Le Mans Université

Master 2 Project Report

Mechanical protection algorithms for loudspeakers

Eliot Deschang & Florian Marie

Supervised by

Antonin Novak

Master 2 IMDEA

Academic Year : 2024 - 2025

Contents

Acknowledgement

The authors extend their gratitude to professor Antonin Novak. His availability as long as his reactivity provided the best environment to lead this project.

Thanks to professors Manuel Melon, Bruno Gazengel as well as Laurent Simon for their support.

1 Introduction

Over the last decades, the evolution of music genres has significantly reshaped the frequency content of popular tracks, with a noticeable increase in bass frequencies. From the funk and disco era to the techno boom of the 90s and the current dominance of rap featuring deep basslines, low-frequency content has become a hallmark of modern music. This shift, however, has posed new challenges for loudspeakers designers, as most of their membrane displacement is driven by low-frequency signals, increasing the risk of mechanical overload. Compounding this issue, the rise of smartphones and laptops has pushed manufacturers to design micro-speakers that deliver impactful bass, a feature often associated with good audio quality. Yet, these small drivers face a fundamental limitation: their compact surface necessitate large membrane excursion to produce bass, increasing their susceptibility to mechanical failure/overload. To ensure speaker longevity and reliability, engineers focused on digital signal processes with the goal of controlling driver's membrane excursion. In general, too high level signals can damage both driver coil made of thin copper wire, suspension components often composed of rubber plastic and/or membrane.

Actually, two solutions exist to evaluate the membrane displacement: either using a sensor (accelerometer or laser typically) or either using a model to assess the displacement knowing another quantity, such as speaker voltage or current. The model-based approach is considered in this work, and compute an estimation of the membrane displacement as a function of the input voltage, knowing the linear parameters of the driver. After introducing the state-of-the-art, a simple linear model is studied in order to derive the transfer function linking the input voltage to membrane displacement. Building on this foundation, the theoretical aspects of various protection algorithms implemented in this project are reviewed. These include a feedback-based algorithm inspired by physics behind electrodynamic driver, an improved version including a delay line, and two feedforward algorithms: the first one is a limiter acting on the estimated displacement signal while the second one is a dynamic low-shelf filter directly processing the audio signal.

To evaluate the algorithms performances, measurement are carried out on a 5-inch woofer driver, implementing the algorithms either as an embedded solution with a microcontroller, or as a software extension that can be integrated into Digital Audio Workstations (DAWs), known as plugin. The driver membrane displacement is measured using various signals and a vibrometer is used to objectively check the algorithms performances.

2 Context

2.1 State of the art

To control the membrane excursion in order to protect it from mechanical overload, it is trivial that its displacement has to be known. This can be achieved through two approaches: using a physical sensor or deriving the displacement from an input quantity, such as voltage or current [1].

In sensor-based methods, a secondary coil can measure velocity by detecting variations in inductance caused by eddy currents. The displacement is then calculated by integrating this velocity [2]. Alternatively, optical methods [3] can provide continuous displacement estimation, or contact switches can serve as indicators for reaching the displacement limit [4].

In the case of a model-based approach, a linear model of a loudspeaker can be used to deduce the membrane's excursion from the music signal voltage, by expressing the linear filter linking the displacement to voltage. Incorporating non-linearities into the model is also feasible but requires a more detailed description of the driver. Specifically, this includes modelling the dependencies of the force factor, stiffness, and inductance as a function of the membrane excursion [5]. A non-linear displacement model can then be implemented using a state-space approach, for instance. However, non-linear models may underestimate the actual displacement as it lead to a compression behaviour introduced by geometric non-linearities (higher stiffness and a reduced force factor, as the excursion increases). Moreover, it has been shown that in practice, non-linearities due to the viscoelastic properties of the driver's suspension components can lead to an expansion behaviour as driver's voltage increases [6].

Up to now, most common solutions for limiting driver's membrane excursion are based on variable high-pass filter with an adjustable cut-off frequency, processing the audio signal [1, 7, 8]. The concept is straightforward: when the membrane's excursion exceeds the threshold, the high-pass filter's cut-off frequency increases until the excursion falls below the limit, after which it gradually returns to its initial state.

Stepping back to signal dynamics control, the audio industry already offers numerous solutions to control signal's dynamic, such as compressors [9, 10, 11], limiters [12] (which are particular type of compressors) and dynamic parametric equalizers [13, 14]. To the knowledge of the authors, these solutions are not that much considered when it comes to control membrane's displacement, and it is in this context that two of these methods will be adapted to the problem.

2.2 Project specificities

In this work, and in the literature, X_{max} is defined as the displacement limit not to exceed. It is important to bear in mind that they are several definitions of X_{max}, either provided by the driver manufacturer or derived from standards [15]. In this work, X_{max} is set arbitrary, as an input parameter for the algorithms.

Moreover, in this project, the protection solutions do not rely on a physical sensor. Instead, the displacement is estimated using the displacement transfer function related to the input voltage, assuming free-field radiation. This approach is straightforward, as it only requires a linear model of the driver, based on lumped parameters that can be easily measured. The next section reviews the linear loudspeaker model used in this work.

3 Loudspeaker model

In this work, the linear model of an electrodynamic driver is considered. The equation characterizing the coupling between the electrical, mechanical and acoustical part of an electrodynamic driver are given in the frequency domain by

$$\begin{cases} U(\omega) = (R_{ec} + j\omega L_{ec})\, I(\omega) + BlV(\omega), \\ BlI(\omega) = \left(j\omega R_{mm} - \omega^2 M_{mm} + \frac{1}{C_{mm}}\right) X(\omega) + F_{air}(\omega), \\ F_{air}(\omega) = S_d(Z_{af} - Z_{ar})Q(\omega), \end{cases} \tag{1}$$

where $\omega = 2\pi f$ is the angular frequency corresponding to frequency f, $U(\omega)$ is the voltage at the driver terminal, $I(\omega)$ is the current flowing through it, and where $X(\omega) = \frac{V(\omega)}{j\omega}$ is the displacement of the moving part with respect to rest position. R_{ec} and L_{ec} denote resistance and inductance, and M_{mm}, R_{mm} and C_{mm} denote the mechanical mass of the moving part, viscous losses, and compliance of the suspension, respectively. Z_{af} and Z_{ar} represent the front and rear acoustical impedance, defined as the pressure over the normal volume velocity Q to the membrane of surface S_d.

The equivalent electrical circuit, representing this system of equations in the three domains, is depicted in Fig. 1.

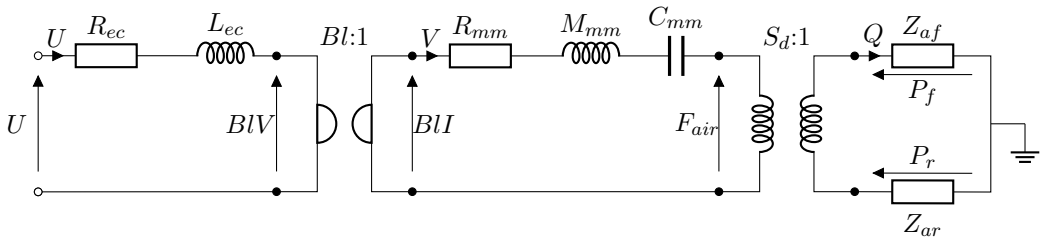


Figure 1: Equivalent electrical network of a loudspeaker, in the electrical, mechanical and acoustical domain.

In the case of a driver positioned on an infinite baffle, acoustical radiation impedances can be written in low frequency approximation as the sum of a mass and a resistance (see Appendix A). These lumped element can be translated in the mechanical domain to the corresponding purely mechanical constant, resulting in equivalent moving mass $M_{ms} = M_{mm} + S_d^2(M_{af} + M_{ar})$ and mechanical resistance $R_{ms} = R_{mm} + S_d^2(R_{af} + R_{ar})$. C_{ms} is equal to C_{mm} in this case. Taking into account these considerations, Eq. (1) can be written as

$$\begin{cases} U(\omega) = (R_{ec} + j\omega L_{ec})\, I(\omega) + BlV(\omega), \\ BlI(\omega) = \left(j\omega R_{ms} - \omega^2 M_{ms} + \frac{1}{C_{ms}}\right) X(\omega) \end{cases} \tag{2}$$

Using Eq. (2), it is possible to express the exact displacement transfer function as

$$\frac{X(\omega)}{U(\omega)} = \frac{Bl}{(R_{ec} + j\omega L_{ec})\left(-\omega^2 M_{ms} + j\omega R_{ms} + \frac{1}{C_{ms}}\right) + j\omega Bl^2}. \tag{3}$$

At low frequency, the impact of L_{ec} is negligible. It is thus possible to write the final displacement estimator filter $H_{XU}(\omega)$ starting from Eq. (2) and neglecting L_{ec} in canonical form as

$$H_{XU}(\omega) = \frac{K}{1 + j\frac{\omega}{\omega_0 Q_s} - \left(\frac{\omega}{\omega_0}\right)^2}, \tag{4}$$

with $K = \dfrac{BlC_{ms}}{R_e}$, $\omega_0 = \frac{1}{\sqrt{M_{ms}C_{ms}}}$ and $Q_s = \frac{1}{R'_{ms}}\sqrt{\frac{M_{ms}}{C_{ms}}}$, where $R'_{ms} = R_{ms} + \frac{Bl^2}{R_{ec}}$.

This second-order transfer function is a low-pass filter, whose cut-off frequency is defined by the driver mass and compliance, while the quality factor depends on all the driver parameters (expect L_{ec}). In this context, a driver whose quality factor is higher than $1/\sqrt{2}$ is qualified as resonant.

In the Laplace domain with $s = j\omega$, $H_{XU}(s)$ is written in the form of a second-order section as

$$H_{XU}(s) = \frac{B_0 s^2 + B_1 s + B_2}{A_0 s^2 + A_1 s + A_2} = \frac{Bl/R_{ec}}{s^2 M_{ms} + s\left(R_{ms} + \frac{Bl^2}{R_e}\right) + \frac{1}{C_{ms}}}, \tag{5}$$

As the algorithms are implemented on numerical devices, it is needed to digitize this filter in order to work in the discrete-time domain. A common solution to digitize continuous-time transfer functions like the one in Eq. (5) is to use the bilinear transform, a mapping from the s to the z plane according to the substitution [16]

$$s \leftarrow 2F_s \frac{1 - z^{-1}}{1 + z^{-1}}, \tag{6}$$

with F_s the sampling frequency. The corresponding digital filter expresses as

$$H_{XU}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}. \tag{7}$$

For second order analogue filter (s-domain), it is easy to analytically apply the bilinear transform (see appendix B). Once the digital coefficient of a filter are obtained, biquad structures are used to compute the output at each step-time. Digital coefficients have to be normalized, such that $a_0 = 1$ in order to avoid important numerical issues. In this project, Direct Form I (DFI) is used to compute the predicted displacement, as it provides the highest stability for constant digital coefficient over time.

4 Feedback algorithms

Feedback algorithm for drivers mechanical protection are based on a filter that try to compensate for an exceed in displacement, estimated from the output of the filter itself [7]. As a result, the threshold defined by the user must be set lower than the absolute displacement limit not to be exceeded. This precaution accounts for the delay inherent to the feedback structure, as the filter only activates once the defined threshold in the algorithm is surpassed.

From literature, feedback algorithms for displacement control are most often based on high-pass filters [7, 8], knowing that the low frequency content of a signal is mostly responsible for high excursions. The idea of such an algorithm is that when the membrane's excursion exceeds the threshold, the high-pass filter's cut-off frequency increases until the excursion falls below the limit, after which it gradually returns to its initial state.

In this section, two algorithms are introduced following a feedback structure. The first one uses a filter that virtually modify the compliance and mechanical resistance parameters of the driver (C_{ms} and R_{ms} respectively). The second one uses two instances of that same filter, while introducing a secondary delayed path to compensate inherent latency introduced by the feedback control algorithm.

4.1 Physical-based filter

The diagram of the feedback algorithm implemented in this work is depicted in Fig. 2.

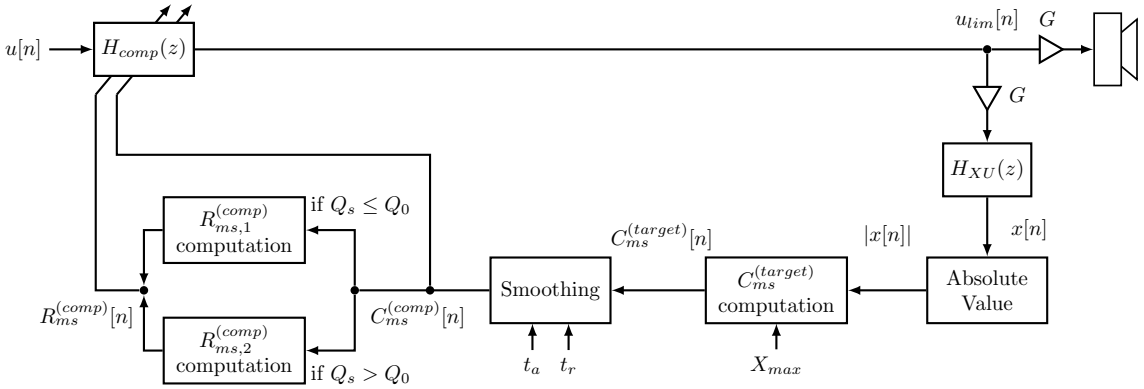


Figure 2: Diagram of the feedback algorithm.

Compensation filter

The compensation filter H_{comp} is based on the physical knowledge of the driver, i.e. its Thiele Small parameters. This compensation filter is design to virtually modify the driver's properties in order to reduce the displacement when the absolute displacement reaches the displacement threshold (X_{max}). Since the displacement transfer function, $H_{XU}(s)$, is a second-order low-pass filter (see Eq. (4)), and its behaviour in low frequency region is primarily stiffness-driven, the objective is to assign a new virtual compliance, denoted as $C_{ms}^{(comp)}$, along with a new mechanical resistance $R_{ms}^{(comp)}$, to control both displacement transfer function gain at low frequency and quality factor. To achieve this, the numerator of the compensation filter must match the denominator of $H_{XU}(s)$, as shown in Eq. (5), ensuring the desired virtual modifications are applied effectively. Such a filter is thus defined as

$$H_{comp}(s) = \frac{s^2 M_{ms} + s\left(R_{ms} + \frac{Bl^2}{R_e}\right) + \frac{1}{C_{ms}}}{s^2 M_{ms} + s\left(R_{ms}^{(comp)} + \frac{Bl^2}{R_e}\right) + \frac{1}{C_{ms}^{(comp)}}}. \tag{8}$$

The product of $H_{XU}(s)$ and $H_{comp}(s)$ is then

$$H_{tot}(s) = \frac{Bl/R_{ec}}{s^2 M_{ms} + s\left(R_{ms}^{(comp)} + \frac{Bl^2}{R_e}\right) + \frac{1}{C_{ms}^{(comp)}}}. \tag{9}$$

Intuitively, examining Eq. (9), reducing $C_{ms}^{(comp)}$ compared to the default compliance of the loudspeaker effectively increases the suspension stiffness. As a result, for the same excitation level $u[n]$, the membrane displacement decreases because the increased stiffness generates a greater restoring force. The ratio of compliances can then be defined: $C = C_{ms}^{(comp)}/C_{ms}$, varying between 0 and 1. The new quality factor of the speaker starting from Eq. (9) can now be defined as

$$Q_s^{(comp)} = \frac{1}{R_{ms}^{(comp)} + Bl^2/R_{ec}} \sqrt{\frac{M_{ms}}{C_{ms}^{(comp)}}}. \tag{10}$$

As shown in Fig. 3 (a), if R_{ms} was not compensated ($R_{ms}^{(comp)} = R_{ms}$), lower values of $C_{ms}^{(comp)}$ compare to C_{ms} could increase the $Q_s^{(comp)}$ factor (dashed line). $R_{ms}^{(comp)}$ is then adjusted alongside $C_{ms}^{(comp)}$ to further control the resonance (continuous line). In this way, and in the case of a non-resonant speaker (for which $Q_s \leq Q_0 = 1/\sqrt{2}$), $Q_s^{(comp)}$ won't exceed $1/\sqrt{2}$. When no compensation is applied, $Q_s^{(comp)} = Q_s$.

For a resonant speaker ($Q_s > Q_0 = 1/\sqrt{2}$), a mapping function (see Appendix C) is employed to gradually decrease $Q_s^{(comp)}$ toward $1/\sqrt{2}$ as $C_{ms}^{(comp)}$ decreases (Fig. 3 (b), continuous line). This approach ensures that the compensation filter does not alter the loudspeaker's compliance or quality factor when no displacement reduction is required.

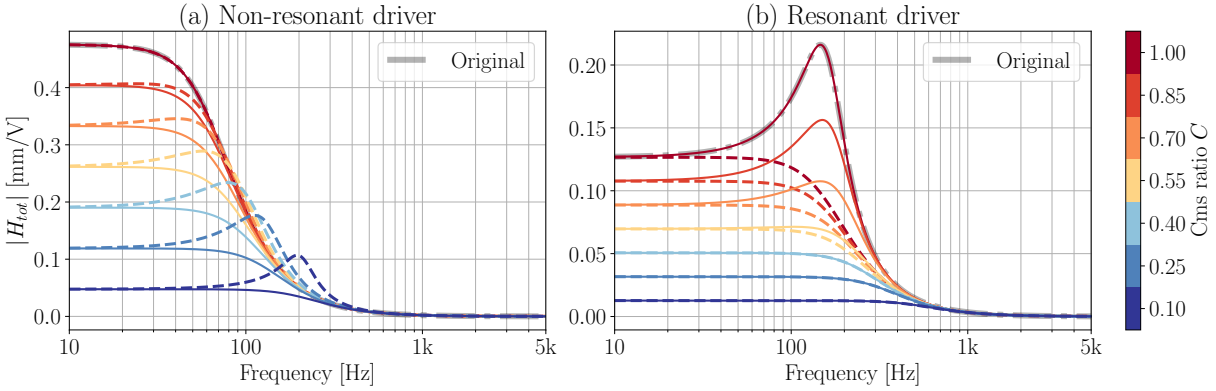


Figure 3: Magnitude of H_{tot} for (a) a non-resonant driver and (b) a resonant driver, shown for various compliance ratios $C = \frac{C_{ms}^{(comp)}}{C_{ms}}$. In (a), dashed lines represent $R_{ms}^{(comp)} = R_{ms}$, while continuous lines show $R_{ms}^{(comp)}$ adjusted to maintain $Q_s^{(comp)} \leq 1/\sqrt{2}$. In (b), dashed lines maintain $Q_s^{(comp)} = 1/\sqrt{2}$, whereas continuous lines use a mapping function to adjust $R_{ms}^{(comp)}$ gradually.

Displacement estimator

Initially, the driver's membrane displacement is estimated using the transfer function $H_{XU}(z)$, which calculates the displacement based on the voltage limited by the compensation filter and scaled by the amplifier's gain G, such that $u_{lim}[n] \cdot G = u_{hp}[n]$ is equal to the speaker terminal voltage. Refer to Eq. (4) and Eq. (5) for details.

Absolute value

The absolute value of the estimated displacement at instant n, $|x[n]|$, is computed to provide a clear criterion for comparison against the set displacement limit X_{max}, in order to update $H_{comp}(s)$ filter coefficients. An alternative approach, as suggested in [7, 8], would involve calculating the envelope of $x[n]$. However, since the definition of a signal's envelope can vary (e.g., using the

Hilbert transform or peak signal values), this feedback algorithm opts for the absolute value of the estimated displacement, as it has shown satisfactory results.

Computation of $C_{ms}^{(target)}$

The main variable of the compensation filter, $C_{ms}^{(comp)}$, has now to be computed. $C_{ms}^{(comp)}$ varies between two limits defined by $C_{ms}^{(target)}$: the lower one being $C_{ms}^{(min)}$, and the upper one C_{ms} (speaker compliance). $C_{ms}^{(min)}$ could have been set to 0, but to limit ripples, it is computed knowing the maximum voltage that can be applied to the speaker, denoted V_{max}. $C_{ms}^{(min)}$ has then to satisfy

$$|H_{tot}(s)| \leq \frac{X_{max}}{V_{max}}, \ \forall s. \tag{11}$$

As this transfer function product exhibits a horizontal asymptote at low frequency, evaluating it in $s = 0$, leads to

$$\frac{Bl/R_{ec}}{1/C_{ms}^{(min)}} = \frac{X_{max}}{V_{max}} \quad \Leftrightarrow \quad C_{ms}^{(min)} = \frac{X_{max} R_{ec}}{V_{max} \cdot Bl}. \tag{12}$$

As a safety margin, the definitive minimum compensation compliance, fix over time, is 90% of $C_{ms}^{(min)}$. The output of the $C_{ms}^{(target)}$ computation block is then

$$C_{ms}^{(target)}[n] = \begin{cases} C_{ms}, & \text{if } |x[n]| \leq X_{max}, \\ C_{ms}^{(min)}, & \text{otherwise.} \end{cases} \tag{13}$$

Smoothing

Since $C_{ms}^{(target)}[n]$ can fluctuate significantly over short time intervals, oscillating between C_{ms} and $C_{ms,\min}^{(comp)}$, $C_{ms}^{(target)}$ must first be smoothed before being applied to the compensation filter. This process offers two key benefits: it reduces sharp filter coefficient transitions, minimizing numerical sensitivity to instabilities, and ensures smoother transitions from a psychoacoustic point of view.

The smoothing of $C_{ms}^{(target)}$, conducing to $C_{ms}^{(comp)}$ is defined as a difference equation given by

$$C_{ms}^{(comp)}[n] = (1 - k) \cdot C_{ms}^{(comp)}[n-1] + k \cdot C_{ms}^{(target)}[n], \tag{14}$$

with

$$\begin{cases} k = \text{AT}, \text{ if } \quad C_{ms}^{(target)}[n] < C_{ms}^{(comp)}[n-1] \\ k = \text{RT}, \text{ otherwise} \end{cases} \tag{15}$$

where AT and RT denote the attack and release coefficients, defined by time constant t_a and t_r respectively, as shown in Fig. 2. The explicit expressions for these coefficients are provided in Appendix D.1.

This smoothing technique, originally introduced by McNally [9], can be intuitively understood through the parameter k, which acts as a potentiometer balancing the contributions of the past value, $C_{ms}^{(comp)}[n-1]$, and the current target value, $C_{ms}^{(target)}[n]$. A smaller value of k gives more weight to the past, effectively creating a slower response to changes in the target compliance. Alternatively, Eq. (14) can be interpreted as the equation of a first-order digital low-pass filter, where the pole location dynamically varies according to the value of k (see Appendix D).

Computation of R_{ms}

$R_{ms,1}^{(comp)}$ and $R_{ms,2}^{(comp)}$ computation blocks are activated depending on the nature of the driver (resonant or non-resonant). The path do not change over time as Q_s is computed only one time during initialization step.

$R_{ms,1}^{(comp)}$ computation block is activated in the case of a non-resonant driver to process the comparison

$$R_{ms,1}^{(comp)} = \max\left(R_{ms},\ \frac{1}{Q_0}\sqrt{\frac{M_{ms}}{C_{ms}^{(comp)}}} - \left(Bl^2/R_{ec}\right)\right), \tag{16}$$

with $Q_0 = 1/\sqrt{2}$, the quality factor of a critically damped system.

$R_{ms,2}^{(comp)}$ computation block is activated in the case of a resonant driver. In order not to modify the original quality factor of the speaker, a mapping function is defined. Its purpose it to gradually decrease Q_s toward Q_0 while $C_{ms}^{(comp)}$ decreases. This function returns a factor $Q_s^{(comp)}$ between Q_s and Q_0, for $C \in [C_{treshold}, 1]$. For $C \in [0, C_{treshold}]$, $Q_s^{(comp)} = Q_0$. $C_{treshold}$ defines the knee location of the mapping function (see appendix C). This block then computes $R_{ms,2}^{(comp)}$ as

$$R_{ms,2}^{(comp)} = \frac{1}{Q_s^{(comp)}}\sqrt{\frac{M_{ms}}{C_{ms}^{(comp)}}} - \left(Bl^2/R_{ec}\right). \tag{17}$$

4.2 Improvement with duplicated compensation filter and preliminary delay

Classical feedback architecture algorithms as the one presented in Fig. 2 have one big advantage: the exact amount of correction to apply at a given time is unimportant. At the same time, this strength represents its own weakness: because of delay in the feedback chain, the compensation gain at low frequency cannot be reduced until after an excessive displacement level has appeared at the output (which is the reason why the displacement threshold in the feedback chain must be set lower than the real limit not to be exceeded). It is therefore inevitable that overshoots will occur at the output.

An alternative to overcome this problem without needs for a margin, is to duplicate the compensation filter and add a preliminary delay line in front of this secondary compensation filter, on which the output will be sent to the amplifier and loudspeaker, as depicted in Fig. 4. This improved algorithm as been adapted from a limiter algorithm presented in [9].

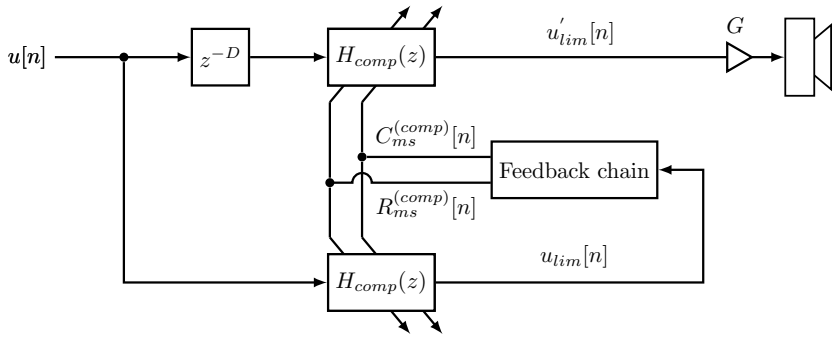


Figure 4: Diagram of the feedback algorithm with duplicate compensation filter and preliminary delay.

From a digital implementation point of view, both H_{comp} filters share the same digital coefficient at instant n, but not the same buffer due to the delay line in the upper path. The feedback chain gathers all the process described lately in Fig. 2. The block preceding H_{comp} delays the signal by D samples. This delay is referred to as the look ahead delay, and is let as an input parameter for the user to set. In this work, D is set to $2\lfloor t_a \cdot F_s \rfloor$.

5 Feedforward algorithms

Feedforward algorithms, unlike feedback algorithms, are based on a control of the input signal that does not rely on real-time feedback from the system's output. Instead, the input signal is processed through a so-called side-chain, which aims to estimate the necessary correction to modify the input signal as desired. This makes the design of a feedforward system significantly more challenging, as these algorithms do not monitor the system's output to verify or adjust the applied correction in real time.

Feedforward protection algorithms, such as those proposed in [17, 18], typically apply a gain to the voltage signal, often preceded by a delay to account for attack time. An algorithm of this type is proposed in Appendix E. This corresponds to a classical limiter, as those used in audio. However, this approach modifies the entire signal spectrum, despite high frequencies rarely contributing to large excursions.

In this section, two feedforward algorithms are introduced. The first algorithm employs a limiter that processes the estimated displacement rather than the input voltage. The second algorithm utilizes a dynamic low-shelf filter designed to adjust the gain applied to the low-frequency content of the voltage signal.

5.1 Displacement limiter

Limiter dynamic processes are common tools to control the level of a music signal. Although limiters use a broadband gain to achieve it, what is proposed in this algorithm is to apply this gain on the displacement signal instead than on the voltage signal.

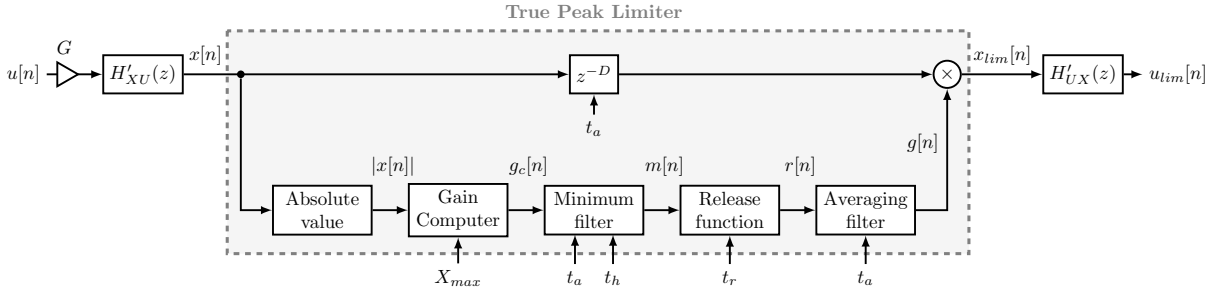


Figure 5: Diagram of the limiter algorithm.

The approach illustrated in Fig. 5 involves converting the voltage signal into displacement $x[n]$, applying a true peak limiter to constrain the displacement to $x_{lim}[n]$, and then converting it back into voltage $u_{lim}[n]$, that is sent to the amplifier and loudspeaker. These swaps of domain from voltage to displacement and vice versa are based on reciprocal filters H'_{XU} and H'_{UX}, such that the product leads to a filter of unitary gain with pure delay (due to the delay line between them) when there is no need for displacement limitation (i.e $g[n] = 1$).

Basic transfer function $H_{XU}(z)$ can't be used directly because it exhibits zeros on the unit circle close to $F_s/2$. As the reciprocal of $H_{XU}(z)$ is its inverse, it is needed to set these zeros inside the unit circle to assure inverse filter stability. This is achieved by applying a transformation on zeros, while ensuring that $H_{XU}(z)$ and $H'_{XU}(z)$ share the same magnitude plateau at low frequencies (see appendix F for details), where it's mandatory to get a good estimation of the displacement.

With the aim of better understanding how this limiter algorithm works, the signals at each stage of the elementary blocks of the limiter's side-chain in Fig. 5 are shown in Fig. 6 for a specific input signal. The input $x[n]$ (blue curve in the top plot) is a Dirac delta function followed by a 600 Hz sine wave modulated by a step change in amplitude. Explanations of how the various blocks work are given below.

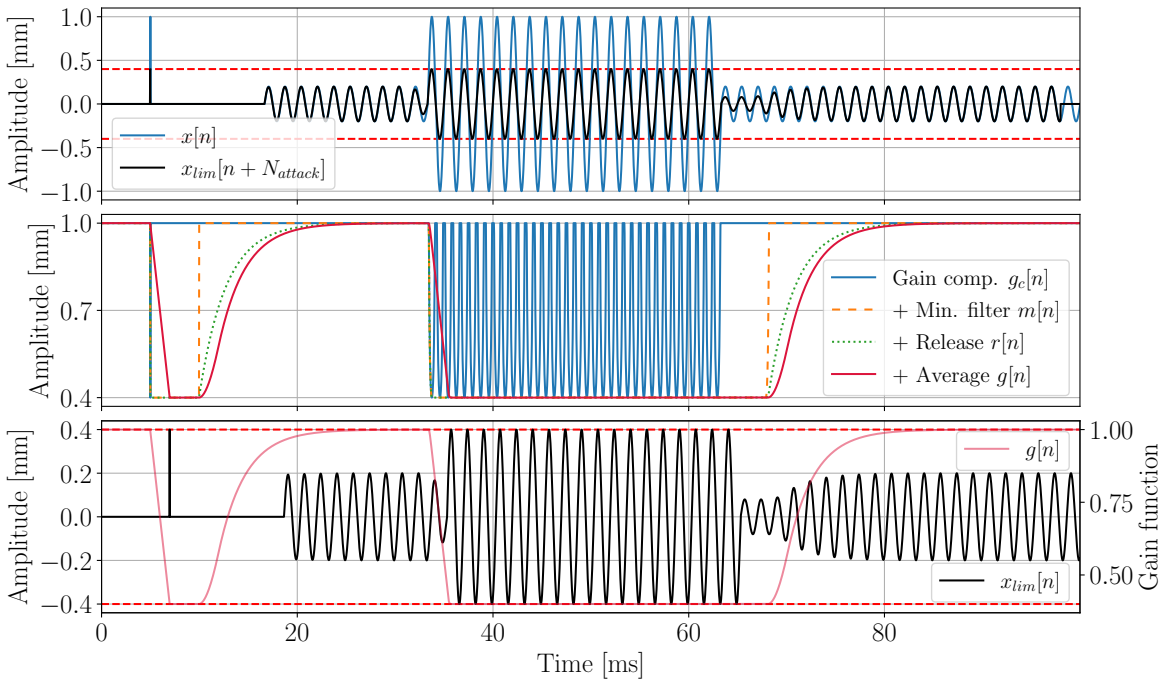


Figure 6: Output signal of the limiter $x_{lim}[n]$ for a given input $x[n]$, for a threshold $X_{max} = 0.4$ mm, and time constants $t_a = 2$ ms, $t_h = 3$ ms, and $t_r = 6$ ms. The central plot illustrates the output signals from each block of the limiter's side-chain.

Gain computer

The role of the gain computer function is to deduce, from the absolute displacement value, what should be the gain $g_c[n]$ to apply to this displacement in order to not exceed the given threshold X_{max}. Thus, the gain computer output is simply defined as

$$g_c[n] = \min\left(1, \frac{X_{max}}{|x[n]|}\right). \tag{18}$$

Therefore, $g_c[n] \in]0, 1]$. A more advanced function can be implemented to achieve smoother transitions between the two subdomains of the gain computer function by including a knee width parameter (see Appendix G).

Minimum filter

The gain value $g_c[n]$ is the maximum gain that can be applied to avoid the displacement going above the limit, but sticking exactly to that limit means the gain curve changes really quickly, and this distorts the signal. The minimum filter actually acts as an extreme smoother applied on the gain computer $g_c[n]$. This block computes the minimum value of a moving rectangular window, whose length is defined by the attack time plus hold, i.e. $N_{attack} + N_{hold}$. This will make more sense when introducing the averaging filter block. The minimum moving filter output is

$$m[n] = \min\left(x[n - (N_{attack} + N_{hold})], \ldots, x[n]\right), \tag{19}$$

with $N_{attack} = \lfloor t_a \cdot F_s \rfloor$ and $N_{hold} = \lfloor t_h \cdot F_s \rfloor$. The filter output is valid $\forall n \in \mathbb{N}$, since values of $x[n]$ for negatives n values are filled with 0, using a circular buffer logic.

For the following explanation, keep in mind that by convention, when the value of $m[n]$ decreases with respect to the previous value, it will be said that the system is in the attack phase. Conversely, when $m[n]$ goes to a higher value, the system enters the release phase, reverting to its previous state where a lower gain reduction was required.

Release function

Since $m[n]$ can vary rapidly between consecutive samples, transitioning abruptly from a low value to a higher one, the release function smoothed this transition. This function uses the release time t_r as an input parameter. The release coefficient, RT, is defined similarly to the smoothing function (refer to Appendix D.1). The release function is expressed as

$$r[n] = \min\left(m[n],\ (1 - \mathrm{RT}) \cdot m[n-1] + \mathrm{RT} \cdot m[n]\right). \tag{20}$$

The use of the min function aim to discard falling from rising edge: the release function will not touch to falling edge of $m[n]$ (passing from a given value to a lower one) and only apply the release when $m[n]$ goes from a low value to a higher one.

Averaging filter

The averaging filter primarily serves to smooth the falling edge of $r[n]$, as the rising edge (release phase) is already smoothed by the release function. Incorporating the averaging filter allows for different attack and release times, enabling the gain reduction shape to adapt based on the nature of the edge (rising or falling). The output of the averaging filter is obtained by convolving a rectangular window of N_{attack} samples with $r[n]$, resulting in

$$g[n] = \frac{1}{N_{attack}} \sum_{i=0}^{N_{attack}-1} r[n-i]. \tag{21}$$

This averaging procedure can be implemented as a finite impulse response (FIR) filter with N_{attack} taps, which is the approach used in the plugin implementing this algorithm. However, this method requires N_{attack} operations per sample. For embedded solutions, where t_a (and consequently N_{attack}) can be declared as constant, a recursive implementation of the moving average can be used to reduce computational complexity. Using this approach, the averaging is computed as

$$g[n] = g[n-1] + \frac{1}{N_{attack}}\left(r[n] - r[n - N_{attack}]\right). \tag{22}$$

This recursive method minimizes the number of operations required, though it necessitates a delay line to access $r[n - N_{attack}]$ for each computation of the average.

Delay line

Applying a moving average filter over N_{attack} samples produces a smoothly varying gain value, which is always superior to or equal to the minimum gain $m[n]$ from N_{attack} samples ago. To ensure proper alignment of the smoothed gain curve with the original signal, the input signal $x[n]$ is delayed by the same number of taps used to compute the averaging output. Therefore, the delay in the delay line block of Fig. 5 is given by $D = N_{attack}$. The output of the limiter is then expressed as

$$x_{lim}[n] = g[n] \cdot x[n - N_{attack}]. \tag{23}$$

5.2 Dynamic low-shelf filter

Another solution based on a low-shelf filter, still following a feedforward implementation, is proposed in this project. Again, this choice of filter has been motivated by the shape of the displacement transfer function, similar to a second-order low-pass (see Eq. (4)). The main advantage of this algorithm is that no reciprocal filter is needed, as the low-shelf is directly applied to the voltage signal to be sent to the speaker. A disadvantage of this solution is that overshoots of the membrane excursion with respect to the X_{max} threshold can occur. In this work, the gain of the low frequency shelf is adjusted, while the cut-off frequency of the low-shelf is fixed at a value equal to the double of the driver's resonant frequency. Varying the cut-off upward, or adding a feedback loop, are possible improvements of this algorithm that could guarantee a better protection for drivers. The process chain, starting from the absolute value computation block to the averaging filter block, remains the same as with the limiter described previously.

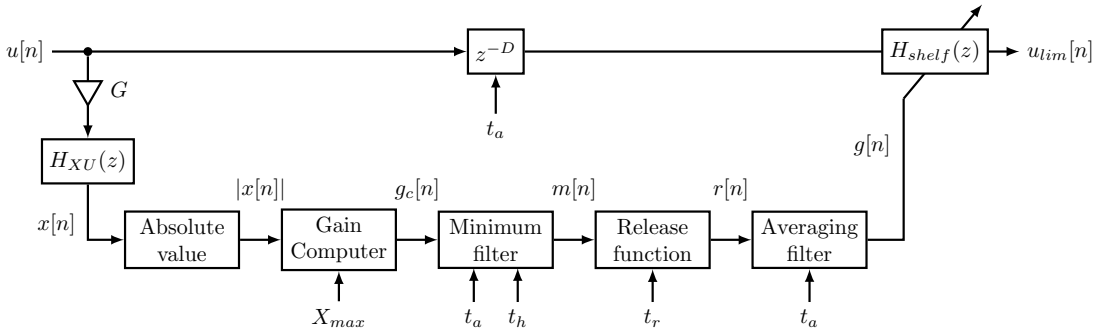


Figure 7: Diagram of the low-shelf algorithm.

In the Laplace domain, a second order low-shelf filter transfer function expresses as [16]

$$H_{LS}(s) = A\frac{s^2 + \frac{\omega_c}{Q_{LS}}\sqrt{A}\ s + A\omega_c^2}{As^2 + \frac{\omega_c}{Q_{LS}}\sqrt{A}\ s + \omega_c^2}, \tag{24}$$

with ω_c the cut-off angular frequency, Q_{LS} the quality factor and A the linear gain. The logarithmic gain A_{dB} (level of the shelf in dB), defined as $20\log(g[n])$, is linked to A by

$$A = 10^{\frac{20}{40}\log_{10}(g[n])} = \sqrt{g[n]}. \tag{25}$$

6 Implementation and experimental setup

This section describes first the algorithm implementation over two supports, i.e. embedded and audio plugin that can be used within a Digital Audio Workstation (DAW). The measurement setup used for both the embedded and the plugin solutions are described. The goal is to evaluate each algorithm performances on a given driver, for which details are provided afterward.

6.1 Algorithm implementation

Arduino IDE is used to implement the feedback, and feedback with a lookahead, in the Teensy 3.6 microcontroller. The board used in this project were designed by Oliver Monroe, Antonin Novak and Stephan Letourneur [19], providing a simplified real-time (sample by sample) digital processing unit, incorporating a dedicated digital to analogue converter (DAC MAX 5717).

To enable anyone with a computer to experiment with the algorithms in real-time, adjusting parameters such as the loudspeaker model, audio chain gain, time constants, and displacement threshold, three distinct plugins were developed in C++ using the JUCE framework [20]. These plugins not only allow users to auralize the signal required to limit the loudspeaker's excursion but also serve as a valuable tool for conducting research on these algorithms. The Graphical User Interface (GUI) of the plugin implementing the two feedback algorithms is shown in Fig. 8.

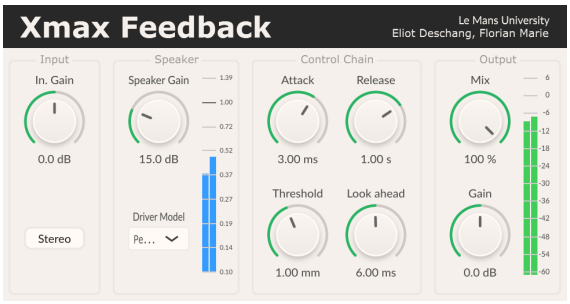
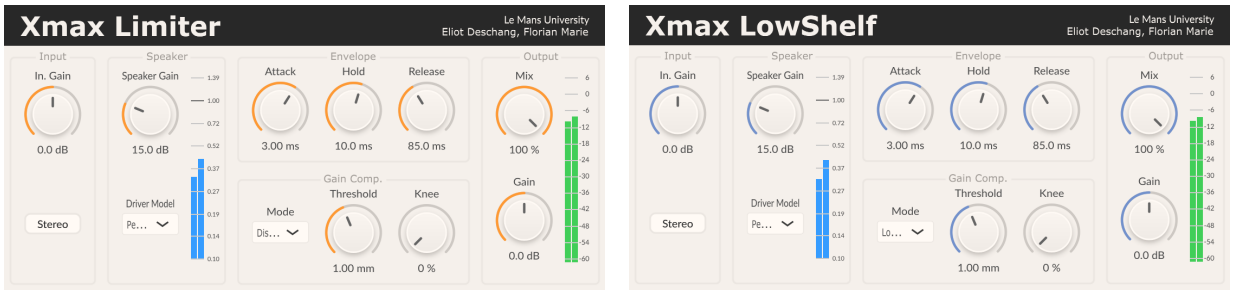


Figure 8: Plugin implementation of both feedback algorithms.

This audio plugin implements in reality the feedback diagram of Fig. 4, but setting the Look Ahead potentiometer to 0 ms allow to use the feedback diagram of Fig. 2. On both 3 plugin GUI's, a speaker section is present to allow the user to set the gain of the audio chain, to choose a driver model among a given list, and monitor the displacement level through a blue level meter. The time constants of the envelope, as well as the threshold limit and the look ahead time (delay block of Fig. 2) are controllable in the control chain section. Finally, the output section presents a dry/wet setting (mix knob) and output gain. The GUI of the plugins implementing the feedforward algorithms are shown in Fig. 9.



(a) Limiter.

(b) Dynamic low-shelf filter.

Figure 9: Plugin implementation of both feedforward algorithms.

Unlike the plugin implementing the feedback approaches, these two plugins feature an envelope section for controlling the side-chain's time constants, and a gain computation section for controlling the threshold, as well as the knee width (see Appendix G) used by the gain computer block of Fig. 5 and Fig. 7. The limiter plugin also features a mode for bypassing reciprocal filters, while the low-shelf plugin can replace the low-shelf filter with a broadband gain. All plugin codes are available at [21].

6.2 Algorithm validation setup

As this work explores two different implementation of the discussed algorithms above, two different measurement setups sharing the same basis were utilized, as illustrated in Fig. 10, to assess algorithms performances.

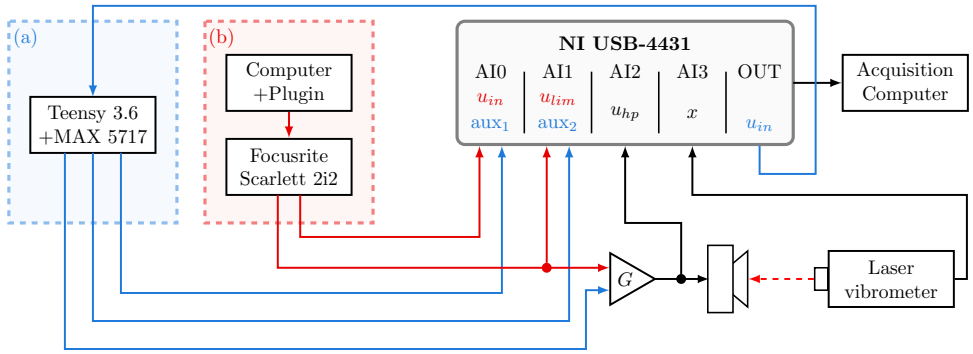


Figure 10: Algorithm test setup using an embedded solution (a) (here Teensy 3.6 plus dedicated serial SPI DAC) or plugins (b).

The measurement setup, is composed of a National Instrument (NI) USB-4431 acquisition board, carrying four analogue input channels and one output. The sampling rate is set at $F_s = 48$ kHz. The NI board is driven by a dedicated acquisition computer running Python scripts. The clamped woofer is driven by a Canford compact power amplifier (20-311). The displacement of the driver membrane is measured by a Polytec OFV 3000 vibrometer using the displacement decoder. Voltage at the driver terminals and output of the vibrometer are recorded as u_{hp} and x respectively. See appendix I for pictures of the setup. Both setup particularities are detailed below.

Case (a) - Embedded solution setup

Concerning the measurements conducted with the embedded solution setup (a) in Fig. 10, the N.I board output plays an audio sequence composed of various test signals. The Teensy microcontroller's default 12-bit DAC provides auxiliary outputs: aux_1 for the compliance ratio C and aux_2 for the estimated displacement x_{est}, enabling monitoring and validation. The processed voltage signal, generated via the MAX 5717 DAC (16-bit resolution through SPI protocol), is sent to the driver through the amplifier, as the Teensy's default DAC resolution is limited to 12 bits.

Case (b) - Plugin-based solution setup

In the case of plugin-based solution, the output of the N.I is not used as the audio sequence is played by the computer within a DAW, thought a Focusrite Scarlett 2i2 second generation sound card. The headphone output is used, where the left channel send the unprocessed input u_{in}. The right channel output the processed signal u_{lim} obtained via the plugin.

6.3 Driver parameters measurement

To evaluate the four algorithm performances, measurements are carried out on a 5-inch woofer driver whose reference is Peerless SDS-P830656. Both linear and non-linear parameters are measured using a Klippel measuring bench, composed of Distortion Analyser 2, a displacement sensor (Panasonic LM10 ANR1282 laser) and power amplifier Dynacord SL 1800. The linear parameters, summarized in Tab. 1, were used as input for the displacement models in all algorithms.

Parameter	**Symbol**	**Value**	**Unit**
Electrical			
DC resistance	R_{ec}	7.00	Ω
Voice Coil Inductance	L_{ec}	0.515	mH
Mechanical			
Moving part resistance	R_{ms}	1.45	$\mathrm{kg\,s^{-1}}$
Moving part mass	M_{ms}	10.0	g
Moving part compliance	C_{ms}	595	$\mu\mathrm{m\,N^{-1}}$
Coupling			
Force factor	Bl	5,59	$\mathrm{T\,m^{-1}}$

Table 1: Measured linear parameters of the Peerless HDS-P830860 woofer driver.

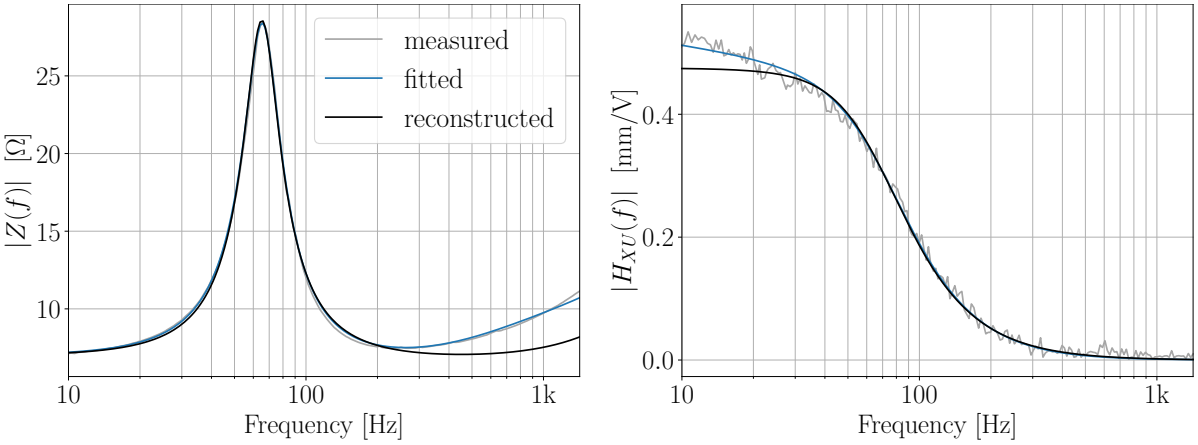


Figure 11: Electrical impedance modulus (left) and displacement to voltage transfer function modulus (right). The measurement and fit (performed with creep modelling) are realized by the Klippel bench. The reconstructed curves are based on the linear parameters of Tab. 1.

7 Results

In this section, the results of the measurements are presented. First, the measured displacement is compared to its estimation, from u_{in} and u_{hp}. The algorithm performances are then compared for various excitation signals. The algorithm time parameters, gathered in Tab. 2, are left constant during measurements.

Algorithm	t_a [ms]	t_h [ms]	t_r [ms]
Feedback	3	-	500
Delayed Feedback	3	-	500
Limiter	6	10	85
Low-shelf	6	10	85

Table 2: Time constant parameters t_a (attack), t_h (hold) and t_r (release) used for the different algorithms for all the measurement realized in this work.

7.1 Displacement comparison

In order to access algorithm performances, it is first needed to access the accuracy of the estimated displacement. Overestimated displacement leads to an overprotection. The opposite happens with underestimated displacement.

To assess displacement estimator accuracy, measurements were conducted using the embedded solution setup, where the Teensy microcontroller outputs the estimated displacement through auxiliary output 2, knowing amplifier gain G and using the input quantity u_{in}. Since all algorithms share the same displacement estimation method, it was considered reasonable to conduct this measurement only once with this setup. A reverse exponential swept sine signal ranging from 10 kHz to 20 Hz is used as a test signal. The advantage of such a signal is that it is easy to link the temporal and frequency domain, as represented in Fig. 12, where both time and instantaneous frequency related to the signal x are represented. Another comparison based on a music signal is shown in Appendix. J.1.

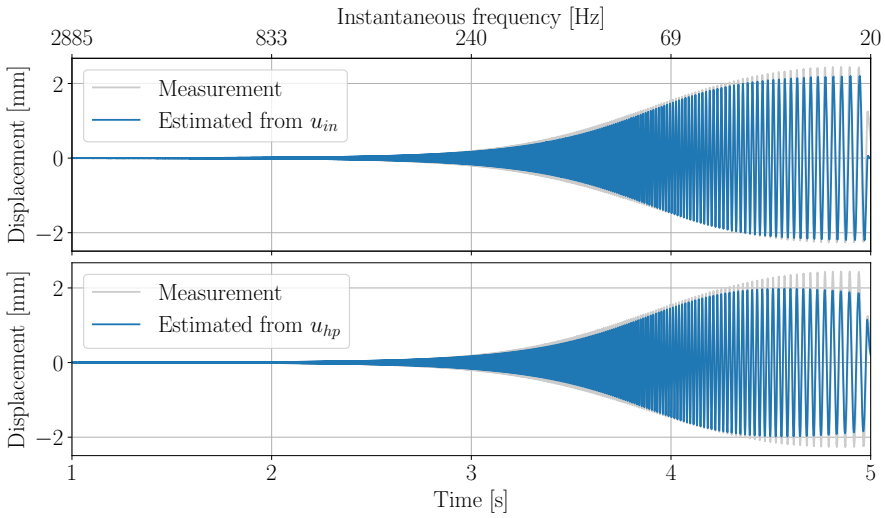


Figure 12: Comparison between the measured displacement x and the estimated displacement on a reverse swept sine signal, derived from (top) u_{in} using the Teensy microcontroller with the amplifier gain G and (bottom) post-processed from u_{hp}.

From Fig. 12, the displacement is a bit underestimated, comparing the measurement and both estimations from u_{in} and u_{hp}. It is worth noting that the estimations from u_{in} differs from the one based on u_{hp}, which have been realized applying $H_{XU}(z)$ filter on u_{hp} in post-processing. This

shows that the amplifier frequency response drops a bit at low frequency, which is actually common. The reason why the measured displacement is higher at low frequency than the estimation from u_{hp} can be explained by the creep effect, inherent to the viscoelastic properties of the driver suspension. This is confirmed by the measurement from the Klippel bench, depicted in Fig. 11, highlighting the creep effect increasing the displacement below 30 Hz.

In this work, the viscoelastic properties of the driver suspension and the amplifier's low-frequency cut-off appear to compensate effectively for this excitation level, resulting in reasonably accurate displacement estimation from u_{in}. However, if the amplifier response or the viscoelastic properties of the suspension were less favourable, it might be necessary to replace the gain block with the amplifier's transfer function and/or incorporate additional poles and zeros at low frequencies to account for creep effect.

7.2 Algorithm performances comparison

As for the displacement estimation measurement, a reverse swept sine signal ranging from 10 kHz to 20 Hz is used as a test signal (see Fig. 13). On all the following figures of this section, the grey signals represent the measured displacement when the corresponding algorithm is bypassed. Please note that due to the delay line in limiter, low-shelf and feedback algorithms, the displayed signals have been time shifted during the data processing stage.

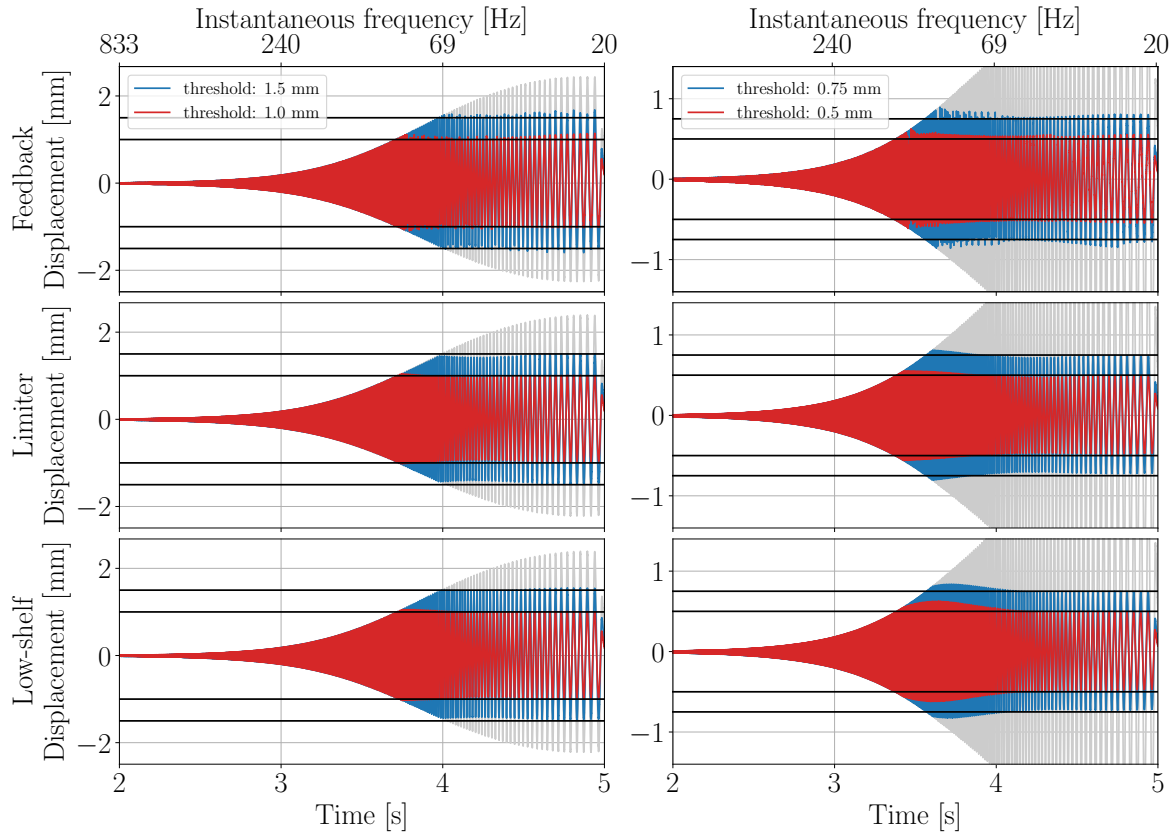


Figure 13: Measured membrane excursion with a chirp. Feedback (top plots), limiter (middle plots) and low-shelf (lower plots) algorithms are tested, for threshold set at 1.5 and 1.0 mm (left plots) then 0.75 and 0.5 mm (right plots).

In general, the three algorithms limit the signal between the X_{max} target. Overshoot of the limit occur with the feedback for threshold set at 1.5 and 1.0 mm. This is even more explicit with threshold of 0.75 and 0.5 mm. The feedback algorithm is also the only one to show a non-uniform attenuation, characterize by small peaks resulting from oscillations between the attack and release phases of the compensation filter. This is avoided by feedforward algorithms through the usage of the hold function/parameter.

Although small overshoot occur with threshold of 0.75 and 0.5 mm, the limiter is the one limiting the more accurately the membrane excursion. The low-shelf provide a good control for threshold of 1.5 and 1.0 mm, but exhibit the largest overshoot for threshold of 0.75 and 0.5 mm. This is due to the cut-off frequency of the low-shelf filter that is set equal to twice the loudspeaker resonance frequency, i.e. $f_r = 138$ Hz. As a result, frequencies above this point cannot be adequately attenuated.

In addition to the parametric study on the threshold, a similar measurement, detailed in Appendix J.2, was conducted by varying the amplifier gain while keeping a constant threshold.

To evaluate algorithm performances in more realistic conditions, two music signals are then used. The first one is *ShookOnes Pt.III* by Mobb Deep & Nick Morgan. This music exhibit heavy low frequency content on its chorus, in a repetitive pattern. This can highlight the potential negative pumping effect of algorithms. The second one is *Take My Breath* by The Weeknd. This one exhibits a less regular pattern with tights transients, pushing algorithms at their extreme. Results with *ShookOnes* and *Take my breath* are first introduced in the following figures, for threshold of 2.0 and 1.5 mm, 1.0 and 0.75 mm.

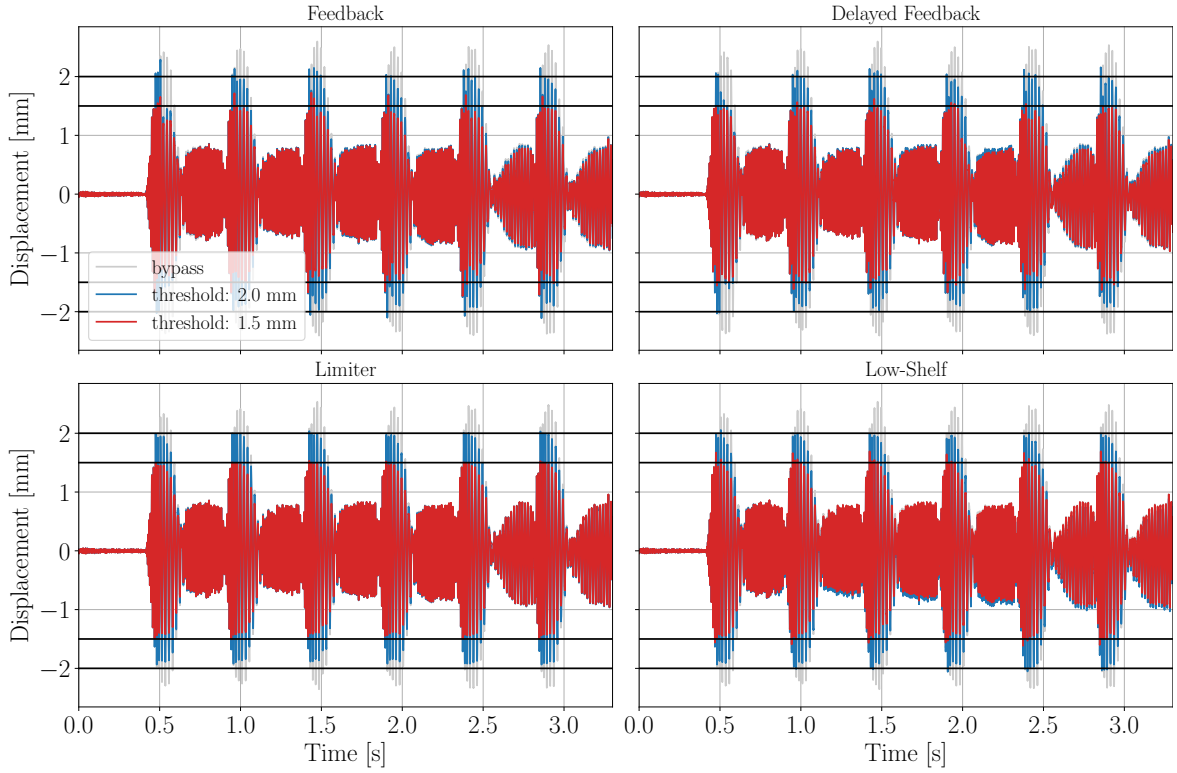


Figure 14: Measured membrane excursion with *ShookOnes* and all algorithm. The threshold is set at 2.0 and 1.5 mm.

From Fig. 14, the same trend as the one discussed earlier with the chirp signal is observable: the classical feedback exhibits few overshoots, the limiter is very accurate and the low-shelf struggle to limit some transient compare to the limiter. The delayed feedback is tested and results show that most of the overshoot are attenuated compared with the classical feedback. For lower threshold values on the same track, the previous observation remains valid, except that the delayed feedback get less precise (see Appendix J.3).

Algorithms are then tested on an extract of *Take My Breath*. From Fig. 15, the interest of the delayed feedback becomes now obvious, as it avoid the feedback structure to miss out on fast transients, leading to a better protection, for both thresholds of 1.5 and 1 mm. However, in general, feedback algorithm lower the signal dynamic more than needed. This is highlighted

by the signal dynamic from Fig. 15 between 0.5 and 1.0 sec. This is due to the release time parameters, set higher for the feedback than limiter (see Tab. 2). Feedback algorithms require a longer release setting to lower ripples on the C-ratio and insure stability. For the lowest threshold of 0.75 mm the previous observation remains valid (see Appendix J.3).

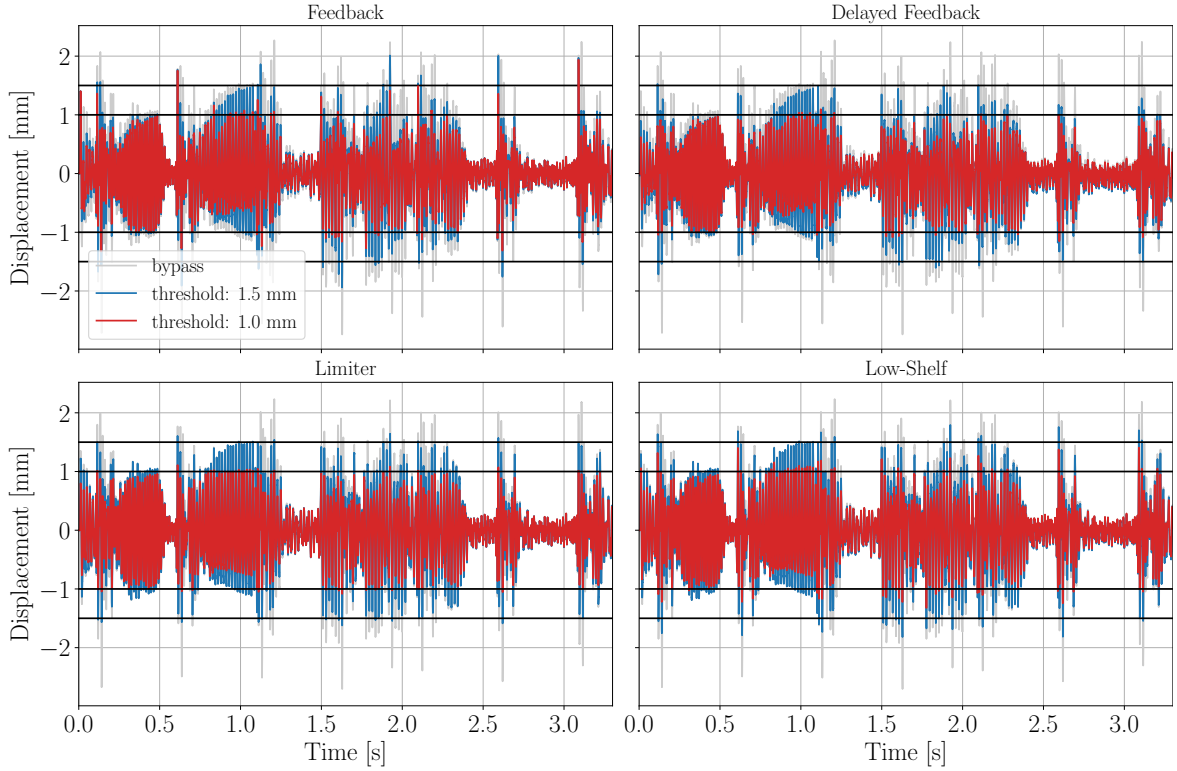


Figure 15: Measured membrane excursion with *Take My Breath* and all algorithm. The threshold is set at 1.5 and 1.0 mm.

7.3 Total harmonic distortion comparison

This section examines the harmonic distortion introduced by each algorithm on the displacement signal at a given frequency. A 40 Hz sine wave, played for 3 seconds, serves as the excitation signal. The signal is calibrated to produce a peak amplitude of 5.6 V at the loudspeaker terminals when the protection algorithms are bypassed, resulting in a peak displacement of approximately 2.85 mm. A 2.0 s analysis window is applied, starting 0.5 s after the sine wave begins, in order to exclude the step response of the system from the spectrum visualization.

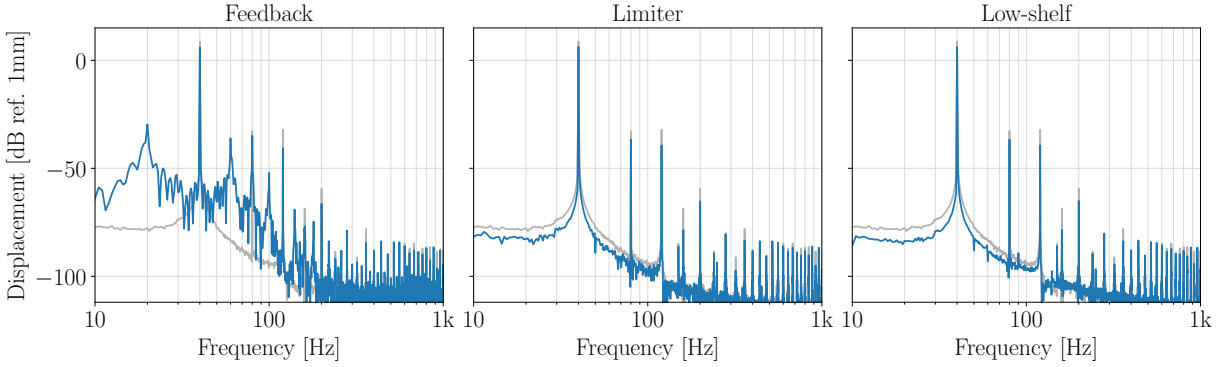


Figure 16: Spectrum of the measured displacement for feedback, limiter and low-shelf algorithms, and threshold of 2.0 mm (blue curves). Excitation signal is a 40 Hz sine of peak amplitude 5.6 V on loudspeaker terminals when protection algorithms are bypassed (gray curves).

Fig. 16 shows the spectrum of the measured displacement x for the three main algorithms and a threshold (X_{max}) of 2.0 mm. The same analysis for threshold of 2.5, 1.5 and 1.0 mm are depicted in Appendix J.4 with u_{lim} and u_{hp} spectra. The feedback algorithm exhibits higher noise level at low frequencies compared to bypass for all threshold values, likely due to the criteria used to activate/deactivate the compensation filter (based on absolute displacement value). For this reason, distortion appears on the spectrum of the feedback algorithm, for threshold of 2.0 mm, where multiple of frequency 20 Hz are introduced. In general, the feedback algorithm seems less predictable in terms of artefact that it introduces, compare to the limiter and low-shelf. Concerning the feedback-based algorithm, performance improvements might be achieved by adopting a different comparison criterion, such as an envelope or peak follower of the displacement signal, instead of relying on its absolute value.

For the different algorithms, the Total Harmonic Distortion (THD) corresponding to u_{lim}, u_{hp} and x spectra for the different thresholds are computed and results are gathered in Tab. 3. The THD is computed as

$$\text{THD} = \frac{\sqrt{\sum_{n=2}^{10} a_n^2}}{\sqrt{\sum_{n=1}^{10} a_n^2}}, \tag{26}$$

where a_n is the amplitude of the n_{th} harmonic component, harmonic 1 being the fundamental.

		Threshold [mm]				
Algorithms	**Quantity**	**-**	**2.5**	**2.0**	**1.5**	**1.0**
Feedback	u_{lim}	-93	-65	-48	-45	-29
	u_{hp}	-71	-66	-48	-45	-29
	x	-38	-39	-40	-38	-33
Limiter	u_{lim}	-93	-80	-80	-81	-81
	u_{hp}	-71	-72	-73	-75	-76
	x	-38	-39	-41	-43	-46
Low-shelf	u_{lim}	-93	-91	-94	-87	-95
	u_{hp}	-71	-72	-74	-74	-76
	x	-38	-39	-41	-43	-46

Table 3: THD (dB ref 100%) for the limited signal u_{lim}, the loudspeaker terminal signal u_{hp} and the displacement signal x, evaluated as a function of the displacement thresholds, for a 40 Hz sinusoidal signal of peak amplitude 5.6 V on loudspeaker terminals when protections are bypassed (denoted by '-' threshold).

The primary objective of these algorithms is to reduce the excursion of the driver's membrane. Since a driver's harmonic distortion increases with greater membrane excursion, it becomes evident that even though the algorithms themselves may introduce some distortion through the voltage signal u_{lim} and so u_{hp}, the overall distortion on the displacement can still be significantly reduced. The reduction in distortion achieved by protecting the driver outweighs the minor artefacts introduced by the algorithms, leading to a net improvement. However, it should be noted that for the feedback algorithm, beyond a certain threshold, the THD cannot be reduced further and ends up to increase on x, as the algorithm itself ends up introducing a non-negligible distortion on the voltage signal u_{lim} - and so on u_{hp} - for threshold of 1.5 and 1.0 mm. This increase in distortion as the threshold is decreased is avoid by all feedforward algorithms, which take advantage of the hold function, allowing to maintain a constant reduction gain $g[n]$ over the measurement time (only valid for this type of excitation). Nevertheless, this increase in distortion might not be as bad as it seems, as the generated distortion could create a psychoacoustic effect that virtually enhances bass presence, even though the protection mechanism reduces the actual displacement.

Also, one should note that for the limiter, the signals u_{lim} and u_{hp} exhibit higher noise levels compared to the low-shelf algorithm at the same threshold values (see Fig. 30 and 31 in Appendix J). This appears to stem from the addition of reciprocal filters H'_{XU} (low-pass) and H'_{UX} (high-pass). Consequently, high-frequency content is attenuated and then re-amplified, potentially amplifying noise at high frequencies. While quantization error should theoretically be negligible with 32-bit floating-point processing (as used in the plugin), a noticeable noise increase is observed. Further investigation is required to determine whether this noise originates from floating-point precision issues or the buffer structure of the biquad filters.

It is worth noting that the vibrometer can also add distortion, because of quantisation resolution, as its sensitivity is not adjusted during measurements and that the displacement range of the membrane is reduced when threshold decrease. Finally, it should be clear that the test signal for measuring the THD is not realistic as protection algorithms are usually used to catch transients, and limit displacement supposed to reasonably overshoot the target threshold. In other word, if the sound system is initially well design, the protection algorithms are not supposed to remove a significant part of the signal. It is possible to use these algorithms in such extreme condition, but lowering the power amplifier gain is a much better choice.

The authors' observation is that all the algorithms lower the low frequency content of any music signal, the low-shelf and feedback ones being the most transparent algorithms, and the limiter the one introducing the most noticeable *pumping effect*.

8 Conclusion

Limiting electrodynamic driver's membrane excursion is a topic whose interest is growing, motivated by the evolution of audio content during the last decades as well as by user demands in terms of low frequencies content to reproduce. The preliminary state-of-the-art study confirmed that the most common approaches are based on a variable gain in time, and dynamic high-pass filter which are applied to the voltage signal sent to the driver to prevent large excursion. Dynamic filters, as well as physics behind electrodynamic drivers, were then investigated in order to meet the objective. Throughout this project, two algorithms were designed using a feedback structure, while two others were developed following a feedforward approach. The four algorithms were implemented as user-friendly plugins, enabling anyone to test them on their own. Feedback algorithms were also implemented in a Teensy 3.6 microcontroller in order to test them on embedded hardware. All of these algorithms were tested in realistic condition to access their performances.

Each of the algorithms introduced in this project were based on an estimator of the driver's membrane displacement, which allow to estimate the displacement knowing the loudspeaker terminals voltage. This estimator is based on the linear model of a driver, i.e. its Thiele Small parameters, which are easy to assess. The strength of this type of algorithm also represents its main limitation: by eliminating the need for a physical sensor to estimate displacement, it becomes entirely dependent on the accuracy of its estimator model. Accordingly, an over estimation of the membrane displacement would lead to an overprotection process. The opposite would happen for an underestimated displacement.

The feedback algorithm virtually modify the mechanical parameters of the driver. In this way, the driver compliance and viscous losses are modified, resulting in an adapted driver sensitivity at low frequency. Unlike protection equalizer such as high-pass filter, based on varying cut-off frequency between arbitrary limits, the feedback algorithm parameters range are set in an optimal way, knowing the input signal amplitude and amplifier gain. An advanced version of this algorithm incorporating a delay line (look ahead) on the input allows to reduce excursion overshoot, while introducing a global delay of few milliseconds on the system.

Concerning the feedforward algorithms, the first one developed was based on a limiter algorithm, that prevent mechanical overload by directly applying a gain on the estimated displacement signal instead than on the voltage signal, as it is commonly realized in some patents [17, 18]. The second algorithm developed was based on a dynamic low-shelf filter which can be seen as a modification of a classical voltage-based limiter, where the gain block was replaced by a low-shelf filter.

To access algorithms performances, a chirp, pure tone and two music signals have been played on a real driver, whose parameters were measured. Its membrane displacement was measured using a vibrometer. In this study, threshold (X_{max}) and input level were set as parametric inputs.

After comparison, it appeared that the feedback algorithm were less predictable in terms of performances, as X_{max} overshoot must occur to activate the protection. Based on the spectrum analysis of the measured displacement signal for a sine signal, they also introduce higher harmonics compared to feedforward implemented algorithms which benefit from a hold parameter. The main advantage of the feedback without delay line is that no latency is introduced by the protection system, which is the main inconvenient of feedforward-based algorithms. However, feedforward-based limiters are more precise thanks to the look ahead delay, which guarantees, in theory, no overshoot of the X_{max}. With the feedforward dynamic low-shelf, overshoot can occur if some of the frequencies responsible for the exceed in displacement are higher than the one of the low-shelf filter frequency cut-off. Transients parts of a music signal are commonly responsible for overshoots, as their spectrum are broad. However, its advantage - with respect to the limiter-based algorithm - is that only the low part of the voltage spectrum is modulated in amplitude, mitigating the *pumping* effect which can be heard with the feedforward-based limiter.

As future improvements, using an envelope follower on the estimated displacement, instead of its absolute value, could improve feedback algorithms stability. Adjusting automatically the time parameters, notably the release time, could help to lower the *pumping* effect from the limiter. This is referred to as an *auto-release* parameter on some compressor/limiter [22]. Another major alternative in the way of evaluating algorithm performances would be to conduct a perceptive study, involving volunteers listening audio files and rating which one their prefer depending on a given criteria or situation. This could also allow to understand the acoustic pressure evolution as a function of the threshold.

A Radiation impedance

- Volumic mass of the air: $\rho_0 = 1,204$ kg/m^3 ;
- Adiabatic gaz constant: $\gamma = 1,4$;
- Membrane radius: a;
- Wave number: $k = \frac{\omega}{c}$. ω is the angular frequency and c the sound celerity.

Radiation impedance

Here, the assumption is made that the driver is a flat rigid piston, which is positioned in an infinite baffle. Obviously in practice, this can never happen, but this hypothesis still remains fair as long as the baffle dimensions are big with respect to the driver membrane radius, and as long as edge effect (diffraction) can be avoided.

For a baffle piston of radius a, the radiation impedance is given by

$$Z_{BP} = \frac{\rho_0 c}{\pi a^2}\left(1 - \frac{J_1(2ka)}{ka} + j\frac{S_1(2ka)}{ka}\right), \tag{27}$$

where J_1 is Bessel function of the first kind, of order 1. S_1 is Struve function. In low frequency ($ka \ll 1$), the radiation impedance is written as

$$Z_{LF} = \frac{\rho_0 c(ka)^2}{2\pi a^2} + j\frac{8\rho_0 cka}{3(\pi a)^2}, \tag{28}$$

which can be simplified as

$$Z_{LF} \approx \frac{\rho_0 \omega^2}{2\pi c} + j\omega\frac{0.85\rho_0}{\pi a}, \tag{29}$$

as $8/(3\pi) \approx 0.85$. It is thus possible to define R_{rad} and M_{rad} such as

$$Z_{LF} = R_{rad}(\omega) + j\omega M_{rad}, \tag{30}$$

with $R_{rad}(\omega) = \frac{\rho_0\omega^2}{2\pi c}$ and $M_{rad} = \frac{0.85\rho_0}{\pi a}$. The expression of front and rear radiation impedance in the lumped element diagram of Fig. 1 can thus be expressed as $Z_{af} = Z_{ar} = R_{rad} + j\omega M_{rad}$.

B Bilinear transform of second order transfer functions

To implement any of the proposed algorithm on an embedded system such as Teensy, it is either possible to call a `bilinear` function from a library, or to process the transformation explicitly. As the maximum degree of transfer function is two in the frame of this report, it is here convenient to derive the transformation analytically. The goal is then to express the digital filter coefficient b_i and a_i of a general digital filter

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}, \tag{31}$$

from the analogue filter coefficients B_i and A_i

$$H(s) = \frac{B_0 s^2 + B_1 s + B_0}{A_0 s^2 + A_1 s + A_0}. \tag{32}$$

Applying the bilinear transform [16]

$$s \leftarrow 2F_s \frac{1 - z^{-1}}{1 + z^{-1}}, \tag{33}$$

to Eq. (32) lead to the following assignation

$$\begin{cases} b_0 = B_0 4F_s^2 + B_1 2F_s + B_2, \\ b_1 = -8B_0 F_s^2 + 2B_2, \\ b_2 = B_0 4F_s^2 - B_1 2F_s + B_2, \end{cases} \qquad \begin{cases} a_0 = A_0 4F_s^2 + A_1 2F_s + A_2, \\ a_1 = -8A_0 F_s^2 + 2A_2, \\ a_2 = A_0 4F_s^2 - A_1 2F_s + A_2. \end{cases} \tag{34}$$

It is worth noticing that a_i and b_i coefficients only depends on A_i and B_i ones, respectively.

C Mapping function

The mapping function shown in Fig. 17 stand a relation between the compliance ratio $C = \frac{C_{ms}^{(comp)}}{C_{ms}}$ and the quality factor $Q_s^{(comp)}$, this last quantity enabling $R_{ms}^{(comp)}$ computation in the case of a resonant speaker.

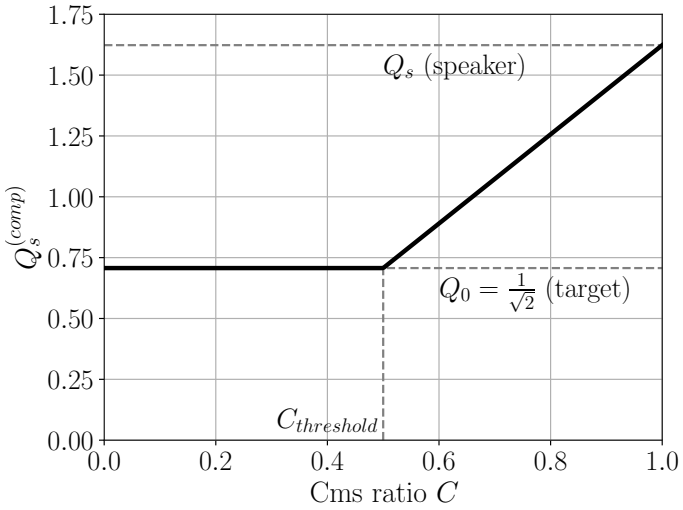


Figure 17: Mapping function.

The mathematical expression of the mapping function is a piece-wise function defined as

$$Q_s^{(comp)} = \max\left(Q_0, \quad \Gamma \cdot (C - C_{tresh}) + Q_0\right), \tag{35}$$

where

$$\Gamma = \frac{Q_s - Q_0}{1 - C_{treshold}}, \tag{36}$$

is the slope of the mapping function for $C \in [C_{treshold}, 1]$. Γ is adjusted such that $Q_s^{(comp)} = Q_s$ at $C = 1$.

D Smoothing

In the frame of the first feedback algorithm presented, a smoothing function is used to derive $C_{ms}^{(comp)}[n]$ as a function of a target compliance $C_{ms^{(target)}}[n]$, which oscillates between two constant values, C_{ms} and $C_{ms,\min}^{(comp)}$. For the sake of clarity, the smoothing function is reviewed here using $f[n]$ to represent $C_{ms}^{(target)}[n]$ and $g[n]$ to represent $C_{ms}^{(comp)}[n]$.

The difference equation for smoothing a function $f[n]$ is given by

$$g[n] = (1-k)\cdot g[n-1] + k\cdot f[n], \quad \text{with} \begin{cases} k = \text{AT}, & \text{if } f[n] < g[n-1], \\ k = \text{RT}, & \text{otherwise}, \end{cases} \tag{37}$$

where AT and RT are the attack and release coefficients, respectively, determined by time constants t_a and t_r. At first glance, it may appear counterintuitive that the attack coefficient applies when $f[n] < g[n-1]$ rather than when $f[n] > g[n-1]$, which is typically associated with an "attack phase" involving an increase in $f[n]$. In this case, however, the smoothing function is applied to $C_{ms}^{(target)}[n]$, and the attack phase is defined as the period when $C_{ms}^{(target)}[n]$ drops to its minimum value, $C_{ms}^{(min)}$, and release phase correspond to the period when $C_{ms}^{(target)}[n]$ goes back to its initial state, i.e. default speaker's C_{ms}, as depicted in Fig. 18). This inverted behaviour results in the observed sign convention for the attack phase.

The corresponding transfer function associated to Eq. (37) leads to

$$H(z) = \frac{G(z)}{F(z)} = \frac{k}{1-(1-k)z^{-1}}, \tag{38}$$

which is a classical one-pole low-pass filter, for which the pole location varies according to k.

D.1 Time constants

If the step response of a continuous-time system is given by

$$g(t) = 1 - e^{-t/\tau}, \quad \forall t \geq 0, \tag{39}$$

where τ is the time constant, its discrete-time counterpart, obtained by sampling the continuous-time step response, becomes

$$g[n] = g(nT_s) = 1 - e^{-nT_s/\tau} = 1 - (z_\infty)^n, \quad \forall n \geq 0, \tag{40}$$

where $z_\infty = e^{-T_s/\tau}$, according to Zölzer's notations [23]. The Z-transform of $g[n]$ is expressed as

$$G(z) = \sum_{n=0}^{\infty} g[n] z^{-n}, \tag{41}$$

$$= \sum_{n=0}^{\infty} z^{-n} - \sum_{n=0}^{\infty} (z_\infty)^n z^{-n}, \tag{42}$$

$$= \frac{z}{z-1} - \frac{1}{1-z_\infty z^{-1}}, \tag{43}$$

$$= \frac{1-z_\infty}{(z-1)(1-z_\infty z^{-1})}. \tag{44}$$

The time constant τ used in the smoothing process is defined in terms of the rise time for the step response to transition from 10% to 90% of its final value. This leads to the conditions

$$g(t_{10}) = 0.1 = 1 - e^{-t_{10}/\tau}, \tag{45}$$

$$g(t_{90}) = 0.9 = 1 - e^{-t_{90}/\tau}. \tag{46}$$

From these equations, t_{10} and t_{90} are derived as

$$t_{10} = -\ln(0.9)\tau, \tag{47}$$
$$t_{90} = -\ln(0.1)\tau. \tag{48}$$

Thus, the attack/release time, $t_{a|r}$, defined as the difference between t_{90} and t_{10}, is

$$t_{a|r} = t_{90} - t_{10}, \tag{49}$$
$$= -(\ln(0.1) - \ln(0.9))\tau, \tag{50}$$
$$= \ln(9)\tau, \tag{51}$$
$$= 2.2\tau. \tag{52}$$

The corresponding discrete-time pole, z_∞, is thus given by

$$z_\infty = e^{-2.2T_s/t_{a|r}}. \tag{53}$$

To design a system implementing this step response, the transfer function $H(z)$ of this digital filter can be obtained by multiplying the Z-transform of the step response, $G(z)$, to the Z transform of the Dirac function (impulse response):

$$H(z) = \frac{z-1}{z}G(z). \tag{54}$$

Substituting $G(z)$, one get

$$H(z) = \frac{(1-z_\infty)z^{-1}}{1-z_\infty z^{-1}}. \tag{55}$$

This corresponds to the transfer function of the smoothing filter (i.e. Eq. (38)), where k would be $1 - z_\infty$ in Eq. (55). Here, the pole z_∞ determines the attack and release times. The coefficients for the time constants associated with attack (AT) and release (RT) are then

$$\text{AT} = 1 - e^{-2.2T_s/t_a}, \tag{56}$$
$$\text{RT} = 1 - e^{-2.2T_s/t_r}. \tag{57}$$

Fig. 18 present an example of the input $C_{ms}^{(target)}[n]$ oscillating between two states, $C_{ms} = 1$ m/N and $C_{ms}^{(min)} = 0.5$. The output of the smoothing function $C_{ms}^{(comp)}[n]$, is shown for time constants $t_a = 30$ ms, and $t_r = 100$ ms.

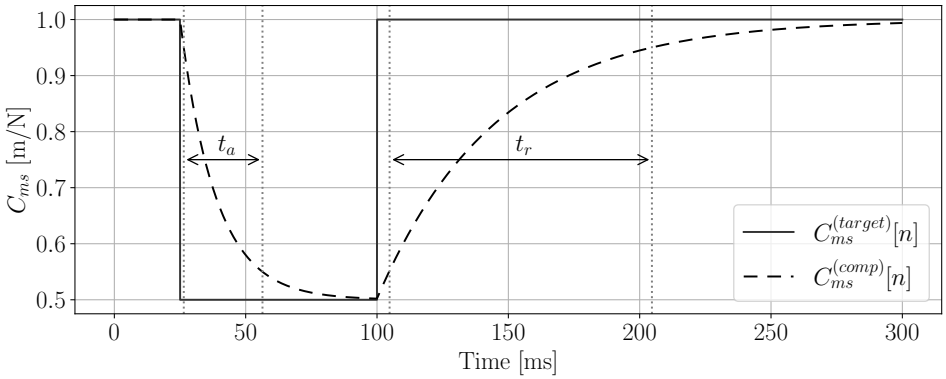


Figure 18: Example of the smoothing filter output $C_{ms}^{(comp)}[n]$ as function of input $C_{ms}^{(target)}[n]$ (oscillating between $C_{ms} = 1$ m/N and $C_{ms}^{(min)} = 0.5$ m/N), for time constants settings $t_a = 30$ ms, and $t_r = 100$ ms.

E Classical limiter approach

A classical voltage-based limiter algorithm structure used in various patents [7, 17] is depicted in Fig. 19. The side-chain depicted in this figure is not something which have been described in these patents, so the implementation may differ from one patent to another. Moreover, the hold parameter doesn't seem to be something that these companies have implemented either. In all cases, this algorithm is based on an estimate of the displacement, and directly applies the compensation gain $g[n]$ which should be applied to this displacement signal to the voltage signal, unlike the algorithm shown in Fig. 5, which applies it to the displacement itself.

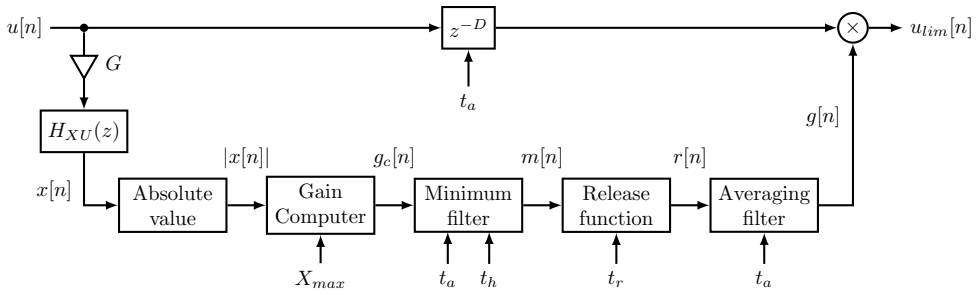


Figure 19: Diagram of a voltage limiter-based algorithm.

A valid question to ask is: does the fact of applying a voltage or displacement signal change anything? In other words, do the two algorithms do exactly the same thing?

The goal of the following calculus is to highlight the difference between the limiter based on displacement control (using reciprocal filters), whose diagram is depicted in Fig. 5, and this limiter version based on voltage control. First, the output u_{lim} of the limiter in Fig. 5 is derived. For all $n > D$

$$u_{lim}[n] = x_{lim}[n] * h_{UX}[n], \tag{58}$$

with h_{UX} the discrete impulse response associated to IIR filter $H_{UX}(z)$, supposing that $H'_{UX}(z) = H_{UX}(z)$. As $x_{lim}[n]$ is the delayed input $x[n]$, multiplied by the output of the side-chain $g[n]$

$$u_{lim}[n] = \big(g[n] \cdot x[n - N_{attack}]\big) * h_{UX}[n]. \tag{59}$$

In the case where $g[n]$ is varying very slowly, it can be approximated as a constant such that it is possible to express $u_{lim}[n]$ as

$$u_{lim}[n] = g[n] \cdot \big(x[n - N_{attack}] * h_{UX}[n]\big) \tag{60}$$

$$u_{lim}[n] = g[n] \cdot \qquad u[n - N_{attack}] \tag{61}$$

which corresponds to the output with the voltage-based limiter. Please note that in both limiter versions, the gain function $g[n]$ is indeed the same as the displacement estimator is yet included within the side-chain of the voltage based limiter.

In conclusion, for stationary signals exceeding the given threshold level, both voltage-based and displacement based limiter should perform almost identically. For musical signals where the limiting process is more susceptible to happen on transients, this should not be the case.

F Stabilization for inverse filtering

In the frame of the limiter-based algorithm, it is essential to estimate the displacement using a displacement filter before applying the limiter and subsequently converting it back into a voltage signal through an inverse filter. However, this conversion cannot rely on the $H_{XU}(z)$ filter used in the feedback algorithms and in the dynamic low-shelf algorithm. The reason is that $H_{XU}(z)$ has zeros typically have zeros on the unit circle in the z-plane, which would result in an inverse filter, $H_{UX}(z)$, with poles on the unit circle, rendering the filter unstable/marginally stable.

Since the objective is to design an inverse filter such that the product of the two filters results in a unity gain filter with a pure delay (due to the delay line between them) when no displacement limitation is required (i.e., $g[n] = 1$), it is not possible to simply relocate the poles of the inverse filter inside the unit circle. Instead, the zeros of the original filter $H_{XU}(z)$ must first be adjusted accordingly to ensure the stability of the inverse filter, as they will be the poles of the latter. The stabilization procedure is detailed below.

First, the poles and zeros of the z-domain transfer function of the displacement filter must be assessed, as it is needed to apply a transformation on these zeros. The displacement filter $H_{XU}(z)$, expressed as a second order transfer function

$$H_{XU}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}, \tag{62}$$

can be rewritten in its pole-zero form as

$$H_{XU}(z) = q\frac{(z - z_0)(z - z_1)}{(z - p_0)(z - p_1)}, \tag{63}$$

where p_0 and p_1 denote the poles, z_0 and z_1 denote the zeros, and $q = b_0$ is the gain. These zeros and poles are easily derived by looking for the root of the second order polynomial function of Eq.(62) of the numerator and denominator, respectively, which are quite handy to obtained as exact solution can be found for second order polynomial.

After getting the poles and zeros of H_{XU}, the zeros are putted inside the unit circle, by applying a transformation on them. This transformation, as far as it is known by the authors, haven't been presented in any book/article - either because it is trivial or because it is not relevant enough. This transformation consists in setting that the new zeros z'_i as

$$z'_i = (1 - \alpha)\frac{|z_i|}{z_i^*}, \quad \text{or} \quad z'_i = (1 - \alpha)e^{i\arg(z_i)}, \tag{64}$$

where $z_i = z_0,\ z_1$ are the zeros of $H_{XU}(z)$, where * denote the conjugate and where $\alpha \in [0, 1]$ is the stabilization coefficient (see Fig. 21 for further understanding). The new tension to displacement filter using the new zeros inside the unit circle is:

$$\hat{H}_{XU} = q\frac{(z - z'_0)(z - z'_1)}{(z - p_0)(z - p_1)}. \tag{65}$$

A consequence of putting the zeros inside the unit circle of the z-domain, is that it change the response of the filter, affecting both its magnitude and phase (see Fig. 20 (a)). To maintain an accurate estimation of the displacement magnitude plateau at low frequencies, one have to apply a compensation gain on this modified filter in order to preserve the same magnitude plateau at low frequency as the original one. To do so, the difference in magnitude of the new filter $\hat{H}_{XU}$ with respect to the original one H_{XU} is computed at $z = 1$ (i.e. $f = 0$ as $z = e^{j2\pi f/Fs}$) on the z-domain such that

$$\Delta_q = \frac{|H_{XU}(z = 1)|}{|\hat{H}_{XU}(z = 1)|} = \frac{z_0 z_1 - z_0 - z_1 + 1}{z'_0 z'_1 - z'_0 - z'_1 + 1} \tag{66}$$

correspond to the gain difference between the two filters at $f = 0$ (as $z = e^{j2\pi f/Fs}$). The final corrected tension to displacement transfer function can thus be expressed as

$$H'_{XU} = \Delta_q \cdot q \cdot \frac{(z - z'_0)(z - z'_1)}{(z - p_0)(z - p_1)} \tag{67}$$

and the reciprocal filter to this one can now be defined as

$$H'_{UX} = 1/H'_{XU}. \tag{68}$$

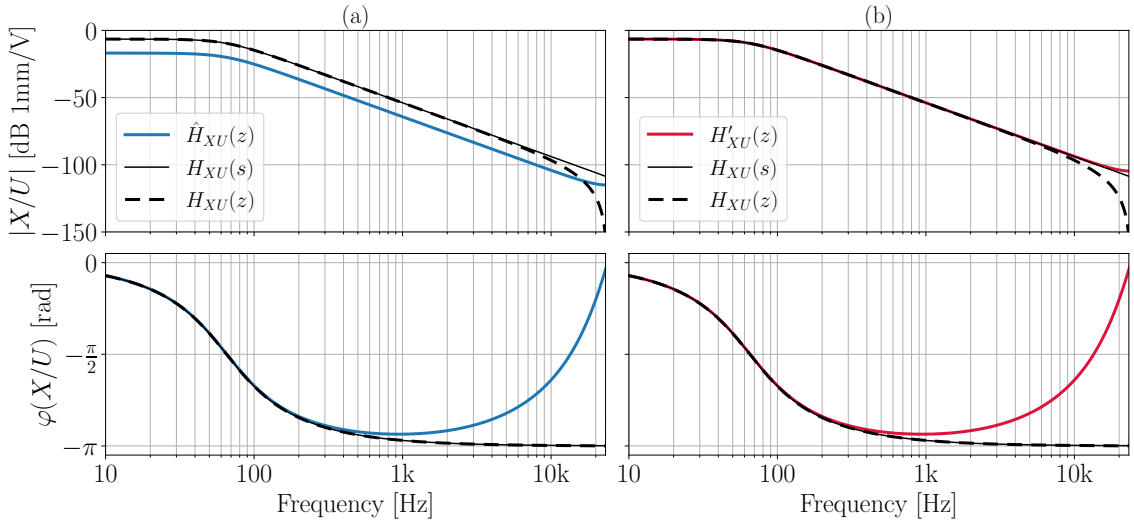


Figure 20: Example of X/U transfer function in the s-domain $H_{XU}(s)$, and in the z-domain obtained using bilinear transform $H_{XU}(z)$, with (a) transformation on the zeros of $H_{XU}(z)$ with $\alpha = 0.9$ leading to $\hat{H}_{XU}(z)$ and (b) additional gain compensation on $\hat{H}_{XU}(z)$ to retrieve the modulus plateau of $H_{XU}(z)$ at low frequency, giving $H'_{XU}(z)$.

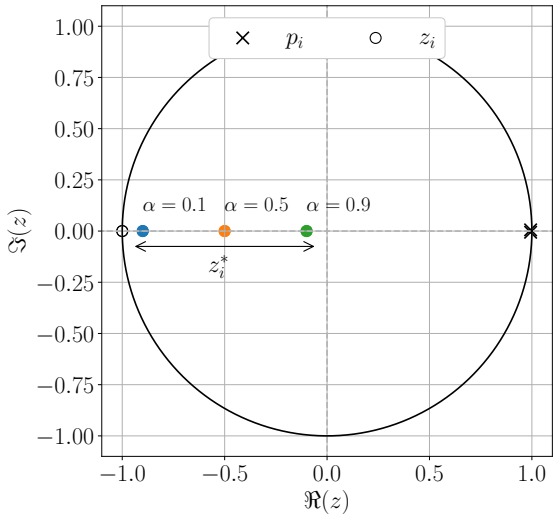


Figure 21: Example of $H_{XU}(z)$ second order transfer function poles p_i and zeros z_i, on the z-plane. $z_i = -1, \forall i$. Additional transformation of these zeros for various stabilization coefficients α leading to z_i^*.

G Gain computer functions

A gain computer, as its name suggests, is a function that determines the gain to be applied to a signal, so that it does not exceed a certain limit, as in the case of a limiter. From a mathematical point of view, a gain function can be calculated by knowing, for an input signal x, the output signal $f(x)$ such that

$$g(x) = \frac{f(x)}{x}, \tag{69}$$

is the gain function. For example, in the case of a brick wall limiter, it is desired by definition to satisfy the condition

$$f(x) = \begin{cases} x, & \text{if } x \leq T, \\ T, & \text{if } x > T, \end{cases} \qquad \Longleftrightarrow \qquad f(x) = \min(x, T), \tag{70}$$

where T is the threshold to not exceed for the output signal. Thus, remembering Eq. 69, the gain function for a brick wall limiter can be deduced dividing Eq.(70) by x, leading to

$$g(x) = \min(1, \frac{T}{x}). \tag{71}$$

This piecewise function can be considered as being too abrupt at the transition between the two sub-intervals (i.e., $x = T$). To address this, it may be desirable to smooth the transition, introducing a supplementary parameter W that defines the width of the "knee" of the function while ensuring the output does not exceed the specified threshold T. The smoothed function can be expressed as:

$$f(x) = \begin{cases} x & \text{if } x < T(1 - W/2), \\ x - \frac{\left[x - T\left(1 - W/2\right)\right]^2}{2WT} & \text{if } 2|x - T|/T \leq W, \\ T & \text{if } x > T(1 + W/2), \end{cases} \tag{72}$$

with $W \in [0, 1]$ and for which $W = 1$ correspond to a smooth transition region with a width equal to T. The gain computer including a knee width parameter can easily be derived using Eq. 69, and is left up to the reader. An example of this function, with varying knee widths, is shown in Fig. 22.

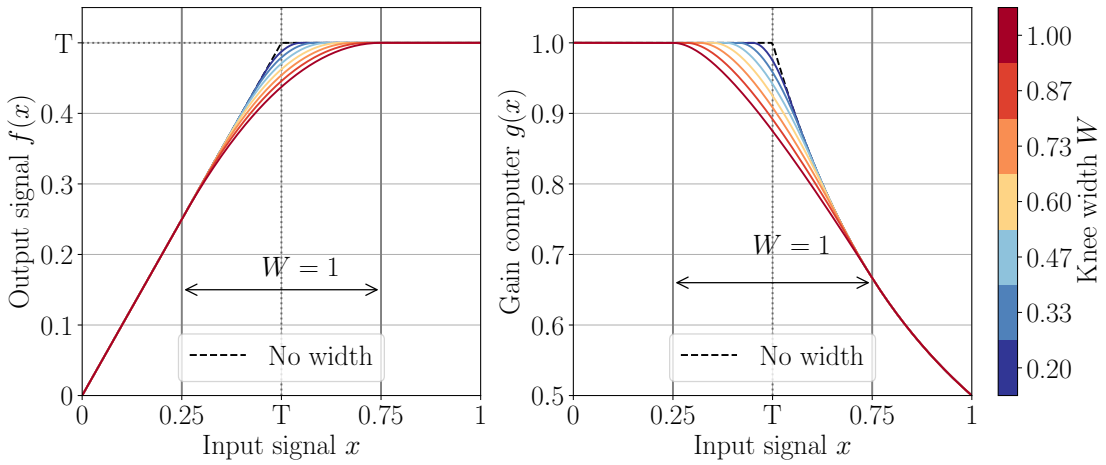


Figure 22: Illustration of the output signal $f(x)$ (left) and the corresponding gain computer function $g(x)$ (right) for a given input signal x, showcasing the effect of varying knee widths W for a fixed threshold $T = 0.5$.

H Driver non-linear parameters

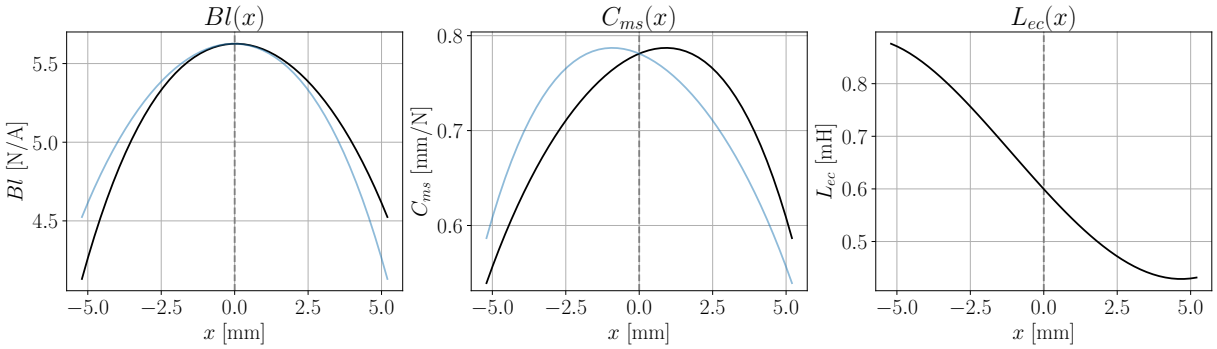


Figure 23: Non-linear parameters Bl, C_{ms}, L_{ec} as function of displacement x, measured using Klippel LSI module.

Coefficient order	Value	Unit
$Bl(x=0)$	5.63	$\mathrm{T\,m^{-1}}$
1^{st}	2.62×10^{-3}	T
2^{nd}	-4.12×10^{-2}	$\mathrm{Tm^{-2}}$
3^{rd}	1.30×10^{-3}	$\mathrm{Tm^{-3}}$
4^{th}	2.53×10^{-4}	$\mathrm{Tm^{-4}}$

Table 4: Coefficients of the polynomial function modelling the force factor Bl as a function of membrane excursion.

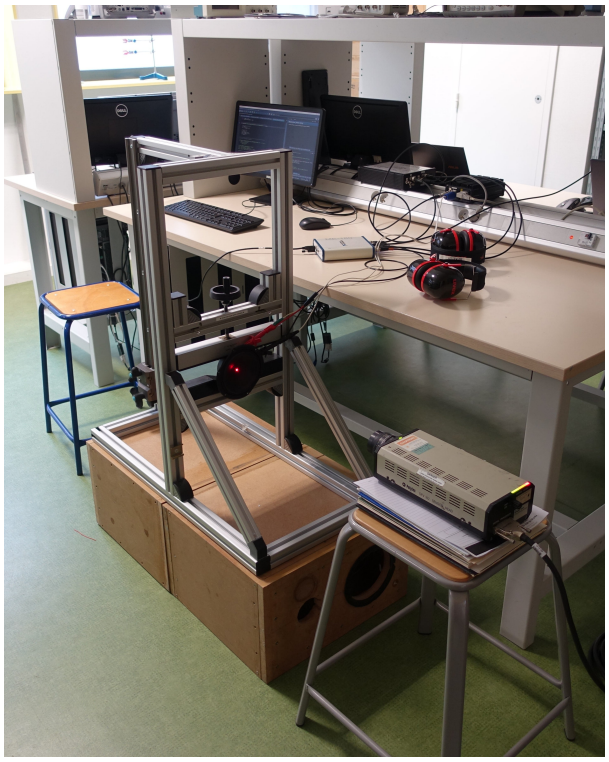
Coefficient order	Value	Unit
$C_{ms}(x=0)$	7.81×10^{-1}	$\mathrm{mm\,N^{-1}}$
1^{st}	1.29×10^{-2}	$\mathrm{N^{-1}}$
2^{nd}	-6.53×10^{-3}	$\mathrm{Nm^{-1}}$
3^{rd}	3.08×10^{-4}	$\mathrm{Nm^{-2}}$
4^{th}	5.67×10^{-5}	$\mathrm{Nm^{-3}}$

Table 5: Coefficients of the polynomial function modelling the compliance C_{ms} as a function of membrane excursion.

Coefficient order	Value	Unit
$L_{ec}(x=0)$	6.00×10^{-1}	mH
1^{st}	-6.12×10^{-2}	$\mathrm{mH\,mm^{-1}}$
2^{nd}	2.42×10^{-3}	$\mathrm{mH\,mm^{-2}}$
3^{rd}	6.84×10^{-4}	$\mathrm{mH\,mm^{-3}}$
4^{th}	-1.56×10^{-5}	$\mathrm{mH\,mm^{-4}}$

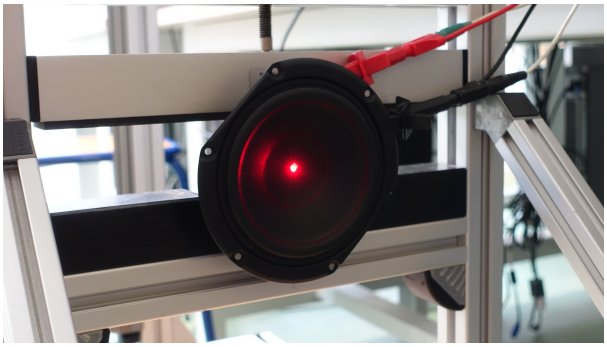
Table 6: Coefficients of the polynomial function modelling the coil inductance L_{ec} as a function of membrane excursion.

I Setup pictures



(a) Global view of the measurement setup with the acquisition computer, National Instrument board, driver stand and laser vibrometer.

(b) Interface of the Polytec OFV 3000 vibrometer controller.



(c) Clamped driver during measurement.

Figure 24: Measurement setup.

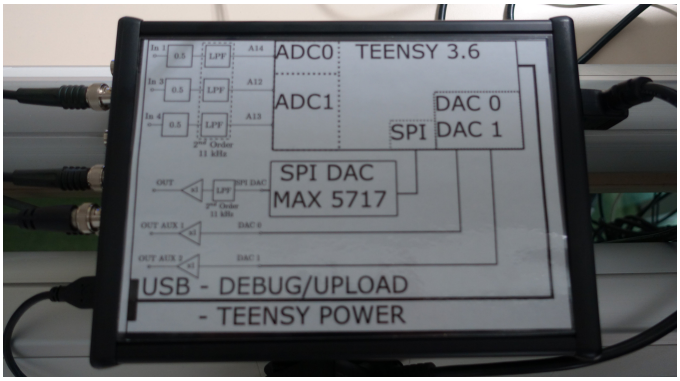


Figure 25: Teensy board.

J Result

J.1 Displacement estimator comparison

Beside the comparison between the measured and estimated displacement from u_{in} and u_{hp} based on a chirp excitation signal, as shown in Fig. 12, the comparison is also carried out based in a music signal (*Take My Breath*), as shown below.

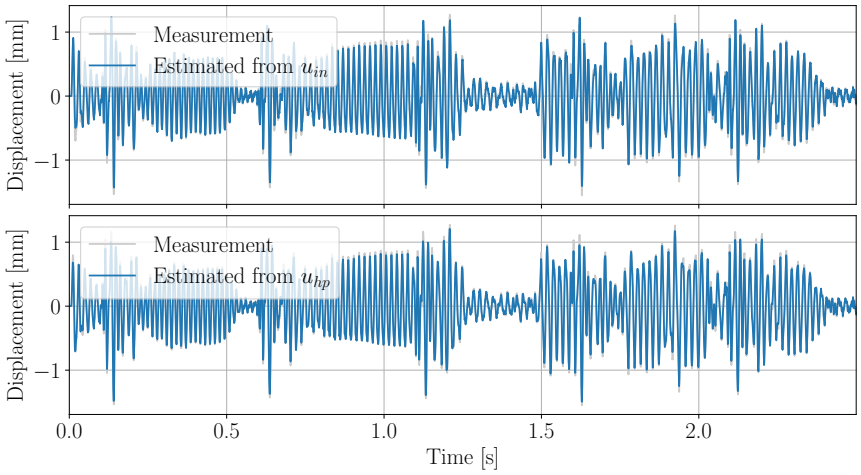


Figure 26: Comparison between the measured displacement x and the estimated displacement on a music signal, derived from (top) u_{in} using the Teensy microcontroller with the amplifier gain G and (bottom) post-processed from u_{hp}.

J.2 Algorithm performances - gain comparison

Fig. 27 compares, for a varying loudspeaker terminal voltage u_{hp}, the displacement between bypass and activated protections with a fixed threshold, here set at 0.75 mm, for all the algorithms. The measurement procedure is realized using the plugin measurement setup which is depicted in Fig. 10 (b). The excitation signal is a reverse exponential swept sine ranging from 1 kHz to 10 Hz. Plugins estimated displacement is calibrated through speaker gain knob, which is set to the corresponding bypass value of u_{hp} in dB (ref 1V), when plugin - and so protection - is deactivated.

To obtain the top and bottom displacement values as a function of frequency for all the different curves, the displacement envelopes were constructed by connecting all local maxima and minima of the time displacement signal. The local extrema were identified by analysing the zero-derivative points of the displacement signal. Additionally, since the instantaneous frequency law of the excitation signal was known, the top and bottom displacement envelopes were mapped over time to their corresponding frequencies, resulting in accurate curves.

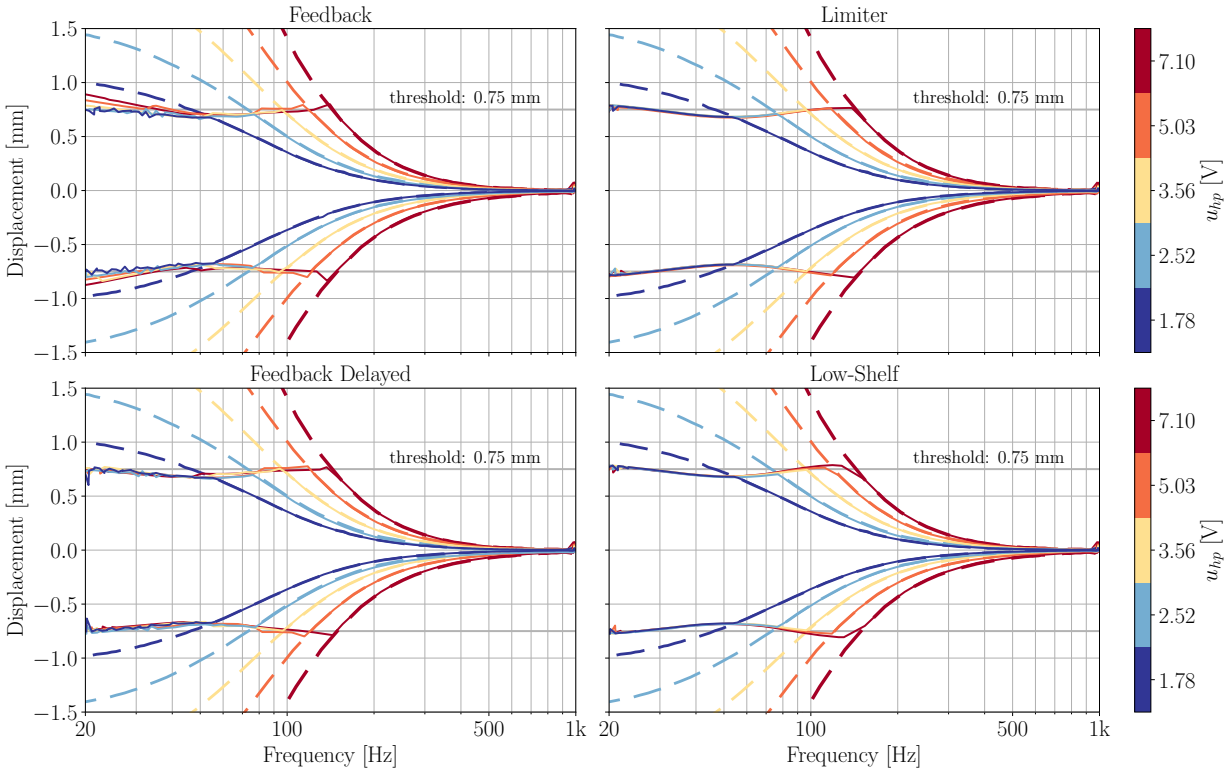


Figure 27: Measured top and bottom displacements as function of frequency for varying driver voltage with protection bypassed (dashed lines) and activated (continuous lines) at a threshold of 0.75 mm, for all algorithm. NB: in bypass mode, for $u_{hp} = 7.1$V, top displacement is around 4 mm at 20 Hz.

As it can be seen, for all algorithms, the same trend is visualized: qualitatively, the displacement is well limited to 0.75 mm, regardless of the initial excitation voltage. Reading more in detail, the limited displacement, for all algorithms, is a bit lower than the threshold around 50 Hz. As this does not depend on the algorithm and on the voltage level, it is probably due to an overestimation of the displacement in this region. It should also be noted that the top and bottom displacements obtained with the protection activated in the case of the limiter/low-shelf lead to limited displacement curves which overlap, which is not the case for the feedback/feedback delayed case, where the top and bottom limited displacement curves are a bit more chaotic and does not overlap properly.

This measurement should surely be made at a more realistic threshold in terms of the maximum excursion that the loudspeaker can accept.

J.3 Algorithm performances - threshold comparison

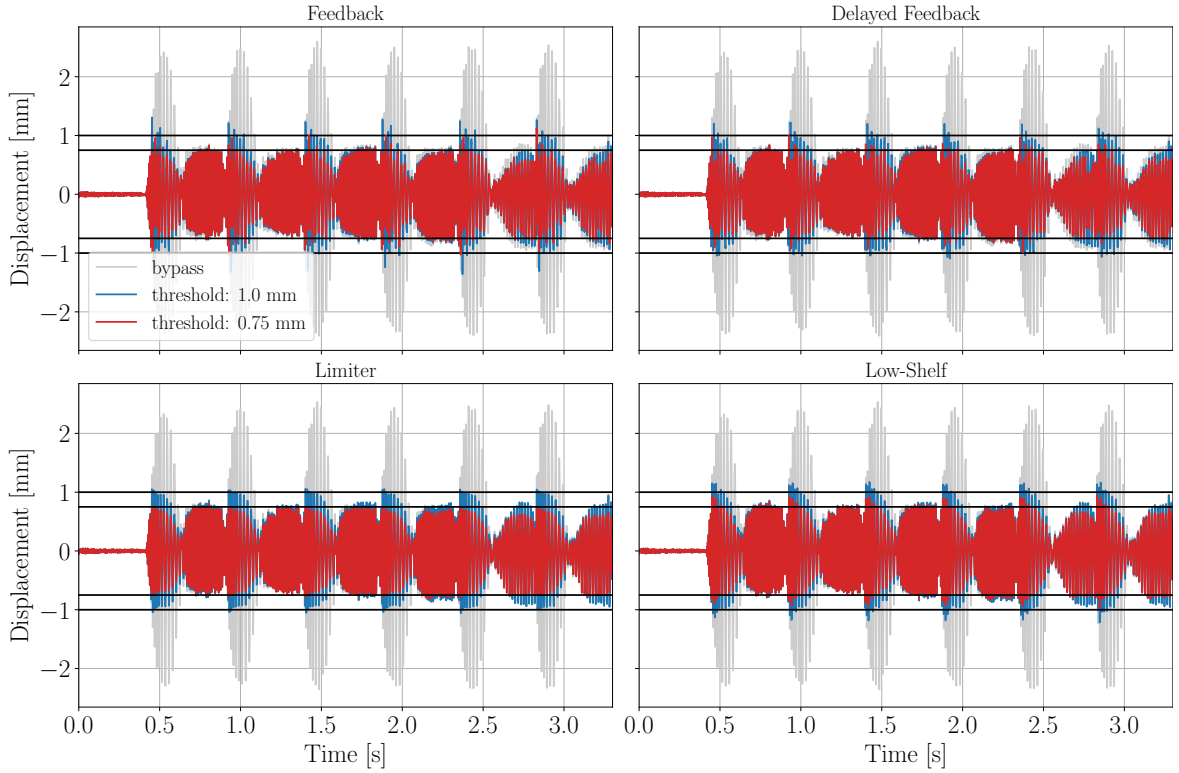


Figure 28: Measured membrane excursion with *ShookOnes* and all algorithm. The threshold is set at 1.0 and 0.75 mm.

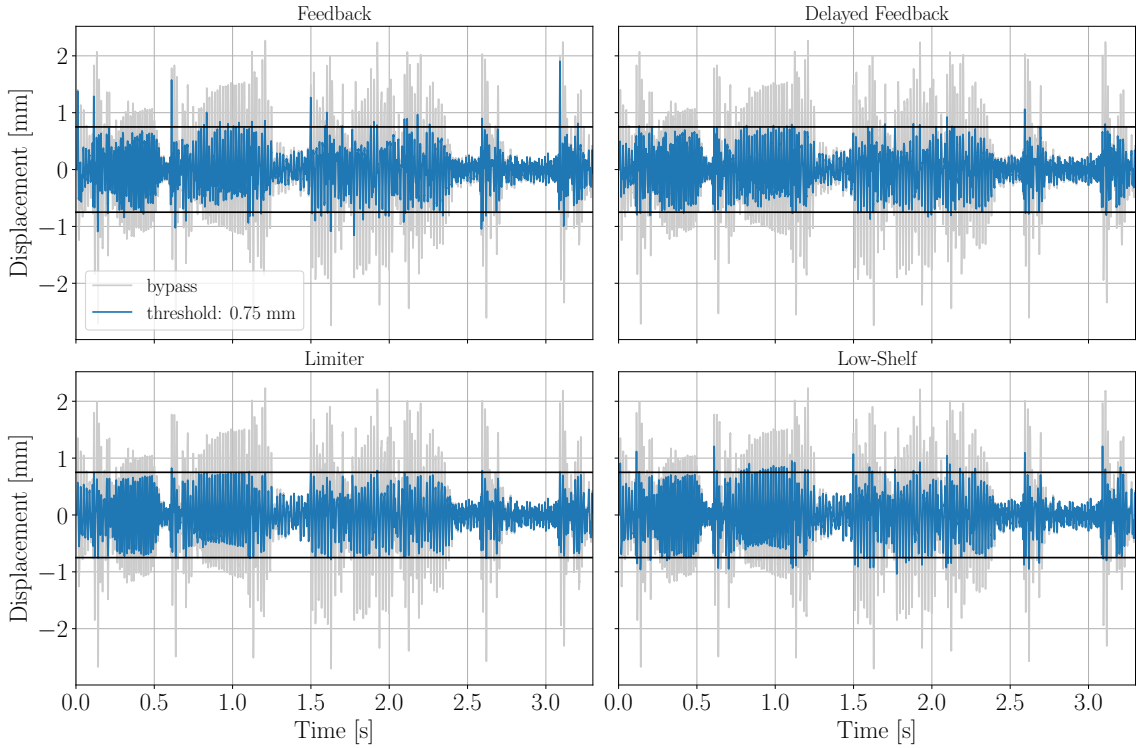


Figure 29: Measured membrane excursion with *Take My Breath* and all algorithm. The threshold is set at 0.75 mm.

J.4 Algorithm performances - spectrum comparison

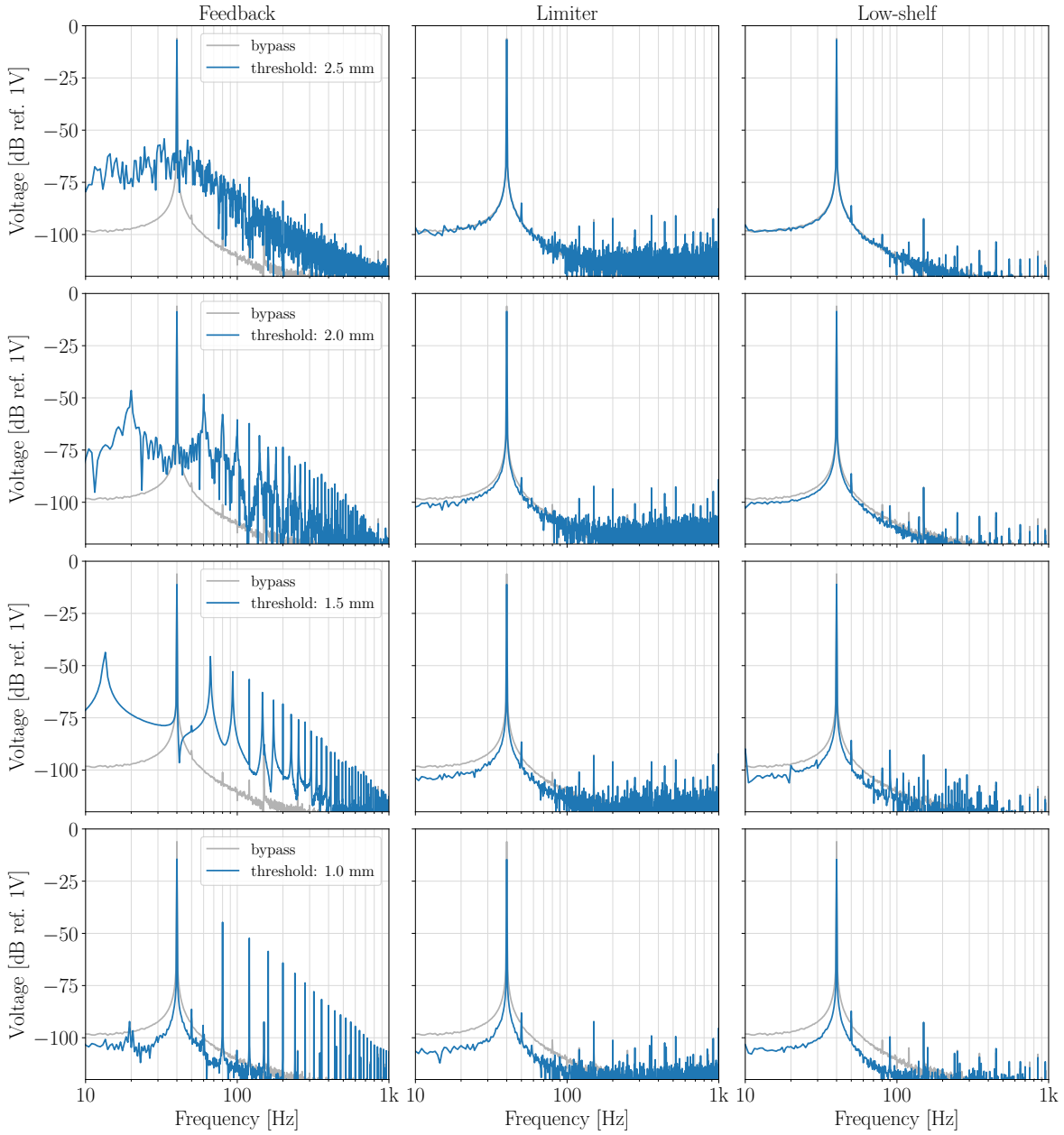


Figure 30: Spectrum of the measured limited voltage signal u_{lim}, for feedback, limiter and low-shelf algorithms, and threshold of 2.5, 2.0, 1.5 and 1.0 mm. Excitation signal is a 40 Hz sine of peak amplitude 5.6 V on loudspeaker terminals when protection algorithms are bypassed (gray curves).

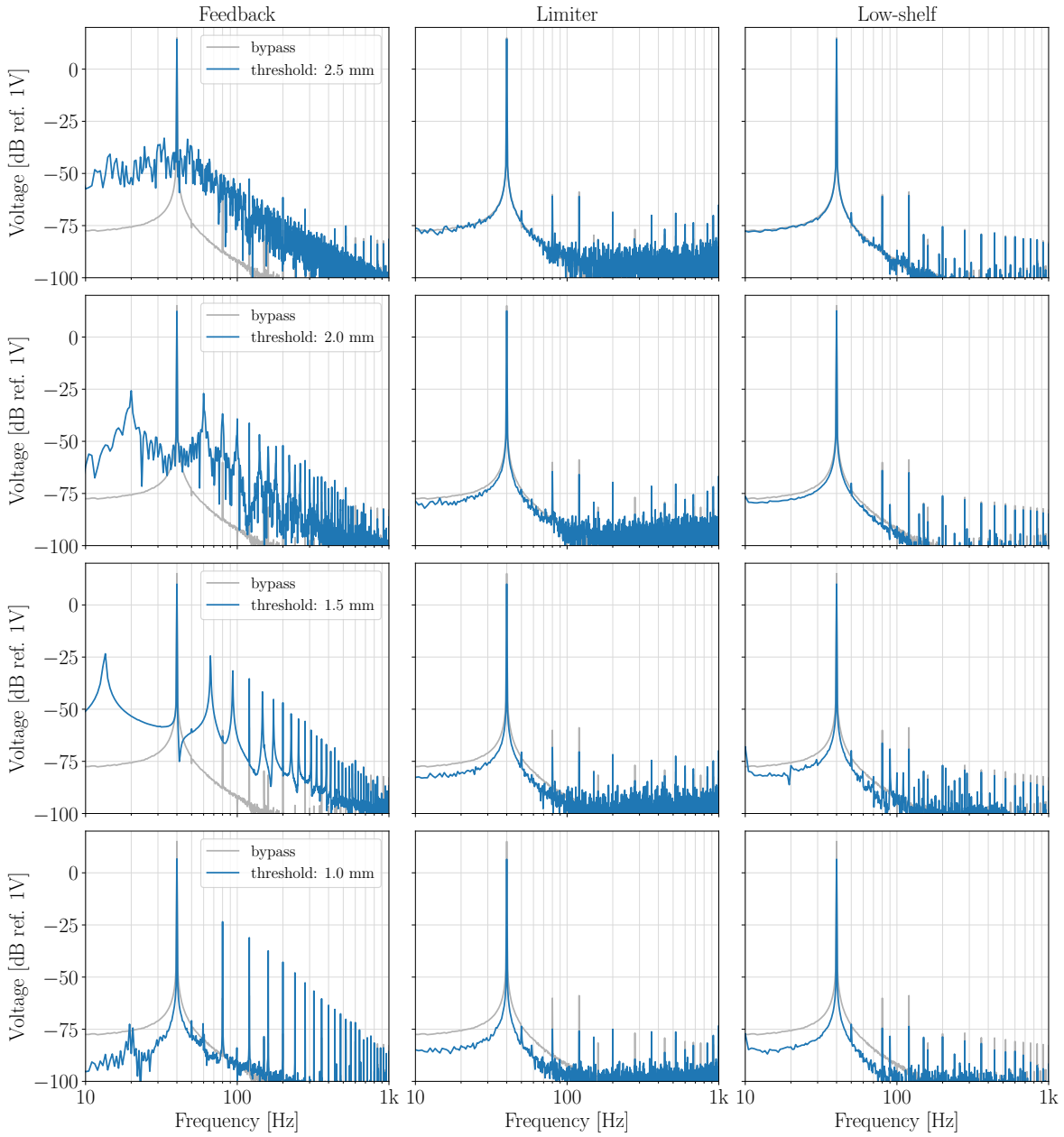


Figure 31: Spectrum of the loudspeaker terminal voltage u_{hp}, for feedback, limiter and low-shelf algorithms, and threshold of 2.5, 2.0, 1.5 and 1.0 mm. Excitation signal is a 40 Hz sine of peak amplitude 5.6 V on loudspeaker terminals when protection algorithms are bypassed (gray curves).

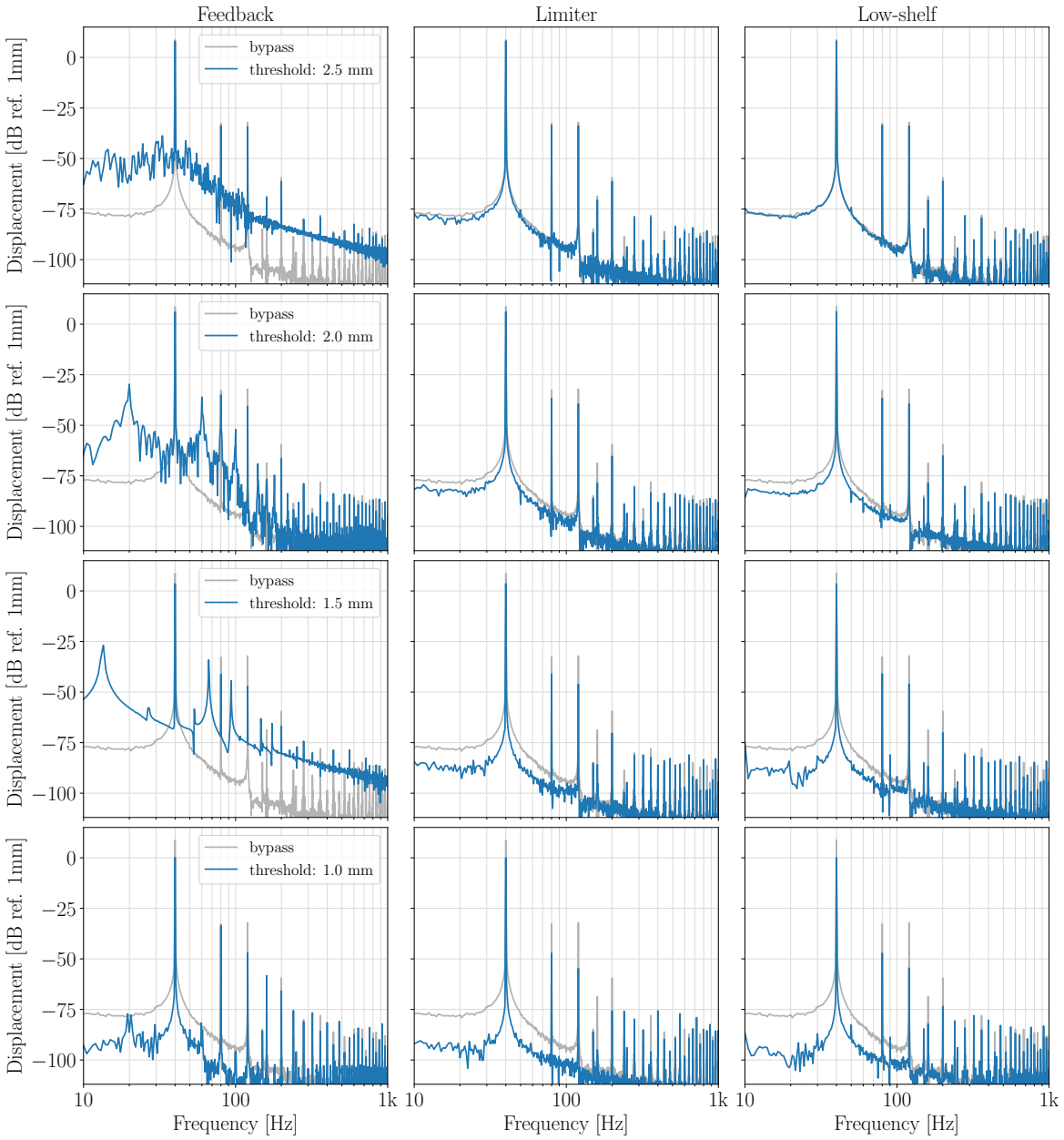


Figure 32: Spectrum of the measured displacement x for feedback, limiter and low-shelf algorithms, and threshold of 2.5, 2.0, 1.5 and 1.0 mm. Excitation signal is a 40 Hz sine of peak amplitude 5.6 V on loudspeaker terminals when protection algorithms are bypassed (gray curves).

Bibliography

[1] Wolfgang Klippel. "Active transducer protection part 1: Mechanical overload". In: *Audio Engineering Society Convention 139*. Audio Engineering Society. 2015.

[2] Paolo La Tocarra. "Feedback control of a dynamic loudspeaker with embedded sensor coil". MA thesis. Politecnico di Milano, 2014.

[3] Wolfgang Geiger. "Servo control of loudspeaker cone motion using an optical linear displacement sensor". In: *Journal of the Audio Engineering Society* 53.6 (2005), pp. 518–524.

[4] Paul Kohut. "Frequency dependent excursion limiter". US Patent 6,931,135 B1. Aug. 2005.

[5] Wolfgang Klippel. "Tutorial: Loudspeaker nonlinearities - causes, parameters, symptoms". In: *Journal of the Audio Engineering Society* 54 (10 Oct. 2006), pp. 907–939.

[6] Antonin Novak. "Compression and expansion nonlinear effects in an electrodynamic loudspeaker: experiments vs. model failure". In: *Proc. Forum Acusticum*. EAA. France, 2020.

[7] Wolfgang Klippel. "Predictive protection arrangement for electroacoustic transducer". US Patent 5,528,695. June 1996.

[8] Bruno Putzeys and Lars Risbo. "A method of controlling loudspeaker diaphragm excursion". EP 3,453,186,B1. Oct. 2020.

[9] G.W McNally. "Digital audio: dynamic range control of digital audio signals". In: *NASA STI/Recon Technical Report N* 84 (1983), p. 14374.

[10] Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. "Digital dynamic range compressor design—A tutorial and analysis". In: *Journal of the Audio Engineering Society* 60.6 (2012), pp. 399–408.

[11] Germán Ramos. "Block processing strategies for computationally efficient dynamic range controllers". In: *Proc. Int. Conf. Digital Audio Effects, Paris, France*. 2011, pp. 253–256.

[12] Perttu Hämäläinen. "Smoothing of the control signal without clipped output in digital peak limiters". In: *International Conference on Digital Audio Effects (DAFx)*. 2002, pp. 195–198.

[13] Duane K Wise. "Concept, design, and implementation of a general dynamic parametric equalizer". In: *Journal of the Audio Engineering Society* 57.1/2 (2009), pp. 16–28.

[14] Vesa Välimäki and Joshua D Reiss. "All about audio equalization: Solutions and frontiers". In: *Applied Sciences* 6.5 (2016), p. 129.

[15] Wolfgang Klippel. "Assessment of voice-coil peak displacement Xmax". In: *Journal of the Audio Engineering Society* 51.5 (2003), pp. 307–324.

[16] U. Zölzer et al. *DAFX - Digital Audio Effects*. John Wiley & Sons, 2002.

[17] Pascal M. Brunet and Glenn S. Kubota. "Energy limiter for loudspeaker protection". US Patent 10,701,485 B2. June 2020.

[18] Thomas Holm Hansen et al. "Measurement-based loudspeaker excursion limiting". US Patent 20,362,541 A1. June 2023.

[19] O. Munroe, S. Letourneur, and A. Novak. "Design of an electronic circuit for loudspeaker real-time digital signal processing". In: *16eme Congrès Français d'Acoustique*. SFA. Marseille, France, 2022.

[20] Storer Jules. *JUCE: Jules' Utility Class Extensions*. `https://github.com/juce-framework`.

[21] Deschang Eliot. *Xmax Protection Plugins : Audio plugins for mechanical overload protection of loudspeakers*. `https://github.com/eliot-des/Xmax-Protection-Plugins/tree/main`.

[22] Dimitrios Giannoulis, Michael Massberg, and Joshua D Reiss. "Parameter automation in a dynamic range compressor". In: *Journal of the Audio Engineering Society* 61.10 (2013), pp. 716–726.

[23] U. Zölzer. *Digital Audio Signal Processing*. Wiley, 2022.