DEVELOPMENT OF AN IMPULSE RESPONSE PROCESSING SOFTWARE

Tomás Gamalerio

Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
Email: tgamalerio@gmail.com

Martín Flaiban

Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
Email: flaibanmartin@gmail.com

Juan Cruz Ojeda

Universidad Nacional de Tres de Febrero, Ingeniería de Sonido, Caseros, Buenos Aires, Argentina
Email: juan_ojeda@hotmail.com.ar

This paper describes the development of an impulse response processing software. The program is developed in Matlab, with a dynamic user interface that has the capability of processing a set of room impulse responses, stereo or mono. This software calculates the acoustical parameters EDT, T20, T30, C50, C80, IACCe, Tt, EDTt using two possible smoothing methods. Also new parameters IACCt and Ct are calculated; the results are shown with the corresponding deviation in tables and graphics. To validate the performance of the software the results are compared with Aurora Acoustical Parameters plug in for Audacity, using an IR from Teatro Colón. The software developed showed an optimal and efficient performance calculating the acoustical parameters, although the type of smoothing showed high sensibility on the early decay and also some pre-processing of the IR stages need to be adjusted.

Keywords: ISO 3382-2, Impulse response (IR), reverberation time RT, smoothing, Clarity, IACCe, Mixing Time (Tt).

1. Introduction

In signal processing the impulse response h(t) or also the transfer function is a fundamental property of a linear and time invariant dynamic system; it represents the output or reaction when it is excited with an ideal impulse, like the delta of Dirac. Once this property of the LTI system is determined, its response to any type of excitation can be calculated by means of the convolution. This has big implications since it allows to anticipate the response of the system before it is built, using computational models for simulation.

The impulse response function can be used to describe the characteristics and behavior of an electro acoustic system, the response of a filter or an electrical circuit, but most of the time it is used to analyze the acoustical behavior of enclosures, such as control rooms or home studios, theaters, concert halls and auditoriums, or a conference or class room. This is because this impulse response can be used to calculate many parameters that can describe and characterize the acoustical quality and subjective perception of a certain auditorium or room enclosure. Some of these parameters are the reverberation time, the early decay time EDT, the clarity parameters like C50 and C80, spatial parameters like IACC, or subjective parameters such as LEV or ASW. Thus these parameters are also used in simulation to anticipate the acoustic characteristics of the rooms and to achieve the design that best suits for the future application of the room.
The aim of this study is to develop a software with an user interface capable of processing a number of impulse responses obtained from logarithmic sine sweep recordings and their corresponding inverse filter, and calculating different acoustical parameters, with their corresponding deviation, which are the most frequently used for characterization of rooms.
The user interface and processing are designed with MATLAB software and then as a method for validation and testing of the software developed the results are compared with other similar softwares such as Aurora from Audacity and also Texture v.~~5~~6 another Matlab software made by A. Bidondo and L. Pepino et al. [6]

2. Background theory

2.1. Impulse response of a room

There are many ways to measure the impulse response of a room or enclosure. Using a delta of Dirac as a measurement signal is not an appropriate method. This is because the duration of this signal is too short, it can not generate impulse responses with enough signal to noise relation, and the emission of such a signal at high levels can damage the loudspeaker or even produce high amounts of distortion. Therefore other techniques of impulse response generation were developed. Some of these techniques use impulsive sources that create a pulse of very short duration similar to a Dirac delta. A few examples of this kind of sources could be a balloon, an applause, a gunshot, wooden bars, or the origamis. But these sources have a few problems: with the sound level reproduced, the bandwidth of the impulses, and most of all with their reproducibility and repeatability.
This work is focused on another and more recently measurement method. The *logarithmic sine sweep (LSS)*. This signal is reproduced in the room with ideally an omnidirectional sound source, and it is recorded by a microphone capturing the room response and its acoustical characteristics. Then, the impulse response of the room can be obtained through the convolution of the sine sweep recorded with its proper inverse filter, which consists of the input signal inverted along the time axis. This process is illustrated in ~~the next~~ figure 1.

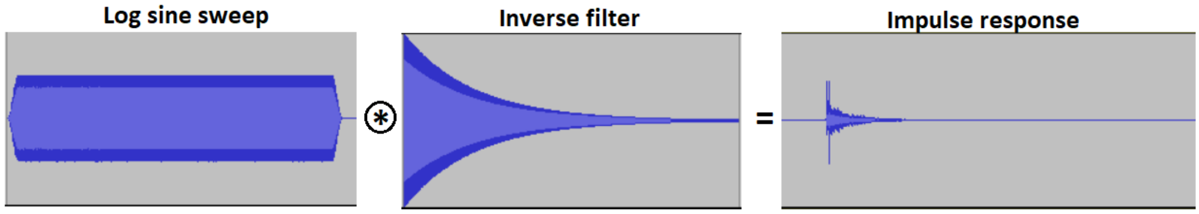

Figure 1: Calculation of an impulse response by convolution.

This method provides a serial number of advantages. First of all, this signal allows a repeatable and reproducible measurement method. Then, the exponential sine sweep provides a considerable advantage with regards to SNR compared with linear sine sweep [7], periodic pulse or MLS techniques, given that this test signal excites each frequency at a time.
Using traditional methods the non-linear response can not be separated from the linear response. With this method the non-linear effects can be perfectly separated from the linear response applying a window, given that deconvolved output presents a clean separation of linear response and harmonic distortion, which allows the linear response to be delineated. So this allows to analyze the impulse response with no influence of the non linear response of the sound source, and thus to use the source at a sufficient level to accomplish the appropriate signal to noise relation worrying less for the nonlinear limits.

2.2. Reverberation Time and Early Decay Time

This is one of the best known parameters, a sort of fundamental of acoustic science. The *reverberation time (RT or T60)* is defined as the time (in seconds) needed by the sound pressure level to decrease by 60 dB; the rate of decay is measured by the linear least-squares regression of the measured decay curve from a level 5 dB below the initial level to 35 dB below; in this case this one is called T30 [8].
It is important to be aware that to have a correct measurement of the reverberation time the signal to noise relation should be enough. To measure T60, this should be at least 60 dB. Sometimes, because of the background noise of the room and the capabilities of the sound source, this is hard to accomplish, so T30 and T20 can be used in this case. If the decay of the curve is linear, then T60 can be obtained from them.
In other hand, the *early decay time (EDT)* is a more specific parameter, it measures the time taken by the energy curve to attenuate 10 dB. Unlike the T20 and T30, which are measured from a starting point of -5 dB, the EDT is measured from 0dB to -10 dB.

2.3. Clarity parameters

These parameters express a balance between early and late arriving energy of the impulse response that is useful to measure the clarity of the sound as perceived by human ears. The Cte parameter is used with te = 50 ms (C50), or te = 80 ms (C80) depending on the destination of the room, speech or music listening, respectively. Cte is defined in equation (1).

$$C_{te} = 10\, log \frac{\int_{0}^{te} p^2(t)\, dt}{\int_{te}^{\infty} p^2(t)\, dt}\ [dB] \tag{1}$$

where p(t) is the measured impulse response ~~in case~~.

2.4. Interaural cross-correlation

The human spatial perception is because of the natural stereo human sound hearing system, given by the difference between the signals arriving to each one of the two ears. If no difference were between left and right, then no spatial information will be received and the listener won't be able to locate the sound source.
This parameter measures the similarity between the left and right channels in a stereo signal, and it can be attributed to the spatial perception in the human hearing system. So with a binaural microphone and a dummy head it is possible to record what arrives at two ears, and a cross correlation between these two signals can show the spatial degree of the sound perceived by a human. A low value of IACC shows a high degree of spatial information.
The Interaural Cross Correlation function (IACF) is defined in Equation (2).

$$IACF_{t1/t2}(\tau) = \frac{\int_{t1}^{t2} p_t(t)\cdot p_r(t+\tau)dt}{\sqrt{\int_{t1}^{t2} p_t^{\,2}(t)\, dt \cdot \int_{t1}^{t2} p_r^{\,2}(t)dt}} \tag{2}$$

and the Interaural Cross Correlation Coefficient (IACC) is defined as the maximum of the IACF and it showed correlation with the subjective spatial impression of the human in a concert hall (REF). IACC is defined in Equation (3).

$$IACC_{t1/t2} = max\left(\left|IACF_{t1/t2}(\tau)\right|\right) \tag{3}$$

for $-\ 1\, ms\ \leq\ \tau\ \leq +\ 1\, ms$

It has also been defined another variant as the *early interaural cross correlation coefficient* (IACCe) which is calculated over the first 80 ms of the signal.

2.5. Moving Median Filter

The moving median filter is a non-linear digital filtering technique which is often used to remove noise from an image or signal. This noise reduction is a typical pre-processing way to improve the results of later processing, for example in the case of this study it can be used to realize a smoothing to the decay curve.
The median for each sample is defined in Equation (4).

$$Med(X) = \begin{cases} X[\frac{n}{2}] & \text{if } n \text{ is even} \\ \frac{X[\frac{n-1}{2}]+X[\frac{n+1}{2}]}{2} & \text{if } n \text{ is odd} \end{cases} \tag{4}$$

Where *n* is the window size, *X* a sample of the signal to be filtered.

2.6. Noise compensation - Lundeby method

When calculating the reverberation time through the impulse response it is necessary to have enough signal to noise relation, to have a clear separation of the decay curve and the background noise. Authors like Chu, Lundeby and Hirata have designed methods to compensate for background noise [2], these methods find the final point of the impulse response to separate it from the background noise.
In the case of this software the Lundeby method is applied to find the correct right temporal interval from which the integration limit of the Schroeder integral is extracted.

2.7. Schroeder integral

The schroeder integral is an operation that uses backward integration of the squared impulse response to obtain a smooth quantity. It can be also used to generate a smooth decay curve from the impulse response and appropriated for the calculation of the reverberation time and the other parameters [2].
The formula that defines the Schroeder integral is the following

$$E(t) = \int_{t}^{\infty} h^2(t)\,dt = \int_{0}^{\infty} h^2(t)\,dt - \int_{0}^{t} h^2(t)\,dt \tag{4}$$

2.8. Reversed filtering of impulse response

The reverberation time along the frequency range is often analyzed in an octave band or in a third octave band. These bandpass filtering, mostly in third octave bands, can introduce error when measuring short reverberation times, due to the filter ringing. This filter ringing, which can be observed in ~~the~~ figure 2, can influence the original decay curve of the sound.
According to Jacobsen [9], a representative decay curve is obtained only if the next condition is accomplished.

$$B.T60 > 16$$

This phenomenon can be observed in ~~the following~~ figure 2.

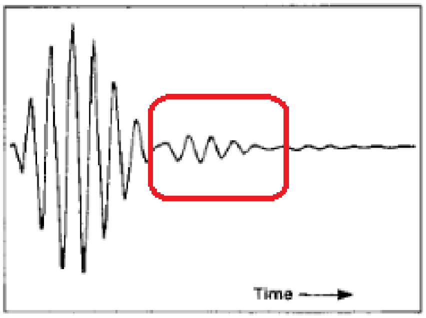

Figure 2: Filter ringing.

In investigations from the authors Jacobsen and Rindel and also Rasmussen [9] it has been demonstrated that reversing the impulse response along the time axis brings an improvement to this issue of the response of the filters. This is possible due to the fact that the impulse response of the filters are asymmetrical. This process will be applied for the third band filtering.

3. State of the Art

3.1. Mixing Time

The mixing time (T_t) is an acoustical parameter which divides the IR in two temporal regions. The first one contains the direct sound, usually called early reflections. This region is delimited between the initial time of the IR and this instant Tt. The second region contains the tale of the IR, also called late reflections. After Tt, the reverberation tale can be considered as an exponentially decay gaussian noise. This parameter denotes the dynamical behavior of the system under analysis; before Tt the behavior of the system is deterministic, and after Tt the behavior of the system is stochastic.
In this work Tt parameter is calculated following a new method proposed by A. Bidondo and L. Pepino et al. [6]. According to this paper Tt is defined as the instant when the cumulative outliers function reaches 99% of its maximum.
This new definition of the term "Early" suggests a new way to calculate traditional acoustical parameters like “Early Decay Time” , “Clarity” or even “Interaural Cross Correlation Early” which have always been arbitrarily calculated between the first 80ms.
In this work three new version parameters are calculated related to the new definition of Mixing time, according to this new definition the term “Early”. These parameters are “EDT_t”, “C_t” and “$IACC_t$”.

4. Procedure and development

As mentioned before, the software for processing impulse responses is developed in Matlab. The GUI user interface is designed using the tool App Designer from matlab. This tool allows to do first the graphical design of the user interface and once it's finished, the signal processing of the impulse response can be done by communicating the scripts of the main functions with the different parts of the interface (buttons, graphics, etc) through the Callbacks functions of each of these GUI parts.
A first observation of the user interface designed for this purpose can be done in figure 4 of the section 3.2.

4.1. Signal processing

In figure 3 it is presented a block diagram of the different stages that conform the processing of the impulse response signal necessary to obtain the different acoustical parameters mentioned before.
This software has the capability of processing multiple impulse responses, and the shown acoustical parameters in the GUI interface are the result of the main average and they are presented with its corresponding deviation.

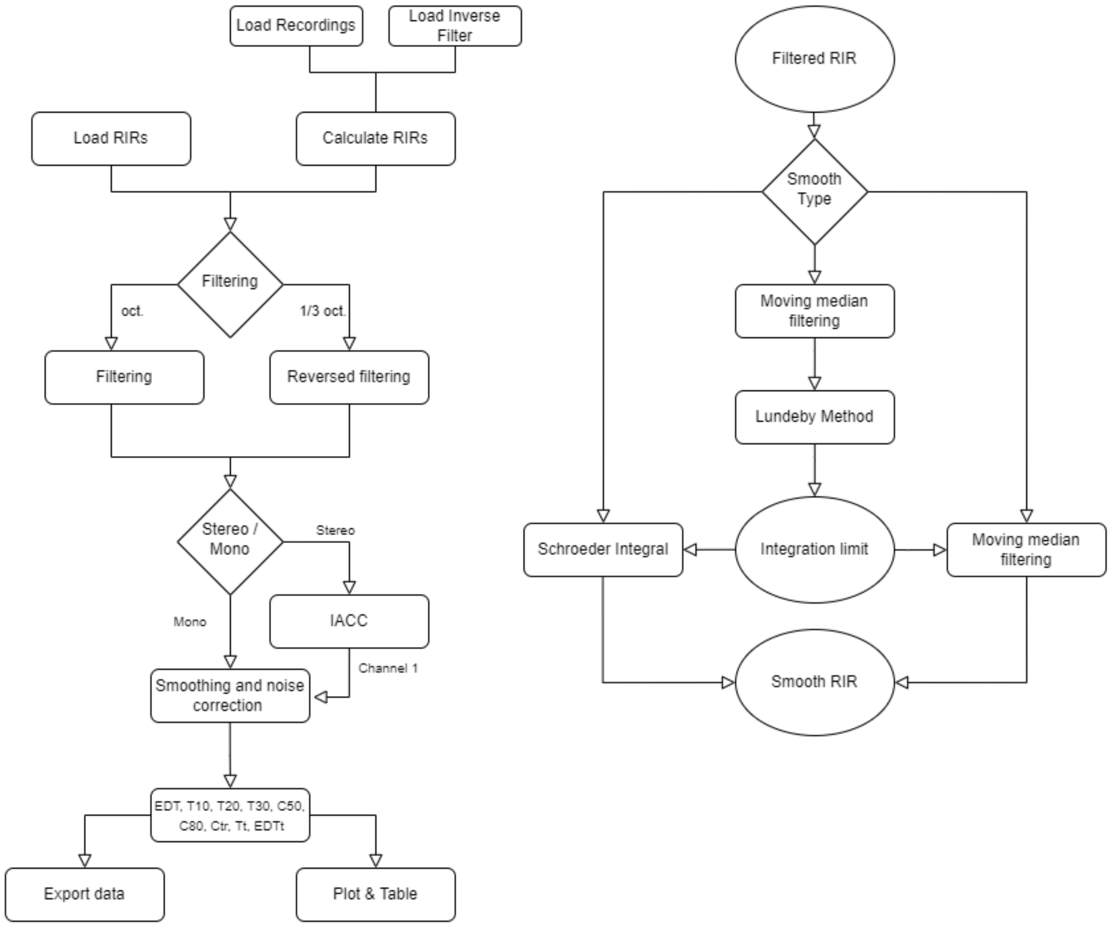

Figure 3: Block Diagram of the impulse response processing.

The first step of this process is to obtain the impulse responses. The software allows two options: one is to upload the impulse responses of the room already calculated in another software or either recorded, and the other one is to upload the recorded logarithmic sine sweeps and the corresponding inverse filter. In this last case the software will calculate the IR's using FFT taking advantage of the convolution properties to improve the code efficiency. The IR's uploaded by the user can be either mono or stereo.

The next stage in the pre-processing of the impulse response is the determination of the instant where the IR starts, and the temporal cutout. This is done over the broadband IR, before filtering. The final point of the IR is calculated with the noise correction method.

Next step is the filtering of the IR. The user has the possibility to choose between octave filters and third octave band filters, which have been designed according to standard IEC 61260 and using the functions filtfilt and deisgnfilt provided by Matlab. Third octave filtering is done following the method proposed by Rasmussen et al.[9]

If the IRs are stereo, the parameter IACC is calculated using both channels and then the first channel is used to calculate all the other acoustical parameters. If the RI's are mono, then the signal passes to the next stage.

The next stage is the noise correction made with the Lundeby method (detection of the final point of the IR) [2] and the smoothing of the IR. In this last case the user can choose between the two methods explained before, using the Moving Median Filter (MMF) or the Schroeder Integral. In case of using the MMF, the size of the window can be configured.

Once the IRs are filtered according to IEC 61260, the acoustical parameters C50, C80, Ct IACCe and IACCt (these last ones if the load files are stereo) are calculated without smoothing the IRs, but with noise correction applied. Then Tt is calculated using both energy curve decay of the IRs, without smoothing and smoothing with MMF and Schroeder, and both with noise correction.[2]
Then EDT, EDTt, T10, T20 and T30 are calculated using linear regression over the smooth IRs.
Finally, once the processing is done the results are plotted into graphics and tables for each band, each parameter with its corresponding deviation. In addition to this, the software gives the possibility to export data in an excel sheet format.

4.2. User Interface

In ~~the next~~ figure 4 the GUI interface design of the software developed is presented. An example of an impulse response already loaded and processed is used for this.

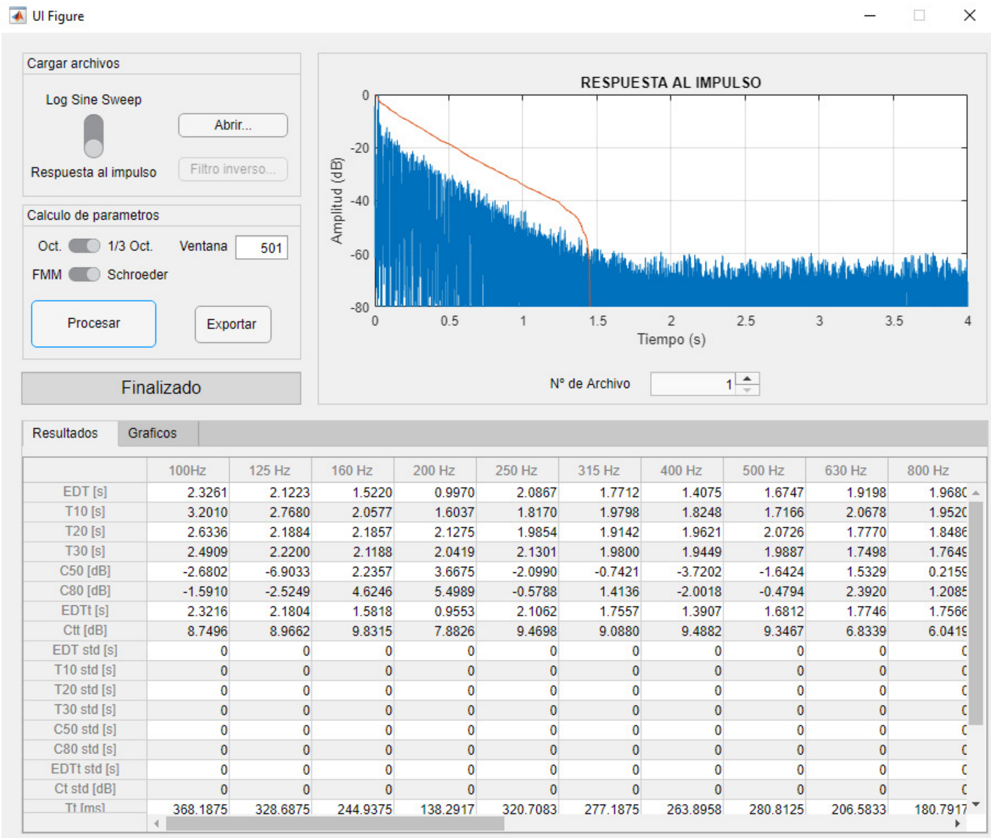

Figure 4: GUI interface with IRs from Teatro Colón.

The first panel in the upper left is to upload the files, containing a switch for the two possible options mentioned in the previous section.
Once the IRs are obtained, the interface will display a graphic of each decay curve in dB, without any filtering, over the upper right panel which has the graphic window. This allows the user to observe and verificate all the IRs uploaded before doing the whole processing, by changing the number file on the spinning number.

Then the user can run the processing of the IR after establishing the configuration of the parameters calculation: type of filtering (Octave band or Third band) and type of smoothing (MMF or Schroeder Integral, with window size in case of MMF).
When the processing is finished, the results will be presented in the below panel that has two tabs. In the first tab, the acoustical parameters results and its deviation (in case of loading more than one impulse response) will be presented in a table that fits to the type of filtering. In the second tab, all the results are presented in function of the frequency by an interpolated curve of the average and the deviation.
Then the user has the possibility to export the table containing the results of all the acoustic parameters, or images of the graphics.

5. Results and Validation

In this section the results and discussion of all the acoustical parameters calculated by the software are presented, in octave bands. The results in third octave bands can be observed in the Appendix.
For testing the functionality of the software designed, mono and stereo impulse responses of measurements from Teatro Colón are used. The mono impulse response was measured with an Earthworks microphone, and the stereo impulse response is binaural recorded with a Kemar head and torso simulator (HATS).
A validation is done by comparing these results with the given by other softwares like *Aurora* from *Audacity* using the *Acoustical Parameters* plug-in, and also another Matlab software developed by A. Bidondo and L. Pepino (*Texture v6.3*) is used for some other parameters.
The acoustical parameters are obtained using both smoothing methods (Schroeder and moving median filter), and ~~they are~~ compared with the results given by the *Acoustical Parameters* module of Aurora. In the case of the MMF, a window size of 501 samples is used, following the criteria of the minimum frequency, which is equivalent to 10,5 ms with a sampling frequency of 48000 Hz.
Performance of the software was evaluated by measuring the processing time required using another set of six mono IRs from a garage, in third octave band and using a Schroeder smoothing. The elapsed time was 56 seconds. The same test was done for the MMF, resulting in an elapsed time of 68 seconds.

5.1. Reverberation time: EDT, T20, T30

The first results shown and analyzed are the ones respecting the reverberation time. This results corresponds to the mono impulse response of Teatro Colón. In each bar graphic of each figure the results from both smoothing methods of the software developed and from *Aurora Acoustical Parameters* are presented and compared.

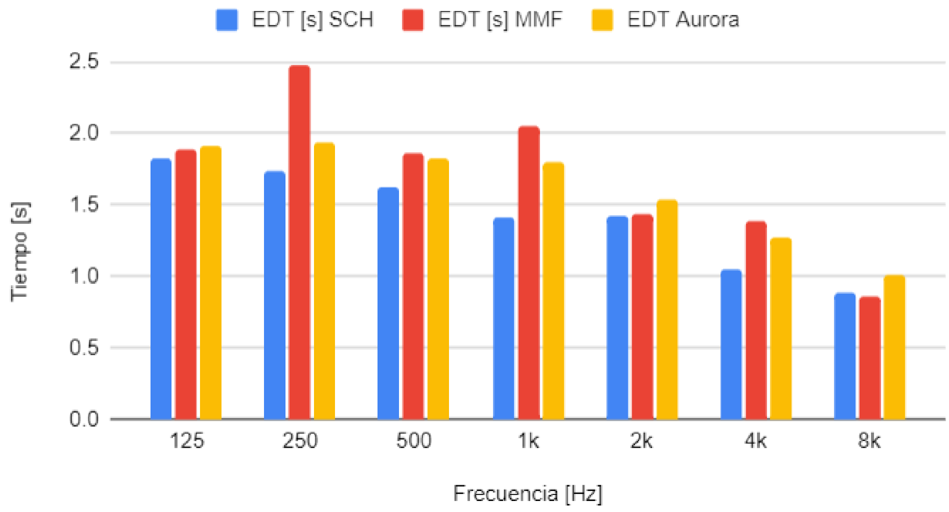

Figure 5: EDT in octave bands (Teatro Colón).

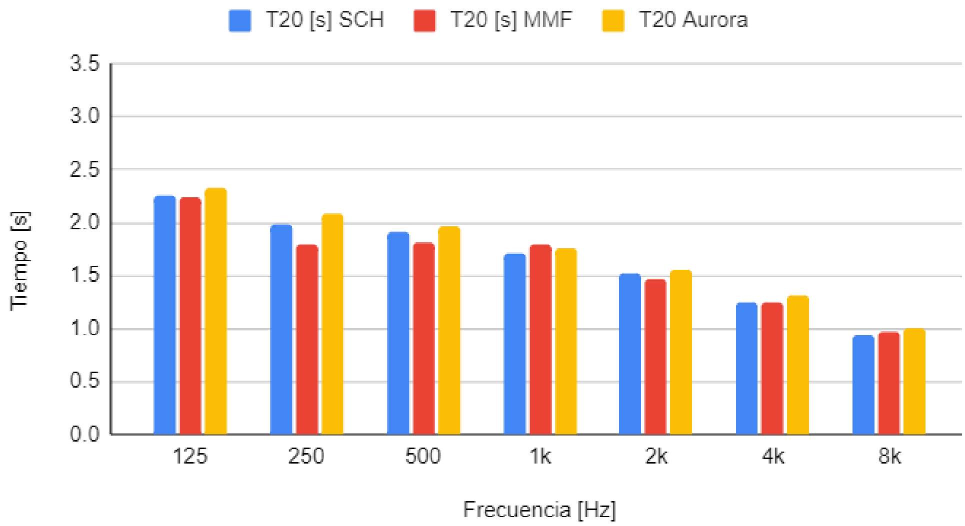

Figure 6: T20 in octave bands (Teatro Colón)

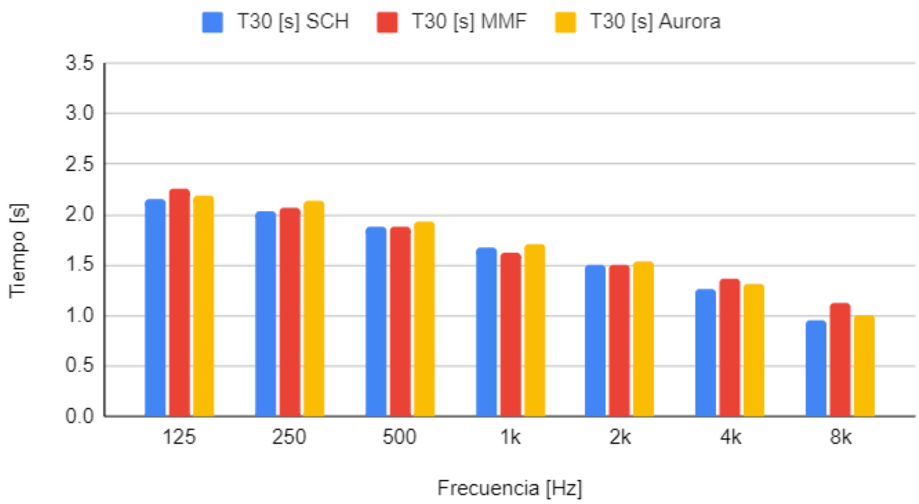

Figure 7: T30 in octave bands (Teatro Colón).

The figure 5 shows the EDT values in octave bands. The major differences between the Schroeder smoothing method and the Moving Median Filtering, and also between these two methods and the values from Aurora, were obtained with this parameter. The MMF is obtaining longer decays in most of the frequency bands. This could happen probably because the early temporal interval of the decay curve (from 0dB to -10 dB) appears to be the most sensitive zone to the smoothing method, obtaining different coefficients from the linear regression.
Much more consistency is obtained with the parameters T20 and T30, regarding figures 6 and 7. Compared with the values from Aurora, the softwares shows more accuracy using the Schroeder smoothing method when measuring reverberation time. There are many stages in the pre-processing of the impulse response, like for example the detection of the beginning or the linear regression, that can introduce differences between this software and Aurora. Nevertheless, the software shows well functioning and accurate results, regarding reverberation time parameters.

5.2. Clarity parameters C50 & C80

In this section, the clarity parameters results of the software are presented and compared to Aurora. Figure 8 shows C50 obtained by both smoothing methods, Schroeder backwards integral and MMF with a window length of 501, and the ones from Aurora.

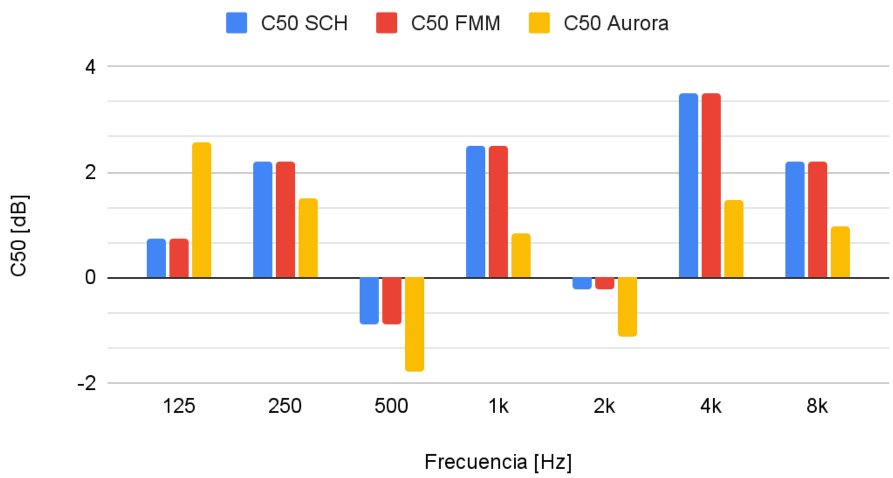

Figure 8: C50 in octave bands (Teatro Colón).

As it can be seen in figure 8, the type of smoothing does not affect the value of clarity parameters, as it was expected, given that no smoothing is needed to calculate them. The results of the software show some discrepancies with Aurora regarding the C50. These clarity parameters calculate the ratio of energy between the early and late portion of the impulse response. So this difference could be mainly by the efficiency or the true effectiveness of each algorithm to detect the t_0 where the impulse response starts, which has influence on the amount of energy of the early portion of the IR; and also where it ends, depending on the noise correction method, and defining the late portion of energy.

Nevertheless, at least the software can confirm either if the IR has more energy in the first 50 ms, or if the tail of the IR has more energy; with low accuracy but pointing in the right direction.
Figure 9 shows the results of C80. In this case the software shows much better concordance between the results.

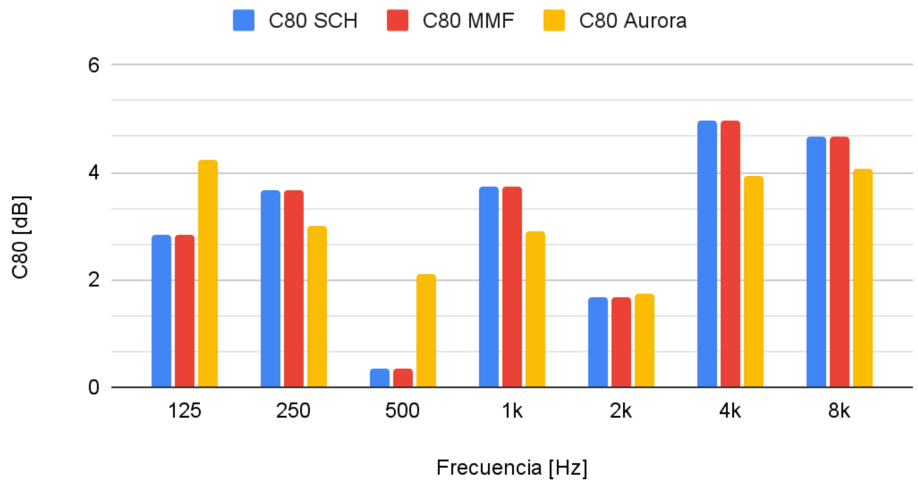

Figure 9: C80 in octave bands (Teatro Colón).

This parameter has shown less error compared to C50, this could be due to the fact that when C80 is calculated, more data points of the IR is required than in C50, naturally because the window size is 30 ms less. This difference can make C80 less sensitive to the true effectiveness of the algorithm to detect when the IR actually starts. It's important to remark that the C80 results also could be affected by the pre-processing octave or third-octave band filter because it shows better results for higher frequency than low frequency, unlike C50, which shows some discrepancies in both low and high frequency.

5.3. IACCe

In this section the results of IACCe are presented and compared to Aurora software results. Figure 10 shows IACCe in octave bands for the IR recorded at teatro colon with HATS in order to obtain binaural room impulse response. The results of the software show great concordance in almost every band with exception of 1 kHz and 8 kHz bands, showing a maximum difference of 0.25 s.

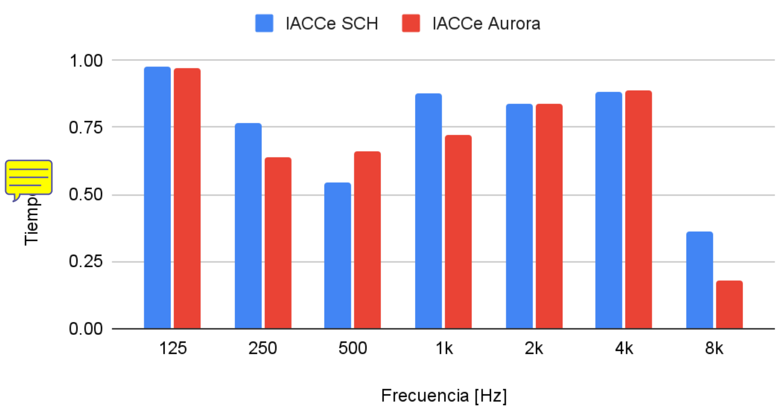

Figure 10: IACCe in octave bands (Teatro Colón HATS).

5.4. Tt, EDTt , IACCt, Ct

In this section the results of Mixing Time T_t are compared to the ones obtained through the software *Texture v6.3* designed by L. Pepino et al., also in Matlab. Figure 11 shows results in the third octave band exhibiting high concordance in the low and mid low frequency range, but some differences in high frequencies. These discrepancies are probably caused by the effectiveness of the algorithm to detect where the impulse response starts or other pre-processing stage, affecting only higher frequencies due to the fact that these are the one which generate high slope change in signals across in time.

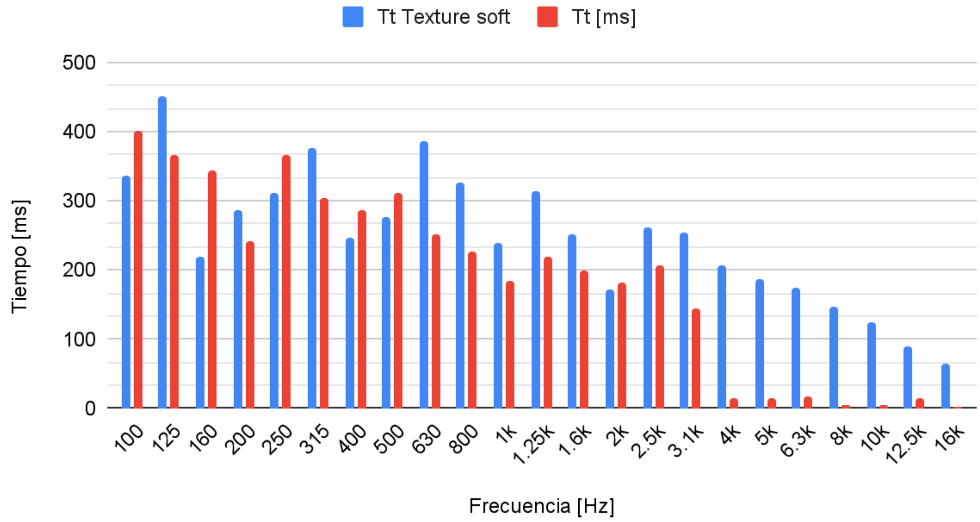

Figure 11: Mixing Time T_t in third octave bands (Teatro Colón).

Previous figure 11 not only shows that the term “Early” also depends on frequency. Moreover, this new parameter brings the possibility to redefine the integration limit in traditional parameters such as the clarity ones, in order to achieve better ways to calculate the energy in the “Early” or the deterministic’s region of the impulse response, and more representative of the sound source and also of the room in question.

Naturally, as the frequency increases, the mixing time T_t decreases in value. This effect is caused by the high density of reflections as a consequence of the scattering phenomena due to the comparable values between the decreasing value of wavelength and porosity of the surface.

In ~~the next~~ figure 12, the redefined clarity parameter called $C_{t;}$ is compared to C50 and C80 to show some differences between them.

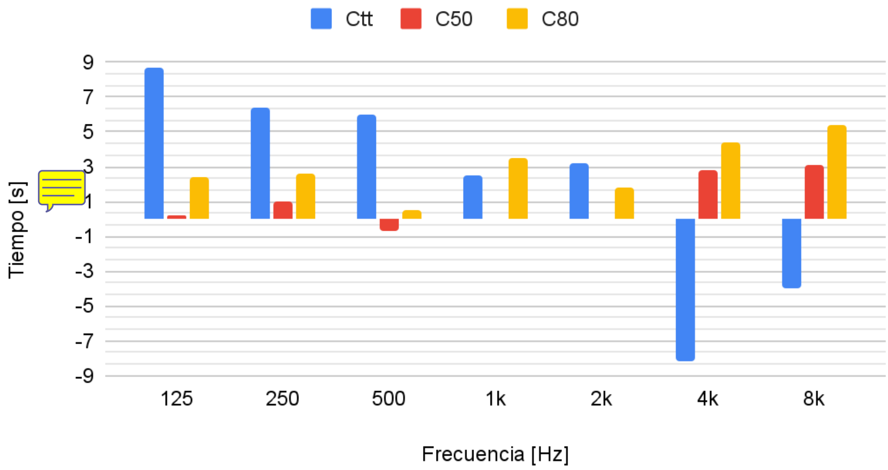

Figure 12: Clarity parameter related to T_t in octave bands (Teatro Colón).

According to figure 12 it looks like the impulse response has much more energy in the early time in low and mid frequency than the one shown by parameters C50 and even C80.
Due to the fact that the mixing time estimated with the software shows some differences in higher frequency compared to Texture v6.3 software, the values from to 4 kHz onwards are considered not valid.
In the next figure 13 Early decay time is calculated using the value of amplitude of the energy decay curve of the impulse response at mixing time in order to obtain EDTt.

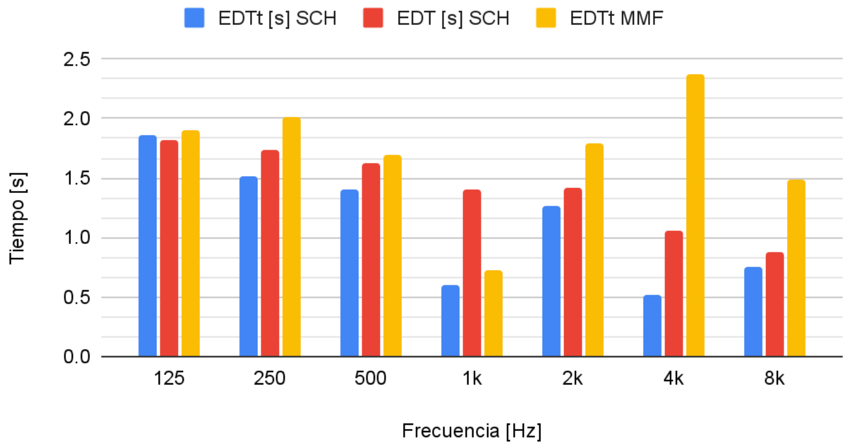

Figure 13: Early decay time parameter related to T_t in octave bands of Teatro Colón.

As it can be seen in figure 13, as the frequency increases the differences between EDTt and EDT become bigger, reaching almost 1s ~~of difference~~ between t~~hem in~~ 1 kHz and 2 kHz where Tt results are valid. It's possible to observe that the value of EDTt depends on the smoothing, and despite the fact that mixing time is the same for both, smoothing by MMF results in higher values than ~~EDTt using smoothing by~~ Schroeder.
Figure 14 shows a comparison between IACCe and IACCt. IACCt is the interaural cross-correlation evaluated from the initial part of the impulse response to the mixing time, instead of the traditional 80 ms corresponding to IACCe.

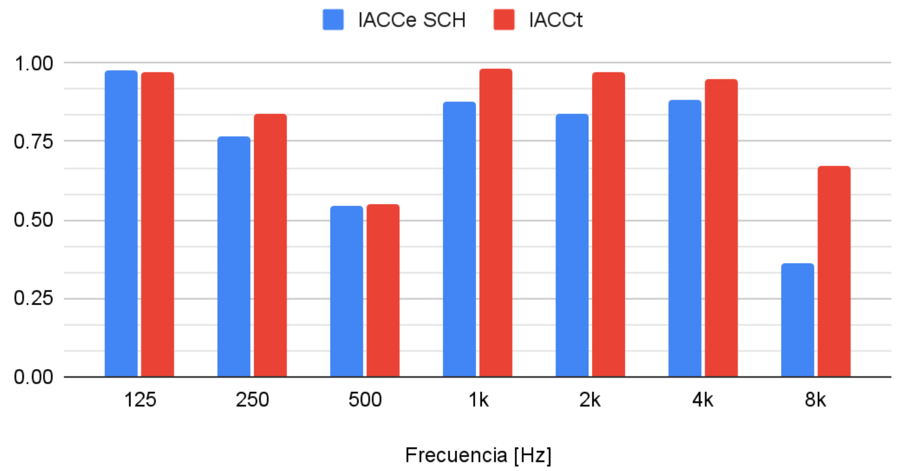

Figure 14: IACC related to T_t in octave bands (Teatro Colón).

In this figure it is shown that IACCt is higher than IACCe in the valid range provided by the comparison of Tt with Texture v6.3 software. This effect is caused because Tt is higher than 80ms, and therefore much more correlated information is taken into account in the process producing higher values of IACCt. But it is observed that the differences between IACCt and IACCe are lower than the one between Ct and C50 and C80.

6. Conclusions

As a first conclusion the acoustical parameters calculated by the software are similar in most cases to the ones shown by Aurora, showing an optimal performance and efficient processing. The software required 56 seconds to process a set of six impulse responses. This is considered a significant advantage of the software because the RT of a room or any of these acoustical parameters should be measured in several positions to do a full survey of the room. Though, it is considered that some signal processing changes could be made to accomplish a higher accuracy, and also adding another software of reference as validation would be more representative.

Respecting the two possible smoothing methods, these showed differences between them on the parameters calculated over the early decay of the IR like EDT, and the results obtained with the moving median filter showed a high dependency on the window size. It is considered that some adjustments need to be done over the preprocessing stages of the IR in order to get more accurate results.

On the other hand, mixing time provides a new method to calculate clarity, early decay time and early IACC. These three parameters are related to the fact that the term “Early”, has been measured and calculated over a constant time interval assuming that the early field was contained in the first 80ms of the impulse response, not taking into account the type of sound source or shape of the room. The time in which the field behavior changes from deterministic to stochastic determines when the field is considered early and when it’s late. But this temporal value is dependent on frequency, and also on the coupling between the sound source and the room.

The new proposed parameters calculated over the mixing time (C_t and $IACC_t$) and the differences obtained with the classical ones, shows that the actual analysis of clarity needs to be reconsidered taking into account the sound source and the room, and also the frequency. So it is considered that this software will be a useful tool in the future analysis of room acoustics.

7. References

[1] ISO, UNE EN. 3382: 2001. Acústica. *“Medición del tiempo de reverberación de recintos con referencia a otros parámetros acústicos”* (ISO 3382: 1997).
[2] A. Farina, A. Venturi, (2013), *“On the effects of pre-processing of impulse responses in the evaluation of acoustic parameters on room acoustics”* ICA montreal june
[3] M Schroeder, *“New method of measuring reverberation time”* Bell Laboratories, Murray Hill, New Jersey. (1964)
[4] IEC 61260 *“Octave band and fractional band filters”* (1995)
[5] ISO 354. Acoustics — Measurement of sound absorption in a reverberation room (2003)
[6] A. Bidondo, L. Pepino, *“Room acoustic texture : a new method for its quantification”* ICA Germany , (2018)
[7] A. Farina.(2000) *“Simultaneous measurement of impulse response and distortion with a swept-sine technique.”* Audio Engineering Society, Paris, France, 19 - 22 February
[8] A. Farina, S. Campanini, *“A new audacity feature: room objective acoustical parameters calculation module .”* Department of Industrial Engineering - University of Parma viale G.P.Usberti, 181/A 43100 Parma, Italy.
[9] B. Rasmussen, Rindel, J. Holger, H. Henriksen “*Design and measurement of short reverberation times at low frequencies in talks studios”* (1991) Denmark University JAES.

8. Appendix

8.1. Results

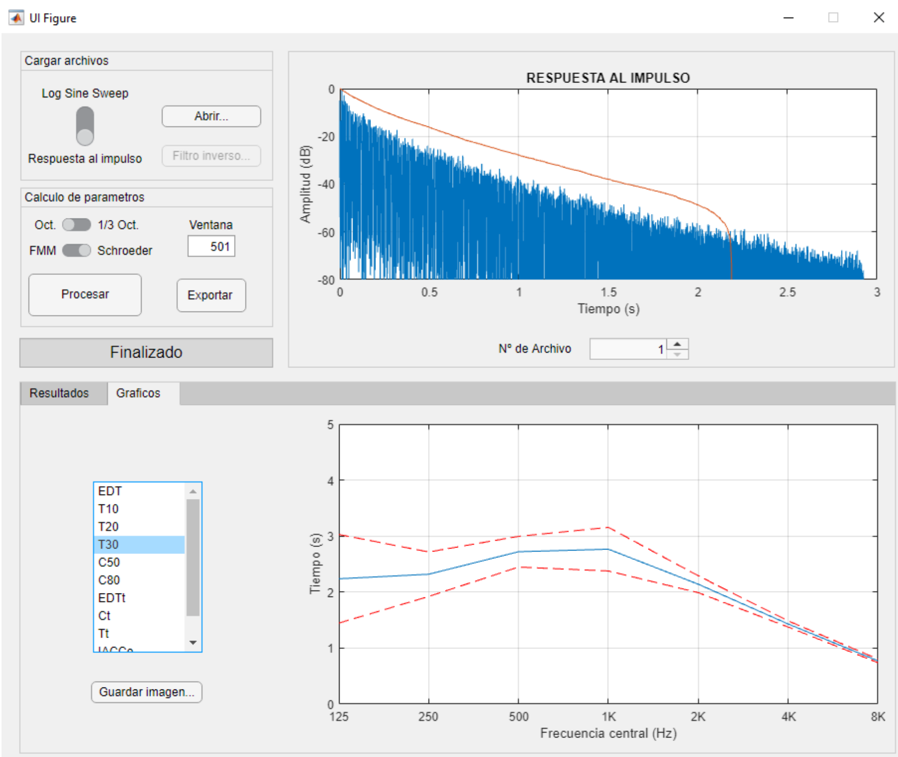

Figure A1: Example of a set of six IR with the deviation.

Table A1: Third octave band results of Teatro Colón IR with Schroeder smoothing.

Frec [Hz]	100	125	160	200	250	315	400	500	630	800	1 k	1.2 k
EDT [s]	2,35	2,11	1,49	0,97	2,09	1,77	1,42	1,68	1,92	1,97	1,13	1,48
T10 [s]	3,28	2,77	2,03	1,59	1,82	1,98	1,83	1,72	2,07	1,95	1,80	1,93
T20 [s]	2,63	2,19	2,17	2,11	1,99	1,92	1,96	2,07	1,78	1,85	1,74	1,65
T30 [s]	2,50	2,22	2,15	2,03	2,13	1,98	1,95	1,99	1,75	1,77	1,71	1,62
C50 [dB]	-3,55	-7,45	2,51	3,97	-2,21	-0,82	-3,84	-1,72	1,55	0,17	3,78	0,28
C80 [dB]	-2,28	-2,62	4,87	5,78	-0,65	1,36	-2,10	-0,54	2,42	1,17	5,15	1,82
EDTt [s]	2,35	2,17	1,55	0,92	2,11	1,77	1,48	1,68	1,77	1,76	0,70	1,37
Ct [dB]	8,78	9,00	9,70	7,76	9,50	9,19	9,85	9,46	6,80	6,08	6,67	6,90
Tt [ms]	373,58	329,40	241,44	133,92	322,13	283,40	291,33	282,35	204,96	181,71	94,06	155,06

Frec [Hz]	*1.6k*	*2k*	*2.5k*	*3.15k*	*4 k*	*5k*	*6.3k*	*8k*	*10k*	*12.5k*	*16k*
EDT [s]	1,54	1,25	1,56	1,43	1,00	0,98	1,03	0,88	0,70	0,50	0,38
T10 [s]	1,59	1,61	1,35	1,50	1,20	1,13	1,07	0,82	0,71	0,58	0,34
T20 [s]	1,56	1,56	1,46	1,42	1,22	1,17	1,03	0,84	0,68	0,57	0,35
T30 [s]	1,53	1,54	1,47	1,42	1,25	1,14	1,02	0,85	0,68	0,57	0,37
C50 [dB]	-0,51	-0,14	-1,10	0,66	2,82	3,63	2,01	2,47	4,30	5,27	12,06
C80 [dB]	0,99	2,23	0,34	2,42	4,34	5,18	3,84	5,45	6,98	8,72	16,33
EDTt [s]	1,37	1,26	1,56	1,29	1,75	0,84	0,69	0,49	0,53	0,79	0,07
Ct [dB]	6,29	7,42	6,23	4,99	-0,12	1,29	0,48	-2,46	-3,04	1,86	2,71
Tt [ms]	155,1	170,3	184,9	128,3	25,8	25,7	31,4	4,4	4,2	27,0	2,0

Table A2: Third octave band results of Teatro Colón IR with MMF smoothing.

Frec [Hz]	*100*	*125*	*160*	*200*	*250*	*315*	*400*	*500*	*630*	*800*	*1k*	*1.2k*	*1.6k*
EDT [s]	0,64	2,19	0,98	1,28	2,09	1,51	0,80	1,85	2,55	2,45	0,59	2,05	0,92
T10 [s]	2,78	2,04	2,32	2,22	2,13	1,40	2,43	1,44	2,44	2,09	1,92	2,98	1,39
T20 [s]	2,67	2,31	2,42	2,25	2,08	1,77	1,42	1,77	2,06	1,98	1,83	1,72	1,71
T30 [s]	2,69	2,22	2,13	2,05	2,10	1,73	1,89	1,83	1,78	1,92	1,79	1,73	1,57
C50 [dB]	-3,55	-7,45	2,51	3,97	-2,21	-0,82	-3,84	-1,72	1,55	0,17	3,78	0,28	-0,51
C80	-2,28	-2,62	4,87	5,78	-0,65	1,36	-2,10	-0,54	2,42	1,17	5,15	1,82	0,99

[dB]													
EDTt [s]	1,98	1,78	1,42	1,88	2,11	1,37	1,15	1,83	2,66	5,17	0,59	1,43	0,88
Ct [dB]	8,78	9,00	9,70	7,76	9,50	9,19	9,85	9,46	6,80	6,08	6,67	6,90	6,29
Tt [ms]	373,5	329,40	241,44	133,92	322,13	283,4	291,3	282,3	204,9	181,7	94,0	155,0	155,1

Frec [Hz]	*2k*	*2.5k*	*3.15k*	*4k*	*5k*	*6.3k*	*8k*	*10k*	*12.5k*	*16k*
EDT [s]	1,10	1,68	1,26	0,98	1,88	1,49	0,89	1,08	0,76	0,58
T10 [s]	1,98	1,16	1,38	0,86	1,24	0,78	0,64	0,66	0,49	0,37
T20 [s]	1,57	1,53	1,56	1,30	1,32	1,19	1,05	0,78	0,68	0,44
T30 [s]	1,58	1,46	1,50	1,32	1,41	1,13	1,14	0,85	0,73	0,44
C50 [dB]	-0,14	-1,10	0,66	2,82	3,63	2,01	2,48	4,30	5,27	12,09
C80 [dB]	2,23	0,34	2,42	4,34	5,18	3,84	5,45	6,98	8,73	16,40
EDTt [s]	1,11	2,04	1,33	1,32	0,74	1,49	11,74	6,59	0,90	2,96
Ct [dB]	7,42	6,23	4,99	-0,12	1,29	0,49	-2,46	-3,04	1,86	2,71
Tt [ms]	170,3	184,9	128,3	25,8	25,7	31,4	4,4	4,2	27,0	2,0

8.2. Code

```
function yw = iirfilter(x, l, fs, plot_response)
%% iirfilter
% Funcíon para realizar un filtrado pasa banda por octava o tercio de octava.
% el filtrado responde a las reglas IEC 61260
% inputs:
% x = (double, cell_array). senal a filtrar
% fs = (int). Frecuencia de muestreo
% l = (int). ancho de banda puede ser de '1', o '1/3'. Se deben ingresar
% dichos valores
%plot_response = ((bool )mostrar grafico de respuesta al impulso si se desea para
%corroborar el funcionamiento del filtro.
% outputs:
% yw = cell array. señal filtrada
%%
        if l == 1
        Fc = [125,250,500,1000,2000,4000,8000]; %frecuencias centrales
        Nfc = length(Fc); %cantidad de bandas

        irFiltbank = cell(1,Nfc);
        for i = 1:Nfc
        f1 = Fc(i)/2^(1/2);
        f2 = Fc(i)*2^(1/2);
```

```
        irFiltbank{i} = designfilt('bandpassiir','FilterOrder',6, ...
                        'HalfPowerFrequency1',f1,'HalfPowerFrequency2',f2, ...
                        'SampleRate',fs);
        end
        yw = {zeros(length(x),Nfc)};
        for i=1:Nfc
        filt = irFiltbank{i};
        yw{:, i} = filtfilt(filt, flip(x));
        yw{:, i} = flip(yw{:, i});
        end

        else
        Fc
=[100,125,160,200,250,315,400,500,630,800,1000,1250,1600,2000,2500,3150,4000,5000,6300,8000
,10000,12500,16000];
        Nfc = length(Fc); %cantidad de bandas

        irFiltbank = cell(1,Nfc);
        for i = 1:Nfc
        l = 1/3;
        f1 = Fc(i)/2^(1/6);
        f2 = f1*2^(l);
        irFiltbank{i} = designfilt('bandpassiir','FilterOrder',8, ...
                'HalfPowerFrequency1',f1,'HalfPowerFrequency2',f2, ...
                'SampleRate',fs);
        end

        yw = {zeros(length(x),Nfc)};
        for i=1:Nfc
        filt = irFiltbank{i};
        yw{:, i} = filtfilt(filt, flip(x)); % se filtra x(-t) para evitar el ringing del filtro.
        yw{:, i} = flip(yw{:, i}); % se vuelve a invertir la señal ya filtrada y se guarda.
        end
        end

        if plot_response == true

        plotter = fvtool(irFiltbank{1}, ...
        irFiltbank{2}, ...
        irFiltbank{3}, ...
        irFiltbank{4}, ...
        irFiltbank{5}, ...
        irFiltbank{6}, ...
        irFiltbank{7}, ...
        'Fs',fs);
        set(plotter,'FrequencyScale','Log')
        end
  end

  function [Et,Etsch] = suave(ht,fm,M,filtertype)
  %% suave
```

```
%   Suavizado y recorte de una RI mediante la integral de schroeder o MMF.
%   Se calcula su limite de integracion aplicando el metodo de Lundeby a la
%   señal previamente suavizada mediante la transformada de Hilbert y un
%   filtro de media movil.
%
%   [Et,Etsch] = suave(ht,fm,M)
%
%   INPUTS:
%       ht = Respuesta al impulso
%       fm = Frecuencia de muestreo
%       M = tamaño de ventana para filtro de mediana movil
%       filtertype = booleano / true realiza proceso schroeder , false
%       aplica mediana movil
%   Outputs:
%       Et = Curva de energia(dB)
%       Etsch = Curva de Schroeder(dB)
%

        hthilb = ht/max(abs(ht));
        hthilb = abs(hilbert(hthilb));
        Ethilb = 10*log10(hthilb.^2/max(hthilb.^2));   %Suavizado de la señal para
        Etlund = medMov(Ethilb,M);
  %     %Limite de integracion
                                        %el metodo de Lundeby con filtro de mediana movil
        Et = 20*log10(abs(ht)/max(abs(ht)));       %Energia del impulso
        if filtertype == 1
        Etsch = schr(ht,Etlund,fm);                 %normalizada en dB
        else
        corte= lundeby(Etlund,fm);

        htmmf = medMov(ht,M);
        htmmfhil = abs(hilbert(htmmf));
        %Moving Average Filter
        windowSize = round(fm/9.6); %Largo optimo para fs
        moving_avg = dsp.MovingAverage(windowSize);
        hMA = moving_avg(htmmfhil);
        hMA=hMA/max(hMA);
        hdBMA=20*log10(hMA);
        index=find(hdBMA==0,1,'first');
        Etsch=hdBMA(index:corte);
        Etsch = transpose(Etsch);

        end
end
%%  Filtro de media movil
function htmm = medMov(ht,M)

        movmedWindow = dsp.MedianFilter(M);
        htmm = movmedWindow(ht);

end
```

```
%%  Integral de Schroeder
function Etsch = schr(ht,Etlund,fs)

        enc = lundeby(Etlund,fs);   %Limite de integracion
        htsch(enc:-1:1) = (cumsum(ht(enc:-1:1).^2)/...
                (sum(ht(1:length(ht))).^2));
        Etsch = 10*log10(htsch/max(abs(htsch)));

end

function [Tt] = trans_time(ir,fs,M)
%% trans_time
%   Calculo del mixing time (Tt) de una respuesta al impulso
%   ir = respuesta al impulso
%   fs = frecuencia de muestreo
%   M = tamaño de ventana
%   Tt = mixing time en ms

        inicio = find(max(abs(ir)),1);
        %fin = lundeby(10*log10(ir.^2),fs);
        ruido = mean(ir(round(.9*length(ir)):end));
        corte=length(ir)-length(ruido);
        ir = ir(inicio:corte);
        DcER = echogram(ir,M);
        Aedf = cumsum(DcER) ./ max((cumsum (DcER)))  ; % eco density
        Tt = find( Aedf >= 0.99 ,1) *1000/fs ;   % Encuentra el 99% de la energia donde se encuentra
Tt
end

%%
function [echogram] = echogram(ir,M)
%% Adaptada de la funcion echogram. Autores: A. Bidondo, L. Pepino
%%
        %Enegy decay curve computation:
        sqx = (ir.^2)/max(ir.^2);
        edc = 10*log10(sqx);
        movmedWindow = dsp.MedianFilter(M);
        envelope = movmedWindow(edc);
        echogram = edc(1:length(envelope)) + envelope; %Envelope is summed to
                                %avoid positive dB.

        echogram = 10.^(echogram./10);

        for i=2:length(echogram)-1
        if echogram(i-1)>echogram(i) && echogram(i+1)>echogram(i)
        idxref = i;
        break
        end
        end
        %Direct sound removal and initial zero energy:
        echogram = echogram(idxref:end);
        echogram(1) = 0;
```

```
        echogram(2) = 0;
end

function [EDT,T10,T20,T30,C50,C80,EDTt,Ctt] = parametros(ht,Etsch,fm,tt)
%% parametros
%   Calcula los tiempos de reverberacion segun la normativa ISO-3382
%   y los energeticos Claridad (C80) y Definicion (D50)
%   Inputs:
%       ht = Vector respuesta al impulso.
%       Etsch = Vector de valores de la señal suavizada.
%       fm = Frecuencia de muestreo.
%       tt = transittion time
%   Outputs:
%       EDT = Early Decay Time en [s]
%       T10 = Parametro T10 [s]
%       T20 = Parametro T20 [s]
%       T30 = Parametro T30 [s]
%       C80 = Claridad (C80) [dB]
%       C50 = Claridad (C50) [dB]
%       EDTt = Early Decay Time Related to Transittion-Time [s]

        [EDT,T10,T20,T30,EDTt] = TR(Etsch,fm,tt);
        [C50,C80,Ctt] = paramEnergeticos(ht(1:length(Etsch)),fm,tt);
end

%%  Tiempos de reverberacion
function [EDT,T10,T20,T30,EDTt] = TR(Etsch,fm,tt)
%%  TR
        T=(length(Etsch))/fm;
        t=0:1/fm:T-1/fm;

        % Definicion de extremos
        [~,t_0db] = min(abs(Etsch));
        [~,t_5db] = min(abs(Etsch + 5));
        [~,t_10db] = min(abs(Etsch + 10));
        [~,t_15db] = min(abs(Etsch + 15));
        [~,t_25db] = min(abs(Etsch + 25));
        [~,t_35db] = min(abs(Etsch + 35));
        t_tt = t_0db + tt;

        %EDT
        x = t(t_0db:t_10db);
        y = Etsch(t_0db:t_10db);
        [a1,~] = polyfit(x,y,1);
        EDT=(-60-a1(2))/a1(1);

        % EDTt
        x = t(t_0db:t_tt);
        y = Etsch(t_0db:t_tt);
        [a1,~] = polyfit(x,y,1);
        EDTt = (-60)/a1(1);
```

```
        % T10
        x = t(t_5db:t_15db);
        y = Etsch(t_5db:t_15db);
        [a1,~] = polyfit(x,y,1);
        T10=(-60-a1(2))/a1(1);

        % T20
        x = t(t_5db:t_25db);
        y = Etsch(t_5db:t_25db);
        [a1,~] = polyfit(x,y,1);
        T20=(-60-a1(2))/a1(1);

        % T30
        x = t(t_5db:t_35db);
        y = Etsch(t_5db:t_35db);
        [a1,~] = polyfit(x,y,1);
        T30=(-60-a1(2))/a1(1);

end

%%  Parametros energeticos
function [C50,C80,Ctt]=paramEnergeticos(ht,fm,tt)

        Et = ht.^2;
        t50 = round(0.05*fm);
        t80 = round(0.08*fm);

        C80 = 10*log10(trapz(Et(1:t80))/trapz(Et(t80:end)));   %C80
        C50 = 10*log10(trapz(Et(1:t50))/trapz(Et(t50:end)));   %C50
        Ctt = 10*log10(trapz(Et(1:tt))/trapz(Et(tt:end))); %Ctt related to transittion time

end

function [dataIACC,dataIACCt, dataTt,Tt] = parametros2(ir_1,ir_2,fs,flag,M)
% Calculo de Tt e IACCe & IACCt
%INPUT
%   ir_1, ir_2 = cell array de RI filtradas
%   fs = frecuencia de muestreo
%   flag = 0 mono / 1 stereo
%OUTPUT
%   dataIACC, dataCtr, dataTt = structs con [media, desvio] por banda
%   Tt = cell array de Tt por ir y por banda


        t80ms = round((80/1000)*fs) ;

        %% Transition time
        Tt = cell(size(ir_1)); % creo cell para guardar los datos
                               %para cada impulso
        for i = 1:size(ir_1,1)
        for n = 1:size(ir_1,2)
        ir_tt= ir_1{i,n} / max(abs(ir_1{i,n}));
```

```
Tt{i,n} = trans_time(ir_tt,fs,M);
end
end

Tt_mean = mean(cell2mat(Tt),1,'omitnan');
Tt_desvio = std(cell2mat(Tt),0,1,'omitnan');

for i = 1:size(ir_1,2)
field = strcat('banda_', num2str(i));
dataTt.(field) = vertcat(Tt_mean(i),Tt_desvio(i));
end

%% IACC
if flag == 1
iacc = cell(size(ir_1)); % creo cell para guardar los datos
        %( L cantidad de datos , ir_1 cantidad de bandas de filtro)
iacct = cell(size(ir_1)); % lo mismo para IACCt
ir_1iacct = cell(size(ir_1));
ir_2iacct = cell(size(ir_1));
for i = 1:size(ir_1,1) %cantidad de tomas
        for n = 1:size(ir_1,2) % cantidad de bandas
        ir_1iacct{i,n} = ir_1{i,n}(1:round((Tt{i,n}) * fs /1000));
        ir_2iacct{i,n} = ir_2{i,n}(1:round((Tt{i,n}) *fs/1000));
        ir_1{i,n} = ir_1{i,n}(1:t80ms);
        ir_2{i,n} = ir_2{i,n}(1:t80ms);
        cross_correlation = xcorr(ir_1{i,n},ir_2{i,n});
        cross_correlation2 = xcorr(ir_1iacct{i,n},ir_2iacct{i,n});
        IACF = cross_correlation ./ sqrt(trapz(ir_1{i,n}.^2).*trapz(ir_2{i,n}.^2)) ; % IACFe
        IACF2 = cross_correlation2 ./ sqrt(trapz(ir_1iacct{i,n}.^2).*trapz(ir_2iacct{i,n}.^2)) ; %
IACFt
        if max(abs(IACF2))>=1
        IACF2=0.99 ;
        end
        iacc{i,n} = max(abs(IACF)); % IACC
        iacct{i,n} = max(abs(IACF2)); % IACC
        end
end

iacc_mean = mean(cell2mat(iacc),1,'omitnan');
desvio = std(cell2mat(iacc),0,1,'omitnan');
iacct_mean = mean(cell2mat(iacct),1,'omitnan');
desvio_iacct = std(cell2mat(iacct),0,1,'omitnan');

for i = 1:size(ir_1,2)
field = strcat('banda_', num2str(i));
dataIACC.(field) = vertcat(iacc_mean(i),desvio(i));
dataIACCt.(field) = vertcat(iacct_mean(i),desvio_iacct(i));
end
else
for i = 1:size(ir_1,2)
field = strcat('banda_', num2str(i));
dataIACC.(field) = vertcat(0,0);
```

```
        dataIACCt.(field) =  vertcat(0,0);
        end

        end
end
```