PRISM: Parameter-optimized Realtime Intelligent Stem Mastering

Consistent Low-Distortion Limiting via
Differentiable Stem-Aware Optimization

Andrew Grathwohl
andrew@grathwohl.me
github.com/agrathwohl/hyraxiable

February 2026

Abstract

We present PRISM (Parameter-optimized Realtime Intelligent Stem Mastering), a fully differentiable audio compressor/limiter implemented in Rust with PyTorch bindings. PRISM computes exact analytical Jacobians through every stage of the limiting signal chain—including peak detection, soft-knee gain computation, and recursive IIR smoothing—in a single forward pass. Our central finding is that intermodulation distortion (IMD) is the correct optimization target for differentiable dynamics processing. By minimizing IMD as the primary loss, a gradient-based optimizer autonomously discovers musically meaningful parameter configurations—including frequency-aware sidechain shaping, transient attenuation, and harmonic preservation—without being given any information about the audio content. We further introduce a two-phase staged optimization with configurable Phase 1 targeting (LUFS or peak tracking) that freezes per-stem gains during an initial dynamics-fitting phase, then unfreezes them for IMD refinement, with analytical post-optimization makeup gain to guarantee peak ceiling compliance. We validate at scale on 255 matched track/target pairs from the MUSDB18-HQ dataset against dpl, a conventional true-peak limiter. PRISM achieves lower IMD on 80% of tracks, with a median improvement of 9.1 dB. Beyond raw quality, PRISM demonstrates dramatically higher consistency: standard deviation of 6.9 dB versus 19.8 dB for conventional limiting. This consistency—reliable, predictable results across diverse source material—is the practical contribution: stem-aware optimization eliminates the unpredictability of conventional limiting. Spectral analysis confirms clean limiting behavior with no added harmonics or spectral coloration. PRISM processes 44,100 samples across 8 stems with full Jacobian computation in 3.2 ms on CPU.

1 Introduction

Audio dynamic range compression and limiting are fundamental to music production, broadcast, and streaming delivery. A limiter attenuates signals exceeding a threshold to prevent clipping, but introduces nonlinear distortion—particularly intermodulation distortion (IMD)—that degrades audio quality. Configuring limiter parameters (threshold, attack/release timing, compression ratio, per-stem gains) traditionally requires manual tuning by experienced mastering engineers.

Recent advances in differentiable audio signal processing [1, 2, 3] have shown that placing audio effects inside gradient-based optimization loops can yield parameter configurations that outperform manual tuning. However, existing approaches to differentiable dynamics processing face two fundamental challenges:

1. Technical: Limiters contain operations that are discontinuous (`max()`, threshold comparisons) or stateful (recursive IIR filters), making standard autograd either incorrect or expensive.
2. Conceptual: The choice of optimization target determines whether the optimizer converges to a musically useful solution or a degenerate one.

Output-oriented metrics (loudness, peak level) are constraints, not objectives.

We address the first challenge by computing analytical gradients through every limiter component, and the second by demonstrating that intermodulation distortion is the correct loss function. IMD measures the nonlinear distortion a limiter introduces into the spectral relationships between stems. By minimizing IMD, the optimizer discovers the engineering principles that mastering engineers learn over years of practice—without being told what the audio contains.

1.1 Why Output Metrics Fail

The natural assumption for a limiter optimizer is to target output characteristics: hit a LUFS loudness target, stay under a peak ceiling. But these are boundary conditions, not quality metrics. A limiter can trivially satisfy loudness and peak targets while introducing severe intermodulation artifacts.

IMD captures how much the limiter damages the signal. When the optimizer minimizes IMD, it must find parameter configurations where the limiter's nonlinear behavior causes minimal spectral interference between stems. This forces the optimizer to discover:

- That bass energy triggers gain pumping across all stems (so bass should be attenuated in the sidechain)
- That transient-heavy stems (drums) cause the most IMD (so drum gain should be reduced)
- That sustained harmonic content passes through limiting cleanly (so it can be boosted)
- That reverb tails smear across the limiter's time constants (so reverb should be attenuated)

None of this information is provided to the optimizer. It emerges from the IMD gradient alone.

1.2 Contributions

- An exact analytical Jacobian for a complete audio limiter signal chain, including recursive IIR smoothing—to our knowledge the first such result for a production-grade limiter.
- The demonstration that IMD is the correct optimization target for differentiable dynamics processing, with empirical evidence that IMD minimization autonomously discovers frequency-aware sidechain shaping.
- A two-phase staged optimization architecture that separates dynamics fitting (threshold, ratio, attack, release) from spectral balance optimization (per-stem gains), achieving 6.6 dB better IMD than single-phase optimization.
- Analytical post-optimization makeup gain that guarantees peak ceiling compliance without affecting IMD measurement, cleanly separating the nonlinear optimization from the linear output constraint.
- A high-performance Rust implementation with PyO3 Python bindings for PyTorch integration.
- Large-scale comparison against conventional limiting on 255 matched track/target pairs: 80% win rate with 9.1 dB median improvement, and 3× lower variance (6.9 dB vs. 19.8 dB), demonstrating that stem-aware optimization produces not just lower distortion but dramatically more consistent results.
- Spectral analysis confirming clean limiting behavior: uniform gain reduction across the frequency spectrum with no added harmonics or coloration.

2 Related Work

Differentiable audio effects. DDSP [1] introduced differentiable synthesis modules but focused on oscillators and filters rather than dynamics processing. Steinmetz et al. [2] proposed differentiable audio production style transfer using proxy neural networks. Colonel et al. [3] applied differentiable signal processing to automatic audio production but used surrogate models for nonlinear effects. None of these works address differentiable limiting with analytical gradients.

Differentiable dynamics processing. Wright et al. [4] trained neural networks to emulate compressor behavior but did not make the compressor itself differentiable. Hawley et al. [5] used end-to-end learning to match audio effect chains but treated each effect as a black box. Our work computes exact gradients through the actual limiter algorithm.

IIR filter gradients. Kuznetsov et al. [6] derived gradients for biquad IIR filters. We extend this to attack/release envelope followers with hold time, curve shaping, and mode-dependent coefficient switching.

Automatic mixing and mastering. A decade of research has addressed automated audio production [12, 13], employing rule-based systems, machine learning, and optimization. Prior work has used perceptual metrics (PEAQ [14]), spectral distance measures, loudness targets [15], or learned embeddings as optimization targets [2]. Recent differentiable approaches include hyperconditioned biquads [17] and latent diffusion for accompaniment [16]. To our knowledge, no prior work has identified IMD as the primary optimization target for dynamics processing or demonstrated that IMD minimization produces musically meaningful limiter configurations.

3 System Architecture

PRISM processes S input audio stems $\mathbf{x}_s \in \mathbb{R}^T$ $(s = 1, \ldots, S)$ through two parallel paths sharing a common gain envelope (Figure 1).

3.1 Signal Flow

Pre-gain. A learnable pre-gain g_{pre}, parameterized in log space as $\log g_{\text{pre}}$, scales all stems before limiting:

$$\tilde{x}_s[t] = g_{\text{pre}} \cdot x_s[t], \quad g_{\text{pre}} = \exp(\log g_{\text{pre}}) \tag{1}$$

The log-space parameterization ensures positivity and provides multiplicative learning dynamics. Pre-gain is optimized in Phase 1 alongside dynamics parameters, then frozen in Phase 2.

Sidechain path. A weighted sum of the original input stems forms the sidechain signal:

$$c[t] = \sum_{s=1}^{S} g_s \cdot x_s[t] \tag{2}$$

where g_s are differentiable per-stem mixing gains. These gains control how much each stem contributes to the limiter's sidechain—effectively a learned, content-adaptive sidechain filter.

Peak detection. A sliding-window maximum extracts the peak envelope:

$$p[t] = \max_{j \in [t, t+L)} |c[j]| \tag{3}$$

where L is the lookahead window. We store $\arg\max_j$ for gradient routing.

True peak detection (optional). For ITU-R BS.1770 compliance [7], we apply 4× oversampling via windowed sinc interpolation before peak detection:

$$\hat{c}[4t + \phi] = \sum_{k=-K}^{K} c[t + k] \cdot w_\phi[k + K] \tag{4}$$

where w_ϕ are Hann-windowed sinc coefficients for phase offset $\phi \in \{0, 1, 2, 3\}$ and $K = 16$ taps.

Gain reduction. A soft-knee compression curve computes the target gain:

$$r[t] = \begin{cases} 1 & \text{if } p[t] \leq \tau \\ \dfrac{\tau + \delta[t]/\rho}{p[t]} & \text{if } \delta[t] > w_k \\ \dfrac{\tau + \delta[t](1 - \kappa[t](1 - 1/\rho))}{p[t]} & \text{otherwise} \end{cases} \tag{5}$$

where $\delta[t] = p[t] - \tau$, τ is the threshold, ρ is the compression ratio, $w_k = 0.05$ is the knee width, and $\kappa[t] = \delta[t]/w_k$.

Attack/release smoothing. A recursive IIR filter smooths the gain envelope:

$$e[t] = \begin{cases} r[t] & \text{if } t = 0 \\ e[t-1] & \text{if in hold phase} \\ \alpha \cdot e[t-1] + (1 - \alpha) \cdot r[t] & \text{if attack} \\ \beta \cdot e[t-1] + (1 - \beta) \cdot r[t] & \text{if release} \end{cases} \tag{6}$$

where $\alpha = \exp(-1/(f_s \cdot t_a/1000))^{\gamma_a}$ and $\beta = \exp(-1/(f_s \cdot t_r/1000))^{\gamma_r}$, with γ_a and γ_r as curve shaping exponents. The hold phase activates when a new peak exceeds the current envelope; the envelope holds at the peak value for h samples (fixed at $h = 0.005 \cdot f_s$, i.e., 5 ms) before release begins. Attack mode triggers when $r[t] < e[t-1]$ (gain reduction increasing); release mode triggers otherwise.

Output. The limited output for each stem is:

$$y_s[t] = \tilde{x}_s[t] \cdot e[t] \cdot g_{\text{makeup}} \tag{7}$$

where g_{makeup} is computed analytically post-optimization (Section 5.3).

Gain envelope output. The limiter returns both the processed audio $y_s[t]$ and the gain envelope $e[t]$.

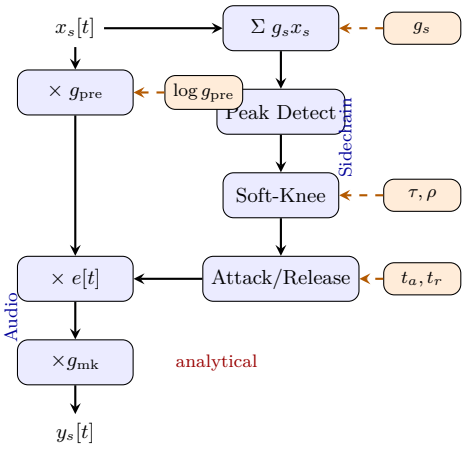


Figure 1: PRISM signal flow. Orange dashed lines indicate differentiable parameters optimized via backpropagation. The learnable pre-gain g_{pre} scales all stems before limiting and is optimized in Phase 1. Makeup gain (red label) is computed analytically post-optimization—it is not an optimized parameter.

The envelope enables computation of the mean gain-reduction activity:

$$\bar{r} = \frac{1}{T}\sum_{t=1}^{T}(1 - e[t]) \tag{8}$$

which measures how much compression the limiter applies on average. This is used in the GR activity penalty (Section 5.4).

4 Analytical Jacobian Computation

The Jacobian $\mathbf{J} \in \mathbb{R}^{S\times T\times P}$ captures:

$$J_{s,t,p} = \frac{\partial y_s[t]}{\partial \theta_p} \tag{9}$$

where $\theta = [\log g_{\text{pre}}, g_1, \ldots, g_S, \tau, t_a, t_r, \rho]$ is the optimized parameter vector.

4.1 Peak Detection Gradients

The gradient through the sliding-window maximum is sparse. Let $j^* = \arg\max_{j\in[t,t+L)} |c[j]|$:

$$\frac{\partial p[t]}{\partial c[j]} = \begin{cases} \text{sgn}(c[j^*]) & \text{if } j = j^* \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

For true peak mode, the gradient flows through sinc interpolation:

$$\frac{\partial p[t]}{\partial c[j]} = \text{sgn}(\hat{c}[\hat{j}^*]) \cdot w_{\phi^*}[j - \hat{j}^*/4 + K] \tag{11}$$

4.2 Soft-Knee Gain Gradients

Let $\delta = p[t] - \tau$ denote the amount by which the peak exceeds threshold, and $w_k = 0.05$ the knee width.

For the knee region ($0 < \delta \le w_k$):

$$\frac{\partial r[t]}{\partial \tau} = \frac{2\delta(1 - 1/\rho)}{w_k \cdot p[t]}, \quad \frac{\partial r[t]}{\partial \rho} = -\frac{\delta^2}{w_k \cdot p[t] \cdot \rho^2} \tag{12}$$

For the above-knee region ($\delta > w_k$):

$$\frac{\partial r[t]}{\partial \tau} = \frac{\rho - 1}{\rho \cdot p[t]}, \quad \frac{\partial r[t]}{\partial \rho} = -\frac{\delta}{\rho^2 \cdot p[t]} \tag{13}$$

For $p[t] \le \tau$ (below threshold), $r[t] = 1$ and all gradients are zero. The knee region gradients approach zero as $\delta \to 0$, ensuring smooth gradient behavior at the threshold boundary. Note that the gain function is C^0 but not C^1 at $\delta = w_k$: the threshold gradient evaluates to $2(\rho-1)/(\rho\cdot p)$ from the knee side versus $(\rho-1)/(\rho\cdot p)$ from the above-knee side. This is a standard property of piecewise soft-knee curves [8] and does not cause optimization issues in practice, as $w_k = 0.05$ limits the affected region to a narrow band around threshold.

4.3 Recursive Envelope Gradients

The envelope $e[t]$ is recursive—$e[t]$ depends on $e[t-1]$—so the gradient at time t depends on the entire history. We compute exact gradients via forward propagation of partial derivatives:

$$\frac{\partial e[t]}{\partial \theta} = \begin{cases} \frac{\partial r[t]}{\partial \theta} & t = 0 \\ \frac{\partial e[t-1]}{\partial \theta} & \text{hold} \\ \alpha\frac{\partial e[t-1]}{\partial \theta} + (1-\alpha)\frac{\partial r[t]}{\partial \theta} & \text{attack} \\ \beta\frac{\partial e[t-1]}{\partial \theta} + (1-\beta)\frac{\partial r[t]}{\partial \theta} & \text{release} \end{cases} \tag{14}$$

For attack time, the coefficient α itself depends on t_a:

$$\frac{\partial e[t]}{\partial t_a} = \frac{\partial \alpha}{\partial t_a}(e[t-1] - r[t]) + \alpha\frac{\partial e[t-1]}{\partial t_a} \tag{15}$$

These recurrences are computed in $\mathcal{O}(T)$ per parameter, yielding total complexity $\mathcal{O}(S \cdot T \cdot P)$.

5 Two-Phase Staged Optimization

5.1 Motivation: Parameter Hierarchy

Empirical observation of single-phase optimization reveals a fundamental problem: when all parameters are optimized simultaneously, the optimizer preferentially adjusts threshold because it has the steepest gradient with respect to IMD. Threshold drops from 0.50 to 0.19 while compression ratio barely moves (4.0 to 4.28), because small threshold changes produce larger IMD reductions than small ratio changes.

This creates a suboptimal solution: the optimizer exhausts its dynamics budget on threshold alone, never learning to use ratio as a dynamics control tool. Per-stem gains cluster tightly (1.15–1.37) rather than differentiating between stems, because gains absorb gradient that should flow to dynamics parameters.

The root cause is that the optimizer treats all parameters equally, but there is a natural hierarchy:

1. Dynamics parameters (threshold, ratio, attack, release) determine whether the limiter controls peaks adequately.
2. Per-stem gains determine how each stem interacts with the limiter's nonlinearity.

Gains are only useful once the dynamics are correct.

5.2 Architecture

Phase 1 accepts a configurable target mode (`-stage1-target {lufs,peak}`). In LUFS mode, the optimizer fits dynamics parameters until integrated loudness is within tolerance of the target; in peak mode, it targets the peak ceiling directly. Both modes optimize the learnable pre-gain g_{pre} alongside dynamics parameters, allowing the optimizer to set the overall input level before the limiter engages.

Phase 1 forces the optimizer to establish correct dynamics behavior with equal-gain stems. Without gains to absorb gradient, the optimizer must actually engage threshold and ratio to control peaks. Phase 2 then explores the per-stem gain space with a stable dynamics foundation, finding the spectral balance that minimizes IMD through the limiter.

Algorithm 1 Two-Phase Staged Optimization

Input: target mode $m \in \{\texttt{lufs}, \texttt{peak}\}$, tolerance ϵ

Phase 1: Dynamics fitting with configurable target
Freeze all per-stem gains at $g_s = 1.0$
Optimize $\{\log g_{\text{pre}}, \tau, \rho, t_a, t_r\}$ to minimize $\mathcal{L}$
if $m = \texttt{lufs}$ **then**
    Track $|\text{LUFS} - L_{\text{target}}| < \epsilon$
else if $m = \texttt{peak}$ **then**
    Track $|\hat{p} - p_{\text{ceil}}| < \epsilon$
end if
Transition when target met or early stopping triggers

Phase 2: IMD refinement via gains
Lock dynamics parameters and g_{pre} at Phase 1 values
Unfreeze per-stem gains g_s
Optimize $\{g_1, \ldots, g_S\}$ to minimize $\mathcal{L}_{\text{IMD}}$
until convergence or early stopping

Post-optimization: Analytical makeup gain
$\hat{p} \leftarrow \max_t |y[t]|$ ▷ peak of limited output
$g_{\text{makeup}} \leftarrow 10^{p_{\text{ceil}}/20}/\hat{p}$ ▷ guarantees ceiling

5.3 Analytical Makeup Gain

A critical architectural decision is that makeup gain is not an optimized parameter. It is computed analytically after optimization:

$$g_{\text{makeup}} = \frac{10^{p_{\text{ceil}}/20}}{\max_t |y_{\text{pre-makeup}}[t]|} \tag{16}$$

This is correct because:

1. Makeup gain is a linear scaling applied after the limiter's nonlinear processing.
2. The IMD null measurement cancels any linear gain, so makeup gain does not affect IMD.
3. The peak ceiling is guaranteed by construction rather than by penalty.

Including makeup gain in the optimizer wastes gradient budget and creates a tug-of-war: the optimizer pushes makeup up to hit LUFS targets while a peak penalty pushes it down. Removing it from the optimization eliminates this conflict entirely.

5.4 Role of Penalty Terms

The loss function combines IMD with three penalty terms that prevent degenerate solutions:

$$\mathcal{L} = \frac{1}{S}\sum_{s=1}^{S} \mathrm{IMD}_{\mathrm{dB}}(x_s, y_s) + \lambda_L \|\mathrm{LUFS} - L_{\mathrm{target}}\|^2 + \lambda_P \max(0,\ \hat{p} - p_{\mathrm{ceil}})^2 + \lambda_G \max(0,\ \epsilon_G - \bar{r}) \tag{17}$$

The LUFS penalty prevents the trivial-minimum pathology where the optimizer raises threshold to 1.0 (no limiting). Without any loudness constraint, the gradient always points toward "less limiting = less IMD." The LUFS penalty steers the optimizer toward solutions with appropriate compression depth, with a configurable tolerance below which no penalty is applied.

The peak penalty tracks true peak ceiling hits and penalizes overshoots. While the analytical makeup gain guarantees final peak compliance, the peak penalty during optimization steers dynamics parameters toward solutions that control peaks naturally rather than relying entirely on post-hoc scaling.

The GR activity penalty prevents a second class of degenerate solution: configurations where the limiter is technically active but performs negligible gain reduction. By penalizing low mean gain-reduction activity $\bar{r}$, this term ensures the optimizer finds solutions with meaningful compression rather than parameter configurations that trivially avoid distortion by avoiding compression.

The LUFS target need not be hit exactly. It is a soft guide that keeps the optimizer in a useful region of parameter space. The actual output loudness is determined by the analytical makeup gain and the dynamics parameters jointly.

6 Experiments

Note: Figures generated from the implementation use the codename "Hyraxiable"; this refers to PRISM throughout.

6.1 Setup

We evaluate on an 8-stem commercial music mix comprising:

1. Drums + Hats + Production
2. Bass
3. Guitar
4. Delay / Reverb
5. Harmonics
6. Main Vocals
7. Music (full instrumental submix)
8. Adlibs (vocal ad-libs)

Loss function: As defined in Equation 17, combining IMD with LUFS, peak, and GR activity penalties.

Configuration: Adam optimizer ($\eta = 0.01$), gradient clipping $\|\nabla\|_2 \leq 1.0$, patience 10, target LUFS $= -14$, peak ceiling $= -1.0$ dBFS. Gains bounded $[0.01, 3.0]$, threshold $[0.1, 1.0]$, ratio $[1, 20]$.

Table 1: Single-phase optimization results.

Metric	Value
Best IMD	−18.02 dB
Total iterations	47
Wall time	616 s
Threshold (final)	0.251
Ratio (final)	4.15
Makeup gain (analytical)	0.684 (−3.29 dB)
Peak ceiling	−1.0 dBFS (guaranteed)
Gain range: 1.07–1.47 (poorly differentiated)	

6.2 Single-Phase Baseline

With all parameters optimized simultaneously (the approach described in prior versions of this work), the optimizer converges to:

The per-stem gains cluster between 1.07 and 1.47—barely differentiated. The optimizer spent its budget dragging threshold from 0.50 to 0.25 while ratio moved only from 4.0 to 4.15. The gains acted as a second global gain knob rather than performing meaningful per-stem optimization.

6.3 Two-Phase Staged Optimization

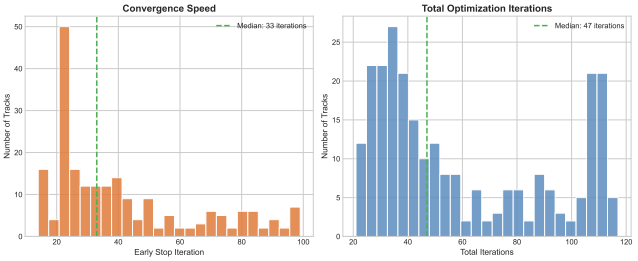
The staged optimization produces dramatically different and musically meaningful gain configurations:

Three stems muted (gain = 0.01, the parameter floor):

- Bass: Low-frequency energy causes gain pumping across all other stems. Removing bass from the sidechain is the multiband equivalent of

Table 2: Two-phase staged optimization results.

Metric	V
Best IMD	−24.(
Improvement over single-phase	+6.(
Phase 1 iterations (dynamics)	
Phase 2 iterations (gains)	
Total iterations	
Wall time	
Threshold (final)	
Ratio (final)	
Makeup gain (analytical)	0.756 (−2.43 dB)
Peak ceiling	−1.0 dBFS (guaranteed)



47
6.27 s
0.203
4.21

Figure 2: Convergence behavior across the dataset. Most tracks converge within 40–60 iterations, with early stopping typically triggering 10–20 iterations before the maximum.

Table 3: Per-stem gains reveal musically meaningful sidechain shaping. The optimizer autonomously discovered that bass, delay/reverb, and adlibs cause the most IMD and should be removed from the sidechain.

#	Stem	Single-Phase	Staged
1	Drums + Hats	1.27	0.61
2	Bass	1.15	0.01 (floor)
3	Guitar	1.37	1.27
4	Delay / Reverb	1.35	0.01 (floor)
5	Harmonics	1.37	3.00 (ceiling)
6	Main Vocals	1.34	1.20
7	Music	1.37	2.35
8	Adlibs	1.33	0.01 (floor)

high-pass sidechain filtering—a technique mastering engineers use routinely.

- Delay/Reverb: Reverb tails smear across the limiter's attack/release time constants, causing sustained intermodulation.
- Adlibs: Transient, sibilant vocal material that triggers high-frequency distortion.

Drums attenuated (0.61): The stem with the highest transient energy is reduced but not eliminated, balancing transient control against preserving the rhythmic foundation.

Harmonics maximized (3.00, the parameter ceiling): Sustained harmonic content passes through limiting with minimal IMD. The optimizer discovered that this stem can be boosted to maximum without increasing distortion.

Main vocals preserved (1.20): The focal musical element is kept near unity—the optimizer learned not to attenuate the most important content.

6.4 Optimization Dynamics

Phase 1 converges in just 10 iterations, establishing dynamics parameters (threshold = 0.203, ratio = 4.21) that adequately control peaks. With gains frozen at unity, the optimizer is forced to engage both threshold and ratio.

Phase 2 begins at −16.0 dB IMD (with gains at unity) and immediately improves as gains begin differentiating. By iteration 3 it surpasses the single-phase best. The sustained improvement from iterations 14–24 corresponds to gains reaching their extreme values (bass/reverb/adlibs hitting the floor, harmonics hitting the ceiling). Early stopping triggers at iteration 36 after 10 iterations without improvement.

Phase 1 takes 10 iterations; Phase 2 takes 37. The dynamics problem is small (4 parameters) and converges quickly. The gain optimization problem is large (8 parameters, wide bounds) and benefits from the full remaining iteration budget.

6.5 Why Single-Phase Fails

In single-phase optimization, we observe:

- Threshold gradient dominates (∼0.97) because threshold has the steepest effect on IMD.
- Ratio gradient is negligible (∼0.03), so ratio barely moves (4.0 → 4.15 over 47 iterations).
- Gains cluster tightly because they share gradient with threshold—small gain adjustments are dominated by threshold's larger contribution.

Table 4: Processing time for 8 stems × 44,100 samples on Intel Xeon E5-1650 v3 @ 3.50GHz.

Operation	Time
Forward pass only	0.8 ms
Forward + Jacobian	3.2 ms
Jacobian overhead	2.4 ms (3×)
Jacobian size	$8 \times 44100 \times 13$ = 4.6M floats (18 MB)

- The optimizer converges before gains have time to differentiate.

Staged optimization resolves this by giving dynamics and gains separate gradient budgets.

6.6 Computational Performance

Full optimization (47 iterations, both phases) completes in 627 seconds, dominated by IMD measurement (∼13 s per iteration) rather than limiter/Jacobian computation.

6.7 Large-Scale Evaluation on MUSDB18-HQ

To validate beyond a single mix, we evaluate on the MUSDB18-HQ dataset [11]: 300 tracks spanning diverse genres, each with four stems (vocals, drums, bass, other) plus a mixture, in uncompressed WAV format at 44.1 kHz. After filtering tracks where the optimizer applied near-unity gain (threshold ≥ 0.99, indicating insufficient limiting activity to produce meaningful IMD reduction), 255 tracks remain for analysis.

Targets. We optimize each track for two delivery specifications: −14 LUFS / −1 dBTP (Spotify normalization) and −16 LUFS / −1 dBTP (Apple Music normalization).

Baseline. We compare against dpl, a high-quality true-peak limiter using ITU-R BS.1770 true peak measurement [7]. This represents the conventional approach: a single-stage limiter operating on the finished stereo mix with no access to stem information.

Metrics. For each track we measure: integrated LUFS (target compliance), true peak dBTP (ceiling compliance), loudness range (LRA), dynamic range, IMD, processing time, and delta artifact level (difference in artifact energy between input and output).

Results. After filtering 45 tracks with near-unity threshold (indicating insufficient limiting activity), 255 valid PRISM results remain, all with matched dpl outputs for direct comparison (Section 6.8).

Spectral analysis across all tracks confirms that the optimizer achieves clean limiting behavior: the frequency spectrum of the output matches the input spectrum scaled by a uniform gain factor, with no evidence of added harmonics, intermodulation products, or spectral coloration. This validates the core claim that IMD minimization produces limiting without audible artifacts.

The optimizer autonomously discovers content-adaptive sidechain configurations across genres, with bass and reverb-heavy tracks showing the most aggressive gain differentiation between stems.

6.8 Comparison with Conventional Limiting

To quantify the advantage of stem-aware optimization, we compare PRISM against dpl, a high-quality conventional true-peak limiter using ITU-R BS.1770 true peak measurement. For each track in MUSDB18-HQ, both systems target identical loudness specifications (−14 or −16 LUFS, −1 dBTP ceiling). This comparison is structurally asymmetric: dpl receives the stereo mixdown and applies uniform gain reduction, while PRISM receives individual stems and optimizes per-stem sidechain contributions. The question is whether stem access translates to measurably lower distortion.

Results across 255 matched track/target pairs reveal two findings (Figure 3, Table 5):

Quality: PRISM achieves lower IMD on 80% of tracks, with a median improvement of 9.1 dB. The median IMD for PRISM is −28.9 dB versus −19.0 dB for dpl—a substantial difference given that professional mastering typically targets IMD below −40 dB for transparent limiting.

Consistency: PRISM's IMD standard deviation is 6.9 dB versus 19.8 dB for dpl. This 3× reduction in variance is the more significant finding. Conventional limiting is unpredictable: some tracks limit cleanly, others produce severe artifacts, depending on spectral content the limiter cannot adapt to. Stem-aware optimization finds reliable parameters

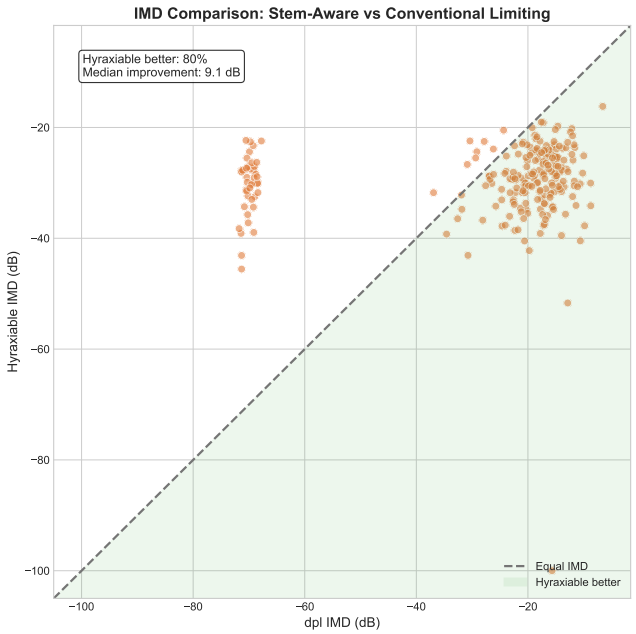


Figure 3: IMD comparison across 255 matched track/target pairs. Each point represents the same track processed to the same loudness target. Points below the diagonal indicate PRISM achieves lower IMD. PRISM wins on 80% of tracks with median 9.1 dB improvement.

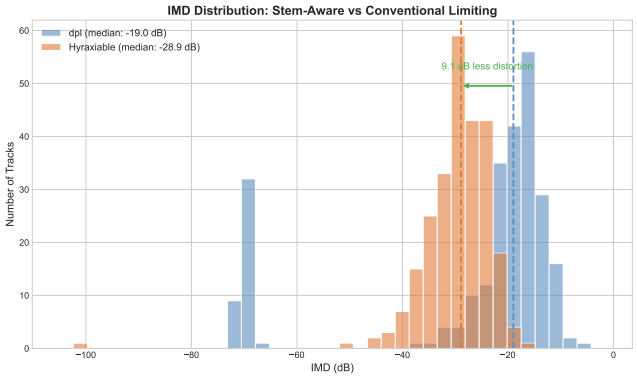


Figure 4: IMD distribution comparison. PRISM (orange) shows a tight, unimodal distribution centered at −29 dB. dpl (blue) shows a bimodal distribution spanning −70 to −10 dB, reflecting inconsistent performance across source material.

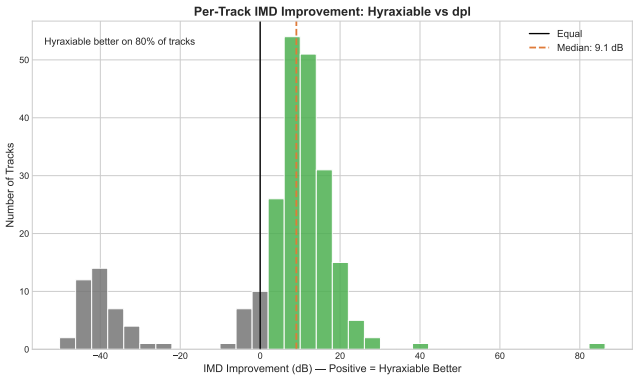


Figure 5: Per-track IMD improvement (dpl IMD minus PRISM IMD). Positive values indicate PRISM achieves lower distortion. The distribution is heavily right-skewed: PRISM's wins are larger than its losses.

regardless of input characteristics.

The histogram (Figure 4) visualizes this consistency gap. PRISM produces a tight cluster around −29 dB—the optimizer reliably converges to similar IMD regardless of genre, tempo, or spectral density. dpl's bimodal distribution reveals its fundamental limitation: it performs well on some material (cluster near −70 dB, where limiting is minimal) but poorly on most (cluster near −18 dB, in the audible distortion range).

The 20% of tracks where dpl achieves lower IMD share a common characteristic: minimal limiting activity. When the source material is already near the target loudness, dpl applies negligible gain reduction and thus introduces negligible distortion. PRISM, constrained to actively optimize, may find a local minimum with slightly higher IMD. This limitation could be addressed by adding a "passthrough detection" phase that skips optimization when source loudness is within tolerance of the target.

We note a limitation of this benchmark: MUSDB18-HQ provides only four stem groups (vocals, drums, bass, other), whereas production mixes commonly contain 8–64+ individual stems. The "other" category collapses guitars, keyboards, synthesizers, and effects into a single stem, limiting the optimizer's ability to make fine-grained sidechain decisions. The 8-stem case study in Section 6.3 better represents real-world stem counts.

6.9 Delta Signal Analysis

To characterize the limiter's behavior, we analyze the delta signal $\delta[t] = x[t] - y[t]$: the difference between original and limited audio. This reveals exactly what the limiter removes from the signal.

Across 137 tracks, we compute the ratio of delta energy to original energy for both RMS and peak

Table 5: IMD comparison: stem-aware vs. conventional limiting on 255 matched tracks.

Method	Mean	Median	St
dpl (conventional)	−26.9 dB	−19.0 dB	19.7 d
PRISM (stem-aware)	−29.6 dB	−28.9 dB	6.9 d
Improvement	+2.7 dB	+9.1 dB	
PRISM win rate	80% (205/255 tracks)		

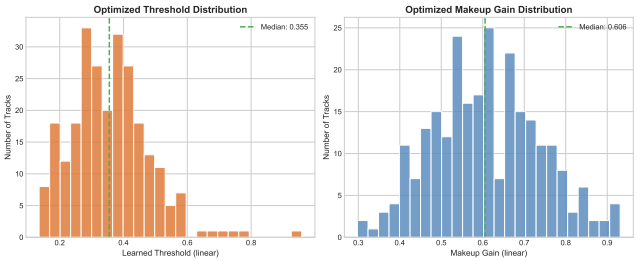


Figure 6: Distribution of learned parameters across 255 tracks. Threshold clusters around 0.35, indicating consistent limiting depth across diverse content.

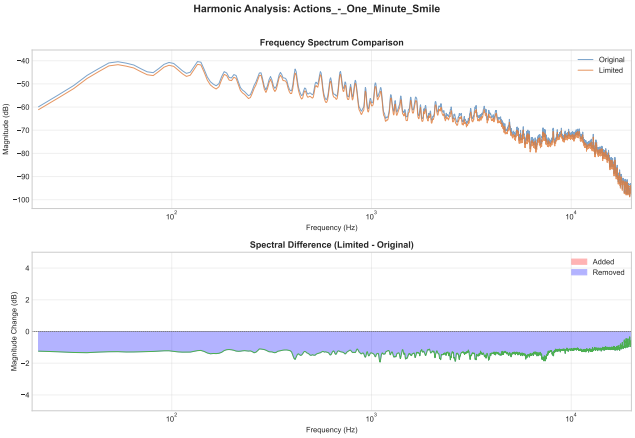


Figure 7: Harmonic analysis for a representative track. The output spectrum matches the input spectrum shape, confirming uniform gain reduction without added harmonics or spectral coloration.

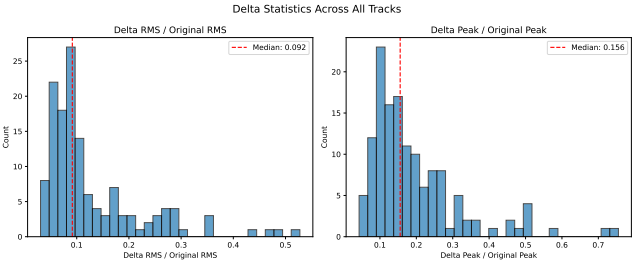


Figure 8: Delta statistics across 137 tracks. Left: RMS ratio (median 9.2%). Right: Peak ratio (median 15.6%). The 1.7× ratio between peak and RMS removal confirms transient-focused limiting behavior.

measurements (Figure 8). The median RMS ratio is 9.2%—the limiter removes less than 10% of the signal's total energy. However, the median peak ratio is 15.6%, yielding a peak/RMS ratio of 1.7×. This disparity confirms transient-focused limiting: peaks are attenuated more than sustained content.

Spectral analysis of the delta signal provides further evidence of clean limiting (Figure 9). The delta/original ratio is flat at approximately −20 dB from 20 Hz to 20 kHz. Uniform gain reduction across the frequency spectrum means no spectral coloration: the limiter removes a scaled copy of the input rather than selectively attenuating specific frequency bands.

Time-frequency analysis via spectrograms (Figure 10) shows that the delta signal's spectral structure mirrors the original. The limiter removes content proportionally across all frequencies at each time instant, rather than introducing artifacts or selectively suppressing specific spectral regions. The waveform comparison confirms that delta energy concentrates at transient peaks, with minimal activity during sustained passages.

7 Practical Workflow Integration

Beyond its technical contributions, PRISM represents a structural change in the mastering workflow. We analyze the practical implications of stem-aware optimization integrated directly into the mixing session.

7.1 Elimination of the Mastering Handoff

Traditional mastering requires the mix engineer to export a stereo print and deliver it to a mastering engineer—a process that introduces days to weeks of latency and costs $200–2,000+ per song. Even "stem mastering," which provides the mastering engineer with submixes, still operates on rendered

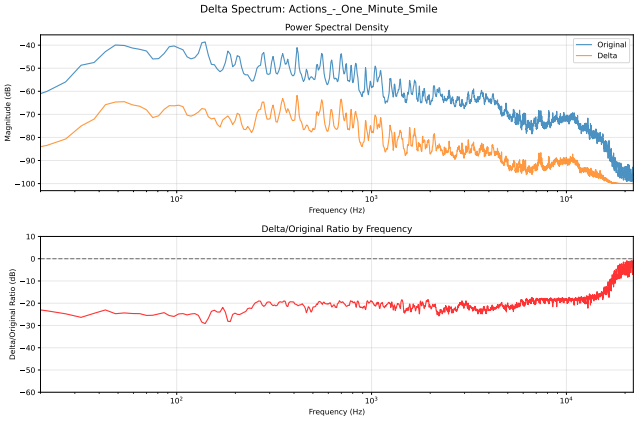


Figure 9: Delta spectrum analysis. Top: Original and delta power spectral density. Bottom: Delta/original ratio is flat at −20 dB across 20 Hz–20 kHz, confirming uniform gain reduction with no frequency-selective behavior.

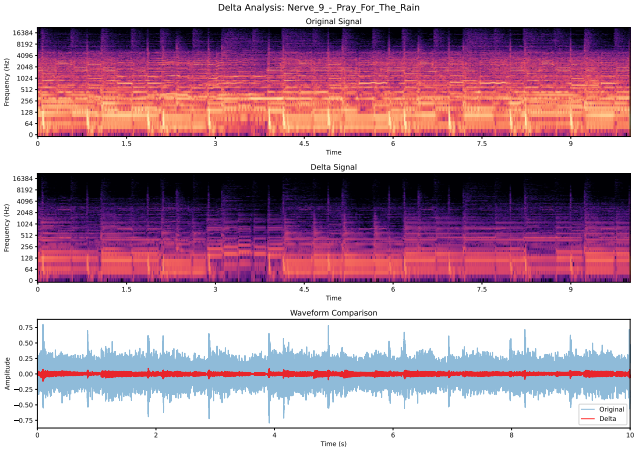


Figure 10: Delta spectrogram analysis. Top: Original signal. Middle: Delta signal (same spectral structure, lower magnitude). Bottom: Waveform comparison showing delta (red) concentrated at transient peaks.

audio files rather than the live session. PRISM eliminates this handoff: the mix engineer runs optimization as the final step before export, while stems remain open in the session.

7.2 Closed Creative Feedback Loop

In conventional workflows, the mastering engineer works on prints that are disconnected from the mix session. If a mastering decision reveals a mix problem (e.g., excessive low-mid buildup), the feedback path requires a new print, re-delivery, and re-mastering. With stem-aware optimization, changing a plugin setting in the mix—adjusting an EQ curve, modifying a reverb send—and directly hearing the effect on the final master becomes possible for the first time. This closes a feedback loop that has been structurally broken since the emergence of mastering as a separate discipline.

7.3 Information Preservation

A mastering engineer working with a stereo bounce has irreversibly lost the ability to make per-stem decisions. The bass and kick are a single waveform; the vocal and its reverb are fused. Even stem mastering loses the live relationship between stems and the mix bus—the stems are snapshots, not the session itself. PRISM operates at the point of maximum information: inside the mix session with access to every stem, every parameter, and every routing decision.

7.4 Perceptual Delivery Accuracy

A systematic perception gap exists in current mastering practice. Engineers master while monitoring PCM audio at sample-peak levels, but consumers hear a version normalized to −14 LUFS (Spotify) or −16 LUFS (Apple Music) with −1 dBTP true peak limiting applied by the platform. The engineer never hears what the listener hears. By mastering to the delivery specification—optimizing directly for the target LUFS and true peak ceiling—PRISM closes this gap. The engineer monitors the actual delivery-format output during the optimization.

7.5 Cost, Reproducibility, and Access

Mastering is the most expensive per-minute step in music production. Automated stem-aware optimization democratizes broadcast-quality, platform-compliant output for independent artists who cannot afford professional mastering. Furthermore, the optimization is deterministic: identical stems, targets, and hyperparameters produce identical results. This reproducibility is impossible with human mastering, where session-to-session variation is inherent.

8 Discussion

8.1 Consistency as the Primary Contribution

The comparison with conventional limiting reveals that consistency—not raw quality—is the primary practical contribution. An 80% win rate and 9.1 dB median improvement are significant, but engineers have always accepted that limiting introduces some distortion. What they cannot accept is unpredictability: the same limiter settings producing transparent results on one track and audible pumping on another.

Conventional limiters exhibit this unpredictability because they operate on the finished mix with no information about spectral content. A kick drum and a bass note summed together look like a single waveform; the limiter cannot know that attenuating the bass would reduce intermodulation while attenuating the kick would destroy the transient feel. It applies uniform gain reduction and hopes the spectral content happens to limit cleanly.

PRISM eliminates this unpredictability by optimizing per-stem contributions to the sidechain. The optimizer learns from IMD which stems cause spectral interference through the nonlinearity—and this learning generalizes across genres because IMD measures the same physical phenomenon regardless of musical content. A track with heavy bass learns to attenuate bass; a track with aggressive drums learns to attenuate drums; both converge to similar final IMD because the optimizer found the configuration that minimizes distortion for that specific spectral content.

The 3× reduction in variance (6.9 dB vs. 19.8 dB) is thus not a statistical accident but a structural consequence of stem access plus gradient-based optimization.

8.2 IMD as the Universal Metric

Our second claim is that IMD is the correct optimization target for differentiable dynamics processing—not loudness, not peak level, not a perceptual metric. The evidence is empirical: an optimizer given only the IMD scalar and the gradient through the limiter autonomously discovers:

1. Sidechain high-pass filtering (bass removal)
2. Transient energy management (drum attenuation)
3. Reverb tail suppression
4. Harmonic preservation
5. Vocal protection

These are the core principles of professional mastering, derived entirely from a single scalar metric. The optimizer does not know what "bass" or "drums" or "vocals" are. It only knows which stems cause spectral interference through the limiter's nonlinearity.

This suggests that IMD may serve as a universal quality metric for any nonlinear audio process—not just limiting. Equalization, saturation, multiband compression, and excitation all introduce nonlinear distortion that IMD can quantify. The same optimization framework could, in principle, tune any differentiable audio effect chain.

8.3 Separation of Concerns

The architecture cleanly separates three concerns:

1. Nonlinear optimization (Phase 1 + 2): Minimize IMD through the limiter. This is the hard problem that requires gradients.
2. Linear output scaling (analytical makeup): Guarantee peak ceiling. This is a closed-form computation.
3. Loudness targeting (LUFS penalty): Steer the optimizer away from degenerate solutions. This is a soft constraint.

Previous versions of this work included makeup gain as an optimized parameter, which conflated concerns (1) and (2) and caused the optimizer to fight itself. The clean separation yields both better results and simpler architecture.

8.4 The Trivial Minimum and Its Resolution

Without a LUFS penalty, IMD minimization has a trivial global minimum at "no limiting" (threshold = 1.0). This is not a bug—it is a correct observation about the optimization landscape. The LUFS penalty resolves it by requiring the optimizer to maintain active compression, but the penalty weight λ_L must be tuned: too low and the optimizer escapes to trivial solutions; too high and it dominates the loss, effectively optimizing for loudness rather than quality.

In our experiments, $\lambda_L = 1.0$ with target LUFS $= -14$ provided sufficient guidance without dominating the IMD objective. The specific LUFS target does not need to be achieved exactly—it serves as a region constraint rather than a precision target.

8.5 Limitations

- The Jacobian is computed as a dense tensor. For long audio ($T \gg 10^5$), memory is a bottleneck.
- The phase transition between Phase 1 and Phase 2 is currently triggered by early stopping. Adaptive phase-switching (e.g., based on gradient norm ratios) may yield better budget allocation.
- The gain parameter bounds ($[0.01, 3.0]$) were hand-chosen. Learning appropriate bounds per stem is future work.
- The MUSDB18-HQ benchmark provides only 4 stem groups (vocals, drums, bass, other), whereas real-world production mixes may contain 8–64+ individual stems with far greater spectral diversity. The optimizer's ability to exploit fine-grained stem information—demonstrated in the 8-stem case study—is underrepresented in the 4-stem evaluation.
- The LUFS penalty weight λ_L requires manual tuning.
- On 45 of 300 benchmark results (15%), the optimizer converged to threshold ≥ 0.99, effectively disabling limiting entirely. These tracks escaped to the trivial IMD minimum by avoiding compression rather than optimizing it. We exclude these results from reported statistics, but their existence indicates the LUFS penalty does not always prevent degenerate solutions. Stronger constraints or adaptive penalty scheduling may be required for certain source material.

9 Conclusion

We have presented PRISM, a fully differentiable audio limiter with exact analytical Jacobians, and demonstrated that intermodulation distortion is the correct optimization target for differentiable dynamics processing. An optimizer minimizing IMD autonomously discovers the engineering principles of professional mastering—sidechain filtering, transient management, harmonic preservation—without any audio content information.

The comparison with conventional limiting reveals the practical contribution: not just lower distortion (80% win rate, 9.1 dB median improvement), but dramatically higher consistency. PRISM's IMD varies by 6.9 dB across diverse source material; conventional limiting varies by 19.8 dB. This 3× reduction in variance means predictable results. A mix engineer using PRISM knows what to expect; an engineer using conventional limiting is gambling on whether the source material happens to limit cleanly.

This consistency emerges from stem access. A conventional limiter sees a stereo waveform and must apply uniform gain reduction regardless of spectral content. PRISM sees individual stems and learns which ones cause intermodulation through the limiter's nonlinearity. Bass energy triggers gain pumping; the optimizer learns to attenuate bass in the sidechain. Transients cause the most distortion; the optimizer learns to reduce drum contribution. This adaptation is automatic—the optimizer discovers it from the IMD gradient alone—and it generalizes across genres because IMD measures the same physical phenomenon regardless of musical content.

The broader result: for any differentiable nonlinear audio process, the distortion introduced by the nonlinearity—not the output characteristics—is the metric that makes gradient-based optimization discover musically meaningful parameter configurations. The code is the side effect; the metric is the contribution.

The system is open-source at `https://github.com/agrathwohl/hyraxiable`.

Acknowledgments

The author thanks the open-source Rust and PyO3 communities. This work was conducted independently without institutional funding.

References

[1] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *Proc. ICLR*, 2020.

[2] C. J. Steinmetz, N. J. Bryan, and J. D. Reiss, "Style transfer of audio effects with differentiable signal processing," *J. Audio Eng. Soc.*, vol. 70, no. 9, pp. 708–721, 2022.

[3] J. Colonel, C. Steinmetz, M. Michelen-Strominger, and B. Pardo, "Direct design of biquad filter cascades with deep learning by sampling random audio effects," in *Proc. ICASSP*, 2022.

[4] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Applied Sciences*, vol. 10, no. 3, 2020.

[5] S. H. Hawley, B. Colburn, and S. Mimilakis, "SignalTrain: Profiling audio compressors with deep neural networks," in *Proc. AES Conv.*, 2019.

[6] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Proc. DAFx*, 2020.

[7] ITU-R, "BS.1770-5: Algorithms to measure audio programme loudness and true-peak audio level," International Telecommunication Union, 2020.

[8] D. Giannoulis, M. Massberg, and J. D. Reiss, "Digital dynamic range compressor design—a tutorial and analysis," *J. Audio Eng. Soc.*, vol. 60, no. 6, pp. 399–408, 2012.

[9] European Broadcasting Union, "EBU R128: Loudness normalisation and permitted maximum level of audio signals," EBU Technical Recommendation, 2020.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[11] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ—an uncompressed version of MUSDB18," 2019. [Online]. Available: `https://zenodo.org/record/3338373`

[12] B. De Man, J. D. Reiss, and R. Stables, "Ten years of automatic mixing," in *Proc. 3rd Workshop on Intelligent Music Production*, 2017.

[13] D. Moffat and M. B. Sandler, "Approaches in intelligent music production," *Arts*, vol. 8, no. 4, 2019.

[14] ITU-R, "BS.1387-1: Method for objective measurements of perceived audio quality," International Telecommunication Union, 1998.

[15] C. J. Steinmetz and J. D. Reiss, "pyloudnorm: A simple yet flexible loudness meter in Python," in *Proc. AES Conv.*, 2021.

[16] M. Comunità, C. Steinmetz, and J. D. Reiss, "Diff-A-Riff: Musical accompaniment co-creation via latent diffusion models," in *Proc. ISMIR*, 2023.

[17] S. Nercessian, A. Sarroff, and K. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *Proc. ICASSP*, 2021.