DIFFERENTIABLE FEEDBACK DELAY NETWORKS VIA CLOSED-FORM TIME-DOMAIN BACKPROPAGATION

Antoine SALMONA[1] and **Rémi MIGNOT**[1]

[1]**STMS Lab - IRCAM, Sorbonne Université, CNRS (UMR9912)**, Paris, France

ABSTRACT

The Feedback Delay Network (FDN) is a widely used structure for artificial reverberation, but its Infinite Impulse Response (IIR) nature makes differentiable training non-trivial. The dominant approach, Frequency Sampling (FS), approximates gradients in the frequency domain, introducing frame-independence and possibly temporal-aliasing artifacts that can be detrimental for reverberant systems. In this paper, we derive the exact backward pass of the FDN in the time domain as a closed-form recursive formula, including a frequency-dependent feedback matrix. Unlike FS, the proposed method computes exact gradients without approximation and preserves the true IIR dynamics across successive input segments via a memory buffer, enabling streaming processing of arbitrarily long sequences. We describe a full differentiable FDN implementation and evaluate it on the task of room impulse response matching, showing that the proposed method is significantly faster than FS for comparable performance and uniquely supports streaming processing, where FS fails in this setting.

1. INTRODUCTION

In the field of digital reverberation, the Feedback Delay Network (FDN) [1, 2] is nowadays one of the most widely used approaches for simulating artificial reverberation. The FDN is a linear delay-based digital recursive structure with Infinite Impulse Response (IIR) that encompasses parallel delay lines interconnected by a feedback mixing matrix. The FDN can be regarded as an evolutionary advancement of earlier digital artificial reverberation models, including Schroeder and Logan model [3], which combines parallel comb filters and serial all-pass filters, and later, Moorer's model [4] which introduces additional one-pole filters, thereby affording greater control over the reverberation time.

Differentiable Digital Signal Processing (DDSP) [5] refers to a family of methods which consists in reimplementing classic Digital Signal Processing (DSP) techniques in a differentiable manner, typically using an automatic differentiation framework [6] such as PyTorch or Tensorflow. This enables gradient computation for the backpropagation algorithm which allows optimizing the DSP module parameters and backpropagating the gradient to elements before the DSP module. The goal is the integration of domain knowledge from signal processing into audio-related deep learning methods. Over the last past years, DDSP has been applied to various audio tasks, including speech synthesis [7–9] instrument modelling [10–12], singing voice synthesis [13, 14], or filter design [15], just to name a few.

Developing efficient differentiable implementations of linear IIR systems is not as straightforward as for linear Finite Impulse Response (FIR) systems. Indeed naive implementations of IIR systems into an automatic differentiation framework, would result in deep computational graphs and slow down training speeds [16–18]. A commonly used approach to differentiate efficiently linear IIR systems is to evaluate their transfer functions in the Fourier domain on a finite number of frequency bins [19]. This is known as the Frequency Sampling (FS) method. This approach allows to directly obtain differentiable implementations of any linear IIR system as soon as an analytical expression of its transfer function is available. However, the FS method presents some potential drawbacks. First, each frame is processed independently, which results in the loss of the IIR system's nature, constituting a significant approximation. Second, frequency-domain processing may introduce time-aliasing artifacts that become particularly pronounced for slowly decaying impulse responses, as encountered in reverberation. Maintaining a relatively good approximation in these cases renders the method computationally inefficient.

A distinct approach for implementing IIR systems is to derive the closed-form solution of the gradients needed for the backpropagation algorithm in the time domain. In the context of filters, Bhattacharya et al. [20] have derived the instantaneous backpropagation algorithm for peak and shelving biquads. More general solutions have been derived for any IIR filter of any order by Forgione and Piga [21] and Yu and Fazekas [14]. They have shown notably that backpropagation through an all-pole filter can be implemented with two filtering passes of the same all-pole filter. More recently, Yu et al. [22] have generalized these results for time-varying all-pole filters, obtaining as a byproduct a more efficient implementation of the backward pass for time-invariant filters, requiring only one single filtering pass of the all-pole filter. We refer to these methods as the Time Domain (TD) methods. Unlike the FS method, TD methods compute the exact gradient of the IIR filter without approximation. A further advantage is that the filter state can be stored in a memory buffer and carried across successive segments of the input, so that

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

streaming processing is possible while preserving the true IIR dynamics of the system.

In this paper, we derive the exact time-domain backward pass of the FDN as a closed-form recursive formula including the frequency-dependent feedback matrix case, and we apply it to Room Impulse Response (RIR) matching and estimation. The rest of the paper is organized as follows. Section 2 recalls the FDN structure and its related parametrizations. Section 3 presents the derivation of the time-domain backward pass for the FDN and its extension to frequency-dependent feedback. Section 4 describes the tested FDN model along with the training setup used for the experiments. Finally, Section 5 reports experimental results demonstrating that this TD method is substantially faster than the FS method and enables streaming processing of arbitrarily long sequences.

2. BACKGROUND

2.1 Feedback delay network

A FDN with N channels is a recursive linear translation-invariant system, and the architecture of the Single-Input Single-Output (SISO) case is illustrated in Figure 1. It can be characterized by the following equations :

$$\begin{aligned} \mathbf{y}_{t+\mathbf{m}} &= \mathbf{A}\mathbf{y}_t + \mathbf{b}v_t, \\ w_t &= \mathbf{c}^\mathsf{T}\mathbf{y}_t + dv_t, \end{aligned} \tag{1}$$

where $v_t, w_t \in \mathbb{R}$ denote the input and output signals, respectively, at sample index t; $\mathbf{b}, \mathbf{c} \in \mathbb{R}^N$ are vectors of input and output gains, $d \in \mathbb{R}$ is a direct scalar gain, $\mathbf{A} \in \mathbb{R}^{N\times N}$ is referred to as the feedback matrix, and $\mathbf{m} = (m_1, \dots, m_N)^\mathsf{T} \in \mathbb{N}^N$ is a vector containing the lengths of the N delay lines in number of samples. The delay lines output is denoted by $\mathbf{y}_t \in \mathbb{R}^N$, and we use the notation $\mathbf{y}_{t+\mathbf{m}} = (y_{1,t+m_1}, \dots, y_{N,t+m_N})^\mathsf{T}$ for convenience. Note that Equation (1) corresponds to the case of the frequency-independent feedback matrix. The frequency-dependent case will be studied later in the paper. The transfer function of the FDN, expressed in terms of the complex variable z, is written

$$\mathsf{H}(z) = \mathbf{c}^\mathsf{T}(\mathbf{D}_\mathbf{m}(z)^{-1} - \mathbf{A})^{-1}\mathbf{b} + d, \tag{2}$$

where $\mathbf{D}_\mathbf{m}(z)$ is a diagonal matrix of size N, with diagonal $z^{-\mathbf{m}} = (z^{-m_1}, \dots, z^{-m_N})^\mathsf{T}$. The order of the system is the sum of the delay lengths $(m_i)_{1\le i\le N}$ [23].

2.2 Feedback matrix parametrization

A typical way to parametrize the feedback matrix $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{U}\mathbf{\Gamma}, \tag{3}$$

where $\mathbf{U}, \mathbf{\Gamma} \in \mathbb{R}^{N\times N}$ are respectively: an *orthogonal* matrix, and a diagonal matrix whose diagonal entries are all between 0 and 1. The matrix $\mathbf{U}$ is often referred to as the *mixing matrix* and $\mathbf{\Gamma}$ as the *attenuation matrix*. With this choice, the stability of the system is proven regardless of the choice of the delay vector $\mathbf{m}$, see [24]. Commonly adopted design choices encompass matrices such as the Hadamard matrix [25] or the Householder matrix [2].

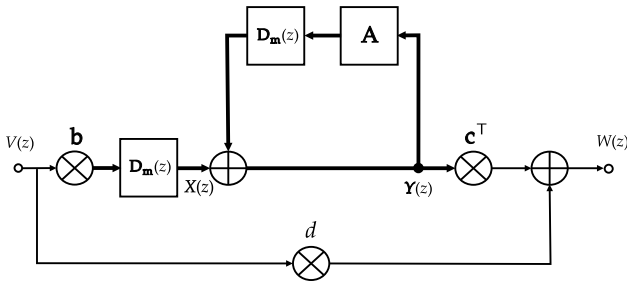


Figure 1: *Block diagram of a SISO FDN. Thin lines represent single-channel connections, while thick lines denote multichannel connections. Note that in the literature,* $\mathbf{D}_\mathbf{m}(z)$ *is most of the time factorized after the sum of the feedback loop, our formulation is equivalent but more convenient for the derivations that follow.*

In the homogeneous setting, the diagonal entries of the matrix $\mathbf{\Gamma}$ are delay-proportional absorption coefficients, i.e. $\mathbf{\Gamma} = \mathrm{diag}(\gamma^{-\mathbf{m}})$ where γ is referred to as the gain-per-sample ($0 \le \gamma \le 1$). With $\mathbf{U}$ orthogonal, γ controls the decay time from

$$20\log_{10}(\gamma) = \frac{-60}{f_\mathrm{s} T_{60}}, \tag{4}$$

where f_s denotes the sampling rate in Hz, while T_{60}, the reverberation time, is the time required for a 60 dB decay.

The feedback matrix can be made frequency-dependent via more complex parametrizations. A widely used approach is to rewrite it as

$$\mathbf{A}(z) = \mathbf{U}\mathbf{\Lambda}(z), \tag{5}$$

where $\mathbf{\Lambda}(z)$ is a diagonal matrix of *attenuation filters*—often parametrized by parametric or graphic equalizers with negative gains (in decibel scale)—enabling frequency-dependent control of T_{60} [26]. All-pass filters can also be inserted in the feedback loop to improve perceptual quality and reduce metallic resonances [27, 28].

2.3 Related works on differentiable FDNs

The majority of the differentiable FDN implementations relies on the Frequency Sampling (FS). Lee et al. [28] used such a differentiable FDN within a multi-scale spectral loss to train a neural network to estimate the reverberator parameters from reference RIRs or reverberant speech. Dal Santo et al. [29] applied the FS approach to optimize the FDN parameters using a spectral flatness loss designed to improve colorlessness; they later refined their method as in [30]. Nevertheless, this work does not aim to approximate a target response or reverberated signals. Mezza et al. [31] proposed a differentiable FDN for the RIRs approximation. The architecture of FDN is directly implemented in time-domain with an automatic differentiation framework, thus involving the unrolled computation graph for the backpropagation. The computation graph is made tractable by truncating the FDN output to a fixed window (up to the T_{60} of the target RIR) before evaluating the loss.

Their work has recently been extended to the Multiple-Input Multiple-Output (MIMO) case for modelling the responses of microphone arrays by Giampiccolo et al. [32]. Finally, let's mention the recent work of Götz et al. [33], where the FDN parameters are blindly estimated from reverberated speech. The proposed method relies on a variational autoencoder with separate speech and RIR encoders, a differentiable FDN (FS method), and a regression model that predicts the FDN parameters from the latent RIR representation inferred from the reverberated speech.

In contrast, the present work derives the exact backward pass of the FDN in the time domain as a closed-form recursive formula, circumventing first, the construction of a large unrolled computation graph and second, the frame-independence and temporal-aliasing limitations inherent to FS. This enables training on audio sequences of arbitrary length and processing in streaming mode (with state transfer between consecutive chunks) while preserving the true IIR dynamics of the model.

3. DIFFERENTIATING THE FDN FEEDBACK LOOP IN THE TIME DOMAIN

Because the differentiable implementations of the gains $\mathbf{b}$, $\mathbf{c}$, and d, and the delays of $\mathbf{D}_\mathbf{m}(z)$ are straightforward, here we only consider the subsystem containing the feedback loop. Therefore, let us define the following MIMO subsystem, illustrated in Figure 2 left, and characterized by

$$\mathbf{y}_{t+\mathbf{m}} = \mathbf{A}\mathbf{y}_t + \mathbf{x}_{t+\mathbf{m}}, \tag{6}$$

where $\mathbf{x}_t, \mathbf{y}_t \in \mathbb{R}^N$ denote, respectively, the multichannel input and output of this subsystem at time t. Observe that Equation (6) corresponds to the first row of Equation (1) where $\mathbf{x}_t = \mathbf{b}v_{t-\mathbf{m}}$. With $a_{i,j} \coloneqq (\mathbf{A})_{i,j}$ for all $1 \leq i, j \leq N$, Equation (6) can also be written as

$$y_{i,t} = \sum_{j=1}^{N} a_{i,j} y_{j,t-m_i} + x_{i,t}. \tag{7}$$

Let the matrices $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{N\times T}$ denote the input $(\mathbf{x}_t)_{0\leq t<T}$ and output $(\mathbf{y}_t)_{0\leq t<T}$ audio signals with N channels. Suppose that our objective is to backpropagate a differentiable loss function $\mathcal{L}$ through this system. During the backward pass, the system takes as input $\nabla_\mathbf{y}\mathcal{L}$, the gradient of $\mathcal{L}$ with respect to $\mathbf{y}$, whose entries are $(\nabla_\mathbf{y}\mathcal{L})_{i,t} = \partial\mathcal{L}/\partial y_{i,t}$. And it produces the gradients with respect to the sequence $\mathbf{x}$, $(\nabla_\mathbf{x}\mathcal{L})_{i,t} = \partial\mathcal{L}/\partial x_{i,t}$, and the matrix $\mathbf{A}$, $(\nabla_\mathbf{A}\mathcal{L})_{i,j} = \partial\mathcal{L}/\partial a_{i,j}$, for $0 \leq t < T$ and $1 \leq i, j \leq N$. The following subsections derive the closed forms of these two gradients.

3.1 Gradient with respect to the input sequence $\mathbf{x}$

Let us denote $\mathbf{H}(t) = [h_{i,j}(t)]_{1\leq i,j\leq N}$ the impulse response matrix of the MIMO system. $h_{i,j}(t)$ represents the i-th component of the output response vector at time t to an impulse in the j-th channel at time 0. Observe that for all $1 \leq i, j \leq N$ and all $0 \leq t < T$, by derivative of the convolution, we get

$$\frac{\partial y_{i,t}}{\partial x_{j,\tau}} = h_{i,j}(t-\tau). \tag{8}$$

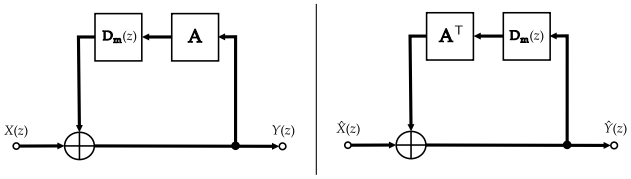


Figure 2: *Left: block diagram of the system applied during the forward pass (Equation* (6)*). Right: block diagram of the system used for the backward pass (Equation* (10)*).*

The chain rule gives then

$$\frac{\partial\mathcal{L}}{\partial x_{j,\tau}} = \sum_{i=1}^{N}\sum_{t=\tau}^{T-1} \frac{\partial\mathcal{L}}{\partial y_{i,t}} h_{i,j}(t-\tau). \tag{9}$$

The lower bound τ of the time sum is because of the causality of the system, $\mathbf{H}(t < 0) = 0$, and the upper bound is due the finite length of the sequence, $0 \leq t < T$.

To find an efficient computation of Equation (9), let us define the auxiliary MIMO system, illustrated in Figure 2 right and characterized by:

$$\hat{\mathbf{y}}_t = \mathbf{A}^\mathsf{T}\hat{\mathbf{y}}_{t-\mathbf{m}} + \hat{\mathbf{x}}_t, \tag{10}$$

where $\hat{\mathbf{x}}_t, \hat{\mathbf{y}}_t \in \mathbb{R}^N$ denote respectively the input and the output of this system. Denoting $\hat{\mathbf{H}}(t) = [\hat{h}_{k,l}(t)]_{1\leq k,l\leq N}$ its impulse response matrix, one can observe—see Appendix 8 for full proof—that the following identity holds for any $t < T$,

$$\hat{\mathbf{H}}^\mathsf{T}(t) = \mathbf{H}(t). \tag{11}$$

Plugging this in Equation (9) gives

$$\frac{\partial\mathcal{L}}{\partial x_{j,\tau}} = \sum_{i=1}^{N}\sum_{t=\tau}^{T-1} \frac{\partial\mathcal{L}}{\partial y_{i,t}} \hat{h}_{j,i}(t-\tau), \tag{12}$$

and with a matrix formulation, it can be simplified by

$$\nabla_{\mathbf{x}_\tau}\mathcal{L} = \sum_{t=0}^{T-\tau-1} \hat{\mathbf{H}}(t)\, \nabla_{\mathbf{y}_{t+\tau}}\mathcal{L}. \tag{13}$$

Similarly to [21], one can observe that Equation (13) corresponds to the cross-correlation operation between the input gradient sequence of the backward pass $\nabla_\mathbf{y}\mathcal{L}$ and the impulse-response $\hat{\mathbf{H}}(t)$ of the MIMO system defined in Equation (10). As in [21], this can be efficiently implemented in three steps: reverse the input gradient sequence $\nabla_\mathbf{y}\mathcal{L}$ in time, apply the system of Equation (10) — since time-reversal turns the cross-correlation into a convolution — and finally reverse the output in time again. This yields

$$\nabla_\mathbf{x}\mathcal{L} = \mathrm{FLIP}\left(\mathcal{T}_{\mathbf{A}^\mathsf{T}}\mathrm{FLIP}\left(\nabla_\mathbf{y}\mathcal{L}\right)\right), \tag{14}$$

where $\mathcal{T}_{\mathbf{A}^\mathsf{T}}$ denotes the linear operator applying the recursive system defined in Equation (10) and where $\mathrm{FLIP}(.)$ denotes the reversing-time operation. Note that this latter operation is trivial to compute using memory indexing. Finally, note that Equation (14) generalizes the results of SISO systems, see [14, 21], to the MIMO setting with the use of a *transposed* version for the backward pass.

3.2 Gradient with respect to the feedback matrix A

In [21] and [14], for the filter case, the gradient with respect to the filter coefficients is obtained by differentiating the recurrence equation, equivalent to Equation (7). This results in a backpropagation that requires a second pass through the filter. Applying the same approach in the FDN case would yield a recursion for $\partial y_{i,t}/\partial a_{k,l}$, which involves N^2 additional passes, for $1 \leq k, l \leq N$. To mitigate this computational complexity, we adopt an approach inspired by [22], in which only one pass is required for both gradients. Considering that a time-invariant system can be regarded as a time-varying system with parameters constant in time, we examine the convolution and the recursive formulations of such a time-varying system:

$$y_{i,t} = \sum_{j=1}^{N} \sum_{\tau} \tilde{h}_{i,j}(t,\tau) x_{j,\tau}, \tag{15}$$

$$= \sum_{j=1}^{N} \tilde{a}_{i,j,t} y_{j,t-m_i} + x_{i,t}, \tag{16}$$

where $\tilde{h}_{i,j}(t,\tau)$ are the impulse responses of this time-varying system, for an impulse at τ, and $\tilde{a}_{i,j,t}$ the time-varying coefficients of the feedback matrix, for all $1 \leq i, j \leq N$. First, considering a perturbation analysis, Equation (15) gives $\partial y_{i,t}/\partial y_{k,\tau} = \tilde{h}_{i,k}(t,\tau)$. Second, the derivation of Equation (16) provides $\partial y_{k,\tau}/\partial \tilde{a}_{k,l,\tau} = y_{l,\tau-m_k}$. And the chain rule gives

$$\frac{\partial y_{i,t}}{\partial \tilde{a}_{k,l,\tau}} = y_{l,\tau-m_k} \tilde{h}_{i,k}(t,\tau), \tag{17}$$

because only $\tilde{a}_{k,l,\tau}$ locally acts on $y_{k,\tau}$, see Equation (16). For causal systems, $\tilde{h}_{k,l}(t,\tau) = 0$ when $t < \tau$, the chain rule gives the loss derivatives

$$\frac{\partial \mathcal{L}}{\partial \tilde{a}_{k,l,\tau}} = \sum_{i=1}^{N} \sum_{t=\tau}^{+\infty} \frac{\partial \mathcal{L}}{\partial y_{i,t}} y_{l,\tau-m_k} \tilde{h}_{i,k}(t,\tau). \tag{18}$$

Remark that this formulation makes appear $\tilde{h}_{i,k}(t,\tau)$ which represents the dynamic of the system, and this will be used later for simplification. Now, going back to the time-invariant system

$$\frac{\partial \mathcal{L}}{\partial a_{k,l}} = \sum_{\tau=-\infty}^{+\infty} \frac{\partial \mathcal{L}}{\partial \tilde{a}_{k,l,\tau}} \tag{19}$$

$$= \sum_{\tau=m_k}^{T-1} y_{l,\tau-m_k} \sum_{i=1}^{N} \sum_{t=\tau}^{T-1} \frac{\partial \mathcal{L}}{\partial y_{i,t}} h_{i,k}(t-\tau),$$

where the sum limits have been changed to take into account limited sequences, for $1 \leq t < T$. One can remark the expression of $\partial \mathcal{L}/\partial x_{k,\tau}$, see Equation (9), leading to

$$\frac{\partial \mathcal{L}}{\partial a_{k,l}} = \sum_{\tau=m_k}^{T-1} y_{l,\tau-m_k} \frac{\partial \mathcal{L}}{\partial x_{k,\tau}}. \tag{20}$$

In summary, for any finite length audio sequence, we can calculate all the required derivatives for backpropagation with one single pass of the system defined in Equation (10), which is computationally equivalent to the forward pass, plus the multiplications in Equation (20).

3.3 Frequency-dependent feedback matrices

Suppose now that the feedback matrix is frequency-dependent, and parametrized as $\mathbf{A}(z) = \mathbf{U}\mathbf{\Lambda}(z)$, where $\mathbf{U} = [u_{k,l}]_{1 \leq k,l \leq N} \in \mathbb{R}^{N \times N}$ and $\mathbf{\Lambda}(z)$ is a diagonal matrix. Each diagonal entry $\lambda_k(z)$ corresponds to a linear system of parameters $\boldsymbol{\theta}_k$, and are usually made of filters.

Note that this parametrization encompasses both cases discussed in Section 2.2. In addition to the gradient with respect to the input $\nabla_{\mathbf{x}}\mathcal{L}$—which remains unchanged—the outputs of the backpropagation algorithm include, the gradients $\nabla_{\mathbf{U}}\mathcal{L}$ and $\nabla_{\boldsymbol{\theta}_k}\mathcal{L}$. The system is now characterized by the following recursive equation, for all $1 \leq i \leq N$,

$$y_{i,t} = \sum_{j=1}^{N} u_{i,j} s_{j,t-m_i} + x_{i,t}, \tag{21}$$

where $s_{j,t}$ is the output signal at time t of the system characterized by $\lambda_j(z)$ for the input $y_{j,t}$. The derivatives $\partial \mathcal{L}/\partial u_{k,l}$ can be obtained by applying the same reasoning as previously used for $\partial \mathcal{L}/\partial a_{k,l}$ in Section 3.2, leading to

$$\frac{\partial \mathcal{L}}{\partial u_{k,l}} = \sum_{t=m_k}^{T-1} s_{l,t-m_k} \frac{\partial \mathcal{L}}{\partial x_{k,t}}. \tag{22}$$

To compute the gradients $\nabla_{\boldsymbol{\theta}_k}\mathcal{L}$, we must run the backpropagation algorithm through the recursive system characterized by $\lambda_k(z)$. This requires, first, a differentiable implementation of this system, and, second, a closed-form expression for the gradient sequence $\nabla_{s_k}\mathcal{L}$, which serves as the input to the backward pass.

As in Section 3.2, first considering a perturbation analysis, we get $\partial y_{i,t}/\partial y_{j,\tau+m_j} = h_{i,j}(t-\tau-m_j)$, and second, the derivation of Equation (21) gives $\partial y_{j,\tau+m_j}/\partial s_{k,\tau} = u_{j,k}$. With the chain rule:

$$\frac{\partial y_{i,t}}{\partial s_{k,\tau}} = \sum_{j=1}^{N} u_{j,k} h_{i,j}(t-\tau-m_j). \tag{23}$$

Then, we obtain

$$\frac{\partial \mathcal{L}}{\partial s_{k,\tau}} = \sum_{i=1}^{N} \sum_{t=-\infty}^{+\infty} \frac{\partial \mathcal{L}}{\partial y_{i,t}} \frac{\partial y_{i,t}}{\partial s_{k,\tau}}, \tag{24}$$

$$= \sum_{j=1}^{N} u_{j,k} \sum_{i=1}^{N} \sum_{t=\tau+m_j}^{T-1} \frac{\partial \mathcal{L}}{\partial y_{i,t}} h_{i,j}(t-\tau-m_j),$$

$$= \sum_{j=1}^{N} u_{j,k} \sum_{i=1}^{N} \sum_{t=\tau+m_j}^{T-1} \frac{\partial \mathcal{L}}{\partial y_{i,t}} \hat{h}_{j,i}(t-(\tau+m_j)),$$

where the sum limits take into account the causality of the system and the finite sequences of duration T. Let us remark that $h_{i,j}(t)$ has been replaced by $\hat{h}_{j,i}(t)$, the responses of the transposed system, because Equation (11) still holds in this case. And for the third sum, with t, note that if $\tau + m_j \geq T$, its result is 0.

We now recognize the expression of $\partial \mathcal{L}/\partial x_{j,\tau+m_j}$ given by Equation (12) and so it follows, for all $0 \leq \tau < T$,

$$\frac{\partial \mathcal{L}}{\partial s_{k,\tau}} = \sum_{\substack{j \in [\![1,N]\!] \\ \tau+m_j<T}} u_{j,k} \frac{\partial \mathcal{L}}{\partial x_{j,\tau+m_j}}. \tag{25}$$

Once we got an expression of $\partial\mathcal{L}/\partial s_{k,\tau}$ we can run the backpropagation algorithm through the linear system characterized by $\lambda_k(z)$ to obtain $\nabla_{\boldsymbol{\theta}_k}\mathcal{L}$. For example, if the $\lambda_k(z)$'s are filters, the parameter gradient expression can be found in [14, 21]. In addition to the previously mentioned computational costs, the backward pass now requires N times the multiplications in Equation (25) (one for each index k), along with N backward passes through the systems $\lambda_k(z)$.

4. TESTED FDN MODEL

Here we present the implementation of the FDN model tested in Section 5. We parametrize the model with *differentiable input, output and direct gains* $\mathbf{b}$, $\mathbf{c}$ and d, along with a *differentiable frequency-dependent feedback matrix* $\mathbf{A}(z)$ of the form $\mathbf{A}(z) = \mathbf{U}\mathbf{\Lambda}(z)$, where $\mathbf{\Lambda}(z)$ is a diagonal matrix whose diagonal entries are the transfer functions of attenuation filters. This parametrization enables frequency-dependent control of the reverberation time. Finally, we *fix the delay lengths* $(m_i)_{1\leq i\leq N}$ as distinct logarithmically distributed co-prime numbers, as in [30], in order to obtain a smooth reverberation tail and avoid degenerate modal patterns.

4.1 Feedback matrix parametrization

For the mixing matrix $\mathbf{U}$, we adopt the same skew-exp parametrization as [29]. The matrix $\mathbf{U}$ is constrained to the set of orthogonal matrices, thereby ensuring unilosslessness regardless of the choice of the delay vector $\mathbf{m}$. At each optimization step, a *learnable weight matrix* $\mathbf{W} \in \mathbb{R}^{N\times N}$ is mapped to a skew-symmetric matrix via its upper triangular part $\mathbf{W}_{\mathrm{Tr}}$, and we set $\mathbf{U} = \exp(\mathbf{W}_{\mathrm{Tr}} - \mathbf{W}_{\mathrm{Tr}}^{\mathsf{T}})$. This mapping implicitly ensures orthogonality of $\mathbf{U}$ and can be used with standard gradient-based optimizers without creating spurious local minima [29]. Note that since the skew-exp parametrization is fully differentiable, the gradient with respect to $\mathbf{W}_{\mathrm{Tr}}$ can be computed via automatic differentiation.

Each attenuation filter, i.e. each diagonal entry of $\mathbf{\Lambda}(z)$, is the transfer function of a differentiable parametric EQ comprising K second-order biquad filters: one low-shelf filter, one high-shelf filter, and $K-2$ peak filters. To obtain differentiable IIR filters, we adopt the time-domain method proposed in [22]. Each filter is parametrized by *three optimizable parameters, namely the gain in dB, the centre frequency, and the quality factor*. We constrain the gain of each filter to be strictly negative (in dB) so as to ensure the stability of the system.

4.2 Summary of trainable parameters

The full set of trainable parameters of the model consists of: the upper triangular weights $\mathbf{W}_{\mathrm{Tr}}$ (from which $\mathbf{U}$ is derived via the skew-exp map), the input and output gain vectors $\mathbf{b}, \mathbf{c} \in \mathbb{R}^N$, the direct gain scalar $d \in \mathbb{R}$, and the parameters of the N attenuation filters, each comprising $3K$ scalar parameters (gain, centre frequency, and quality factor for each of the K biquad sections). The total number of trainable parameters is $N(N-1)/2 + 2N + 1 + 3KN$.

4.3 Parameters initialization

We initialize the mixing matrix weights $\mathbf{W}$ and the input and output gains $\mathbf{b}$, $\mathbf{c}$ in the same way as [29], i.e. by drawing from the normal distribution $\mathcal{N}(0, N^{-1})$. The direct gain d is initialized as 0, corresponding to a fully wet signal. Finally, for each i, we initialize the gain $g_{\mathrm{dB}}^{(i)}$ of the i-th channel attenuation filter as

$$g_{\mathrm{dB}}^{(i)} = -\frac{60 \cdot m_i}{f_{\mathrm{s}} T}, \tag{26}$$

where f_{s} is the sampling frequency and T is a hyperparameter of the model. The frequencies are initialized randomly using a uniform distribution on the mel scale.

4.4 Loss function

The training objective combines a multi-resolution STFT loss with a regularization term. The main term, inspired by [29], reads as

$$\begin{aligned}\mathcal{L}_{\mathrm{MR}}(y, y_{\mathrm{target}}) &= \mathcal{L}_{\mathrm{STFT}}(y, y_{\mathrm{target}}) \\ &\quad + \frac{1}{N}\sum_{k=1}^{N} \mathcal{L}_{\mathrm{STFT}}(y^k, y_{\mathrm{target}}),\end{aligned} \tag{27}$$

where $\mathcal{L}_{\mathrm{STFT}}$ denotes a multi-resolution STFT loss with ℓ_1 magnitude distance across multiple FFT sizes and hop lengths, y denotes the FDN output, y^k the k-th channel output, and y_{target} the target signal.

In addition, we add a regularization term proposed in [30] that encourages the echo density by controling the sparsity of the mixing matrix $\mathbf{U}$, thereby mitigating metallic resonances, a common pitfall of FDNs. It is given by

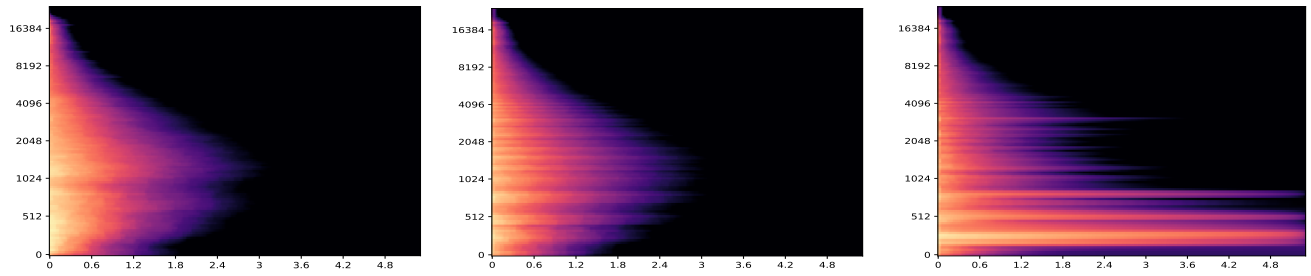
$$\mathcal{R}_{\mathrm{sparse}}(\mathbf{U}) = \frac{N\sqrt{N} - \sum_{ij}|\mathbf{U}_{ij}|}{N(\sqrt{N}-1)}. \tag{28}$$

The total loss is $\mathcal{L} = \mathcal{L}_{\mathrm{MR}} + \alpha\mathcal{R}_{\mathrm{sparse}}(\mathbf{U})$, where α is the regularization weight. In our experiments, we set $\alpha = 0.1$.

5. EXPERIMENTS

We evaluate our differentiable FDN on the task of RIR matching, following a setup inspired by [34]. In this experiment, the input excitation is formed from the first 50 ms of the target RIR ($v(t) = h_{\mathrm{RIR}}(t)$ for $0 \leq t < 50 \times 10^{-3}/f_{\mathrm{s}}$, and 0 elsewhere). This corresponds to the physical interpretation in which the early reflections feed the diffuse reverberant field. The FDN parameters (mixing matrix, gains, and attenuation filters) are then optimized so that the FDN output $\tilde{w}(t) = \mathrm{FDN}\{v(t)\}$ matches the full target response $w^*(t) = (v * h_{\mathrm{RIR}})(t)$. This effectively fits the model to the measured room's reverberation characteristics, according to the spectral loss $\mathcal{L}$. We compare the proposed TD method against the FS approach on this task.

In addition to this physical interpretation, using the first 50 ms of the RIR provides two practical benefits: first, it implicitly applies a frequency weighting adapted to the spectral content of the early reflections; second, it improves gradient-based optimization by distributing the excitation energy over time. While this slightly relaxes



(a) Target (b) Recursive (TD) (c) Frequency Sampling (FS)

Figure 3: Mel-frequency Energy Decay Reliefs (EDRs): target impulse response (a), FDN output with the proposed TD method (b), and FDN output with the FS method (c). Both FDNs were trained in streaming mode with chunks of duration 1 second.

the identifiability of the target impulse response, it provides a significant improvement in the conditioning of the gradient-based optimization. This has been confirmed by comparative experiments using a Dirac impulse as excitation, which resulted in less stable convergence.

5.1 Performance comparison

First, we compare the training performances of the proposed TD core method with the classic FS method. To do so, we perform the RIR matching task on IRs from the ChurchIR dataset [35] which includes 90 impulse responses acquired in the church of Saints Marcellino and Pietro in Cremona (Italy) using an omnidirectional microphone. The RIRs last approximately 7 seconds. To reduce computational cost, the RIRs were downsampled to a sampling rate of $f_\mathrm{s} = 24$ kHz. Our FDN implementation uses $N = 8$ delay lines. As described in Section 4.1, each channel is equipped with a parametric EQ composed of $K = 3$ filters (low-shelf, peak, and high-shelf) for frequency-dependent attenuation control. The mixing matrix $\mathbf{U}$ is parametrized using the skew-exponential method and the initial reverberation time parameter is set to $T = 2.0$ seconds. Training is performed using the Adam optimizer [36] with a learning rate of 0.1 for 50 iterations. The loss function combines a multi-resolution STFT loss and the sparsity regularization term described in Section 4.4. To ensure a fair comparison between the proposed TD method and the FS baseline, we use the same initialization for both methods on each target RIR. For the FS method, the number of frequency points is set equal to the total length of the sequence, since it corresponds roughly to the length of the impulse response. All experiments are run on CPU on the same machine (Dell XPS-13, 2020). Results can be found in Table 1.

Core	Time (s)	Memory (MB)	Final loss
Recursive (TD)	$\mathbf{34.89} \pm 0.98$	1078.6 ± 51.0	0.6901 ± 0.0585
Frequency Sampling (FS)	188.11 ± 2.75	1099.9 ± 56.4	0.6955 ± 0.0684

Table 1: FDN training benchmark on CPU. Values are mean $\pm$ standard deviation over 90 target RIRs with ten training runs (with different initialization) per RIR.

The TD method achieves approximately $\mathbf{5.4}\times$ faster training compared to FS, while maintaining comparable memory footprint and achieving similar final loss values. The significant speedup of the TD method can be attributed to two main factors. First, the TD approach processes audio sequences directly in the time domain with state continuity, avoiding the computational overhead of repeated FFT operations required by FS. Second, the efficient backward pass implementation described in Section 3 requires only one additional filtering operation, making it computationally comparable to the forward pass.

5.2 Streaming processing

We evaluate the proposed TD-FDN method in a streaming setup, where long sequences are processed in fixed-size non-overlapping chunks to limit memory use. The time-domain structure enables carrying the final state of each chunk over to the next as initial conditions, thereby preserving the IIR dynamics of the model. By contrast, in the FS method each chunk is processed independently of the others, and the number of frequency points is set equal to the chunk length. For this experiment, we use an impulse response recorded in the *espace de projection* (concert hall) at IRCAM with an omnidirectional microphone, at a sampling rate of $f_\mathrm{s} = 48$ kHz. We use the same FDN configuration as in the benchmark above, and the Adam optimizer with a learning rate of 0.1. The response is processed sequentially in 1 second chunks and the full sequence is repeated 100 times.

Figure 3 shows Energy Decay Reliefs (EDRs) [37] on a mel-frequency scale for the target RIR, the IR obtained with the proposed TD method, and the IR obtained with the FS method. Note that the streaming TD method involves two sources of approximations compared to full-sequence processing. First, during the forward pass of each chunk, the initial hidden state (i.e. the memory of the recursive system) is taken from the output of the *previous* chunk, which was computed with the model parameters *before* the latest gradient update. This introduces a discrepancy between the actual system state and the one that would result from the current parameters. Second, the gradients are computed independently for each chunk. This is a direct consequence of the time-reversal structure of Equation (14): since the backward pass requires running

the transposed system on the *time-reversed* gradient sequence, it cannot propagate gradient information beyond the boundaries of the current chunk. As a result, the contribution of each chunk to the loss is optimized in isolation. Despite these approximations, our experiments show that the streaming TD method converges to a satisfactory solution, whereas FS is fundamentally unable to model cross-chunk dependencies.

6. CONCLUSIONS

In this work, we derived the exact backward pass of the FDN in the time domain as a closed-form recursive formula and extended it to frequency-dependent feedback matrices. Unlike the Frequency Sampling approach, this yields exact gradients without approximation and avoids frame-independence and temporal-aliasing artifacts. On the task of RIR matching, experiments showed that the proposed method achieves a significant speedup over the FS baseline while enabling streaming processing, making it well suited to long sequences. Furthermore, we showed that this result generalizes earlier SISO backpropagation algorithms [14, 21] to the MIMO setting, with the notable appearance of the transposed feedback matrix in the backward pass.

A key limitation of the proposed method is its sequential nature: both the forward and backward passes process time steps one by one, preventing the data-parallel execution afforded by FFT-based methods. A promising direction is the *parallel scan* algorithm [38], which evaluates linear recurrences of the form $y_t = a_t y_{t-1} + x_t$ in $\mathcal{O}(\log T)$ parallel steps. While successfully applied to first-order scalar recurrences [39], extending this approach to FDNs remains, to the best of our knowledge, an open problem and a natural direction for future work.

Acknowledgments

This research was funded by the Agence Nationale de la Recherche (ANR), through the AQUA-RIUS project (ANR-22-CE23-0022).

The python / PyTorch implementation of this differentiable FDN is available in the *aqua-dfx* module at: `https://github.com/AQUA-RIUS/aqua-dfx`. The module also includes other differentiable audio effects, such as filters, dynamic range compression, with different methods and parameterizations.

7. REFERENCES

[1] M. A. Gerzon, "Synthetic stereo reverberation," *Studio Sound*, vol. 13, pp. 632–635, 1971.

[2] J.-M. Jot and A. Chaigne, "Digital delay networks for designing artificial reverberators," in *Proc. of the 90th Audio Eng. Soc. Conv.*, 1991, AES Paper 3030.

[3] M. R. Schroeder and B. F. Logan, ""Colorless" artificial reverberation," *The Journal of the Acoustical Society of America*, vol. 32, pp. 1520–1520, 1960.

[4] J. A. Moorer, "About this reverberation business," *Computer Music Journal*, pp. 13–28, 1979.

[5] J. Engel, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *International Conference on Learning Representations (ICLR)*, 2020.

[6] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," Facebook AI Research, Tech. Rep., 2017.

[7] J. J. Webber, C. Valentini-Botinhao, E. Williams, G. E. Henter, and S. King, "Autovocoder: fast waveform generation from a learned speech representation using differentiable digital signal processing," in *IEEE ICASSP*, 2023.

[8] K. Song, Y. Zhang, Y. Lei, J. Cong, H. Li, L. Xie, G. He, and J. Bai, "DSPGAN: A GAN-based universal vocoder for high-fidelity TTS by time-frequency domain supervision from DSP," in *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2023.

[9] A. Rao and P. K. Ghosh, "SFNet: A computationally efficient source filter model based neural speech synthesis," *IEEE Signal Processing Letters*, vol. 27, pp. 1170–1174, 2020.

[10] L. Renault, R. Mignot, and A. Roebel, "DDSP-Piano: A Neural Sound Synthesizer Informed by Instrument Knowledge," *Journal of the Audio Engineering Society (JAES)*, vol. 71, no. 9, pp. 552–565, sep 2023.

[11] F. Caspe, A. McPherson, and M. Sandler, "DDX7: Differentiable FM synthesis of musical instrument sounds," in *the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[12] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2022.

[13] H.-S. Choi, J. Yang, J. Lee, and H. Kim, "NANSY++: Unified voice synthesis with neural analysis and synthesis," in *International Conference on Learning Representations (ICLR)*, 2023.

[14] C.-Y. Yu and G. Fazekas, "Singing voice synthesis using differentiable LPC and glottal-flow-inspired wavetables," in *the 24th International Society for Music Information Retrieval Conference (ISMIR)*, 2023.

[15] J. T. Colonel, C. J. Steinmetz, M. Michelen, and J. D. Reiss, "Direct design of biquad filter cascades with deep learning by sampling random polynomials," in *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2022.

[16] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 2021.

[17] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, "Style transfer of audio effects with differentiable signal processing," *Journal of the Audio Engineering Society (JAES)*, vol. 70, no. 9, pp. 708–721, 2022.

[18] A. Wright and V. Valimaki, "Grey-box modelling of dynamic range compression," in *DAFx20in22*, 2022, pp. 304–311.

[19] A. Shaw, "Optimal design of digital iir filters by model-fitting frequency response data," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 42, no. 11, pp. 702–710, 1995.

[20] P. Bhattacharya, P. Nowak, and U. Zölzer, "Optimization of cascaded parametric peak and shelving filters with backpropagation algorithm," in *DAFx*, 2020, pp. 101–108.

[21] M. Forgione and D. Piga, "dynoNet: A neural network architecture for learning dynamical systems," *International Journal of Adaptive Control and Signal Processing*, vol. 35, no. 4, pp. 612–626, 2021.

[22] C.-y. Yu, C. Mitcheltree, A. Carson, S. Bilbao, J. Reiss, and G. Fazekas, "Differentiable all-pole filters for time-varying audio systems," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2024.

[23] D. Rocchesso and J. O. Smith, "Circulant and elliptic feedback delay networks for artificial reverberation," *IEEE Trans. on Speech and Audio Proc.*, vol. 5, no. 1, pp. 51–63, 1997.

[24] S. J. Schlecht and E. A. Habets, "On lossless feedback delay networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 6, pp. 1554–1564, 2016.

[25] J. Jullien, E. Kahle, M. Marin, and O. Warusfel, "Spatializer: a perceptual approach," in *Proc. of the 94th Audio Eng. Soc. Conv.*, 1993, AES Paper 3465.

[26] S. J. Schlecht and E. A. Habets, "Accurate reverberation time control in feedback delay networks," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2017.

[27] L. Dahl and J.-M. Jot, "A reverberator based on absorbent all-pass filters," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2000.

[28] S. Lee, H.-S. Choi, and K. Lee, "Differentiable artificial reverberation," *IEEE/ACM TASLP*, vol. 30, pp. 2541–2556, 2022.

[29] G. Dal Santo, K. Prawda, S. Schlecht, and V. Välimäki, "Differentiable feedback delay network for colorless reverberation," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2023.

[30] G. Dal Santo, K. Prawda, S. J. Schlecht, and V. Välimäki, "Optimizing tiny colorless feedback delay networks," *EURASIP Journal on Audio, Speech, and Music Processing*, no. 1, p. 13, 2025.

[31] A. I. Mezza, R. Giampiccolo, E. De Sena, and A. Bernardini, "Data-driven room acoustic modeling via differentiable feedback delay networks with learnable delay lines," *EURASIP Journal on Audio, Speech, and Music Processing*, no. 1, p. 51, 2024.

[32] R. Giampiccolo, A. I. Mezza, M. Pezzoli, S. Koyama, A. Bernardini, and F. Antonacci, "Modeling the impulse response of higher-order microphone arrays using differentiable feedback delay networks," in *Proc. of the Int. Conf. on Digital Audio Effects (DAFx)*, 2025.

[33] P. Götz, G. D. Santo, S. J. Schlecht, V. Välimäki, and E. Habets, "Matching reverberant speech through learned acoustic embeddings," in *IEEE ICASSP*, 2026.

[34] J. Lin, G. Götz, and S. J. Schlecht, "Deep room impulse response completion," *EURASIP Journal on Audio, Speech, and Music Processing*, no. 1, p. 20, 2025.

[35] R. Giampiccolo, S. Parrinelli, and F. Antonacci, "ChurchIR: A dataset of multichannel church impulse responses for spatial audio applications," in *33rd European Signal Processing Conference (EUSIPCO)*, 2025.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[37] J.-M. Jot, "An analysis/synthesis approach to real-time artificial reverberation," in *IEEE Int. Conf. on Acoustics, Speech and Signal Proc. (ICASSP)*, 1992.

[38] G. E. Blelloch, "Prefix sums and their applications," Carnegie Mellon University, Tech. Rep., 1990.

[39] J. T. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," in *ICLR*, 2023.

8. APPENDIX

Let consider the following MIMO recursive system

$$\mathbf{y}_{t+\mathbf{m}} = \mathbf{A}\mathbf{y}_t + \mathbf{x}_{t+\mathbf{m}}, \tag{29}$$

where $\mathbf{x}_T, \mathbf{y}_t \in \mathbb{R}^N$ denotes respectively the input and the output of the system at time t and $\mathbf{A} \in \mathbb{R}^{N\times N}$. This system admits a transfer function of the form,

$$\mathbf{H}(z) = \left(\mathbf{Id}_N - \mathbf{D}_\mathbf{m}(z)\mathbf{A}\right)^{-1}. \tag{30}$$

Now, consider the following MIMO system:

$$\hat{\boldsymbol{y}}_t = \mathbf{A}^\mathsf{T}\hat{\boldsymbol{y}}_{t-\mathbf{m}} + \hat{\boldsymbol{x}}_t. \tag{31}$$

This system admits as transfer function

$$\hat{\mathbf{H}}(z) = \left(\mathbf{Id}_N - \mathbf{A}^\mathsf{T}\mathbf{D}_\mathbf{m}(z)\right)^{-1}. \tag{32}$$

Observe that for all z, $\hat{\mathbf{H}}^\mathsf{T}(z) = \mathbf{H}(z)$ since $\mathbf{D}_\mathbf{m}(z)$ is diagonal. Finally, denoting $\mathbf{H}$ and $\hat{\mathbf{H}}$ the MIMO impulse response matrices of the recursive systems characterized by Equations (29) and (31), we directly get by inverse Fourier transform that for all $t \leq T$,

$$\hat{\mathbf{H}}^\mathsf{T}(t) = \mathbf{H}(t), \tag{33}$$

which concludes the proof.