Acoustical Instruments & Measurements
Universidad Nacional de Tres de Febrero - UNTREF

UNTREF
UNIVERSIDAD NACIONAL
DE TRES DE FEBRERO

SOFTWARE DEVELOPMENT FOR RIR'S PROCESSING

Franco Daniel Areco
Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
email:francoareco53@gmail.com

Juan Martín Rucci
Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
email:rucci45744@estudiantes.untref.edu.ar

Juan Almaraz
Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
email:juan.almaraz097@gmail.com

Calquin Epullan
Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
email:epullan44186@estudiantes.untref.edu.ar

This paper explains the development of software designed for the calculation of key acoustic parameters that characterise a room environment. Using the Python programming language and the *Tkinter* framework, a graphical user interface (GUI) has been constructed to facilitate the processing of monophonic and stereo impulse responses. These response types are essential for deriving binaural acoustic parameters, including interaural cross-correlation (*IACC*). The resulting parameter calculations are compared to those generated by other software for validation purposes. Overall, the results show a remarkable agreement with other software in most of the processes, although some divergences are observed in the clarity parameters, especially in the low frequency range. The T20 and T30 parameters show values consistent with the commercial software and are in line with the expected trends. Furthermore, the investigation highlights the interdependence and sensitivity of the early decay time (*EDT*) to the transition time (T_t). As far as the early *IACC* parameter is concerned, there is an observable similarity with frequency. The *MMF* and *Schroeder* methods are compared and it is found that there is a similarity in the parameters at high frequencies, but there is a difference at low frequencies, due to the limits of the cut-off of the impulse response.

Keywords: Sine Sweep, Acoustical Parameters, Impulse Response

1. Introduction

The logarithmic sinusoidal sweep (LSS) approach became a widely used technique in obtaining the environmental impulse response (RIR) in the field of acoustics.

The objective characterization of a linear time invariant system, such as a room, is achieved through its transfer function, which is translated into its impulse response. The RIR is represented as a curve describing the dissipation of acoustic energy in the room after the termination of a sound

source. After a filtering and smoothing process, this curve becomes a valuable tool for the calculation of highly relevant acoustic parameters.

The measurement of acoustic parameters, such as reverberation time, is essential for understanding the acoustic characteristics of a room and thus for making informed decisions in the design of acoustic environments. Angelo Farina [1] describes how this RIR is typically obtained from a swept sinusoidal signal, and its subsequent processing with specialized software determines the final acoustic parameters.

This work focuses on the development of an open source software in Python to process room impulse responses and calculate key acoustic parameters, such as EDT, T20, T30, C50, C80, TT, IACCE and EDTt. An intuitive graphical user interface (GUI) was created for ease of use.

The software uses the logarithmic sine sweep method to obtain accurate measurements of room impulse responses. These responses are applicable to a variety of environments, such as control rooms, studios, theaters, and auditoriums.

The proposed software was benchmarked against widely used commercial and academic tools for obtaining acoustic parameters to validate the accuracy and effectiveness of the software developed in Python. These tools include the Aurora add-on for Audacity and a software developed by Densil Cabrera in Matlab, thus establishing a solid basis for its reliability and usefulness in acoustic analysis.

2. State of the art

The characterization of listening rooms is a constantly evolving field of study in the field of acoustics. Over the years, various acoustic parameters have been developed to evaluate and understand the acoustic properties of rooms objectively and subjectively. These parameters play a crucial role in optimizing the design of spaces intended for applications such as concert halls, recording studios, theaters and auditoriums.

Reverberation time, initially proposed by W.C. Sabine in 1922 [2], has been a fundamental parameter in room characterization. RT refers to the time it takes for the sound intensity to decrease by 60 decibels (dB) after the sound source has been interrupted. Despite its age, RT is still relevant today and has benefited from technological advances that allow accurate and efficient measurements.

Auditory perception in a room goes beyond RT. To address this complexity, Early Decay Time (EDT) was introduced in 1970 by N.C. Jordan [3]. EDT focuses on the initial phase of the impulse response and correlates closely with the subjective perception of reverberation. It is often used in conjunction with RT to provide a more complete description of the acoustic characteristics of a room.

The intelligibility of speech and music in a room is also of great importance. Clarity parameters, such as C50 and C80, were introduced by Reichard and Abdel Alim in 1974 [4] to measure the ratio of direct sound energy to reflected sound energy in the room. These parameters are essential for assessing sound intelligibility and are used in applications ranging from conference rooms to theaters.

Spatial localization of sound is another crucial aspect of room acoustics. Interaural Cross-Correlation (IACC) was introduced in 1974 [5] and is used to quantify the coherence of sound signals in both ears of a listener. A high value of IACC indicates strong interaural coherence and better perception of sound direction.

Advancing technology has led to the development of specialized acoustic room characterization software. Notable examples include the Aurora plugin developed by Angelo Farina [6] and the AARAE software of Densil Cabrera [7]. These tools allow the accurate calculation of various parameters by analyzing impulse responses, facilitating the evaluation and design of acoustic spaces.

The characterization of acoustic rooms has undergone significant progress over time, leading to the development of a wide variety of acoustic parameters and computational tools. These advances are essential for optimizing the design of spaces for a variety of applications, from concerts to conferences. The combination of traditional parameters such as RT with more recent metrics such as EDT and clarity has enriched the understanding of room acoustic properties.

3. Theoretical framework

3.1 Room Impulse Response

A linear time invariant (LTI) system is a fundamental concept in systems theory and is used to describe a wide range of physical and electronic systems. An LTI system can be characterized by its transfer function $h(t)$, which provides information on how the system responds to different input signals as a function of time.

To obtain the transfer function $h(t)$ of an LTI system, it is necessary to excite the system with an input signal $x(t)$. The input signal must meet certain requirements to ensure accurate characterization of the system. These requirements include that the input signal must be deterministic, periodic and broadband. Deterministic ensures that the input signal is predictable and reproducible, periodicity allows analysis in the frequency domain, and broadband ensures representation of a wide range of frequencies.

The relationship between the input signal $x(t)$ and the system response $y(t)$ is established by a fundamental operation known as convolution in the time domain as expressed in Equation 1.

$$y(t) = x(t) * h(t) \tag{1}$$

An equivalent way to represent this relationship is in the frequency domain, using the Fourier transform. The relationship is expressed in Equation 2.

$$Y(f) = X(f) \cdot H(f) \tag{2}$$

Where $X(f)$ and $Y(f)$ are the Fourier transforms of the signals $x(t)$ and $y(t)$, respectively, and $H(f)$ is the Fourier transform of the transfer function $h(t)$.

3.1.1 Logarithmic Sine-Sweep Method

The logarithmic sine-sweep method, devised by Farina [1][8], serves the primary purpose of extracting the impulse response of an enclosure. In comparison to conventional approaches, such as the Maximum-Length Sequence (MLS) method or the generation of impulses through methods like bursting a balloon, clapping, or employing a Kami-teppo, this method offers a multitude of advantages.

Foremost among its benefits is its exceptional reproducibility, whereby any stimulus generated within the specified constraints consistently yields identical outcomes. This reproducibility stems from the method's self-sufficiency, relying exclusively on the characteristics of the loudspeaker system. Consequently, it ensures the generation of impulsive stimuli, resulting in the creation of a delta function through the convolution of an inverse filter and a logarithmic sine-sweep signal (as depicted in Figure 1.

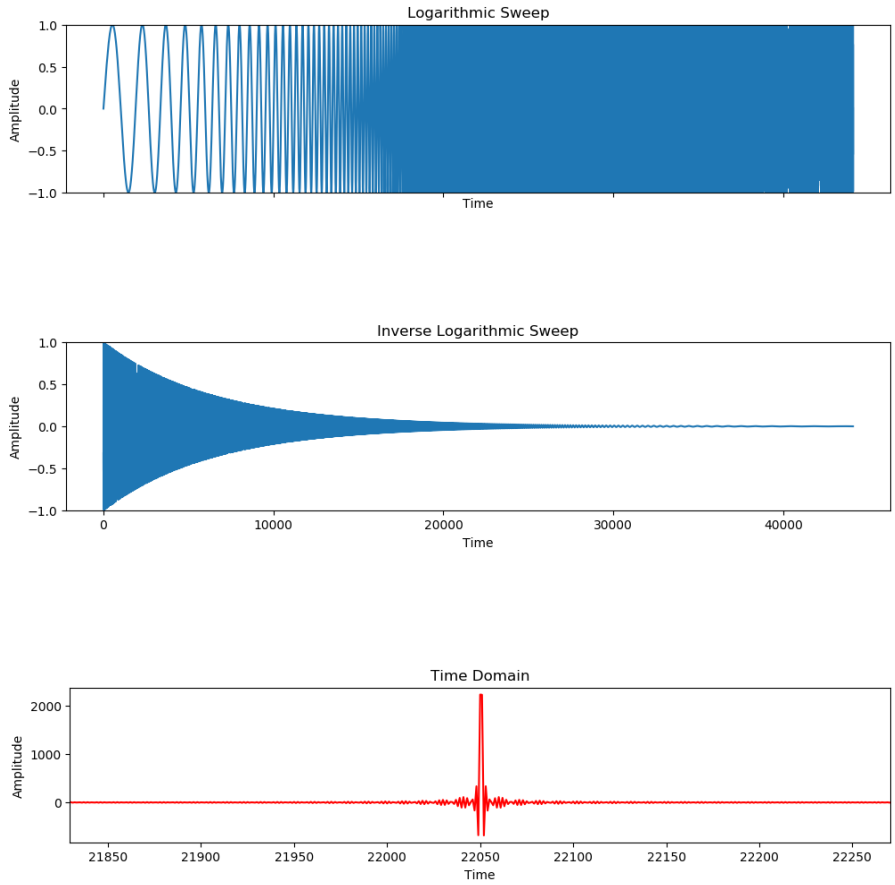


Figure 1: Impulse response generation using the log sine sweep method.

3.2 Acoustical parameters

3.2.1 Reverberation time

The Reverberation Time (RT) is defined as the period required for the sound intensity to decrease by 60 decibels (dB) after the interruption of the sound source. This measurement standard, which uses a 60 dB reduction as a reference, provides an objective metric for evaluating the acoustic characteristics of a room. The RT is essential for the characterization of room acoustics as it influences the sound quality in various applications such as concert halls, recording studios, and classrooms.

To obtain the Reverberation Time T_{30} and T_{20}, dynamic ranges less than 60 dB are considered and extrapolated from T_{60}. T_{30} is defined as the time it takes for the sound decay curve to drop from -5 dB to -35 dB below the initial level, while T_{20} is defined as the time it takes to drop from -5 dB to -25 dB.

These reverberation time parameters provide a detailed understanding of how sound behaves in a space and are essential in the design and evaluation of acoustical environments for various applications.

It is important to note that when the sound decay is linear, i.e. when the decay curve of the sound intensity follows a constant path, the values of T60, T20 and T30 are identical. This means that if one knows the value of T60 in a given room, one can accurately infer the values of T20 and T30, which greatly simplifies the process of acoustic measurement and analysis. This phenomenon is clearly reflected in Figure 2 , where the equality of these parameters is observed when the linear decay condition is met.

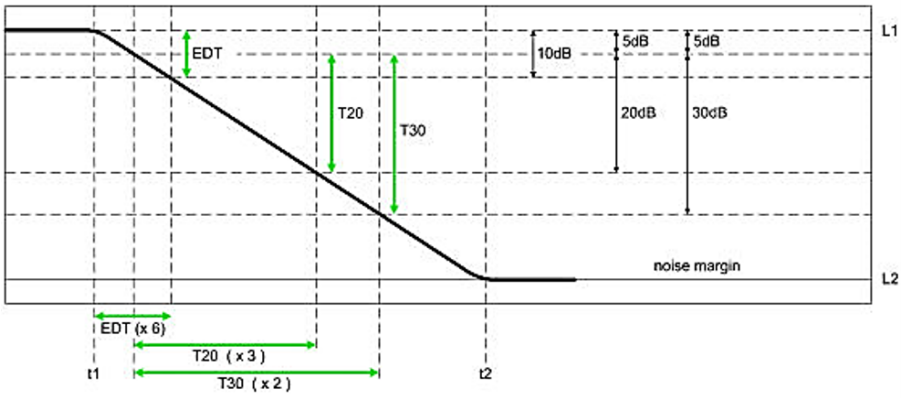


Figure 2: Diagram illustrating the comparison of EDT, T20, and T30.

3.2.2 Early decay time

Early Decay Time (EDT) is an acoustic parameter used to evaluate the quality of reverberation in a given space. EDT refers to the time it takes for the sound pressure level to decrease by 10 decibels (dB) after a sound source has stopped.

EDT provides valuable information about the clarity and intelligibility of sound in a given environment. A shorter EDT indicates that room reverberation fades quickly, which is generally associated with better speech intelligibility and crisper sound quality. On the other hand, a longer EDT suggests that reverberation persists longer, which can negatively affect sound clarity and intelligibility, especially in environments where clear communication is required, such as conference rooms or auditoriums.

3.2.3 Clarity

Clarity is an acoustic parameter that evaluates the intelligibility of sound in a specific environment. It refers to the ratio between the direct sound energy reaching the listener and the reflected sound energy in the room. Higher clarity indicates better speech and music intelligibility.

The clarity parameter (C) expresses the ratio of early to late or reverberant energy, as shown in the Equation 3.

$$C = 10 \cdot \log_{10} \left(\frac{\int_0^{t_e} p(t)dt}{\int_{t_e}^{\infty} p(t)dt} \right), \text{dB} \tag{3}$$

Where C is the clarity in decibels (dB), t_e indicates the integration time limit and $p(t)$ is the instantaneous sound pressure of the measured impulse response.

It is possible to define the clarity for different types of signals as a function of the integration limits. The clarity of a spoken word signal is called C_{50}, and is calculated considering the initial 50 ms. In the case of a musical signal, the clarity parameter C_{80} is defined considering an integration limit of 80 ms.

3.2.4 Interaural cross-correlation

Interaural Cross Correlation (IACC) is a parameter used in acoustics to quantify the coherence of sound signals in both ears of a listener. It measures the similarity between the sound signals reaching both ears and is used to evaluate the spatial localization of sound. The IACC is represented by the Equation 4.

$$IACC = \frac{\int_0^T [R(t) \cdot S(t)]\, dt}{\sqrt{\int_0^T [R^2(t) \cdot S^2(t)]\, dt}} \tag{4}$$

Where $R(t)$ is the sound signal in the right ear as a function of time, $S(t)$ is the sound signal in the left ear as a function of time and T is the analysis time.

This formula quantifies interaural coherence and is used to evaluate the perception of sound direction in an acoustic environment.

3.3 Software Processing

3.3.1 Moving median filter

The Moving Median Filter (MMF) is used as a time domain signal processing technique. It works by taking a time window of N samples of the signal, sorting the samples in ascending order and taking the value of the N/2 sample. It then assigns the median value to that portion of the signal. As the window moves along the signal, this filtering process is applied. MMF is widely used to reduce random noise in signals while preserving the shape of the original signal. Its implementation is efficient and fast, especially when recursive techniques are used. In the developed software, the MMF is used as a filtering option to smooth the signal decay curve.

$$MMF(x) = \begin{cases} x\left[\frac{n}{2}\right] & \text{if n is even} \\ \frac{x[0.5(n-1)]+x[0.5(n+1)]}{2} & \text{if n is odd} \end{cases} \tag{5}$$

3.3.2 Schroeder filter

The Schroeder-Stirling Integral, represented as ETC(t), is a technique used to obtain the energy decay curve of the impulse response of a room. It is defined as the integral of the squared signal from time t to time T, as shown in the Equation 6.

$$ETC(t) = \int_t^T h^2(\tau) d\tau \tag{6}$$

Where $h(\tau)$ represents the room impulse response as a function of time τ, t is the initial integration time, and T is the final integration time, which is determined by the point at which the signal is predominantly background noise.

This technique is used to analyze and characterize the decay properties of the impulse response of a room and is essential for the accurate evaluation of its acoustic behavior. In addition, to avoid overestimation of the impulse response, a background noise compensation method, such as Lundeby's method, is applied.

3.3.3 The Lundeby Method: Estimating Crossover Points in Acoustic Analysis.

The Lundeby Crossover Point Estimation Method is a technique used in acoustics for the purpose of accurately determining the point at which the decay curve of an impulse response intersects the background noise level. This intersection point is subsequently used as the upper limit of integration in the calculation of the Schroeder Integral, leading to a significant improvement in the accuracy of the filtering technique.

The Lundeby estimation process is based on an iterative algorithm consisting of the following four fundamental steps:

1. First, the impulse response is averaged and squared over a short time interval, usually in the range of 10-50 milliseconds.

2. Next, the background noise level is estimated using the last 10% of the signal, which is assumed to be noise-dominated.
3. Then, a slope is calculated from the 0 dBFS reference level to the level corresponding to the background noise.
4. Finally, the algorithm identifies the intersection point between the linear regression obtained in the previous step and the corresponding noise level.

This process is repeated iteratively until a convergence criterion is met, indicating that the crossover point between the decay curve and the background noise level has been accurately determined.

The Lundeby Crossover Point Estimation Method is noted for its effectiveness in separating the decay portion of the impulse response from the background noise, which contributes to a significant improvement in the accuracy of the Schroeder filtering technique. This iterative approach ensures reliable results in acoustic analysis by providing accurate estimates of the relevant parameters.

4. Procedure and software developement

The software developed is aimed at processing impulse response measurements of rooms, as well as calculating their acoustical parameters. It also has other functions such as a graphic display of the impulse response and its decay or the possibility of exporting the results.

The development of both the signal processing functions and the user interface (GUI) was carried out entirely in Python. This section describes the signal processing procedure in order to obtain the acoustic parameter values according to the signalflow in figure 3.

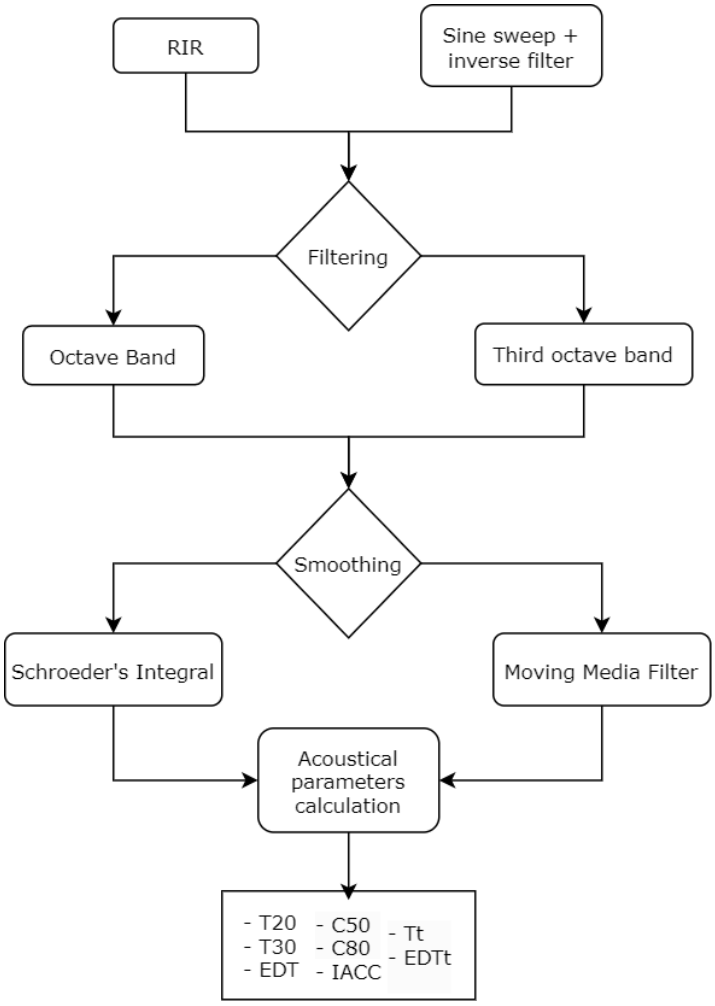


Figure 3: Flowchart of the software developed.

The libraries used for this software are numpy v1.24.3, scipy v1.10.1, pandas v2.0.2, matplotlib v3.7.1, tkinter v0.2, soundfile v0.12.1 and acoustics v0.2.6.

4.1 Load file

The impulse response to be processed and evaluated in order to obtain the acoustic parameters that characterise a room can be obtained in two ways: by loading a .wav audio file containing the impulse response or by loading the signal of a sinusoidal sweep with its respective inverse filter. Once the information is loaded, an automatic detection of whether the file is mono or stereo is performed and allows the selection of the channel to be evaluated and represented.

4.2 Filtering and smoothing

With the RIR loaded, the option of filtering the signal into octave bands or third octave bands is offered following the IEC 61260 standard [9]. The filters implemented are eighth-order Butterworth band-pass filters.

Once the signal has been filtered, a smoothing process is carried out with a choice between two options: the inverse Schroeder time integral or a Moving Media Filter (MMF). For both processes, first a smoothing with the Hilbert transform is applied.

When smoothing with the Schroeder integral is applied, the Lundeby method is then used to find the end of the tail of the impulse and the background noise component. On the other hand, when smoothing is done using the Moving Media Filter, the GUI enables a box in which the time value in milliseconds of the desired window can be introduced. By default, the value of the window is 20 ms. This window length allows an analysis of the impulse response starting at 50 Hz, which is enough for the first third octave band calculated (63 Hz).

4.3 Acoustical parameters calculation

For the calculation of *EDT*, *T20* and *T30*, the index corresponding to the decay range necessary for each of them is searched: the range between -1 dB and -11 dB for *EDT*; the range from -5 dB to -25 dB for *T20*; and from -5 dB to -35 dB for *T30*. Finally, a linear regression is performed to obtain the slope corresponding to each parameter.

On the other hand, the parameters *C50* and *C80* corresponding to clarity differ in the integration time according to equation (3). Thus, by entering the corresponding time, the desired parameter is obtained.

If the IRs are stereo, the *IACC* parameter can be calculated, for which both the right and left channels of the input signal are taken and equation (4) is used with a time limit equal to 80 milliseconds.

Finally, for the calculation of *EDTt* and *Tt*, the smoothed IR signal is taken from its maximum value and the index of the interval in which the signal reaches 99% of its energy is sought. This index marks the transition time *Tt* in seconds. Subsequently, the signal is cut between the mentioned limits and a linear regression is performed to obtain the *EDTt*.

4.4 Graphic user interface

In Figure 4 the grafic user interface of the Impulse Analyzer software is shown. Mainly two areas are distinguished, on the left a panel with the necessary options for IR processing, and on the right a panel where the information obtained after processing is displayed. On the left there are two tabs that allow importing an impulse response or a sine sweep. Then, the 'browse...' button opens a file dialog where the .wav file must be selected. When you do so, the name of the selected file, its sample rate, its duration and the number of channels are displayed. If the file is stereo, you can select the left or right channel for processing. In the case of importing a sine sweep, the option is given to import an inverse

filter, or to generate one given a start frequency, end frequency and duration. In this way the impulse response is obtained by convolving the two vectors. In the meantime, the imported ir is displayed in the plot area on the right. Then the option to filter the IR by thirds of octave or by octave bands is offered. It is also possible to smooth the signal with the Schrodinger integral or with a MMF, option that allows to choose the length of the window in ms. Finally, four buttons are displayed: the first one performs the IR processing and displays the results in the table on the right. The second (Clear) returns all GUI information to default values. The last two buttons allow for exporting the table as CSV or copy to clipboard.

Finally, when clicking on a parameter on the table, a new window emerges showing a plot representing the parameter in every frequency band.

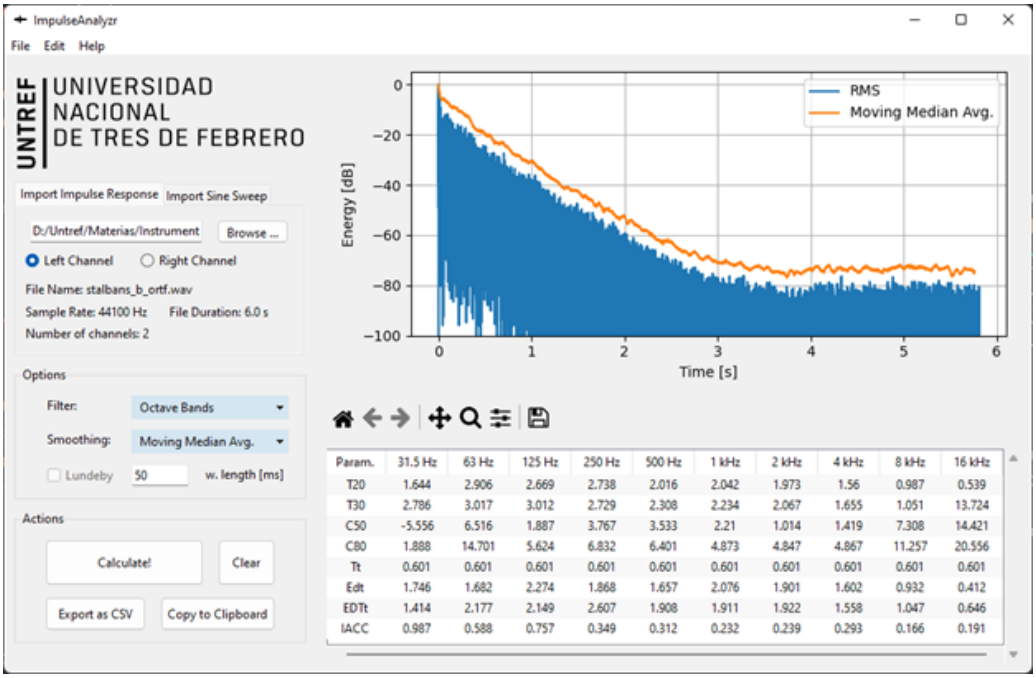


Figure 4: Graphic user interface of the software developed.

5. Results

The results obtained through the Room Impulse Response (RIR) processing software developed are compared with those obtained through the AARAE software, developed by Densil Cabrera and Aurora by Angelo Farina.

Since it is difficult to obtain a significant signal-to-noise ratio at low frequencies, the analysis is performed starting from the 63 Hz band.

In the case of the aurora, for the calculation of the acoustic parameters, the values of -1 and -11 dB were defined to determine EDT, and the correction for background noise was enabled. C50, C80, T20, T30, and EDT results were obtained with this software. On the other hand, with the *AARAE* software.

5.1 T20

In the graphs of Figure 5, it can be seen that the values obtained by applying Schroeder's method show significant agreement with those obtained through commercial software. These plots reveal the existence of three distinctive regions. In the region above 1250 Hz, there is a high similarity in the results between the different programs. In the middle region between 250 Hz and 1 kHz, although there is a remarkable agreement, it is lower compared to the higher frequency region. Finally, in the low-frequency region, from 200 Hz onwards, the greatest discrepancy in the results is evident.

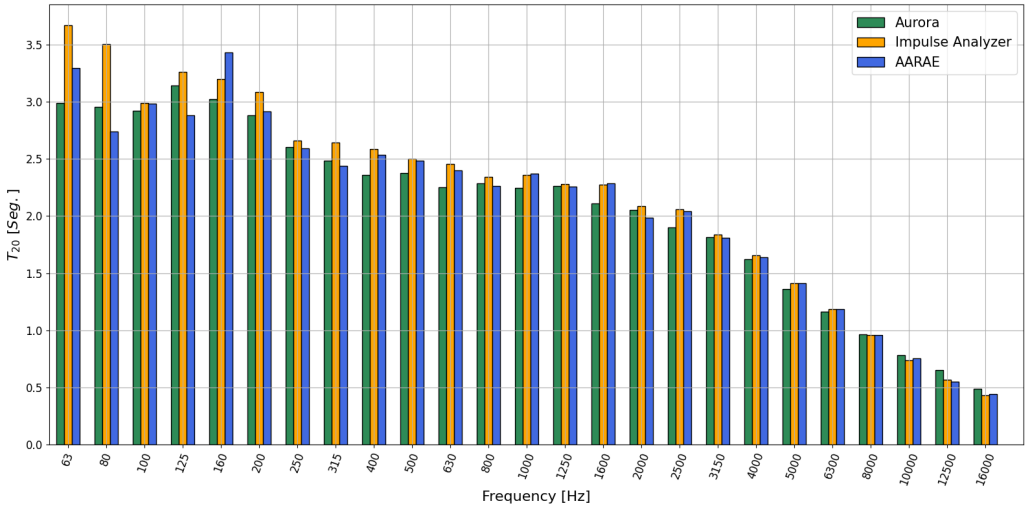


Figure 5: Comparison of T20 results.

5.2 T30

Figure 6 shows the results corresponding to parameter T30. The results show a high level of agreement at frequencies above 500 Hz. However, it can be seen that, at frequencies below this threshold, the developed software tends to overestimate the results at most frequencies compared to commercial solutions. This discrepancy is particularly noticeable at the 63 Hz frequency, where the value obtained by applying the Schroeder method exceeds the results obtained with the other programs by more than 1 second.

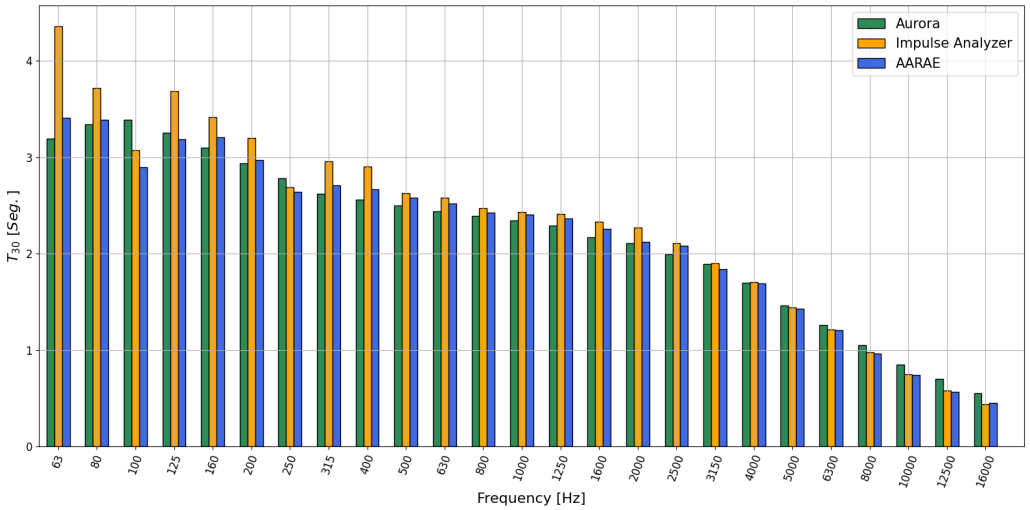


Figure 6: Comparison of T30 results.

5.3 EDT

Figure 7 shows that the results obtained for the calculation of the EDT exhibit a trend similar to that observed for the T30 parameter. This trend is characterized by a greater agreement in the results between different acoustic software when the frequency exceeds 500 Hz, and by more significant discrepancies in values below this frequency. It should be noted that these discrepancies are not only noticeable in the software developed, where mostly higher values are obtained compared to others, but also extend to the differences observed between commercial software, particularly in the low frequency range. Discrepancies in low-frequency results may be the result of a combination of factors related to the physics of sound waves, room characteristics, limitations of measurement equipment

and the complexity of acoustic phenomena in the low-frequency range. Therefore, it is important to consider these limitations and factors when analyzing and comparing acoustic measurements in this range.

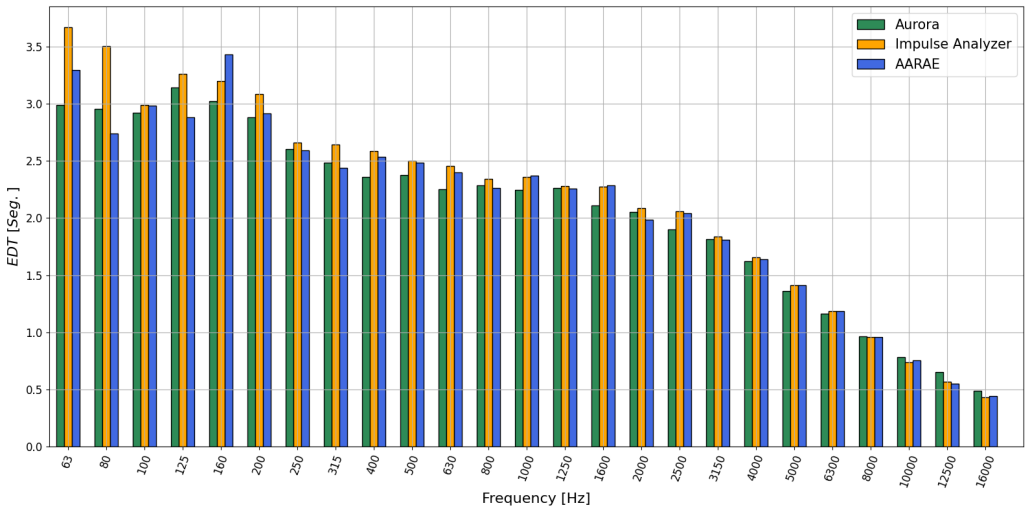


Figure 7: Comparison of EDT results.

5.4 C50

Figure 8 shows the comparative results for the acoustic parameter C50. There is a trend towards convergence of the values at high frequencies; however, in general terms, there is a significant disparity between the results obtained with the software developed and the commercial solutions. In the frequency range between 250 Hz and 2 kHz, the values obtained with the proprietary software show a trend of practically zero. On the other hand, at low frequencies, the values of the three software solutions do not show an appreciable relationship.

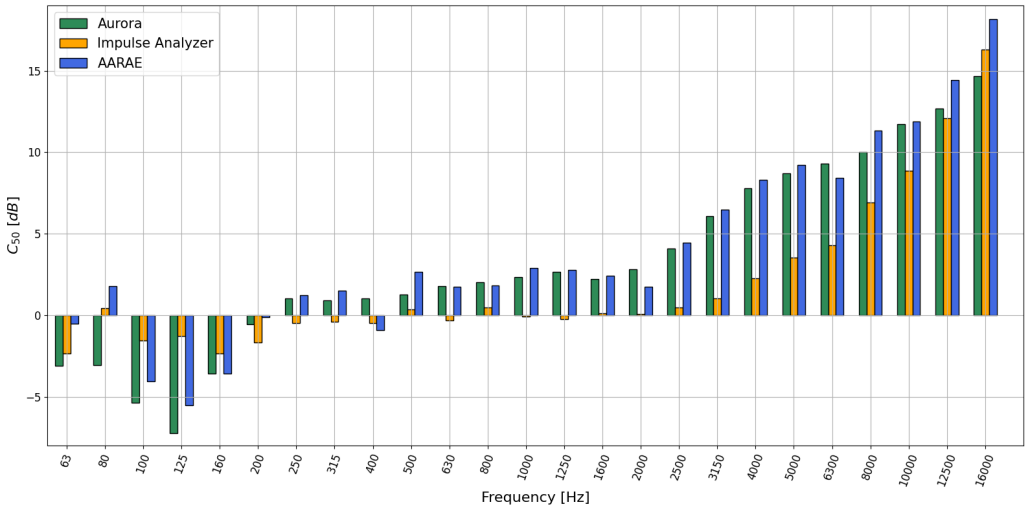


Figure 8: Comparison of C50 results.

5.5 C80

In contrast, the results corresponding to parameter C80, as illustrated in Figure 9, reveal a more coherent relationship. Above 250 Hz, a greater consistency between the values is evident, while below this frequency, it is again found that the values do not correlate between the different softwares, this discrepancy being more noticeable below 80 Hz.

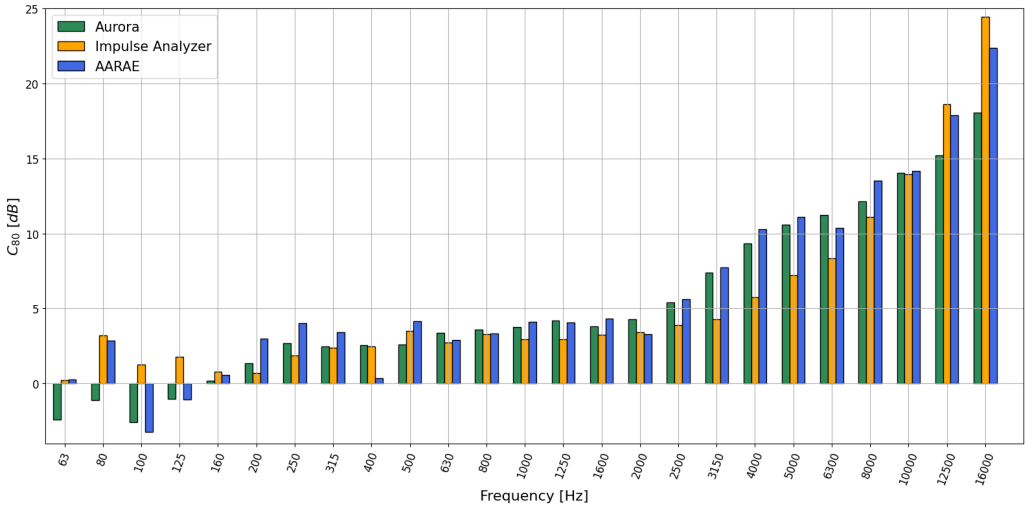


Figure 9: Comparison of C80 results.

5.6 IACC

Regarding the IACC parameter, a comparison was carried out exclusively between the developed software and the AARAE software, as illustrated in Figure 10, since the Aurora software did not provide the values corresponding to this parameter.

A general similarity between both software applications could be observed, with some notable exceptions at specific frequencies, such as 63 Hz, 1 kHz and 1250 kHz. In addition, it was observed that the results obtained using the proprietary software exhibit greater continuity as a function of frequencies, while the AARAE software presents more pronounced discontinuity jumps, particularly in the low frequency range. This difference can be attributed to a different length of the trimmed impulse. In the case of the software developed in this work, a time of 80 ms is used, in the case of AARAE this value is not known.

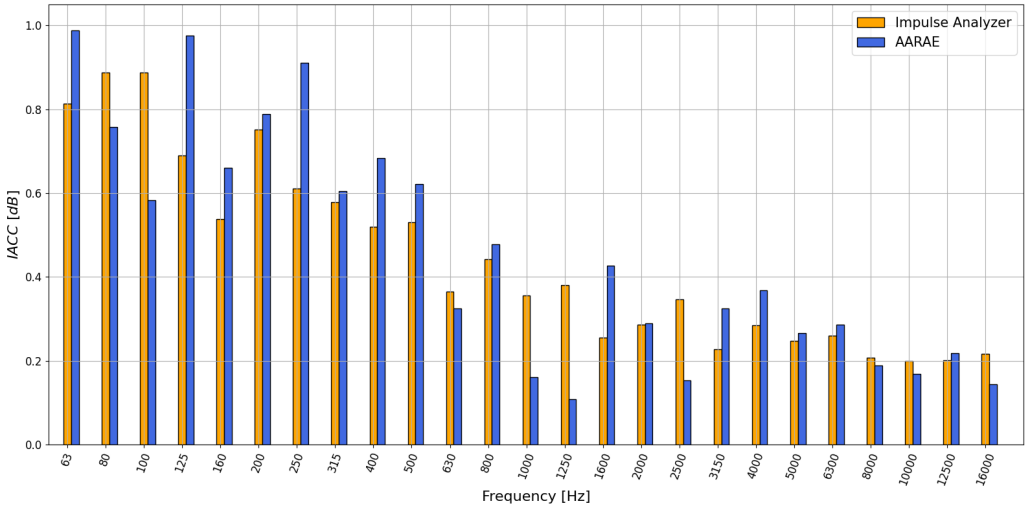


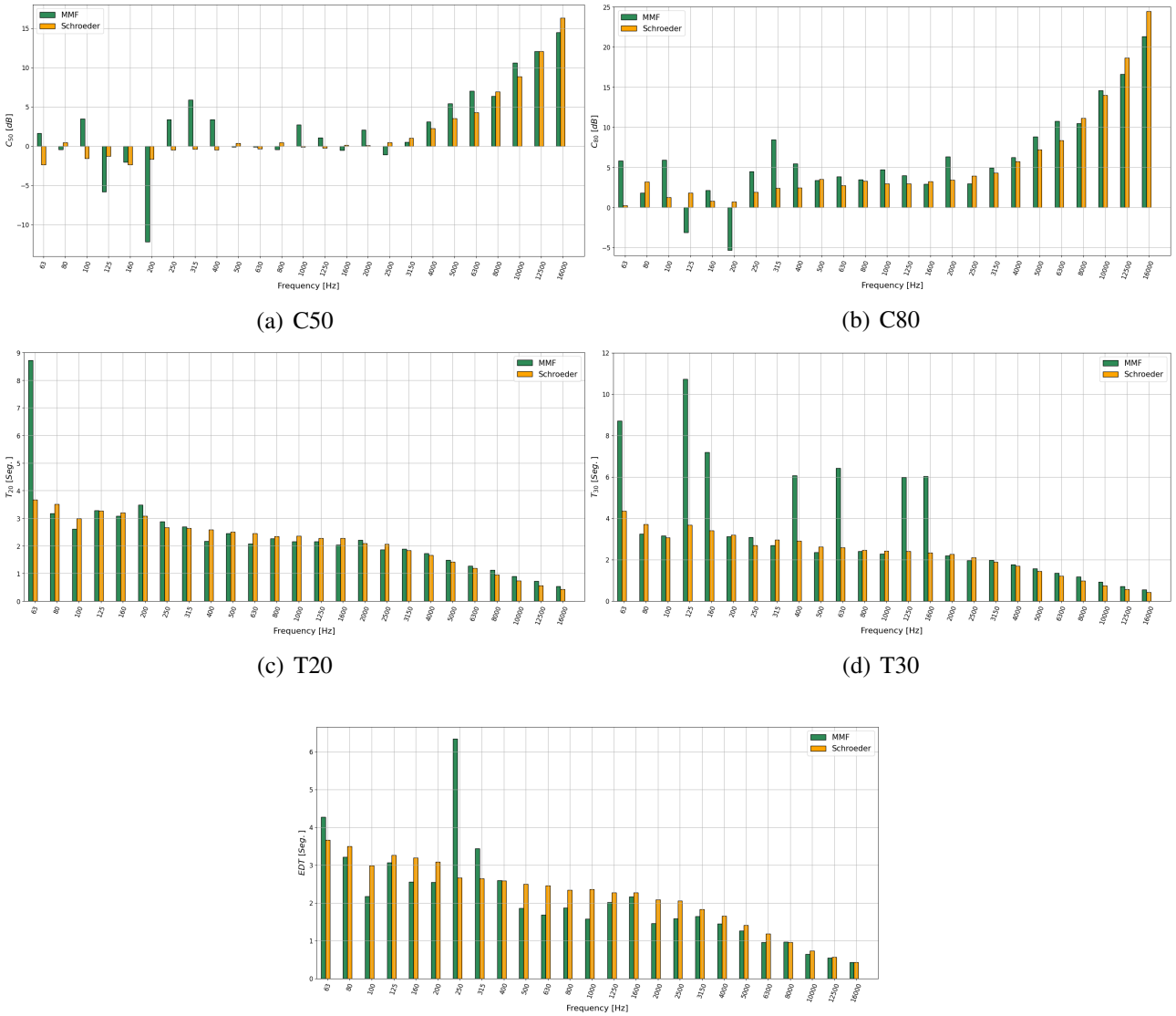
Figure 10: Comparison of IACC results.

It is noteworthy that, in all acoustic parameters, a distinction exists between low-frequency and high-frequency components. This distinction primarily arises from the presence of background noise within the analyzed audio, resulting in a non-optimal signal-to-noise ratios. In the case of the MMF (Moving Median Filter) method, the choice of the analysis window further impacts this difference. Specifically, the selection of a larger window leads to enhanced resolution in low frequencies, while conversely, a smaller window yields improved resolution in high frequencies. Consequently, these

factors give rise to substantial disparities in results between the Shroeder method and the MMF method.

On the other hand, it was possible to corroborate the reliability of the software developed by Angelo Farina and Densil Cabrera. While these software applications yielded varying results in some instances, this discrepancy can be attributed to differences in the functions employed, as well as the characteristics of the analyzed audio file and the methodology applied for truncating the impulse response and determining the starting point for parameter calculation.

5.7 Comparison between Moving Median Average *(MMF)* and Schroeder's method



(a) C50

(b) C80

(c) T20

(d) T30

(e) EDT

Figure 11: Comparison between MMF methods and Schroeder's method.

Figure 11 shows the comparison of the results obtained by applying the schroeder integral or a moving median filter. It can be seen that by smoothing the impulse with a MMF, values appear in some bands that are not the expected ones (taking as a reference the results of AURORA or AARAE). This may depend on the length of the window used. To analyze this behavior, Figure 12 shows a comparison of the T20 obtained with different window lengths.

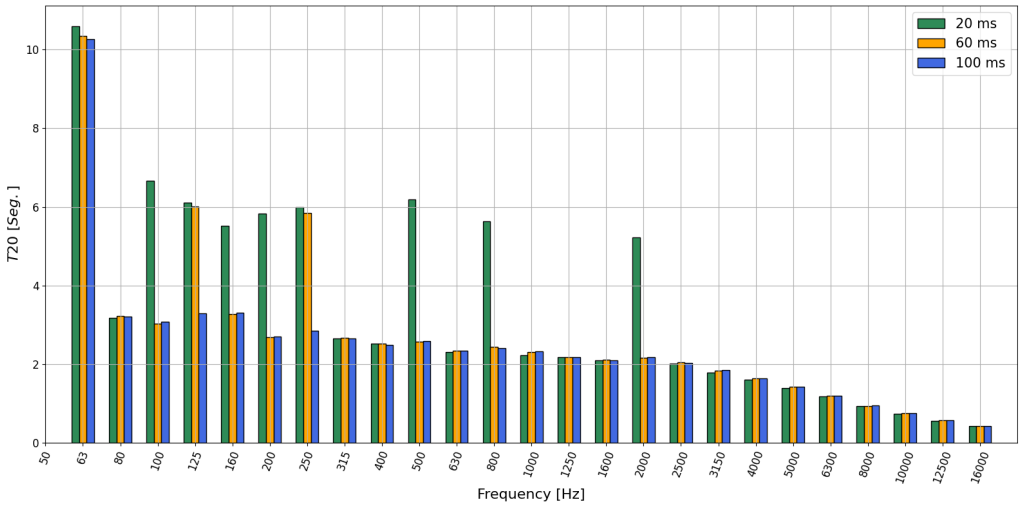


Figure 12: Comparison of T30 with different MMF window length.

It is clear that a 20 ms window is not enough to smooth the IR and obtain valid T20 results. When using a 100 ms window the results are similar to those obtained using Schroeders integral. This shows that the results obtained using MMF smoothing are sensitive to the variation of the window length. Therefore, it is important to be careful and verify the results obtained when using this method.

6. Conclusion

This project has highlighted the importance of accessibility and user-friendliness in the design of specialized software, especially when seeking to reach an audience that spans diverse areas of knowledge and skills. The convergence of acoustics and programming skills has resulted in a more accessible and effective tool, and this interaction between disciplines represents a valuable contribution to the field of acoustics and sound engineering.

It can be inferred that the EDT, RT20 and RT30 calculations exhibit a remarkable similarity with the results obtained using commercial software. On the other hand, the values of C50 and C80 show some divergences, particularly C50, and this disparity tends to be accentuated at lower frequencies.

The comparative analysis of the IACC parameter between the software developed in this study and the AARAE software has revealed a general similarity in the results, although with notable exceptions at specific frequencies. These discrepancies can be attributed in part to differences in the length of the trimmed pulse used in each program, being unknown such value in the case of AARAE. These findings emphasize the importance of careful consideration of technical and methodological particularities when conducting software comparisons and evaluations in acoustics.

In the comparative analysis carried out, it has been observed that the agreement between the results of both software is notoriously consistent in the high frequency spectrum. However, in the low frequency range, substantial discrepancies have been identified, attributable to the inherent limitations of the truncated impulse response. In the case of the Schroeder method, this limitation is defined by the Lundebay method, while the Moving Median Filter (MMF) method incorporates a window that influences the results in this frequency range.

It is relevant to highlight that, as an area of improvement identified, is the optimization of the inverse filter generation process, particularly in relation to the use of the sinesweep. The accurate generation of the sinesweep is essential to obtain more precise and coherent results in the calculation of the acoustic parameters.

In addition, it has been found that there is room for improvement of the error handling mechanisms in the software. Refining these procedures would contribute to strengthening the robustness and reliability of the applications, enabling a smoother and more effective user experience in obtaining

accurate acoustic results.

References

1. Angelo Farina. Simultaneous measurement of impulse response and distortion with a swept-sine technique. In *Audio engineering society convention 108*. Audio Engineering Society, 2000.

2. W.C. Sabine. *Collected Papers on Acoustics*. Harvard University Press, 1922.

3. N.C. Jordan. A smoothing procedure for experimental room transfer functions. *The Journal of the Acoustical Society of America*, 48(2B):528–536, 1970.

4. D.G. Reichard and M.A. Abdel Alim. Speech intelligibility measurements in terms of satisfactory signal-to-noise ratios. *The Journal of the Acoustical Society of America*, 55(5):1139–1142, 1974.

5. M.R. Schroeder, B.S. Atal, and J.L. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *The Journal of the Acoustical Society of America*, 55(2), 1974.

6. Angelo Farina. Aurora: Software for the calculation of room parameters. `https://aurora-plugins.forumfree.it/`, 2000.

7. D Cabrera, D Jimenez, and WL Martens. Audio and acoustical response analysis environment (aarae): a tool to support education and research in acoustics. *INTER-NOISE and NOISE-CON congress and conference proceedings*, 249(6):1667–1676, 2023.

8. Angelo Farina. Advancements in impulse response measurements by sine sweeps. In *Audio engineering society convention 122*. Audio Engineering Society, 2007.

9. IEC 61260-1:2014. Electroacoustics - octave-band and fractional-octave-band filters - part 1: Specifications. Technical report, International Electrotechnical Commission, 2014.

10. Manfred R Schroeder. New method of measuring reverberation time. *The Journal of the Acoustical Society of America*, 37(6_Supplement):1187–1188, 1965.

11. UNE-EN ISO 354:2004. Acoustics - measurement of sound absorption in a reverberation room. Technical report, International Organization for Standardization, 2004.

12. Simone Campanini and Angelo Farina. *A new audicity feature: room objective acoustical parameters calculation module*. na, 2009.

Annex

A. Data.

AURORA																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
stalbans_omni	C50	-3.120	-3.065	-5.357	-7.251	-3.592	-0.568	1.027	0.897	1.015	1.277	1.783	2.032	2.343	2.655	2.215	2.812	4.104	6.086	7.774	8.708	9.288	10.025	11.740	12.675	14.681
stalbans_omni	C80	-2.396	-1.090	-2.582	-1.002	0.207	1.372	2.684	2.477	2.560	2.590	3.366	3.599	3.776	4.178	3.790	4.294	5.407	7.408	9.345	10.604	11.250	12.121	14.057	15.195	18.073
stalbans_omni	D50	32.774	33.054	22.556	15.846	30.425	46.735	55.882	55.144	55.815	57.296	60.124	61.486	63.170	64.827	62.484	65.644	72.012	80.240	85.694	88.133	89.460	90.956	93.722	94.875	96.709
stalbans_omni	Ts	194.37	163.97	179.85	172.69	148.44	116.58	96.70	95.38	92.84	87.60	81.46	80.61	77.38	72.99	72.76	65.37	52.75	37.20	26.25	20.45	18.18	15.75	11.80	9.85	6.30
stalbans_omni	EDT	2.595	2.405	2.356	2.155	2.399	2.169	2.221	1.927	1.787	1.808	1.722	1.801	1.840	1.863	1.765	1.709	1.675	1.531	1.084	0.830	0.626	0.451	0.343	0.375	0.416
stalbans_omni	Tuser	2.968	2.609	2.542	2.779	2.745	2.730	2.784	2.494	2.225	2.094	2.088	2.085	2.010	2.109	2.057	1.953	1.732	1.625	1.414	1.110	0.944	0.771	0.557	0.456	0.358
stalbans_omni	T20	2.990	2.953	2.922	3.142	3.021	2.881	2.606	2.483	2.358	2.375	2.250	2.288	2.247	2.261	2.112	2.055	1.902	1.817	1.621	1.361	1.164	0.966	0.781	0.653	0.490
stalbans_omni	T30	3.195	3.340	3.392	3.255	3.101	2.936	2.780	2.622	2.559	2.497	2.437	2.389	2.346	2.288	2.172	2.111	1.994	1.890	1.696	1.462	1.262	1.048	0.846	0.701	0.551
IMPULSE ANALYZR																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
stalbans_omni	T20	3.668	3.503	2.991	3.262	3.197	3.084	2.663	2.644	2.588	2.502	2.456	2.340	2.358	2.279	2.273	2.090	2.058	1.835	1.656	1.413	1.188	0.957	0.740	0.566	0.433
stalbans_omni	T30	4.357	3.718	3.075	3.685	3.413	3.201	2.691	2.961	2.904	2.629	2.582	2.471	2.434	2.410	2.332	2.268	2.106	1.900	1.703	1.442	1.213	0.977	0.751	0.580	0.436
stalbans_omni	C50	-2.358	0.447	-1.576	-1.289	-2.348	-1.659	-0.484	-0.397	-0.473	0.370	-0.318	0.468	-0.090	-0.251	0.125	0.088	0.487	1.025	2.251	3.551	4.278	6.902	8.879	12.072	16.295
stalbans_omni	C80	0.241	3.196	1.244	1.775	0.803	0.685	1.870	2.395	2.452	3.499	2.739	3.292	2.938	2.950	3.234	3.414	3.914	4.273	5.739	7.208	8.331	11.122	13.964	18.618	24.453
stalbans_omni	Tt	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581	0.581
stalbans_omni	Edt	3.119	2.345	2.833	2.683	3.095	2.946	2.827	2.402	2.301	2.140	2.223	2.165	2.076	1.990	2.283	1.945	1.821	1.753	1.485	1.289	1.114	0.919	0.720	0.512	0.351
stalbans_omni	EDTt	3.106	2.579	2.889	2.766	3.087	2.853	2.642	2.334	2.494	2.169	2.298	2.258	2.238	2.085	2.261	1.975	1.983	1.801	1.632	1.406	1.196	0.975	0.768	0.613	0.496
stalbans_omni	IACC	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
DENSIL CABRERA																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
stalbans_omni	EDT	2.962	2.036	2.599	2.592	2.136	3.184	2.719	2.374	1.855	1.708	2.018	1.744	1.996	2.042	1.890	1.800	1.668	1.623	1.289	1.032	0.988	0.642	0.538	0.379	0.276
stalbans_omni	T20	3.293	2.737	2.985	2.881	3.431	2.913	2.590	2.438	2.537	2.485	2.401	2.263	2.371	2.255	2.287	1.988	2.039	1.809	1.642	1.410	1.183	0.958	0.752	0.550	0.441
stalbans_omni	T30	3.407	3.392	2.897	3.188	3.205	2.975	2.643	2.710	2.667	2.579	2.520	2.426	2.404	2.365	2.255	2.125	2.081	1.842	1.692	1.432	1.209	0.966	0.743	0.564	0.452
stalbans_omni	C50	-0.539	1.788	-4.068	-5.524	-3.600	-0.112	1.245	1.506	-0.939	2.676	1.744	1.819	2.900	2.762	2.418	1.760	4.437	6.462	8.293	9.230	8.427	11.337	11.866	14.447	18.148
stalbans_omni	C80	0.277	2.880	-3.220	-1.046	0.576	2.987	4.042	3.408	0.376	4.176	2.926	3.338	4.100	4.074	4.328	3.306	5.627	7.734	10.269	11.116	10.388	13.526	14.159	17.863	22.347

A 1: Comparison of parameters obtained with the different softwares for a mono impulse response.

AURORA																											
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
stalbans_b_ortf	C50	-2.532	-2.686	-4.988	-2.200	-1.780	-1.150	-1.111	-0.023	-0.427	-1.503	-3.079	-2.802	-2.782	-1.156	0.221	1.034	1.558	2.112	3.535	5.329	6.257	7.237	8.238	9.986	11.737
stalbans_b_ortf	C80	0.095	-1.777	-3.392	-0.069	0.857	1.240	0.559	1.195	1.196	0.362	-0.610	-0.640	-0.587	0.561	1.755	2.512	3.066	3.701	5.132	7.112	8.200	9.322	10.613	12.659	15.400
stalbans_b_ortf	D50	35.82	35.01	24.08	37.60	39.89	43.42	43.64	49.87	47.54	41.43	32.98	34.41	34.51	43.38	51.28	55.92	58.87	61.92	69.30	77.33	80.86	84.11	86.95	90.88	93.72
stalbans_b_ortf	Ts	165.40	185.15	209.44	161.17	139.30	131.49	131.76	119.43	118.22	124.90	138.52	137.48	135.83	118.74	103.58	92.58	82.92	71.96	55.65	40.97	34.52	28.80	24.07	18.81	15.11
stalbans_b_ortf	EDT	2.864	2.513	2.332	2.238	1.953	1.907	2.103	2.168	2.127	2.011	2.106	2.107	2.125	2.012	2.013	1.924	1.768	1.585	1.314	1.005	0.803	0.613	0.465	0.305	0.202
stalbans_b_ortf	T20	3.278	3.140	3.147	2.934	3.049	2.992	2.848	2.602	2.516	2.484	2.383	2.309	2.268	2.209	2.123	2.031	1.948	1.848	1.616	1.368	1.158	1.026	0.888	0.726	0.532
stalbans_b_ortf	T30	3.226	3.134	3.196	3.156	3.089	2.987	2.872	2.771	2.657	2.598	2.518	2.472	2.435	2.360	2.253	2.187	2.096	1.967	1.727	1.453	1.254	1.066	0.921	0.766	0.583

IMPULSE ANALYZR																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
stalbans_b_ortf	T20	2.940	3.235	3.024	3.232	3.058	2.960	2.973	2.673	2.741	2.529	2.516	2.454	2.386	2.338	2.256	2.170	2.078	1.882	1.648	1.430	1.171	0.964	0.767	0.567	0.406
stalbans_b_ortf	T30	2.751	3.036	2.961	3.436	3.348	2.900	2.953	2.706	2.983	2.670	2.630	2.620	2.525	2.486	2.379	2.254	2.189	1.951	1.741	1.489	1.215	0.982	0.788	0.592	0.423
stalbans_b_ortf	C50	-1.377	-3.642	-1.353	-1.910	-1.446	-0.523	-1.783	-0.778	-1.011	-0.438	-0.777	-0.430	-0.302	-0.351	0.046	-0.425	0.354	1.188	1.632	2.749	4.329	5.887	8.176	12.217	16.746
stalbans_b_ortf	C80	2.088	-1.158	1.119	1.212	1.361	2.435	1.240	2.004	1.958	2.419	2.487	2.407	2.709	2.699	2.984	2.540	3.462	4.631	5.260	6.432	8.399	10.295	13.193	18.718	25.389
stalbans_b_ortf	Tt	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613	0.613
stalbans_b_ortf	Edt	2.310	4.227	3.079	2.501	2.866	2.835	2.690	2.295	2.251	2.453	2.125	2.284	2.121	2.131	2.041	2.072	1.920	1.682	1.512	1.291	1.105	0.935	0.741	0.507	0.355
stalbans_b_ortf	EDTt	2.502	4.466	2.999	2.710	2.866	2.758	2.793	2.337	2.421	2.434	2.296	2.239	2.150	2.270	2.087	2.075	1.981	1.829	1.618	1.435	1.192	0.988	0.808	0.656	0.436
stalbans_b_ortf	IACC	0.814	0.887	0.887	0.690	0.538	0.751	0.611	0.578	0.520	0.530	0.365	0.442	0.356	0.381	0.255	0.286	0.346	0.228	0.285	0.248	0.259	0.207	0.200	0.201	0.216

DENSIL CABRERA																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
stalbans_b_ortf	EDT	2.136	3.818	2.573	2.014	2.838	2.529	2.404	2.932	1.828	2.058	2.010	2.284	2.006	1.935	1.974	2.015	1.731	1.596	1.437	1.154	0.930	0.718	0.543	0.304	0.212
stalbans_b_ortf	T20	3.419	3.187	2.920	3.336	2.950	3.044	2.869	2.529	2.637	2.540	2.385	2.355	2.307	2.238	2.124	2.131	2.010	1.812	1.555	1.358	1.157	0.956	0.758	0.545	0.400
stalbans_b_ortf	T30	3.683	3.199	3.065	3.396	3.257	2.867	2.941	2.576	2.746	2.606	2.525	2.490	2.446	2.355	2.242	2.201	2.106	1.902	1.655	1.448	1.199	0.967	0.768	0.572	0.421
stalbans_b_ortf	C50	1.040	3.356	-0.715	-6.812	0.366	-3.769	1.450	1.645	0.941	-1.859	1.376	-1.562	0.751	0.363	0.624	2.400	2.957	3.674	5.595	5.849	7.203	7.632	10.576	14.430	19.548
stalbans_b_ortf	C80	0.803	3.900	-0.856	-2.822	2.764	-1.023	2.296	3.333	3.173	1.193	2.282	0.760	2.156	1.972	2.589	3.913	4.468	4.955	6.975	7.650	8.903	9.937	13.033	17.859	23.924
stalbans_b_ortf	IACC Earl	0.988	0.757	0.582	0.976	0.661	0.788	0.910	0.604	0.684	0.621	0.324	0.478	0.161	0.108	0.427	0.289	0.153	0.325	0.368	0.266	0.285	0.189	0.168	0.218	0.145

A 2: Comparison of parameters obtained with the different softwares for a stereo impulse response.

AURORA																											
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
Convolved 1	C50	9.645	3.138	-0.786	0.241	-0.378	-1.700	-2.284	-1.925	-2.195	-3.020	-3.409	-3.869	-4.760	-5.183	-4.438	-3.130	-1.924	-1.686	-1.266	-0.228	1.538	3.508	4.899	6.210	8.160
Convolved 1	C80	10.40	3.64	0.17	0.92	0.62	-0.56	-0.91	-0.49	-1.06	-1.32	-1.19	-0.95	-1.58	-1.95	-1.85	-0.48	0.58	0.83	1.32	2.57	4.56	6.74	8.35	9.90	13.03
Convolved 1	Ts	34.21	84.89	137.06	124.54	124.10	128.90	129.27	137.23	153.72	170.27	169.09	164.07	174.37	181.52	174.27	141.11	110.77	102.60	93.47	78.37	59.93	45.12	36.91	30.80	23.53
Convolved 1	EDT	0.282	2.292	2.172	2.316	2.287	2.107	1.836	2.029	2.249	2.547	2.619	2.545	2.568	2.579	2.488	2.072	1.631	1.510	1.364	1.177	0.930	0.702	0.564	0.478	0.359
Convolved 1	T20	2.298	2.585	2.561	2.249	2.180	2.073	2.200	2.399	2.564	2.760	2.883	2.833	2.731	2.577	2.413	2.144	1.743	1.560	1.411	1.205	1.019	0.805	0.646	0.533	0.430
Convolved 1	T30	5.075	3.147	2.986	2.490	2.373	2.236	2.482	2.582	2.824	3.050	3.193	3.078	2.947	2.713	2.579	2.214	1.815	1.619	1.467	1.262	1.051	0.851	0.681	0.550	0.460
IMPULSE ANALYZR																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
Convolved 1	T20	7.244	6.967	7.613	3.061	2.318	2.111	2.124	2.745	2.622	3.201	3.286	3.201	2.809	2.783	2.469	2.243	1.836	1.628	1.401	1.187	0.945	0.711	0.553	0.475	0.633
Convolved 1	T30	6.809	5.986	6.589	3.710	2.467	2.068	2.288	3.256	3.022	3.891	3.751	3.617	3.098	3.052	2.663	2.473	1.952	1.700	1.505	1.245	1.019	0.779	0.625	1.794	4.654
Convolved 1	C50	-10.577	-9.372	-5.648	-1.006	-1.852	0.787	0.502	-0.557	-0.817	-2.136	-2.213	-0.684	-1.196	-1.748	-1.105	-0.200	0.995	1.728	2.481	3.606	5.371	8.526	11.344	14.623	15.531
Convolved 1	C80	-8.394	-7.104	-3.296	1.726	1.243	3.483	3.668	2.617	1.912	0.564	0.392	2.007	1.449	1.065	1.606	2.766	4.196	5.199	6.122	7.523	9.893	14.253	18.709	22.987	23.162
Convolved 1	Tt	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646	0.646
Convolved 1	Edt	9.745	8.766	8.242	2.535	2.079	2.446	1.789	2.237	2.358	2.801	2.931	2.867	2.695	2.552	2.444	2.025	1.756	1.542	1.363	1.153	0.925	0.680	0.511	0.409	0.379
Convolved 1	EDTt	21.135	11.529	6.206	2.599	2.042	2.145	1.982	2.323	2.452	2.836	2.885	2.941	2.688	2.515	2.401	2.136	1.802	1.619	1.415	1.210	1.035	0.918	0.955	1.331	1.888
Convolved 1	IACC	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----
DENSIL CABRERA																										
Filename	Params.	63	80	100	125	160	200	250	315	400	500	630	800	1000	1250	1600	2000	2500	3150	4000	5000	6300	8000	10000	12500	16000
Convolved 2	EDT	2.485	6.638	3.422	2.309	2.348	2.288	1.839	1.917	2.249	2.629	2.859	2.515	2.440	2.690	2.568	2.039	1.782	1.487	1.402	1.131	0.906	0.609	0.495	0.339	0.359
Convolved 3	T20	0.779	1.157	4.229	2.634	2.261	2.103	2.057	2.456	2.441	2.832	3.039	2.897	2.735	2.626	2.351	2.165	1.751	1.574	1.315	1.132	0.898	0.677	0.521	0.403	0.344
Convolved 4	T30	0.713	0.731	4.001	2.968	2.353	2.072	2.297	2.743	2.558	3.177	3.299	3.198	2.776	2.825	2.452	2.159	1.813	1.610	1.403	1.138	0.925	0.703	0.528	0.409	0.358
Convolved 6	C50	8.791	10.442	11.932	-0.866	-2.494	-1.919	-3.549	-0.649	-1.949	-3.565	-3.453	-3.213	-5.728	-5.530	-3.980	-3.240	-2.378	-1.601	-1.461	-0.514	1.519	4.198	6.003	8.306	12.565
Convolved 7	C80	8.686	10.447	13.777	0.005	-1.510	-0.331	-1.563	0.185	0.463	-2.546	-0.709	-0.566	-1.233	-2.913	-1.176	-1.213	0.368	0.831	1.133	2.447	4.798	7.635	9.590	13.579	17.051

A 3: Comparison of parameters obtained with the different softwares for an impulse response obtained convoluting a sine sweep with its inverse filter.

B. Code.

Reverberation time (R_{20} y T_{30}) and EDT

```
def RT(signal, fs):
    '''
    Calculates the Reverberation time of signal by T20, T30 and EDT methods.

    Args:
        sig (numpy array): array containing the audio signal.
        fs (int): sample rate of audio file.

    Returns:
        parameter_results (numpy array): array containing each parameter
            result.
        parameters_names (numpy array): array containing each parameter name.
    '''

    parameters_names = ['Edt', 'T20', 'T30']
    parameters_boundaries = [[-1, -11],[-5,-25],[-5,-35]]
    parameter_results = np.zeros(3)

    for idx, bound in enumerate(parameters_boundaries):
        start = np.where(signal >= -1 )[0][-1]

        end = np.where(signal>= bound[1])[0][-1]

        rt_i = 1/fs*np.arange(start, end)
        poly = np.polyfit(rt_i, signal[start:end],1)
        parameter_results[idx] = -60 / poly[0]

    return parameter_results, parameters_names
```

C_{50} & C_{80}

```
def C(signal,fs, ci):

    signal = 10**(signal/10)
    t = int(np.round((ci / 1000.0) * fs))
    c_i = 10 * np.log10((np.sum(signal[:t]**2))/(np.sum(signal[t:-1]**2)))
    if ~np.isfinite(c_i):
        c_i = '---'
    return c_i
```

IACC

```
def IACC(signal_L, signal_R, fs,band ,stereo, stereo_ss):
    iacc_coefficients =[]

    if stereo==0 and band==1 and stereo_ss==0:
        iacc_coefficients = np.array(['----']*10)
    elif stereo==0 and band==3 and stereo_ss==0:
        iacc_coefficients = np.array(['----']*29)
    else:
        try:
            frequency, ir_filter_L = filter(signal_L,fs,band)
            frequency, ir_filter_R = filter(signal_R,fs,band)
```

```
            for ir_L, ir_R in zip(ir_filter_L,ir_filter_R):
                t80 = int(0.08*fs)
                ir_L = ir_L[np.argmax(ir_L):] #Cut form peak beyond
                ir_R = ir_R[np.argmax(ir_R):]

                ir_L = ir_L[:t80] #Cut first 80 ms
                ir_R = ir_R[:t80]
                iacc = np.correlate(ir_L, ir_R, 'full')
                iacc_normalized = iacc/(np.sqrt(np.sum(ir_L**2)*np.sum(ir_R
                    **2)))
                iacc_coefficient = round(np.max(np.abs(iacc_normalized)),3)
                iacc_coefficients = np.append(iacc_coefficients,
                    iacc_coefficient)
        except Exception as e:
            raise ValueError("This is not a stereo file, choose mono. " + str
                (e))

    return iacc_coefficients
```

T_t & EDT_t

```
def Transition_time_and_Edt(ir, fs, ir_smoothing):
    '''
    Calculates the transition time and EDTt values.

    Args:
        ir (numpy array): array containing the audio signal.
        fs (int): sample rate of audio file.
        ir_smooth (numpy array): array containing the smoothed audio signal.

    Returns:
        Tt_time (float): Transition time value.
        edtt (float): EDTt value.
    '''
    ir = ir[np.argmax(ir):] #Cut from max value

    energy_99 = np.sum(ir**2)*0.99
    cummulative_energy = np.cumsum(ir**2)

    Tt = np.where(cummulative_energy <= energy_99)[0][-1] #Tt value in
        samples

    Tt_time = Tt/fs  #Tt value in seconds

    ir_edtt = ir_smoothing[:Tt]

    if Tt_time < 0.001:
        edtt = '---'
    else:
        poly = np.polyfit(np.arange(0, Tt_time, 1/fs)[:len(ir_edtt)], ir_edtt
            , 1)
        edtt = -60/poly[0]

    if ~np.isfinite(edtt):
        edtt = '---'

    return Tt_time, edtt
```

Schroeder, MMF, Hilbert

```
def Schroeder(signal):
    '''Smooth the signal (signal) utilizing the Schroeder integral method'''
    return np.sum(signal) - np.cumsum(signal)


def mmf(signal, window_size):
    """Applies a moving mean filter with a specified window length to a 1D
       signal.
    The values when the window is half empty (beggining and end) are
       discarted."""
    return np.convolve(signal, np.ones(window_size)/window_size, mode='valid'
       )


def Hilbert(signal):
    '''Smooth the signal (signal) utilizing the Hilbert Transform method'''
    return np.abs(sgn.hilbert(signal))
```

Lundeby

```
def Lundeby(signal, fs):
    '''
    Lundeby method as especified in "On the effects of pre-processing of
       impulse responses in the evaluation of
    acoustic parameters on room acoustics" by Venturio and Farina in 2013.

    Lundeby method is based on an iterative algorithm that chooses the right
       temporal interval to which the linear
    regression is calculated. This will leave out unwanted contributions.

    1. Average squared IR in local time intervals (10-50 ms)
    2. Estimate background noise level using tail (last 10% of the signal)
    3. Estimate slope of decay from 0 dB to noise level (the    left    point
        is 0 dB. Search the    right    point 5-10 dB above noise level)
    4. Find the crosspoint defined by regression line and noise level
    5. Find a new local time intervals based on the actual slope (use 3-10
       intervals per 10 dB of decay)
    6. Average squared IR in new local time intervals
    7. Estimate background noise (starting from a time corresponding to a
       decay of 5-10 dB based on actual
    slope after the actual crosspoint but 10% of length of IR should be the
       minimum length)
    8. Estimate late decay slope (a dynamic range of 10-20 dB should be
       evaluated, starting 5-10 dB above
    the noise level)
    9. Find a new crosspoint
    10. Repeat 7, 8 and 9 until convergence of crosspoint is achieved
    '''
    signal = signal[np.argmax(signal):]/np.max(signal)  #Normalize and cut
       the IR from peak and beyond

    #1
    window = int(10*(10**-3)*fs)  #Window in samples chosen as 30 ms

    signal_average_dB = []

    for i in range(0, int(len(signal)/window)):
```

```
        window_average_rms = np.sqrt(np.mean(signal[window*i:window*(i+1)
            ]**2))
        window_average_rms_dB = 10*np.log10(np.abs(window_average_rms)/np.max
            (signal))
        signal_average_dB.append(window_average_rms_dB)

    t = np.linspace(window/2, len(signal), len(signal_average_dB))  # signal
        average time samples
    t = t.astype(int)

    #2
    signal_last10 = signal[int(0.9*len(signal)):]
    background_noise = np.sqrt(np.mean(signal_last10**2))
    background_noise_dB = 10*np.log10(background_noise/max(signal))  # Noise
        level of last 10% of signal

    #3
    if len(np.argwhere(signal_average_dB > background_noise_dB + 10)) < 1:
        return fs*3     # if the method fails a 3 seconds max TR is supossed

    cross_signal_noise = np.argwhere(signal_average_dB > background_noise_dB
        + 10)[-1][0]  # Crosspoint between signal and 10 dB above noise level

    poly = np.polyfit(t[0:cross_signal_noise], signal_average_dB[0:
        cross_signal_noise],1)  #Regression linear (1 degree)

    #4
    cross = (background_noise_dB - poly[1])/poly[0]  #Crosspoint between
        regression and noise in samples
    cross = int(cross)

    for i in range(0, 7):  # Do 7 iterations (it should converge after 5, but
         noisy signals converge after 7 using this script)

        #5
        divisions = 10  # 3 to 10 divisions per 10 dB of regeression slope (
            use 7)
        window_new = int(divisions/poly[0]/-10)  # new window length in
            samples [amount of divisions in segment (7) / slope (-x dB) / per
            (-10 dB) decay]

        #6
        signal_average_dB = []
        for i in range(0, int(len(signal[:int((cross-(poly[0]/-10)))])/
            window_new)):  #calculation with new window
            window_average_rms = np.sqrt(np.mean(signal[window_new*i:
                window_new*(i+1)]**2))
            window_average_rms_dB = 10*np.log10(np.abs(window_average_rms)/np
                .max(signal))
            signal_average_dB.append(window_average_rms_dB)

        t = np.linspace(window_new/2, len(signal[:int((cross-(poly[0]/-10)))
            ]), len(signal_average_dB))  # signal average time samples

        if len(t) < 1:
            return fs*3     # if the method fails a 3 seconds max TR is
                supossed
        t = t.astype(int)

        #7
```

```
        signal_5dB_after_cross = signal[int(cross+(5/-poly[0])):]  # signal
            after 5 dB decay from crosspoint

        if len(signal_5dB_after_cross) < (len(signal)/10):
            signal_5dB_after_cross = signal[int(len(signal)*0.9):]  # It has
                to have at least 10% of the length of the original signal

        background_noise = np.sqrt(np.mean(signal_5dB_after_cross**2))
        background_noise_dB = 10*np.log10(background_noise/max(signal))

        #8
        poly = np.polyfit(t, signal_average_dB, 1)

        #9
        cross = (background_noise_dB - poly[1])/poly[0]  #Crosspoint between
            regression and noise in samples

    return int(cross)
```

Questions

What would be the best method to cut the limits of the impulse response, so that the Schroeder method and Moving Average Filter give similar results? When we talk about results, we are referring to the acoustic parameters calculated through the GUI, for this we are looking for both methods to give similar results.