4.4 Time (in µs) taken to process 128 samples of audio for different FDN implementations.

4.2 Optimization framework

With the real-time FDN implementation provided by the sfFDN library, it is now possible to optimize the various parameters of the FDN to achieve (1) a colorless reverberation and (2) an FDN that matches the characteristics of a target RIR. For this purpose, the *ensmallen* library (Curtin et al., 2021) will be used to perform the optimization. *ensmallen* is a C++library that provides multiple optimization algorithms, including gradient-based and gradient-free methods. Unless specified otherwise, the Adam optimizer (Kingma & Ba, 2015), as implemented in *ensmallen*, was used for the optimization of the FDN parameters in the following sections.

4.2.1 Gradient Approximation using Finite Differences

Since one of the goals of this project was to perform the optimization on a real-time and non-differentiable FDN implementation, the gradients cannot be computed using automatic differentiation and must instead be approximated. The approximation of the gradients was implemented

using the central finite difference method defined as follows:

$$\nabla^{\mathrm{FD}}\mathcal{L}(\mathcal{P})_i = \frac{\mathcal{L}(\mathcal{P} + \epsilon d_i) - \mathcal{L}(\mathcal{P} - \epsilon d_i)}{2\epsilon}, \tag{4.1}$$

where $\nabla^{\mathrm{FD}}\mathcal{L}(\mathcal{P})_i$ is the i-th component of the gradient, $\mathcal{L}$ is the loss function, $\mathcal{P}$ is the vector of parameters, d_i is a standard basis vector with 1 in the i-th position and 0 elsewhere, and ϵ is a small perturbation value. With this approach the loss function must be evaluated twice for each parameter, which can be computationally expensive for large parameter spaces. In this case, it can be beneficial to use a strategy to reduce the number of parameters to optimize, such as using the Householder or circulant matrix for the feedback matrix, which only requires N parameters instead of N^2, as mentioned in Sec. 4.2.2. Despite the need for manual computation of the gradients, the Adam optimizer was found to consistently give the best results while still being faster than most gradient-free optimizers. Details on the comparison of different optimizers can be found in Appendix B.

4.2.2 Optimization for Colorless Reverberation

Parameterization

Probably the biggest challenge when designing an FDN is the task of choosing the right parameters in order to minimize the coloration of the resulting reverberation. Heldmann and Schlecht (2021) showed that the coloration of an FDN is mostly unaffected by the choice of attenuation filters and is mostly determined by the choice of the feedback matrix A, the input and output gains b_i and c_i, and the delay lengths m_i. In this work, the optimization for minimizing coloration will focus on optimizing the parameters A, b_i and c_i of a lossless FDN. For an FDN of size N, the parameters to optimize can be represented as a vector $\mathcal{P}$ of size $N^2 + 2N$ containing the N^2 coefficients of the feedback matrix A and the $2N$ input and output gains.

$$\mathcal{P} = [a_{11}, a_{12}, \ldots, a_{NN}, b_1, b_2, \ldots, b_N, c_1, c_2, \ldots, c_N] \tag{4.2}$$

To ensure orthogonality of the feedback matrix, Equation (3.18) is used to re-parameterize the feedback matrix A in terms of an unconstrained matrix A' of size $N \times N$. If a reduced number of parameters is desired, the number of parameters required for the feedback matrix can be reduced by using the Householder or circulant matrix, which can be both parameterized by a single vector of size N, as discussed in Sec. 3.3. The vector $b = [b_1, b_2, \ldots, b_N]$ and $c = [c_1, c_2, \ldots, c_N]$ are also re-parameterized by dividing them by their respective norm. This step is optional as the magnitude of the input and output gain vectors does not affect the coloration of the FDN, but was found desirable as it ensures a consistent amplitude of the generated impulse response which is useful when comparing the results of different optimization runs. The delay lengths m_i stayed constant during the optimization process. Three sets of delay lengths were used:

$$M_4 = [1499, 1889, 2381, 2999]$$
$$M_6 = [997, 1153, 1327, 1559, 1801, 2099]$$
$$M_8 = [809, 877, 937, 1049, 1151, 1249, 1373, 1499],$$

where M_4, M_6, and M_8 are the delay lengths used for the FDN of size $N = 4$, $N = 6$, and $N = 8$ respectively. These sets are the same one's used by Dal Santo et al. (2023). Figure 4.3 shows a diagram of the FDN structure used for the optimization of the parameters for colorless reverberation.

Loss Functions

Two loss functions were tested for the optimization of the FDN parameters for colorless reverberation: a spectral flatness loss, and a time-domain sparsity loss.

The spectral flatness loss is inspired by the spectral loss function used in the work of Dal Santo et al. (2023) and aims to maximize the spectral flatness of the generated impulse response. Spectral flatness is defined as the ratio between the geometric mean and the arithmetic mean of the power

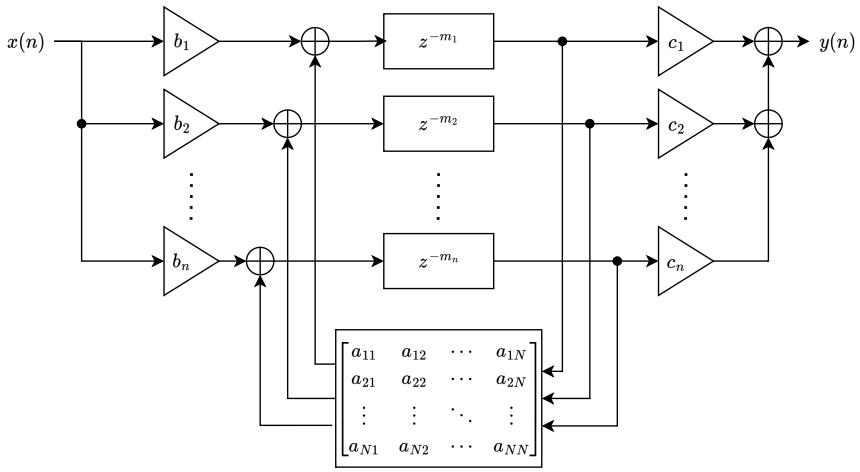


Figure 4.3 Diagram of the FDN structure used for the optimization of the parameters for colorless reverberation. The delay lengths m_i are kept constant during the optimization process.

spectrum and is given by the equation:

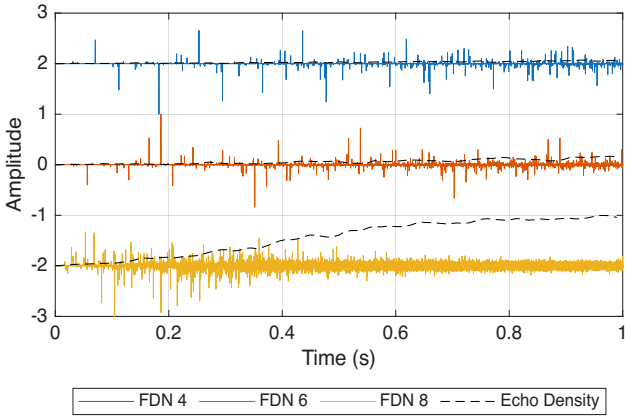
$$\text{Flatness}(X) = \frac{\sqrt[K]{\prod_{k=0}^{K-1} X(k)}}{\frac{1}{K}\sum_{k=0}^{K-1} X(k)}, \tag{4.3}$$

where $X(k)$ is the power spectrum at bin k (Johnston, 1988). The spectral flatness is a measure of how noise-like a signal is, with a value approaching 1 for a perfectly flat spectrum and a value of 0 for a pure tone. Since the optimization framework used in this work expects a loss function to be minimized, the spectral flatness loss $\mathcal{L}_{\text{flatness}}$ is defined as:

$$\mathcal{L}_{\text{flatness}} = 1 - \text{Flatness}(X), \tag{4.4}$$

where X is the power spectrum of the generated impulse response. Figure 4.4 shows the impulse responses generated after optimization using the $\mathcal{L}_{\text{flatness}}$ loss function as well as the initial and final spectral flatness values.

While the resulting IRs were technically more spectrally flat after optimization, they were also noticeably sparse in the time-domain, leading to a low-quality reverberation. Indeed, while it is



(a)

	(N=4)	(N=6)	(N=8)
Initial	0.160	0.112	0.109
Final	0.218	0.249	0.245
Increase (%)	36.25	122.32	124.77

(b)

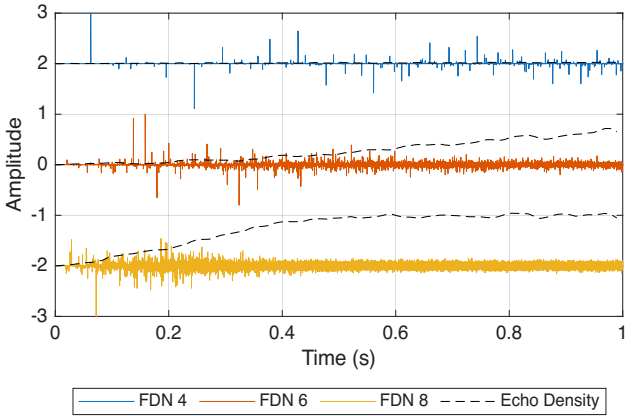
Figure 4.4 Impulse response (a) and spectral flatness (b) of the generated impulse responses after optimization using the spectral flatness loss. The impulse responses were normalized and shifted for better visualization.

often said that white noise should have a spectral flatness of 1, this is not the case in practice when dealing with discrete signals of finite length. Empirically, it was found that the spectral flatness of a randomly generated signal of length $L = 48000$ samples is around 0.56, which is consistent with the findings of Agus et al. (2018). On the other hand, the unit impulse signal has a spectral flatness of 1, which could suggest that the current $\mathcal{L}_{\text{flatness}}$ loss function is actually encouraging the optimization to find a sparse solution in order to maximize the spectral flatness of the generated impulse response, which is undesirable. With this in mind, $\mathcal{L}_{\text{flatness}}$ was modified as follows in order to optimize towards a less sparse, noise-like solution:

$$\mathcal{L}_{\text{flatness}} = \left|\text{Flatness}(X) - 0.56\right|. \tag{4.5}$$

Figure 4.5 shows a sample of the impulse responses generated after optimization using the modified spectral flatness loss. The FDN of size 4 is still sparse, but a slight improvement in the echo density build-up can be observed on the FDN of size 6 and 8.

Figure 4.6 shows the feedback matrix and input/output gains of the optimized FDNs using the modified spectral flatness loss. The matrices are sparse, with most coefficients close to zero except



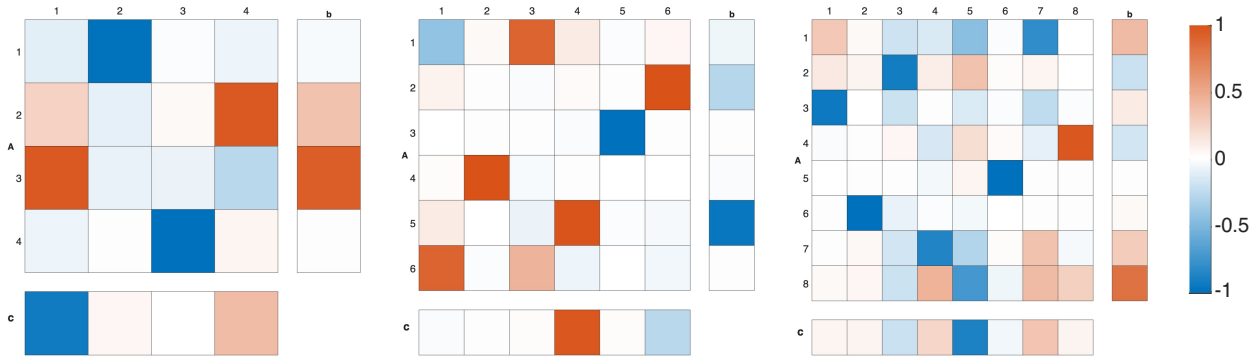
(a)

	(N=4)	(N=6)	(N=8)
Initial	0.125	0.103	0.096
Final	0.211	0.226	0.232
Increase (%)	68.80	119.42	141.67

(b)

Figure 4.5 Impulse response (a) and spectral flatness (b) of the generated impulse responses after optimization using the modified spectral flatness loss. The impulse responses were normalized and shifted for better visualization.

for one coefficient per column. If the non-zero coefficients were located on the diagonal, the resulting FDN would be equivalent to N parallel comb filters, but in this particular case with the non-zero coefficients located off the diagonal, the resulting FDN behaves more like a set of comb filters placed in series, which is still undesirable.

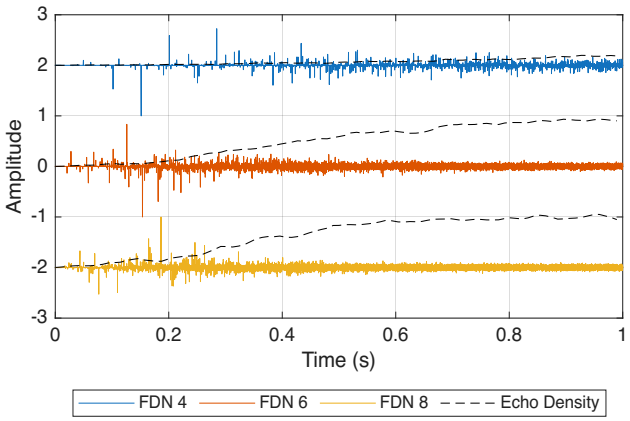


(a) $N = 4$ (b) $N = 6$ (c) $N = 8$

Figure 4.6 Feedback matrix and input/output gains of the optimized FDNs using the modified spectral flatness loss. The $N \times N$ grids show the value of the matrices A. The bottom $1 \times N$ grids show the value of the output gains c_i, and the right $N \times 1$ grids show the value of the input gains b_i.

So far, all of the initial parameters during the optimization process were drawn from a normal

distribution. During development, it was found useful to set the choice of the initial parameters to a set of constant values to make it easier to compare the results of different optimization runs. Coincidentally, it was also found that using the Hadamard matrix (or Householder matrix for non-power-of-two sizes) as the initial feedback matrix and setting all input and output gains to $\frac{1}{\sqrt{N}}$ resulted in a noticeable improvement in the optimization process. This FDN configuration will now be refered to as $\text{FDN}_{\text{static}}$. Figure 4.7 shows the impact of using these fixed initial parameters on the echo density and spectral flatness of the resulting IR.



(a)

	(N=4)	(N=6)	(N=8)
Initial	0.092	0.090	0.085
Final	0.222	0.218	0.237
Increase (%)	141.30	142.22	178.82

(b)

Figure 4.7 Impulse response (a) and spectral flatness (b) of the generated impulse responses after optimization using the modified spectral flatness loss and a fixed set of initial parameters. The impulse responses were normalized and shifted for better visualization.

Figure 4.8 shows the impact of $\mathcal{L}_{\text{flatness}}$ when applied to $\text{FDN}_{\text{static}}$ on the echo density profile of the generated IR compared to an initial FDN with $b_i = c_i = 1/\sqrt{N}$ and the initial feedback matrix set to the Hadamard matrix for FDNs of size 4 and 8 and to the Householder matrix for FDNs of size 6. It becomes apparent that optimizing solely for spectral flatness consistently results in a decrease in the echo density profile of the optimized IR compared to the initial FDN. Nevertheless, these results show that targeting a spectral flatness of 0.56 instead of 1 and setting the initial parameters to a fixed set of values did help to mitigate the issue, even if the echo density profile was not directly optimized in the process.

The issues with sparsity in the optimized parameters and corresponding impulse responses are

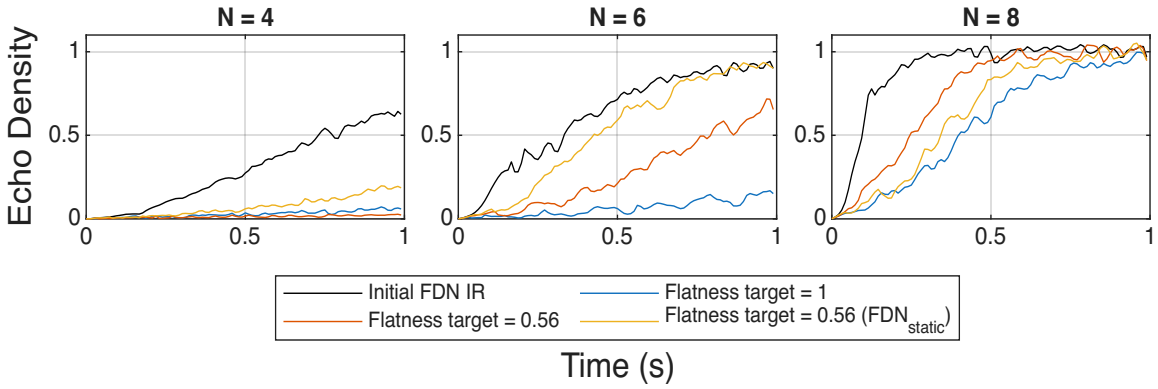


Figure 4.8 Comparison of the echo density profile of the initial and optimized FDNs using the three loss functions discussed in this section.

not new and were also observed by Dal Santo et al. (2023). In their work, the authors proposed the addition of a sparsity loss term used in conjunction with the spectral loss to mitigate the issue. The sparsity loss $\mathcal{L}_{\text{sparsity}}$ is defined as the ratio between the ℓ_2 norm and the ℓ_1 norm of the impulse response:

$$\mathcal{L}_{\text{sparsity}} = \frac{\|x\|_2}{\|x\|_1}, \tag{4.6}$$

where x is the impulse response and the operator $\|\cdot\|_p$ is the ℓ_p norm. The ℓ_1 and ℓ_2 norms are defined as follow:

$$\ell_1 = \sum_{i=0}^{n} |x_i| \tag{4.7}$$

$$\ell_2 = \sqrt{\sum_{i=0}^{n} |x_i|^2}. \tag{4.8}$$

The ℓ_1 in the denominator encourages the optimization to find a solution with more non-zero coefficients, while the ℓ_2 in the numerator encourages the optimization to find a solution with smaller coefficients, or in other words, a solution with less energy concentrated in a few coefficients which would results in a few distinct echoes. Through testing, it was found that calculating $\mathcal{L}_{\text{sparsity}}$ on the first 100 ms of the impulse response was sufficient to mitigate the sparsity issue. The new

loss function is then defined as:

$$\mathcal{L}_{\text{color}} = \alpha_1 \mathcal{L}_{\text{flatness}} + \alpha_2 \mathcal{L}_{\text{sparsity}}, \tag{4.9}$$

where α_1 and α_2 are weighting factors used to balance the contribution of each loss term. Values of $\alpha_1 = 1$ and $\alpha_2 = 0.5$ were found to give good results. Since the magnitude of $\mathcal{L}_{\text{sparsity}}$ varies greatly depending on the length of the portion of the impulse response used for the calculation, the scaling factors might need to be adjusted if a different length is used. Figure 4.9 shows the impact of $\mathcal{L}_{\text{sparsity}}$ on the echo density profile and Table 4.5 shows the impact on the spectral flatness of the generated impulse response.

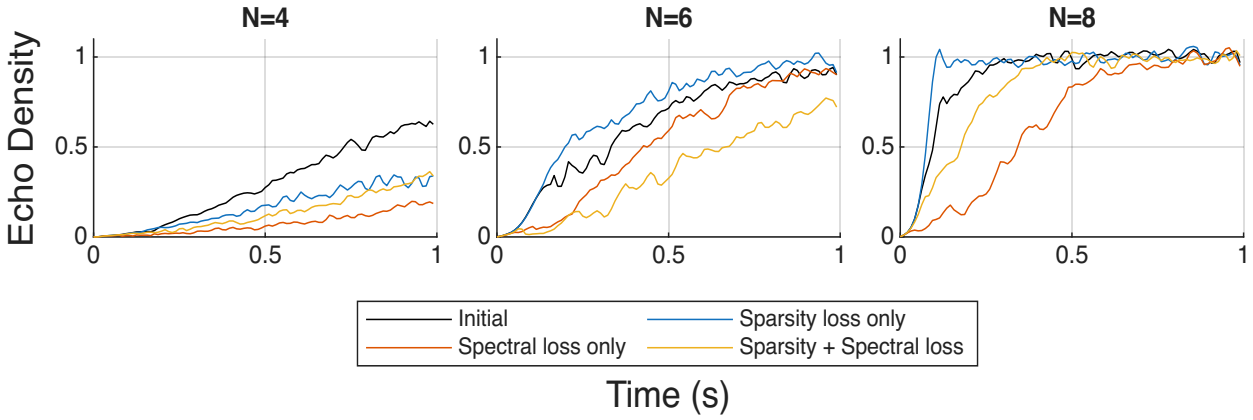


Figure 4.9 Echo density profile of optimized FDNs.

	Initial	$\mathcal{L}_{\text{sparsity}}$	$\mathcal{L}_{\text{flatness}}$	$\mathcal{L}_{\text{color}}$
(N=4)	0.092	0.124	0.222	0.228
(N=6)	0.090	0.123	0.218	0.207
(N=8)	0.085	0.096	0.237	0.230

Table 4.5 Spectral flatness of the generated impulse responses after optimization.

Finally, while it is true that the choice of attenuation filters does not affect the coloration of the FDN, it was found that adding a homogeneous decay using Equation (3.11) with an RT_{60} of 1 s to the previously lossless FDN resulted in a noticeable improvement in the optimization process. This

additional decay helps to emphasize the early part of the impulse response during the evaluation of the loss function, resulting in a faster echo density build-up. Figure 4.10 shows the effect of this additional decay on the optimization process. The additional decay results in a consistent increase in the echo density profile in all FDN sizes tested and also results in a significant increase in the spectral flatness of the generated IR.

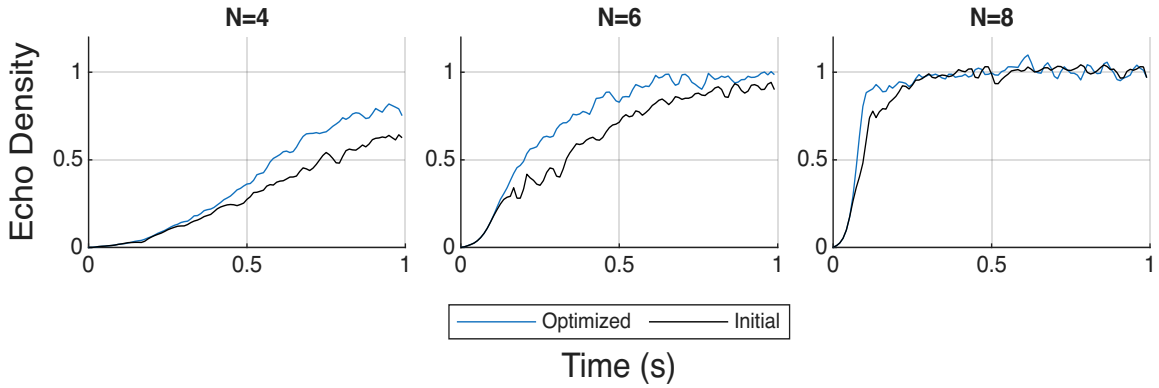


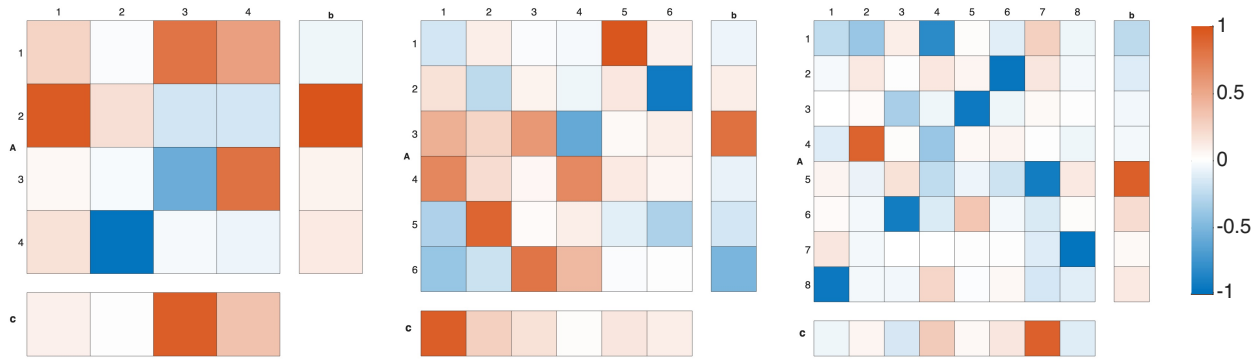
Figure 4.10 Echo density profile of the optimized FDNs using $\mathcal{L}_{\text{color}}$ and an additional homogeneous decay of 1 s. $\text{FDN}_{\text{static}}$ is still being used as the initial configuration.

	(N=4)	(N=6)	(N=8)
Initial	0.092	0.090	0.085
Final	0.391	0.328	0.345
Increase (%)	325.00	264.44	305.88

Table 4.6 Spectral flatness of the generated impulse responses after optimization.

The effect can also be observed on the resulting feedback matrix coefficients, as shown in Fig. 4.11, which are now noticeably less sparse than in Fig. 4.6.

Figure 4.12 shows the probability density function of the modal excitation of the FDNs before and after optimization. To obtain the data shown in Fig. 4.12, the modal distribution was flipped and shifted by $-60\,\text{dB}$ before using the MATLAB function `raylfit` to obtain the scaling factor σ. A smaller value of σ corresponds to a narrower distribution and therefore less coloration. The optimized FDNs show a significant decrease in the value of σ in comparison to the initial state.



(a) $N = 4$ (b) $N = 6$ (c) $N = 8$

Figure 4.11 Feedback matrix and input/output gains of the optimized FDNs using the modified spectral flatness loss and additional decay. The $N \times N$ grids show the value of the matrices A. The bottom $1 \times N$ grids show the value of the output gains c_i, and the right $N \times 1$ grids show the value of the input gains b_i.

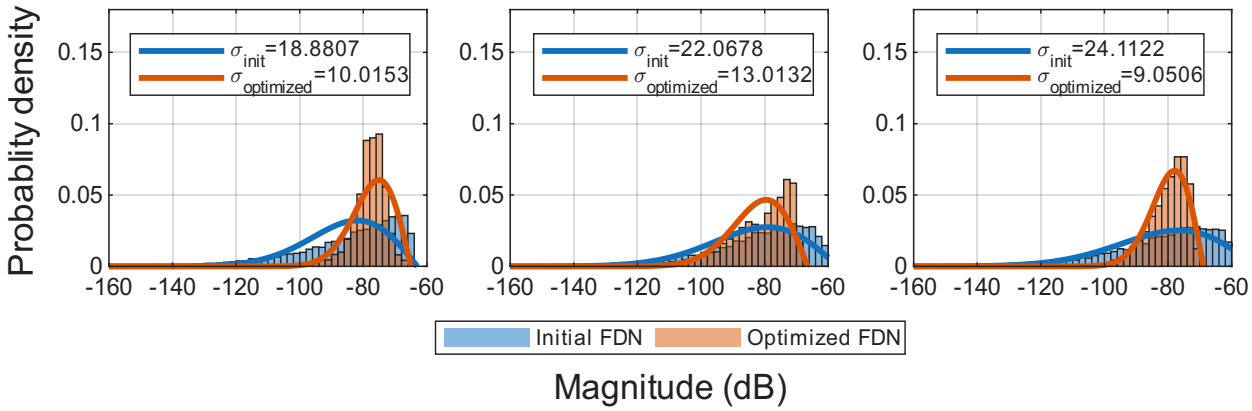


Figure 4.12 Probability density function of the modal excitation of three FDNs of size 4 (left), 6 (center), and 8 (right) optimized for colorless reverberation.

Comparison to previous work

Since the implementation of Dal Santo, Prawda, et al. (2025) is publicly available online, it is possible to compare their results with the results obtained in this work. The same set of delay lengths M_4, M_6, and M_8 were used for the optimization of the FDNs. In the case of the FLAMO library, the initial parameters for the optimization were drawn from a normal distribution, as opposed to the fixed set of parameters used in this work. Figure 4.13 shows the echo density profiles of

three FDNs optimized with the FLAMO library (Dal Santo, De Bortoli, et al., 2025) compared to the implementation presented in this thesis. Figure 4.14 shows the probability density function of the modal excitation of the FDNs optimized using the implementation of Dal Santo, Prawda, et al. (2025). These results show that the optimization process implemented in this thesis is able to achieve similar results to the FLAMO library, with a similar echo density profile and modal excitation. While the modal density of the optimized FDN of size 4 is much higher for the FLAMO implementation, it is interesting to note that the echo density profile did not noticeably improve compared to the initial impulse response. In fact, both implementations struggle to increase the echo density enough to reach the late reverberation stage, suggesting that the FDN topology itself or the choice of delay lengths might be the limiting factor in the 4-channel case.

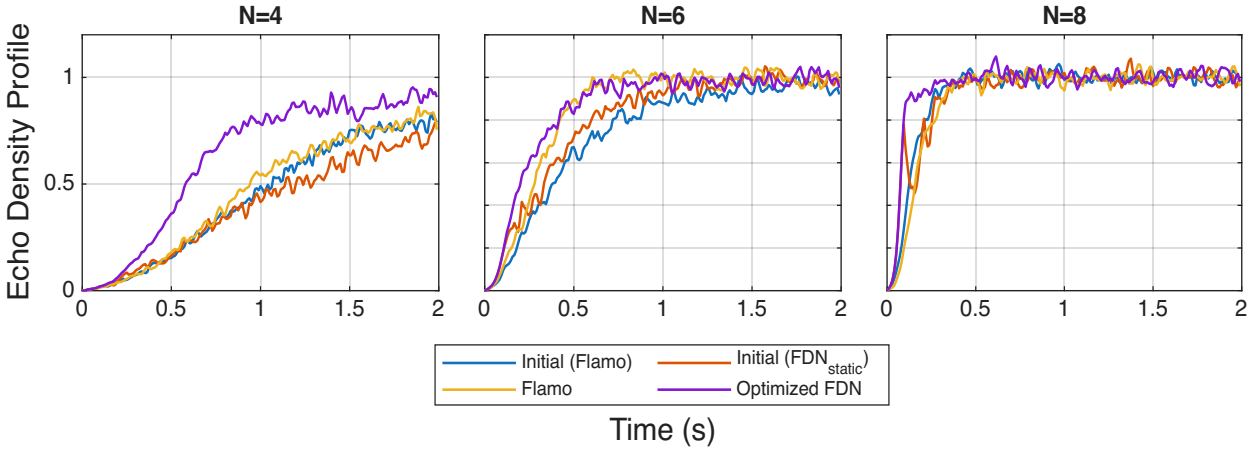


Figure 4.13 Echo density profile at initialization and after optimization using the FLAMO library (Dal Santo, De Bortoli, et al., 2025). In purple, the echo density profile of the optimized FDNs using the implementation presented in this thesis is shown for comparison.

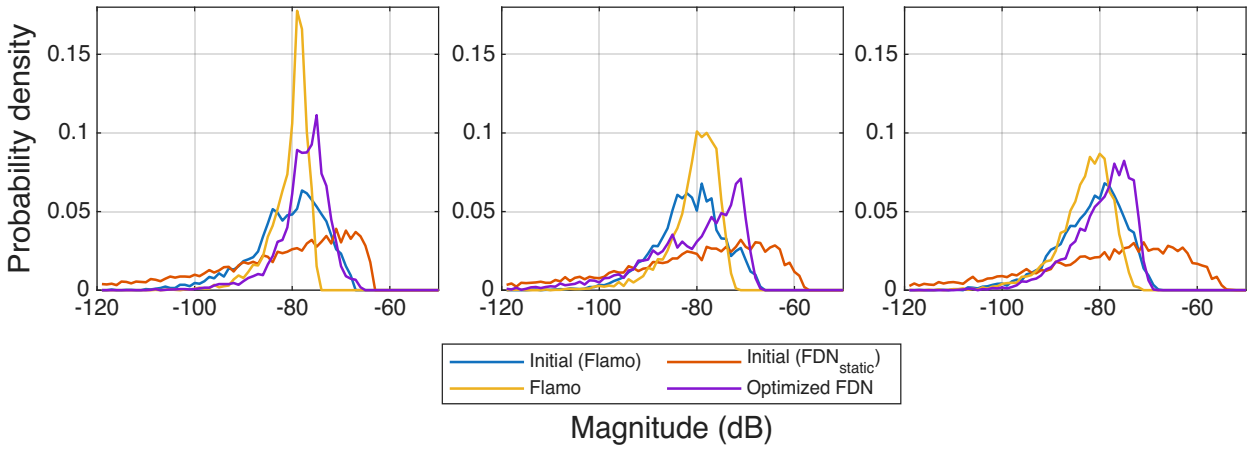


Figure 4.14 Probability density function of the modal excitation of three FDNs optimized for colorless reverberation using the FLAMO library (Dal Santo, De Bortoli, et al., 2025). In purple, the modal density profile of the optimized FDNs using the implementation presented in this thesis is shown for comparison.

Table 4.7 shows the time taken to perform the optimization for colorless reverberation of FDNs of size 4, 6, and 8 for both implementations. The optimization process implemented in this thesis is significantly faster despite not using automatic differentiation and relying on finite differences for the calculation of the gradients.

FDN Size	4	6	8
sfFDN	1.5	2.6	4.7
FLAMO	48.0	81.0	135.0

Table 4.7 Time in seconds taken to optimize the parameters of an FDN for colorless reverberation using the implementation presented in this thesis and the FLAMO library (Dal Santo, De Bortoli, et al., 2025).

4.2.3 Optimization for Impulse Response Matching

Once a colorless FDN is obtained, it is now possible to optimize the FDN parameters to match the characteristics of a target RIR. For this step, the parameters to optimize are the frequency-dependent $RT_{60}(\omega)$ of the attenuation filters, the gains $T(\omega_i)$ of the tone correction filter, and an additional global gain factor G_{global}. Two FDN configurations were tested. The first, referred to as

$\text{FDN}_{\text{10band}}$, used an octave-band GEQ based on the two-stage approach of Välimäki et al. (2024) for the attenuation filters. A single set of 10 $RT_{60}(\omega)$ parameters will be used to design the filters for all channels of the FDN. The 10 bands of the filter are at 32 Hz, 62 Hz, 125 Hz, 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz, 8000 Hz, and 16 000 Hz. The corresponding gains g_i will be calculated using the formula:

$$g_i(\omega) = \frac{-60m_i}{f_s RT_{60}(\omega)}, \tag{4.10}$$

where m_i is the delay length of the i-th delay line, f_s is the sampling frequency in Hz, and $RT_{60}(\omega)$ is the desired reverberation time at frequency ω in seconds. The second FDN configuration, referred to as $\text{FDN}_{\text{3band}}$, will use a 3-band attenuation filter composed of a 2nd order low-shelf filter, a 2nd order high-shelf filter, and a simple gain factor for the mid frequencies. The filter cutoff frequencies for the low and high-shelf filters were set to 800 Hz and 8 kHz, respectively. As opposed to the octave-band GEQ configuration, each channel of the FDN will have its own set of $RT_{60}(\omega)$ parameters, resulting in a total of $3 \times N$ parameters for the attenuation filters. Equation (4.10) will still be used to calculate the corresponding gains $g_{\text{low}}, g_{\text{mid}}, g_{\text{high}}$ for each delay line. Both configurations will use the same tone correction filter, again based on the two-stage approach of Välimäki et al. (2024), with 10 parameters $T(\omega_i)$ corresponding to the gain of each band of the GEQ. Since the optimal range for the gains of the GEQ is ± 12 dB (Välimäki et al., 2024), a global gain factor G_{global} was added to help the parameters $T(\omega_i)$ stay in that range. A parameter vector $\mathcal{P}$ can then be represented as a vector of size $2 \times 10 + 1 = 21$ for $\text{FDN}_{\text{10band}}$ and $3N + 10 + 1$ for $\text{FDN}_{\text{3band}}$, containing the RT_{60} of the attenuation filters, the gains of the tone correction filter, and the global gain factor:

$$\mathcal{P}_{\text{geq}} = [RT_{60}(\omega_1), RT_{60}(\omega_2), \ldots, T(\omega_1), T(\omega_2), \ldots, T(\omega_{10}), G_{global}]. \tag{4.11}$$

$$\mathcal{P}_{\text{3band}} = [RT_{60}(\omega_{11}), RT_{60}(\omega_{12}), \ldots, RT_{60}(\omega_{3N}), T(\omega_1), T(\omega_2), \ldots, T(\omega_{10}), G_{global}]. \tag{4.12}$$

To ensure good results during the optimization process, the $RT_{60}(\omega)$ parameters were constrained to be between 0.1 s and 20 s. The tone correction filter gains $T(\omega_i)$ and global gain factor G_{global} were

left unconstrained. The optimization was performed at a sampling frequency of 48 kHz. An FDN of size $N = 8$ was used for the optimization, with delay lengths M_8 as defined in Sec. 4.2.2. The RIR matching optimization was tested on a set of artificially generated RIRs using the `pyroomacoustics` library (Scheibler et al., 2018). Three RIRs were generated to cover a range of RT_{60} values: a "Concert Hall" (high), a "Meeting Room" (medium), and an "Office Lobby" (low). Synthetic RIRs were used because the presence of noise in real RIRs greatly affects the shape of the EDC and EDR, which would adversely affect the optimization process. While this is not an insurmountable issue and certain methods exist to minimize the impact of noise on the EDC of the RIR, it was deemed out of the scope of this project. Similarly to what was done in (Dal Santo et al., 2024), an FIR filter meant to simulate early reflections was added to the direct path of the FDN. Since the target RIRs were already generated using the image source method, the early reflection filter was generated using the same method with a maximum reflection order of 3. Figure 4.15 shows the RT_{60} of the three target RIRs used for the optimization process and were estimated using DecayFitNet (Götz et al., 2022).

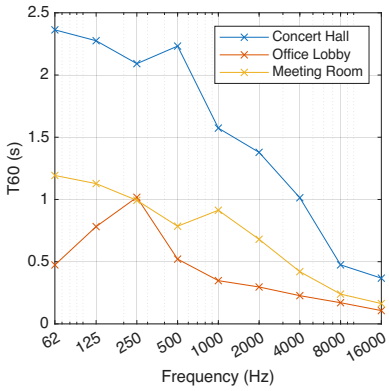


Figure 4.15 RT_{60} of the target RIRs.

Loss Functions

Various loss functions have been proposed in the literature for the goal of matching the characteristics of a target RIR. Mezza et al. (2024) proposed a 3-part loss function based on the echo density

profile, the EDC, and the mel-scale EDR. Since none of the parameters currently optimized have a direct effect on the echo density of the FDN, the echo density profile was not included in the loss function used in this project. The mel-scale EDR is defined as:

$$\mathrm{EDR}_{\mathrm{mel}}(t_n, f_k) = 10 \log_{10} \sum_{m=n}^{M} \left| \mathcal{H}_{\mathrm{mel}}(m,k) \right|^2 , \tag{4.13}$$

where $\mathcal{H}_{\mathrm{mel}}(m,k)$ denotes bin k of the mel-scale spectrogram at time m and is obtained by applying triangular mel-scale filters to the squared magnitude of the STFT of the impulse response. The STFT was computed using an FFT size of 4096 samples, a hop size of 128 samples, a Hann window of 1024 samples, and 32 triangular mel-scale filters. The loss function $\mathcal{L}_{EDR}$ is then defined as the least absolute deviation error between the target and optimized mel-scale EDR:

$$\mathcal{L}_{EDR} = \frac{\sum_k \sum_m \left| \mathrm{EDR}_{\mathrm{mel}}^{target}(m,k) - \mathrm{EDR}_{\mathrm{mel}}^{optim}(m,k) \right|}{\sum_k \sum_m \left| \mathrm{EDR}_{\mathrm{mel}}^{target}(m,k) \right|}. \tag{4.14}$$

The energy decay curve loss function $\mathcal{L}_{EDC}$ is defined as

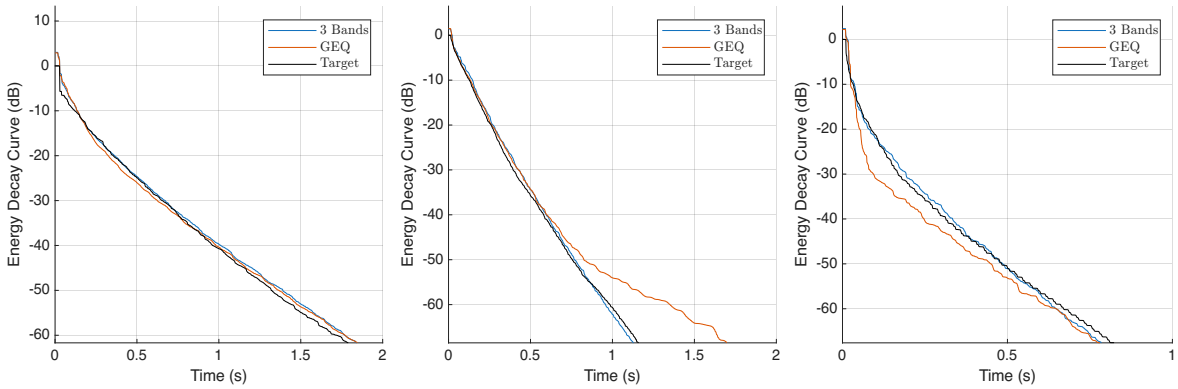
$$\mathcal{L}_{EDC} = \frac{\sum_n \left(\mathrm{EDC}^{target}(n) - \mathrm{EDC}^{optim}(n) \right)^2}{\sum_n \mathrm{EDC}^{target}(n)^2}, \tag{4.15}$$

where $\mathrm{EDC}(n)$ is the energy decay curve at time n as defined in Equation (2.1). The final loss function is then defined as

$$\mathcal{L}_{\mathrm{spectrum}} = \alpha_1 \mathcal{L}_{EDR} + \alpha_2 \mathcal{L}_{EDC}, \tag{4.16}$$

where α_1 and α_2 are weighting factors used to balance the contribution of each loss term. Values of $\alpha_1 = 1$ and $\alpha_2 = 0.1$ were found to give good results. Figure 4.16 shows that the optimized FDNs were able to closely match the EDC of the target RIR for the three different RIR tested, with $\mathrm{FDN}_{\mathrm{3band}}$ showing a slightly better fit than $\mathrm{FDN}_{\mathrm{10band}}$.

Figures 4.17 to 4.19 shows the EDR of the optimized FDNs compared to the target RIR. The EDR was computed by applying an octave-band filter bank to the impulse response and calculating



(a) Concert hall (b) Office lobby (c) Meeting room

Figure 4.16 EDC of the two FDN configurations optimized with $\mathcal{L}_{\text{spectrum}}$. For visualization purposes, the EDCs were normalized such that the target EDC starts at 0 dB.

the energy decay curve of each band. The optimized FDNs were able to closely match the EDR of the target RIR in all three cases, with $\text{FDN}_{\text{3band}}$ showing a slightly better fit than $\text{FDN}_{\text{10band}}$ suggesting that individually tuning the filters for each delay line separately allows for more flexibility when matching the EDR than simply using a single set of filters with more bands. The mel-scale EDR of the optimized FDNs compared to the target RIR is shown in figs. 4.20 to 4.22.

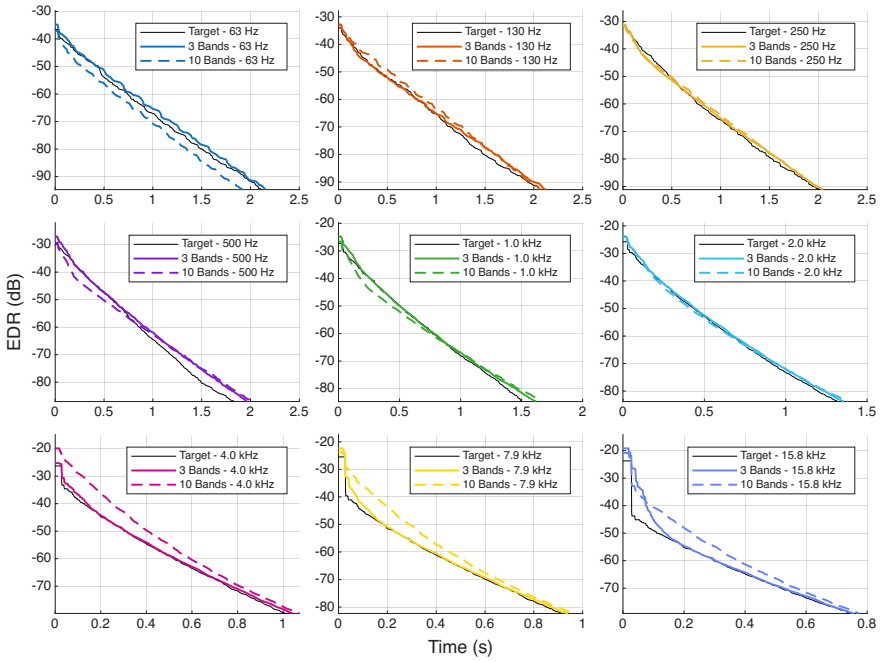


Figure 4.17 Concert hall: EDR of the two FDN configurations optimized using $\mathcal{L}_{\text{spectrum}}$.

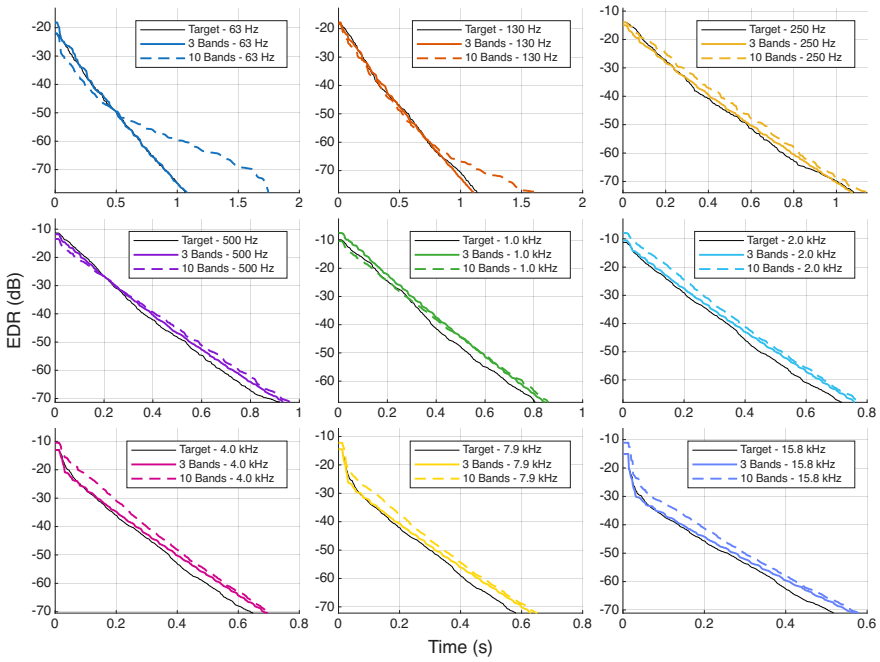


Figure 4.18 Office lobby: EDR of the two FDN configurations optimized using $\mathcal{L}_{\text{spectrum}}$.

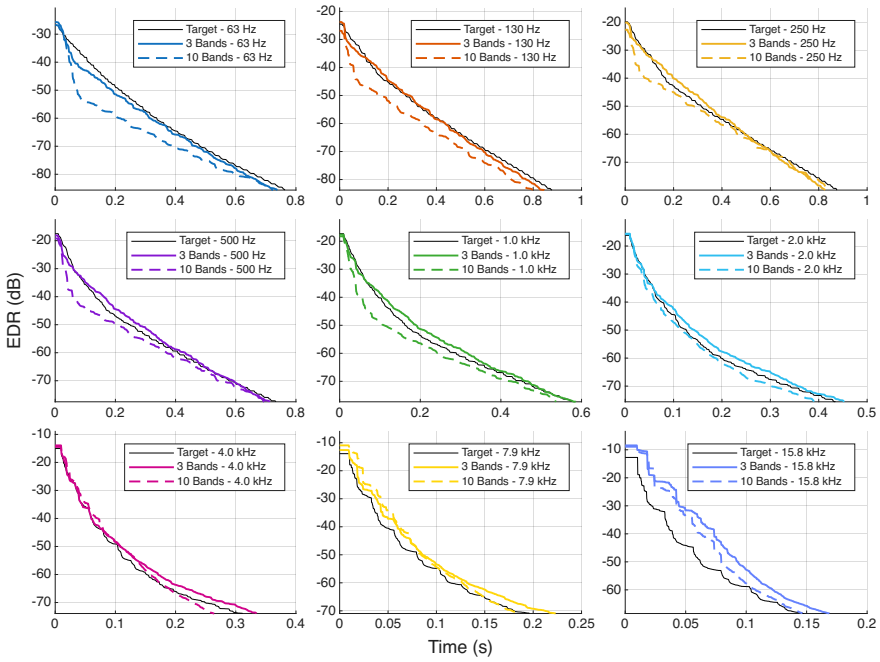
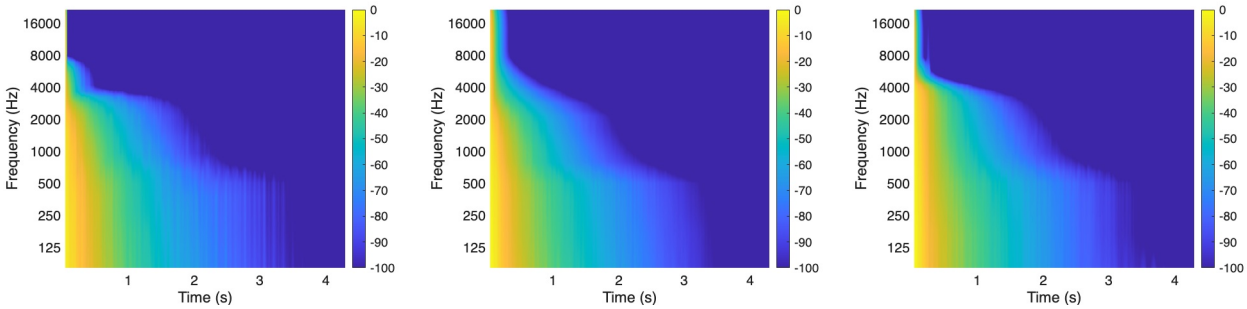
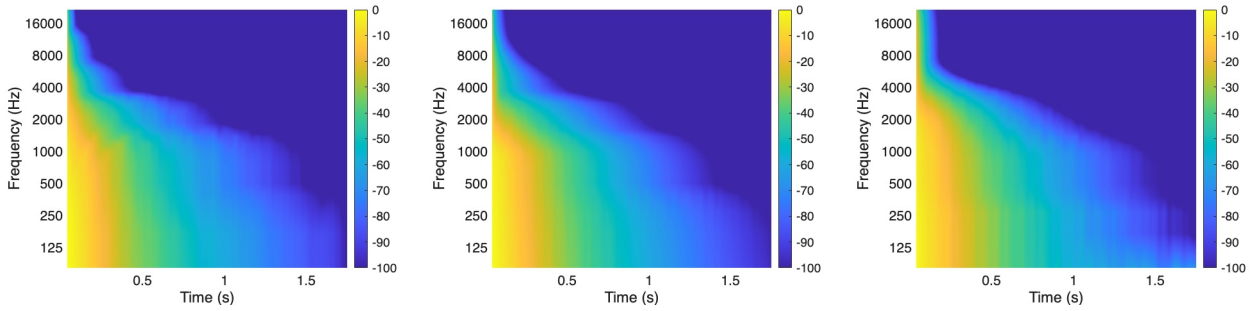


Figure 4.19 Meeting room: EDR of the two FDN configurations optimized using $\mathcal{L}_{\text{spectrum}}$.



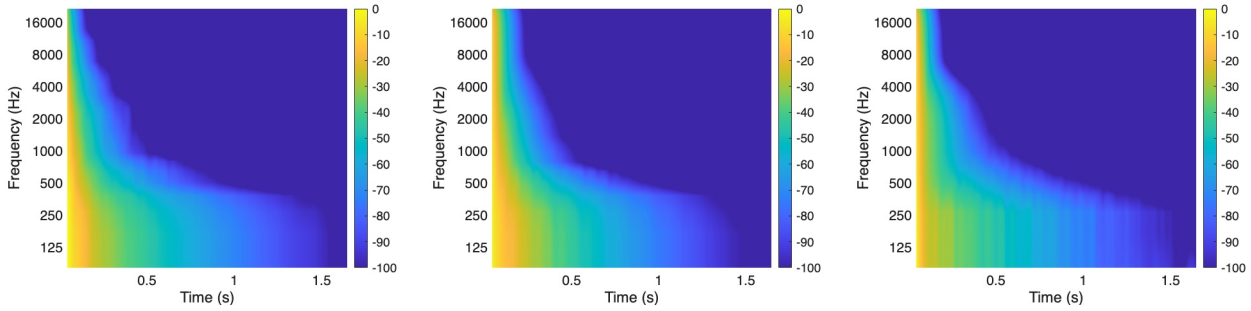
(a) Target (b) $\text{FDN}_{\text{3band}}$ (c) FDN_{geq}

Figure 4.20 Concert Hall: Mel-scale EDR of the two FDN configurations optimized using $\mathcal{L}_{\text{spectrum}}$.



(a) Target (b) $\text{FDN}_{\text{3band}}$ (c) FDN_{geq}

Figure 4.21 Office Lobby: Mel-scale EDR of the two FDN configurations optimized using $\mathcal{L}_{\text{spectrum}}$.



(a) Target (b) $\text{FDN}_{\text{3band}}$ (c) FDN_{geq}

Figure 4.22 Meeting Room: Mel-scale EDR of the two FDN configurations optimized using $\mathcal{L}_{\text{spectrum}}$.

Figures 4.23 to 4.25 shows a comparison of the EDR of the FDNs optimized using the $\text{FDN}_{\text{3band}}$ configuration and two other methods. The first by Ibnyahya and Reiss (2022), referred to as "MatchReverb" and presented earlier in Sec. 3.7.1, and the second by Dal Santo et al. (2024), referred to as "RIR2FDN" and presented earlier in Sec. 3.7.2. While these two methods use genetic algorithms and differentiable DSP techniques, respectively, to optimize some of the parameters of the FDN, the attenuation and tone correction filter parameters are designed using an analysis-synthesis approach in both cases. The results show that the optimization process implemented in this thesis is able to achieve a much closer match to the target EDR than both the MatchReverb and RIR2FDN methods. The "MatchReverb" method fails to accurately match the initial level of

the EDR, which can be seen by the large vertical offset between the target and optimized EDR. "RIR2FDN" is able to accurately match the initial level of the EDR but is unable to match the slope of the EDR with the same accuracy as the $\text{FDN}_{\text{3band}}$ method. "RIR2FDN" also does not include the early reflection FIR filter in the optimization process, which could explain some of the discrepancies between the target and synthesized EDR in the early part of the curve. These results show that the current analysis-synthesis approach for the design of the attenuation and tone correction filters is not sufficient to achieve a close match to the target EDR and that optimizing the parameters of these filters directly can lead to better results.

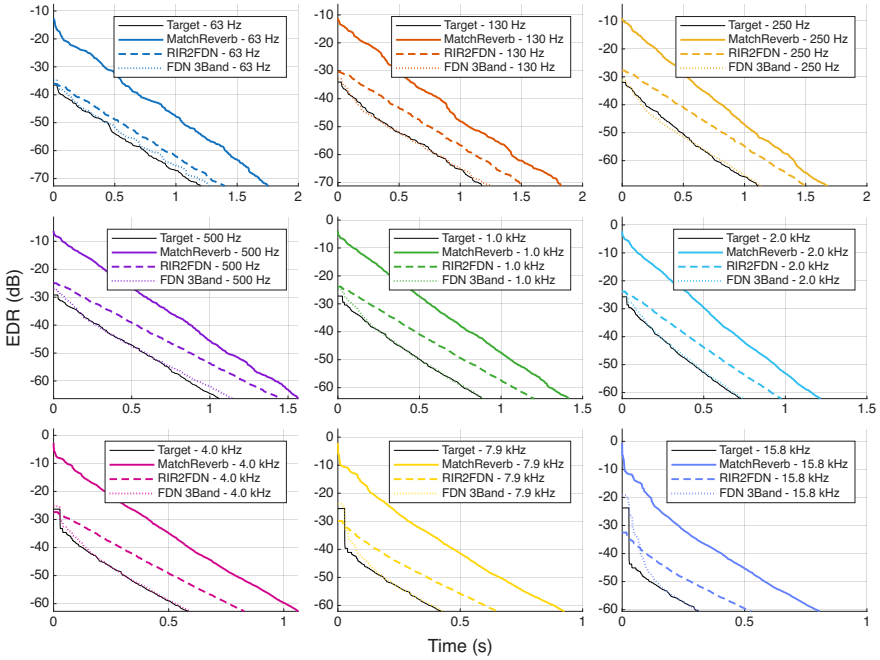


Figure 4.23 Concert hall: EDR of the optimized $\text{FDN}_{\text{3band}}$ compared to MatchReverb and RIR2FDN.

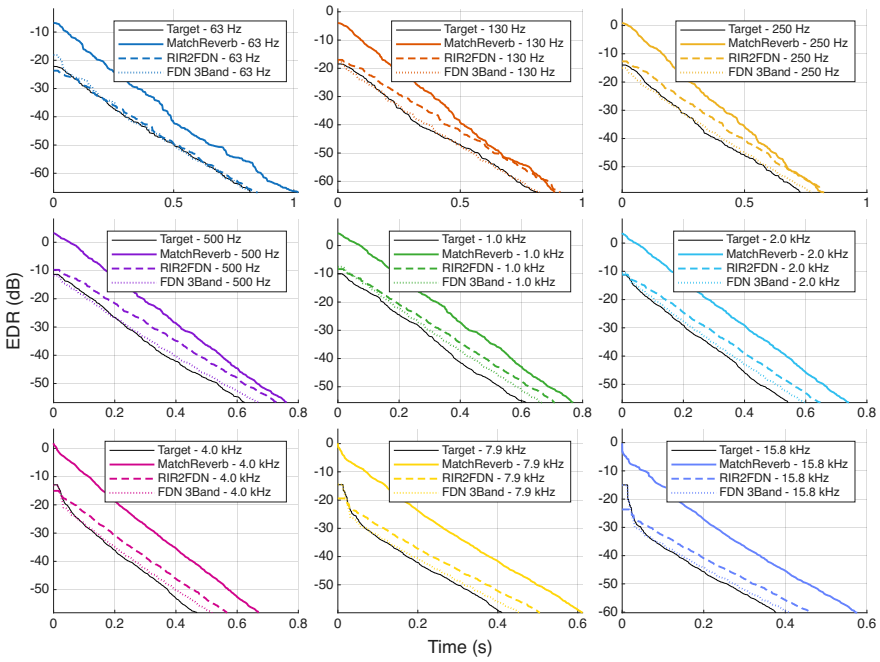


Figure 4.24 Office lobby: EDR of the optimized $\text{FDN}_{\text{3band}}$ compared to MatchReverb and RIR2FDN.

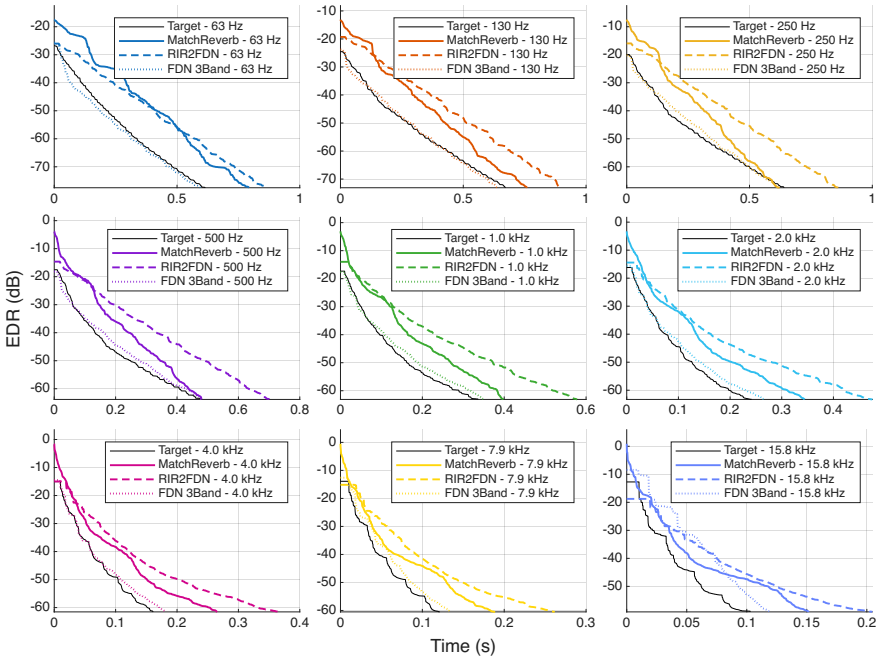


Figure 4.25 Meeting room: EDR of the optimized $\text{FDN}_{\text{3band}}$ compared to MatchReverb and RIR2FDN.

4.3 Graphical User Interface for Experimentation

An application[7] was developed to allow users to experiment with different FDN configurations and parameters in real-time. Figure 4.26 shows a screenshot of the application.

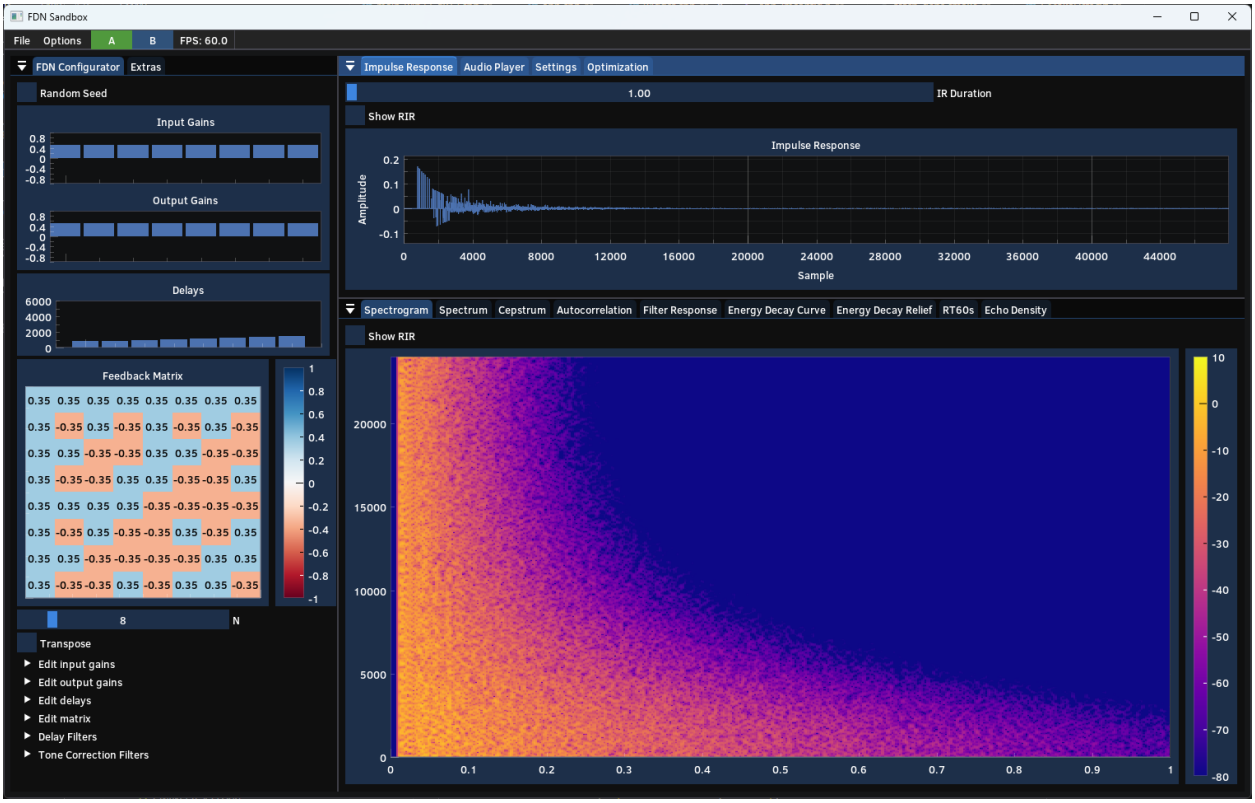


Figure 4.26 Screenshot of the FDN Sandbox application.

The application includes an audio player enabling users to listen to the generated impulse responses as well as auralization of reverberated sound files. Figure 4.27 shows the user interface of the audio player. The application also includes a convolution reverb module to allow the user to auralize external IRs and compare them to the current FDN.

Since the application is meant as a tool for experimentation and research, it also includes a variety of graphs that were found to be useful when analyzing FDNs. These plots include the spectrogram, the spectrum and modal density histogram of the impulse response, the cepstrum, the

[7] https://github.com/Segfault1602/FDNSandbox

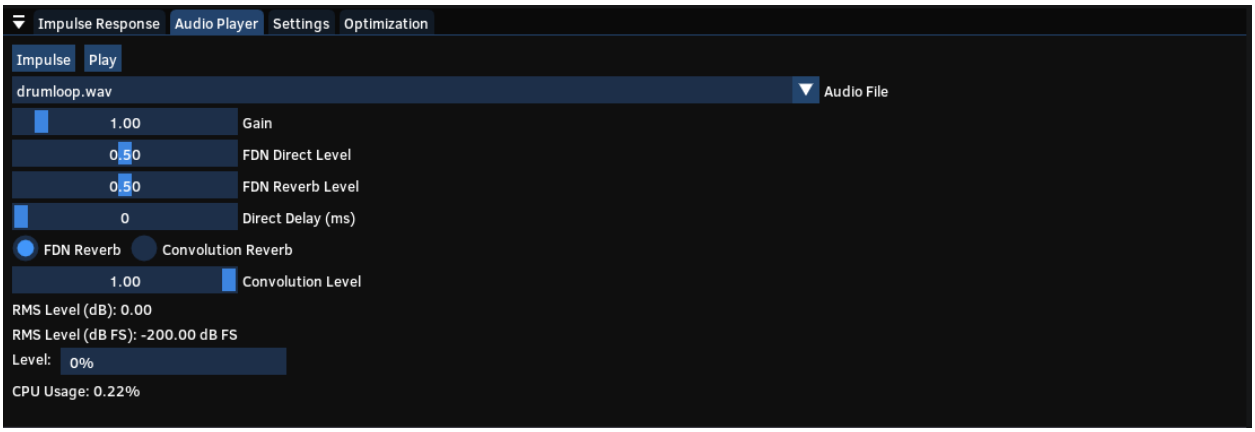
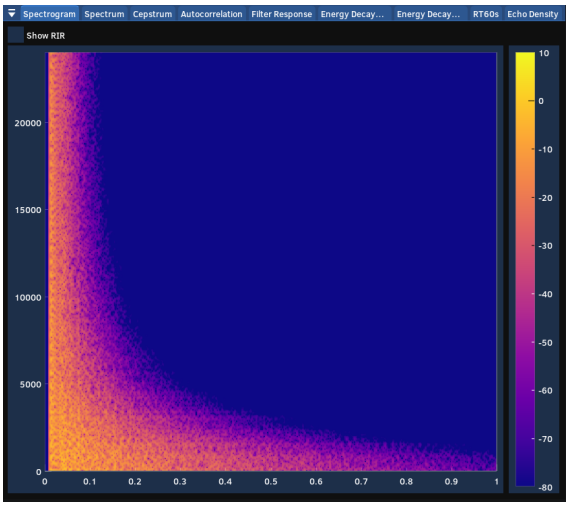
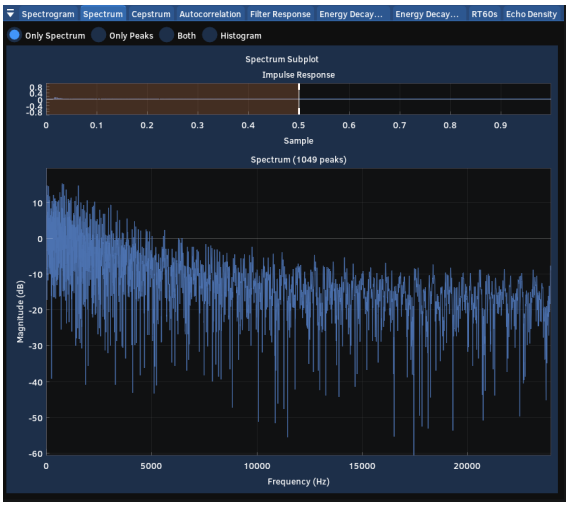


Figure 4.27 Screenshot of the audio player in the FDN Sandbox application.

time and spectral-domain autocorrelation, the frequency response of each individual attenuation and tone correction filter, the EDC and Mel-scale EDR, the RT_{60}s and the echo density. Each plot is updated in real-time as the user changes the parameters of the FDN, allowing for a better understanding of how each parameter affects the different characteristics of the generated impulse response. Figure 4.28 shows a screenshot of the different plots available in the application.

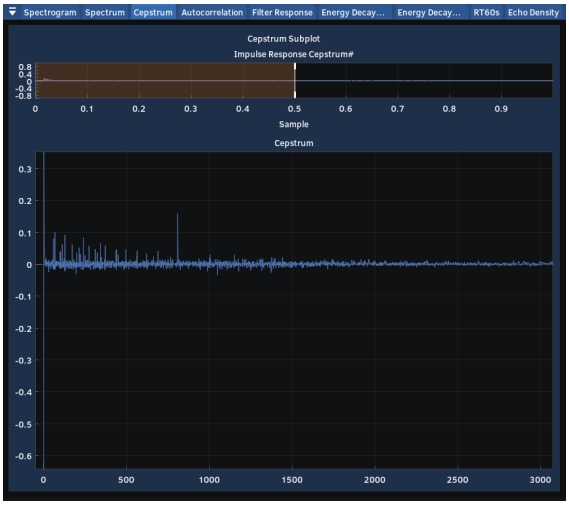


(a) Spectrogram

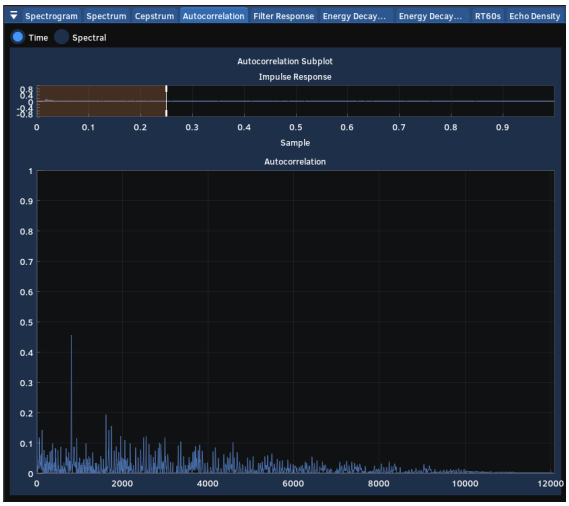


(b) Spectrum

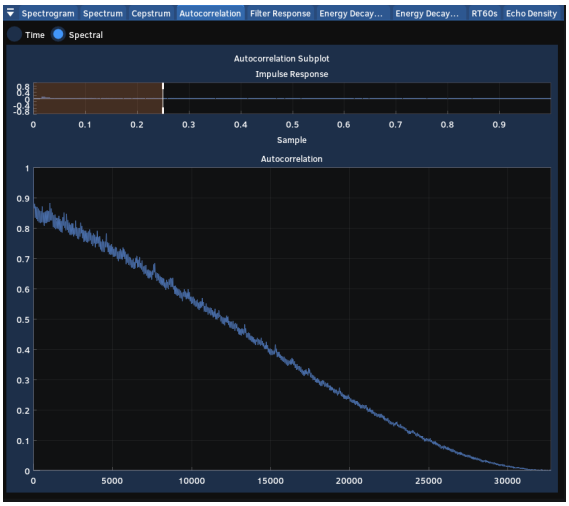
(c) Modal density histogram



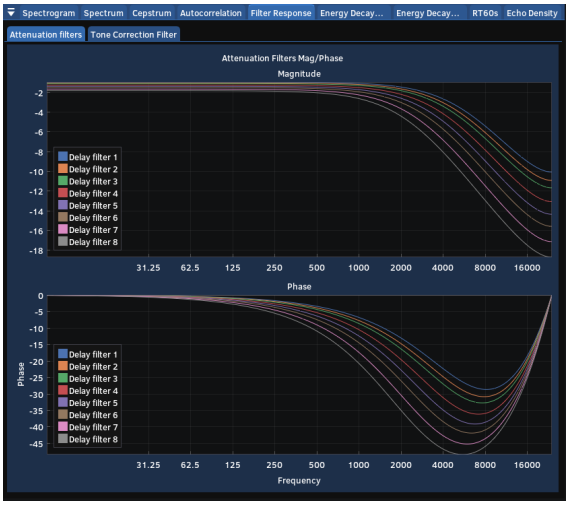
(d) Cepstrum



(e) Autocorrelation

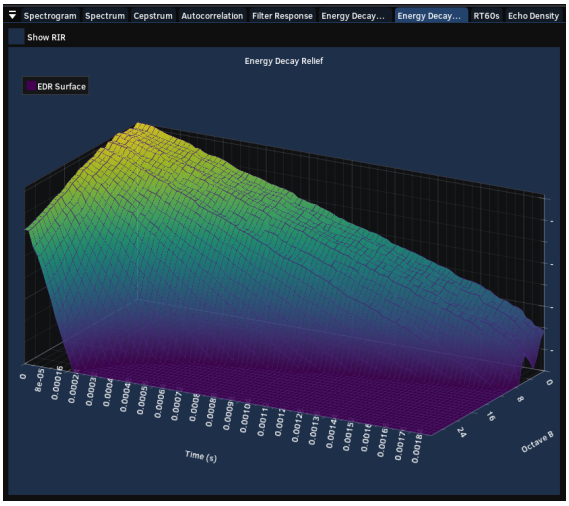


(f) Spectral Autocorrelation

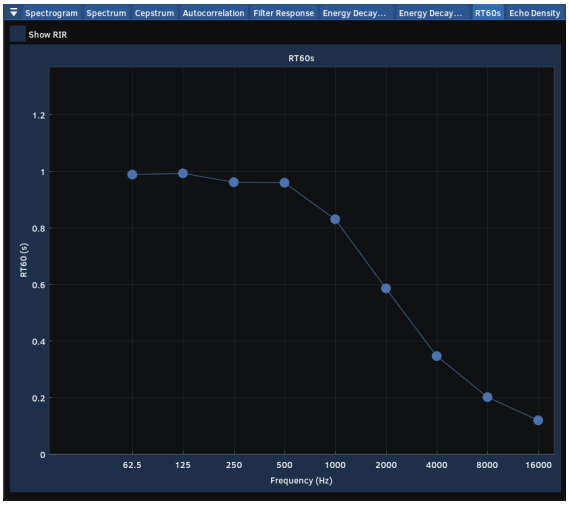


(g) Att. Filter Frequency Response

(h) Energy Decay Curve



(i) Mel-Scale EDR



(j) Reverberation Time (RT_{60})

(k) Echo Density

Figure 4.28 Screenshot of the different plots available in the FDN Sandbox application.

Finally, the application also includes an interface to optimize the parameters of the FDN. Two types of optimization are supported: optimization for colorless reverberation and optimization for RIR matching. For the colorless optimization, the input and output gains as well as the matrix coefficients are optimized. For the RIR matching optimization, the attenuation and tone correction filter gains are optimized. Figure 4.29 shows a screenshot of the optimization interface in the application.

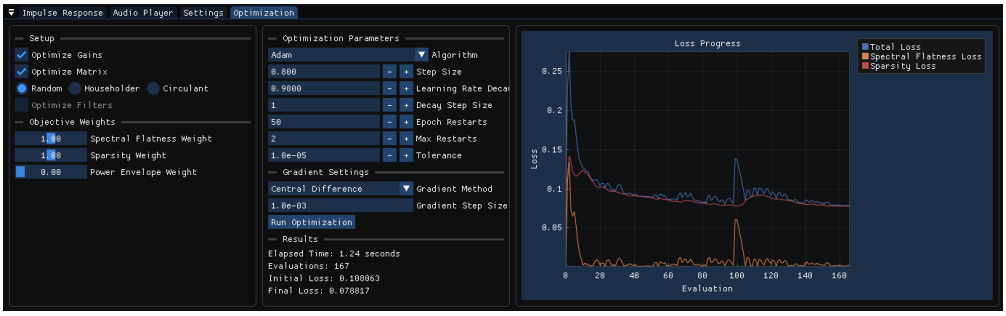


Figure 4.29 Screenshot of the optimization interface in the FDN Sandbox application.

Together, the sfFDN library, the optimization framework, and the FDN Sandbox application form a complete toolchain for designing, tuning, and evaluating FDNs in real time. The library

provides an efficient and extensible implementation whose performance was shown to be suitable for production use, the optimization framework was applied to both colorless reverberation and RIR matching, and the Sandbox application allows the resulting FDNs to be auditioned and their characteristics to be inspected.

Chapter 5

Conclusion

In this thesis, an optimization framework was developed to automatically tune the parameters of a real-time-capable FDN implementation. The framework was tested on two optimization tasks: reducing the coloration of an FDN and matching the characteristics of a target RIR. For the first task, loss functions based on spectral flatness and temporal sparsity were used to optimize the matrix coefficients, input gains, and output gains of the FDN. The results showed a significant reduction in coloration compared to non-optimized FDNs, with performance comparable to a recent method based on DDSP. The framework also accurately tuned the filters of an FDN to match the mel-scale EDR of a target RIR. Two attenuation filter configurations were tested: a 10-band GEQ with identical target RT_{60} values for all channels of the FDN, and 3-band filters for which the target RT_{60} values of each channel were optimized independently. For both configurations, the generated RIR matched the target RIR well, and the 3-band configuration yielded a better match than the 10-band GEQ configuration. Results were also comparable to or better than recent analysis-synthesis methods found in the literature.

5.1 Future Work

While the results presented in this thesis are promising, listening tests are still needed to evaluate the perceptual quality of the optimized FDNs, especially with regard to the RIR matching task. The

FDN topology used in this work was intentionally kept relatively simple to facilitate comparison with recent methods from the literature. In future work, more complex topologies could be explored. As the optimization framework effectively treats the FDN as a black box, it can be applied to any FDN topology, including those using time-varying components. The framework could also be extended to optimize other parameters, such as the delay lengths of the FDN or the coefficients of a filter feedback matrix.

Bibliography

Abel, J. S., & Huang, P. (2006). A simple, robust measure of reverberation echo density. *Proceedings of the 121st Audio Engineering Society Convention (AES)*, 1–10.

Agus, N., Anderson, H., Chen, J.-M., Lui, S., & Herremans, D. (2018). Perceptual evaluation of measures of spectral variance. *The Journal of the Acoustical Society of America*, *143*, 3300–3311.

Allen, J. B., & Berkley, D. A. (1979). Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America*, *65*, 943–950.

Bona, R., Fantini, D., Presti, G., Tiraboschi, M., Engel Alonso-Martinez, J. I., & Avanzini, F. (2022). Automatic parameters tuning of late reverberation algorithms for audio augmented reality. *Proceedings of the 17th Int. Audio Mostly Conference (AM '22)*, 36–43.

Chemistruck, M., Marcolini, K., & Pirkle, W. (2012). Generating matrix coefficients for feedback delay networks using genetic algorithm. *Proceedings of the 133rd Audio Engineering Society Convention (AES)*, 1–6.

Coggin, J., & Pirkle, W. (2016). Automatic design of feedback delay network reverb parameters for impulse response matching. *Proceedings of the 141st Audio Engineering Society Convention (AES)*, 1–10.

Curtin, R. R., Edel, M., Prabhu, R. G., Basak, S., Lou, Z., & Sanderson, C. (2021). The ensmallen library for flexible numerical optimization. *Journal of Machine Learning Research*, *22*, 1–6.

Dal Santo, G., Alary, B., Prawda, K., Schlecht, S., & Välimäki, V. (2024). RIR2fdn: An improved room impulse response analysis and synthesis. *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)*, 230–237.

Dal Santo, G., De Bortoli, G. M., Prawda, K., Schlecht, S. J., & Välimäki, V. (2025). FLAMO: An open-source library for frequency-domain differentiable audio processing. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5.

Dal Santo, G., Prawda, K., Schlecht, S. J., & Välimäki, V. (2023). Differentiable feedback delay network for colorless reverberation. *Proceedings of the 26th International Conference on Digital Audio Effects (DAFx23)*, 244–251.

Dal Santo, G., Prawda, K., Schlecht, S. J., & Välimäki, V. (2025). Optimizing tiny colorless feedback delay networks. *EURASIP Journal on Audio, Speech, and Music Processing*, 13.

Dattorro, J. (1997). Effect design, part 1: Reverberator and other filters. *Journal of the Audio Engineering Society*, *45*, 660–684.

Fagerström, J., Alary, B., Schlecht, S. J., & Välimäki, V. (2020). Velvet-noise feedback delay network. *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx2020)*, 219–226.

Gardner, W. G. (1995). Efficient convolution without input/output delay. *Journal of the Audio Engineering Society*, *43*, 127–136.

Gerzon, M. (1972). Synthetic stereo reverberation, part II. *Studio Sound*, *14*, 24–28.

Götz, G., Falcón Pérez, R., Schlecht, S. J., & Pulkki, V. (2022). Neural network for multi-exponential sound energy decay analysis. *The Journal of the Acoustical Society of America*, *152*, 942–953.

Hayes, B., Shier, J., Fazekas, G., McPherson, A., & Saitis, C. (2024). A review of differentiable digital signal processing for music and speech synthesis. *Frontiers in Signal Processing*, *3*, 1284100.

Heldmann, J., & Schlecht, S. J. (2021). The role of modal excitation in colorless reverberation. *Proceedings of the 24th International Conference on Digital Audio Effects (DAFx20in21)*, 206–213.

Ibnyahya, I., & Reiss, J. D. (2022). A method for matching room impulse responses with feedback delay networks. *Proceedings of the 153rd Audio Engineering Society Convention (AES)*, 1–8.

Johnston, J. D. (1988). Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*, *6*, 314–323.

Jot, J.-M. (1992). An analysis/synthesis approach to real-time artificial reverberation. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, *2*, 221–224.

Jot, J.-M. (1997). Efficient models for reverberation and distance rendering in computer music and virtual audio reality. *Proceedings of the International Computer Music Conference*, 1–8.

Jot, J.-M. (2015). Proportional parametric equalizers - application to digital reverberation and environmental audio processing. *Proceedings of the 139th Audio Engineering Society Convention (AES)*, 1–8.

Jot, J.-M., & Chaigne, A. (1991). Digital delay networks for designing artificial reverberators. *Proceedings of the 90th Audio Engineering Society Convention (AES)*, 1–16.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 1–15. https://arxiv.org/abs/1412.6980

Krokstad, A., Strøm, S., & Sørsdal, S. (1968). Calculating the acoustical room response by the use of a ray tracing technique. *Journal of Sound and Vibration*, *8*, 118–125.

Krokstad, A., Strøm, S., & Sørsdal, S. (1983). Fifteen years' experience with computerized ray tracing. *Applied Acoustics*, *16*, 291–312.

Kuttruff, H., & Vorländer, M. (2024). *Room acoustics* (7th ed.). CRC Press.

Martínez Ramírez, M. A., Wang, O., Smaragdis, P., & Bryan, N. J. (2021). Differentiable signal processing with black-box audio effects. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 66–70.

Mezza, A. I., Giampiccolo, R., & Bernardini, A. (2024). Modeling the frequency-dependent sound energy decay of acoustic environments with differentiable feedback delay networks. *Proceedings of the 27th International Conference on Digital Audio Effects (DAFx24)*, 238–245.

Mitchell, M. (1996). *An introduction to genetic algorithms*. The MIT Press.

Moorer, J. (1979). About this reverberation business. *Computer Music Journal*, *3*, 605–639.

Prawda, K., Välimäki, V., Willemsen, S., & Serafin, S. (2020). Flexible real-time reverberation synthesis with accurate parameter control. *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, 16–23.

Rocchesso, D., & Smith, J. O. (1997). Circulant and elliptic feedback delay networks for artificial reverberation. *IEEE Transaction on Speech and Audio Processing*, *5*, 51–63.

Rubak, P. (2005). Evaluation of artificial reverberation decay quality. *Proceedings of the 118th Audio Engineering Society Convention (AES)*, 1–11.

Rubak, P., & Johansen, L. G. (2003). Coloration in natural and artificial room impulse responses. *23rd International Conference: Signal Processing in Audio Recording and Reproduction*, 1–19.

Savioja, L., Rinne, T. J., & Takala, T. (1994). Simulation of room acoustics with a 3-d finite difference mesh. *International Conference on Mathematics and Computing*, 463–466.

Savioja, L., & Svensson, U. P. (2015). Overview of geometrical room acoustic modeling techniques. *The Journal of the Acoustical Society of America*, *138*, 708–730.

Scheibler, R., Bezzam, E., & Dokmanić, I. (2018). Pyroomacoustics: A python package for audio room simulations and array processing algorithms. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 351–355.

Schlecht, S. J. (2020). FDNTB: The feedback delay network toolbox. *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, 211–218.

Schlecht, S. J. (2021). Allpass feedback delay networks. *IEEE Transactions on Signal Processing*, *69*, 1028–1038.

Schlecht, S. J., & Habets, E. A. P. (2017a). Feedback delay networks: Echo density and mixing time. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *25*, 374–383.

Schlecht, S. J., & Habets, E. A. P. (2017b). On lossless feedback delay networks. *IEEE Transactions on Signal Processing*, *65*, 1554–1564.

Schlecht, S. J., & Habets, E. A. P. (2019). Modal decomposition of feedback delay networks. *IEEE Transactions on Signal Processing*, *67*, 5340–5351.

Schlecht, S. J., & Habets, E. A. P. (2020). Scattering in feedback delay networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *28*, 1–11.

Schroeder, M. R. (1962). Natural sounding artificial reverberation. *Journal of the Audio Engineering Society*, *10*, 219–223.

Schroeder, M. R. (1966). New method of measuring reverberation time. *The Journal of the Acoustical Society of America*, *40*, 428–433.

Schroeder, M. R., & Logan, B. F. (1961). "colorless" artificial reverberation. *Journal of the Audio Engineering Society*, *9*, 192–197.

Siltanen, S., Lokki, T., & Savioja, L. (2010). Rays or waves? Understanding the strengths and weaknesses of computational room acoustics modeling techniques. *Proceedings of the International Symposium on Room Acoustics*, 1–6.

Smith, J. O. (1985). A new approach to digital reverberation using closed waveguide networks. *Proceedings of the International Computer Music Conference*, 45–53.

Stautner, J., & Puckette, M. (1982). Designing multi-channel reverberators. *Computer Music Journal*, *6*, 52–65.

Stewart, G. W. (1980). The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM Journal on Numerical Analysis*, *17*, 403–409.

Väänänen, R., Välimäki, V., Huopaniemi, J., & Karjalainen, M. (1997). Efficient and parametric reverberator for room acoustics modeling. *Proceedings of the International Computer Music Conference*, 200–203.

Välimäki, V., Parker, J. D., Savioja, L., Smith, J. O., & Abel, J. S. (2012). Fifty years of artificial reverberation. *IEEE Transaction on Audio, Speech, and Language Processing*, *20*, 1421–1448.

Välimäki, V., Prawda, K., & Schlecht, S. J. (2024). Two-stage attenuation filter for artificial reverberation. *IEEE Signal Processing Letters*, *31*, 391–395.

Wefers, F. (2015). *Partitioned convolution algorithms for real-time auralization*. Logos Verlag Berlin GmbH.

Appendix A

Online code repository

Source code for the `sfFDN` library is available on Github at:

https://github.com/Segfault1602/sfFDN

Source code for the optimization framework is available on Github at:

https://github.com/Segfault1602/fdn_opt

Source code for the FDN Sandbox is available on Github at:

https://github.com/Segfault1602/FDNSandbox

Appendix B

Optimization results for colorless FDNs

B.1 Optimizer Types

The Adam optimizer was used to perform the optimization in the previous sections, but other optimizers were also tested. The following is a list of the optimizers that were tested and a brief description of how they work, specifically in the context of the *ensmallen* library used in this work.

Gradient Descent Gradient Descent is an optimization technique that iteratively updates the parameters in the direction of the negative gradient of the loss function with respect to the parameters. Specifically, the Momentum Delta-Bar-Delta variant of gradient descent was used, which incorporates momentum and adaptive learning rates to improve convergence.

Adam Adam is an optimization technique based on stochastic gradient descent that computes adaptive learning rates for each parameter. It is a widely used optimizer and can be found in popular machine learning libraries such as PyTorch and TensorFlow.

SPSA Simultaneous Perturbation Stochastic Approximation can be described as a variation of gradient descent optimization where the gradient is approximated using a stochastic approach.

Simulated Annealing Simulated Annealing is a probabilistic optimization technique that explores the search space by accepting both better and worse solutions with a certain probability that

decreases over time.

CNE Conventional Neural Evolution, also known as Genetic Algorithm, is an optimization technique inspired by the process of natural selection where a population of candidate solutions evolves over generations through selection, crossover, and mutation operations.

DE Differential Evolution is similar to CNE but with slightly different mutation and crossover operations.

PSO Particle Swarm Optimization is an optimization technique where candidate solutions, called particles, move through the search space influenced by their own best position and the best position of their neighbors.

L-BFGS Limited-memory Broyden-Fletcher-Goldfarb-Shanno (BFGS) is an optimization technique that determines the search direction using an approximation of the Hessian matrix of the loss function.

CMA-ES Covariance Matrix Adaptation Evolution Strategy is an optimization technique in the family of evolutionary algorithms that estimates a covariance matrix to guide the evolution of the population of candidate solutions.

These optimizers can further be categorized into two groups: gradient-based and gradient-free optimizers. The groups are defined in Table B.1.

Gradient-Based	Gradient-Free
Gradient Descent	SPSA
Adam	Simulated Annealing
L-BFGS	CNE
	DE
	PSO
	CMA-ES

Table B.1 Categorization of the optimizers used for the optimization of the FDN parameters.

B.2 Performance Comparison

Table B.2 shows the results of the optimization of colorless FDNs of size $N = 8$ using the different optimizers.

	Final Loss	Time (s)	Total Evaluations	
Adam	0.036	4.538	679	(108640)
Gradient Descent	0.037	2.204	302	(48320)
CMA-ES	0.038	4.318	11 375	
Differential Evolution	0.038	5.903	152 550	
Simulated Annealing	0.040	9.942	14 972	
L-BFGS	0.041	4.160	600	(96000)
PSO	0.044	2.672	46 795	
SPSA	0.046	0.156	327	
CNE	0.063	1.976	57 202	

Table B.2 Optimization results for colorless FDNs of size $N = 8$. The final loss is the value of $\mathcal{L}_{\text{color}}$ at the end of the optimization process. The time is the total time taken to perform the optimization. The total number of evaluations refers to the number of times the loss function was evaluated during the optimization process. The number in parentheses corresponds to the number of evaluations required for the approximation of the gradients for the gradient-based optimizers.