UNIVERSITÀ DEGLI STUDI DI PADOVA

Master Course in COMPUTER ENGINEERING

Master Course Thesis

Numerical Simulation of a Tube-Delay Audio Effect

Graduate:

Simionato Riccardo
1128267

Supervisor:

prof. Rodà Antonio
UNIVERSITA' DEGLI STUDI DI PADOVA

Co-supervisors:

prof. Avanzini Federico
UNIVERSITA' DEGLI STUDI DI MILANO
prof. Välimäki Vesa
AALTO UNIVERSITY

Advisor:

PhD. stud. Liski Juho
AALTO UNIVERSITY

10 September 2018 Academic Year 2017/2018

Abstract

This thesis investigates the wave propagation in long narrow tubes and the related delay effect produced by such a medium. Several measurements were performed and then analyzed in order to model the energy losses and time delay produced by the tube. Plastic tubes, such as garden hoses, were chosen for this purpose. They were used in the analog tube-based effect analyzed and, consisting of hard walls, fit for the research objectives. The work describes the acoustic measurements conducted and the analysis of the obtained impulse responses performed in MATLAB®. The results of the analysis were used to design delay lines and digital filters, which simulate the propagation delay and losses. A study on the reflection caused by a finite-length tube is also described and the reflections from the end of the tube which may be open, closed, or acoustically attenuated were also modelled. The resulting system was found producing delay effects having a realistic low-pass filtering and was modelled using simple 1st and 2nd order IIR digital filters. A virtual tube delay effect based on the real-time simulation of acoustic wave propagation in a garden hose, including the reflection simulation, is also presented. The stereo effect plugin is implemented in PURE DATA[1] using the C++ language.

This research work was conducted between August 2017 and January 2018, when the author was visiting the Aalto Acoustics Lab within the framework of the Erasmus+ program.

[1] `http://puredata.info`

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Delay is an effect which can be experienced in any acoustical spaces [Zöl11]. A sound wave propagating in space, can be reflected by hitting a wall or an obstacle and then superimposed on the sound wave at the source. Depending on the obstacle distance, therefore, on the distance travelled by the wave, the reflection can be separated in time or can interfere with others reflections. In the first case we will hear an echo, in the second one we will notice a modification of the sound color. Additionally, in the case of parallel boundaries, repeated reflections can appear. Their distance determines the delay that is imposed to each reflected sound wave. A sound propagating in a cylinder is also reflected at both ends, whether they are open or closed. Hence, as discussed above, the distance of the boundaries determines if the cylinder will make an iterative pattern of reflections or a modification of the sound color. Long cylinders will make delayed sound separated in time, the short ones the reflections will be superimposed resulting in a pitched tone.

1.1 Motivation

The sound propagation in long narrow tubes can be exploited to design spatial effects as delay and reverberation. As it will be discussed in Ch. 2, a reverberator and a popular delay were designed using plastic tubes. Garden hoses and vacuum hoses are also used today to build a homemade analog delay and reverb[1] for guitar, drum or other instruments. Due to energy losses and reflections from the ends, these kind of audio effects produce particular manipulation on the sound timbre resulting in attractive sonic qualities. Furthermore, the reverberation and the coloration caused by a long tube has also been shown to be a robust cue for the distance perception of a sound source [FR08]. In a recent study, a digital-waveguide-mesh model of a small tubular shape has been used to simulate distance in a virtual environment [GAF16]. Finally, these two aspects motivated the research having as its aim an accurate digital imitation of waves propagation in long narrow tubes.

[1] Some examples:
Vintage Analog Delay - https://www.youtube.com/watch?v=Bj9ETTqrmys,
Pipe Delay 1 - https://www.youtube.com/watch?v=RlsZnpyZmIo,
Pipe Delay 2 - https://www.youtube.com/watch?v=rJSCQapZ-M4,
Vacuum Hose Drum Delay - https://www.youtube.com/watch?v=z83X5_2Wo1w,
Pooper Time Cube - https://www.youtube.com/watch?v=bYewZHbtrro,
Garden Hose Reverb - https://www.youtube.com/watch?v=jiPE9mYOspo&frags=pl%2Cwn

1.2 Thesis Overview

The thesis is organized as follows. In Ch. 2 the State of the Art and the related work are presented. A mathematical overview of the acoustic concepts, including the sound propagation in a cylinder, is given in Ch. 3, while Ch. 4 focus on the acoustic measurements and in particular on the impulse response measurement techniques. The performed measurements are described in Ch. 5, where their analysis is also presented. Ch. 6 describes the approach used to model the tube model and to design the digital filters, which are then compared to the measurements in order to provide an objective evaluation of the results. Finally, Ch. 7 presents and discusses the implementation of a real-time plugin in the Pure Data environment. Ch. 8 concludes the work discussing the results and an Appendix follows with Matlab® and C++ codes and other additional information.

Moreover, supplementary materials including the plugin, the external libraries for Mac OS X and Linux, the source C++ file, and some dry/wet sounds are available for download at `https://github.com/RiccardoVib/VIRTUAL_TUBE_DELAY-EFFECT-`.

Chapter 2

Related Work

Delay effect is at the basis of several audio effects, including vibrato, flanger, chorus, echo, as well as spatial effects such as reverberation. Being a popular audio effect, used to add space to many instruments or other music production contexts, different type of delay effect were proposed.

In this first chapter, the delay- and the tube-based analog spatial effects will be presented. The latter ones took advantage on the waves propagation in tubes and represent the references for the simulation proposed in this thesis. In the latter sections, the Virtual Analog modelling problem will be introduced and, finally, some delay-based virtual analog model will be discussed.

2.1 Delay Effects

Delay-based audio effects are several. Vibrato effect is a periodical pitch variation of the sound. Pitch deviation in the real life is due to the fact that the distance between the source and our ears is being varied. Varying the distance means varying the time delay. Hence, the vibrato effect can be obtained from varying periodically the time delay. Typical values are between 5 to 10 ms. Flanger, instead, is produced by mixing a delayed signal version with the original one. The signal is delayed by a small and gradually changing period using a low frequency continuous variation such as 1 Hz. Usual values for flanger are shorter than 15 ms. This effect produces peaks and notches in the frequency spectrum, providing a swept comb filter effect. Varying the time delay causes these notches to sweep up and down the frequency spectrum. Similarly, but using several delayed copies of the input signal and small and random variations in the delay times, can be obtained a chorus effect. The range in this case is 10 to 25 ms. Lastly, a delay in the range 10 to 25 ms is called slapback or doubling and the effect is a quick repetition of the input.

Table 2.1: *Typical delay-based effects.*

Delay range (ms)	Modulation	Effect name
0...20	-	Resonator
0...15	Sinusoidal	Flanging
10...25	Random	Chorus
25...50	-	Slapback
> 50	-	Echo

Table 2.1 summarizes these delay-based effects. Values greater than 50 ms create a the delayed signal heard as a distinct echo of a direct sound but the actual amount of delay time required to

create this effect depends on the nature of the audio signal being delayed. Transient, percussive signals can reveal echoes with shorter delay times, less than 30 ms. Whereas sustained, steady-state ones the values required to create an audible echo are much longer and they can be more than 50 ms.

2.2 Tube-based Analog Spatial Effects

Tube-based reverberator The first analog audio effect based on a narrow long tube was proposed in 1960 [OB60]. The authors presented a synthetic reverberator built with a tube, a loudspeaker, transducers, and a microphone delay unit in combination with a feedback system. A horn-loudspeaker coupled to a tube with three microphones located at different distances provided three different delays that, in conjunction with a positive feedback system, created time spaced components.
Reverberation consists of multiple reflections due to walls or other obstacles. Each time the sound is reflected by them it suffers a decrease in intensity and is heard by the listener with a certain delay. Authors' idea was to simulate this condition by passing the reproduced sound through a series of transducers with progressive delay and attenuation.

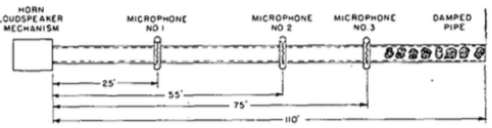

Figure 2.1: *Electroacoustic delay system consisting of loudspeaker, pipe and microphones.*

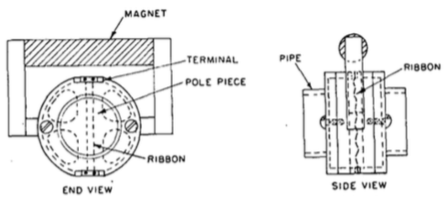

Figure 2.2: *End and side views of the microphone used in the electroacoustic delay system of the Fig. 2.1.*

Synthetic reverberator system is a electroacoustic means used to introduce increments of progressive delay and attenuation in the original signal. Figs. 2.1 and 2.2 show the delay system and the microphones of the reverberator. The delay unit is the key element of the system. It consists of a horn-loudspeaker mechanism coupled to a tube with three ribbon-type microphone units located at distances of 25, 55, and 75 ft from the loudspeaker mechanism, providing delays of 23, 50, and 69 ms. In conjunction with feedback system, that feed back the output of the microphone through the system in a positive feedback fashion, the delay unit provides a series of components with time spacings of 23, 27, 19, 23, 27, 19, and so on ms. In this way, the reverberation time can be also varied by means of the gain in the feedback loop.

The block diagram of the reverberator system is shown in Fig.2.3 and in Fig. 2.5 can be seen the synthetic reverberator without the back, showing amplifiers, loudspeaker, pipe and microphones. The outputs of the three microphones are mixed giving three components unequally spaced with respect to the time and providing a measure of randomness in the components of the reverberant sound. A feedback control potentiometer, changing the amount of feedback, controls the rate of the amplitude decay of the amplitude of the emitted sound components. The artificial reverberation is obtained mixing the direct and reverberant sound as it can be seen in Fig. 2.4. The reverberation time, defined as the time required for the sound to decay 60 dB, can be computed taking in account that the transit time of the sound through the delay system is 69 ms and using the follow formula:

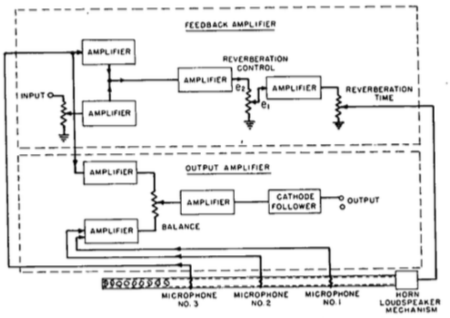

Figure 2.3: *Schematic block diagram of the synthetic reverberator.*

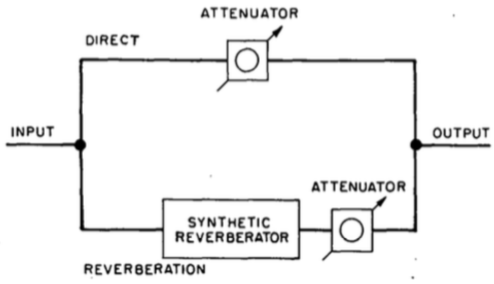

Figure 2.4: *Schematic diagram of the overall system.*

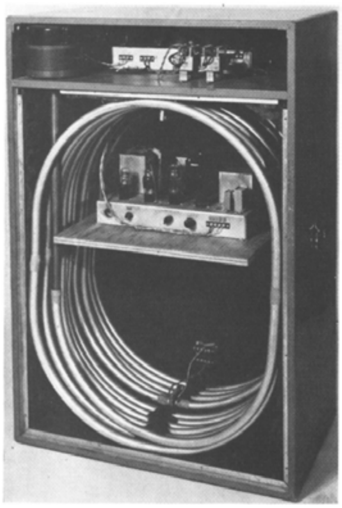

Figure 2.5: *Inside the synthetic reverberator.*

$$T = \frac{0.207}{\log_{10}(e_2/e_1)}[s], \tag{2.1}$$

where e_2 is the output voltage of the delay unit having as input e_1 with the the reverberation control disconnected. Since $20\log_{10}(e_2/e_1)$ represents the loss in gain in decibels through the delay system, the potentiometer provides for any ratio of e_1 to e_2 and hence any reverberation time.

The Cooper Time Cube In 1971, Bill Putman and Duane H. Cooper designed a tube-based mechanical delay[1]. The echo-free acoustic delay device, called Cooper Time Cube, sends audio through long coiled tubing, not unlike a garden hose. It using mic capsules as speakers and pickups, create a time delay unit. Shure mic capsules at both ends of each line was used. In addition, a series of tooled aluminium blocks tune the delay to a relatively flat response. Two coiled rigid polyethylene tubes was used creating a parallel effect and the two coils were spaced to prevent crosstalk. In addition, high frequency pre-emphasis and mid-range equalization were done. Finally, in order to maintain acoustical isolation the whole system was suspended on springs within the housing. The device was less flexible than tape-based delays or later electronic units, it had feature

[1] `https://www.uaudio.com/blog/cooper-time-cube-power/`,
`http://www.musicradar.com/reviews/tech/universal-audio-cooper-time-cube-mk-ii-207730`

limited to 14, 16 or a combined 30 ms delay. But, despite only 1,000 were ever made, it was noted for its ability to always sit perfectly in the mix with its short delay and doubling effects.

2.3 Virtual Analog

The need of replace the disappearing analog technology, characterized by unique design concepts, with virtual versions of the original systems led to the birth of research field called Virtual Analog. This term denotes the digitization of analog components and electronic circuits of old analog music system [Val+10]. Virtual analog modelling presents methodological similarities with the physical modelling of musical instruments. Where physical modeling performs a computational simulations of acoustic systems, virtual analog performs a computational simulations of electronic ones. This research area involves non-linear and time-varying systems and it contains various subtopics, such amplifiers and loudspeaker cabinets simulations, analog effects, electronic music synthesizers and acoustic musical instruments. Digital emulation of vintage electronic and electromechanical effects processors has received a lot of attention recently [Pak+11]. Analog effects devices have their own characteristic timbre and the digital implementation must imitate their response, reproducing a sufficiently similar timbre. Plate and spring reverberators, tape and bucket brigade delays are some examples of common electromechanical devices digitally implemented.

Examples of simulated analog delay system are the Leslie effect [Smi+02; KMV08; HHA09], the Echoplex Tape Delay[2] [AAS08], and the Bucket Brigade Device [RS10].

The Leslie Effect

The Leslie is a popular audio effect consisting of a rotating horn housed in a small cabinet [HHA09]. The cabinet is a wooden box and in addition to the rotating horn radiating high frequencies, contains a rotating speaker port adapted to a woofer radiating low frequencies. Each rotating source is driven by its own motor and mechanical assembly. Hence, the rotating speeds of the sources are different. The crossover frequency of this two speaker system is about 800 Hz. A diffuser is mounted at the end of the horn and approximates an omnidirectional pattern of radiation. The box is almost completely closed and contains only the vents from which the sound radiates. The rotating speed of the horn is fast enough to obtain pitch and amplitude modulations. In the woofer port instead, the main perceptual effect is the amplitude modulation because the frequency changes are not perceptible [Hen81].

The Leslie device provides a kind of chorus effect creating multiple reflections because the rotating horn and rotating speaker. The rotating source causes the changing of the direct path position. In turn, the sound intensity increase (when it points at the listener) and decrease, resulting in a amplitude modulation (AM) effect. By moving closer to the rotating speaker the modulation effect will increase. However, as the source rotates toward the listener its relative velocity increase the pitch of any tone it produces and as it rotates away the pitch is lowered. It creates a Doppler effect. The Doppler effect causes the pitch of a sound source appearing rising or falling due to motion of the source and/or listener relative to each other. The Doppler effect also create, in this way, a frequency modulation (FM) effect. Lastly, in addition to these effects, the rotation of both sources results in a particular spatial modulation effect because the multiple reflections. Since the horn rotates within a cabinet, the listener hears multiple reflections at different Doppler shifts, and giving the chorus effect [Hen81].

[2] https://www.uaudio.com/blog/echoplex-space-echo-and-delay-history/

Approaches to emulating the Leslie effect include separately modeling each arrival with an interpolated write according to the horn's varying position and a digital filter representing the horn radiation pattern [Smi+02]. In another approach [HHA09] impulse responses are tabulated as a function of horn rotation angle. As the horn rotates, a time-varying FIR filter is applied to the input, with each filter drawn from a different table entry according to the horn's evolving rotational state. Rotation rates into the audio bands were produced.

Doppler Shift Simulation

An algorithm for the Doppler effect and the Leslie simulation were proposed in [Smi+02]. The authors used interpolating and de-interpolating delay lines to simulate a rotating horn. Based on measurements from a real Leslie device, angle-dependent digital filters was calibrated, simulating the changing frequency response of the rotating horn. The computational model of Doppler shift accommodated any number of moving sound sources and moving listeners. For Leslie simulation, multiple sources correspond to the direct and reflected signals from the rotating horn, and two fixed listeners correspond to two ears or two studio microphones. Measurements were made in order to calibrate the angle-dependent filters corresponding to each propagation path geometry from horn to listener.

The mathematics formulation of the Doppler shift is given by

$$\omega_l = \omega_s \frac{1 + \frac{v_{ls}}{c}}{1 - \frac{v_{sl}}{c}} \tag{2.2}$$

where ω_s is the radian frequency emitted by the source, ω_l is the frequency received by the listener, v_{ls} and v_{sl} denote the speed of the listener relative to the propagation medium in the direction of, respectively, the source and the listener. The term c indicates the sound speed. This formula represents the apparent change in acoustic frequency content of a sound source due to motion of the source relative to the listener.
As discussed above, frequency shift can be obtained by time-varying delay line. In fact, time-varying delay is often used for vibrato and chorus effects. Considering the magnetic tape as the delay line, the tape read-head as the read-pointer of the delay line and the write-head as the delay-line write-pointer, the modulation of the delay by changing the read-pointer increment from 1 to $1 + \frac{v_{ls}}{c}$ corresponds to listener motion away from the source at speed v_{ls}. It also follows that changing the write-pointer increment from 1 to $1 + \frac{v_{sl}}{c}$ corresponds source motion toward the listener at speed v_{sl}. With this aim interpolating writes into the delay memory (also called de-interpolation [Väl95]) were used. If $x(t)$ denotes the input to a time-varying delay, the output can be written as

$$y(t) = x(t - Dt), \tag{2.3}$$

where D_t is the time-varying delay in seconds. If D_t is not an integer multiple of the sampling interval, $x(t-D_t)$ may be approximated using fractional delay [Roc00]. Writing $e^{\jmath\omega_s t}$ as the complex sinusoid input at frequency ω_s, the frequency shift caused by a time-varying delay can be analyzed and the output will be

$$y(t) = x(t - D_t) = e^{\jmath\omega_s(t-D_t)} \tag{2.4}$$

and its instantaneous phase

$$\theta(t) = \angle y(t) = \omega_s(t - D_t), \tag{2.5}$$

which can be differentiated to give the instantaneous frequency $\omega_l = \omega_s(1 - \dot{D}_t)$, where ω_l denotes the output frequency and $\dot{D}_t$ the time derivative D_t. Hence, the delay growth-rate, $\dot{D}_t$, equals the relative frequency downshift,

$$\dot{D}_t = \frac{\omega_s - \omega_l}{\omega_s} = -\frac{v_{ls}}{c}. \tag{2.6}$$

The time-varying delay simulates Doppler shift caused by a moving listener. In this way, a Doppler effect simulation using fractional delay lines can be done.

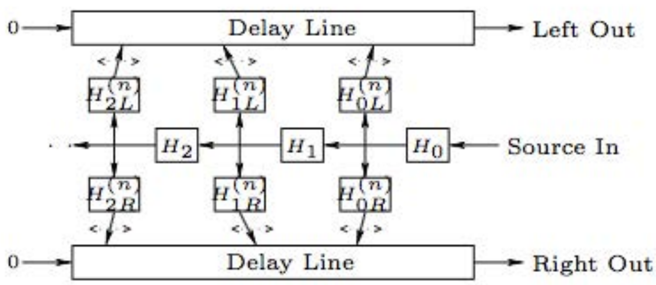

Figure 2.6: *Block diagram of the Doppler effect stereo simulator.*

Fig. 2.6 shows the schematic diagram of the Doppler effect stereo multiple-source simulation. $H_0(z)$ provides time-invariant filtering common to all propagation paths. $H_{0L}^{(n)}(z)$ and $H_{0R}^{(n)}(z)$ are the left and right channel filters implementing the time-varying characteristics of the shortest time-varying propagation path from the source to each listener. These filter outputs sum into the delay lines at arbitrary time-varying locations using interpolating writes and correspond to the "direct signal" from the moving source. These filters, in the Leslie simulation, will incorporate modulation of losses due to the changing propagation distance from the moving source to each listener and the dynamic equalization corresponding to the changing radiation strength in different directions from the moving source toward each listener. $H_1(z)$, $H_{1L}^{(n)}(z)$ and $H_{1R}^{(n)}(z)$ correspond to the next-to-shortest acoustic propagation path, the "first reflection" due to a wall close to the source. $H_1(z)$ is a fixed component, $H_{1L}^{(n)}(z)$ and $H_{1R}^{(n)}(z)$ are, instead, the time-varying ones.

Leslie Simulation

After deriving a theoretical model of the predicted Doppler shift, the Leslie simulation was performed, focusing on the rotating horn. The Leslie rotating horn can be seen in Fig. 2.7.
For a circularly rotating horn, the source position was approximated as

$$\vec{x}_s(t) = \begin{bmatrix} r_s \cos(\omega_m t) \\ r_s \sin(\omega_m t) \end{bmatrix} \tag{2.7}$$

where r_s is the circular radius and ω_m is the angular velocity. This expression approximates the horn as an omnidirectional radiator located at the same radius for all frequencies. The diffuser in the Leslie is inserted into the end of the horn in order to make the radiation pattern closer to uniform [Hen81]. Leaving aside the mathematical derivations (that can be found in [Smi+02]), the approximation valid for the simulation of the Leslie effect is given by:

$$\omega_l = \frac{\omega_s}{r_s \omega_s \sin(\omega_m t) c} \approx \omega_s \Big[1 - \frac{r_s \omega_m}{c} \sin(\omega_m t)\Big]. \tag{2.8}$$

It shows that, in the far field, a rotating horn causes an approximately sinusoidal multiplicative frequency shift with the amplitude given by horn length r_s times horn angular velocity ω_m divided

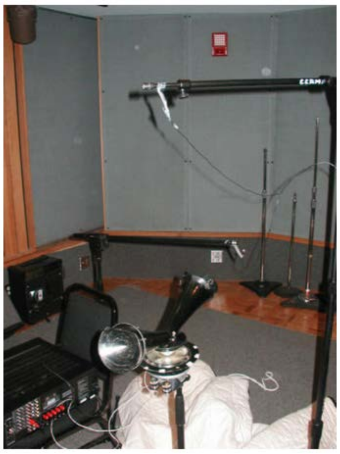

Figure 2.7: *Leslie rotating horn.*

by sound speed c. The free-field radiation pattern of the rotating horn was also measured. A matched pair of microphone elements (Crystal River Snapshot system) were used, both in the plane of rotation and along the axis of rotation because no variation in the Doppler effect or radiation was expected. The horn was set manually to fixed angles from -180 to 180 degrees in increments of 15 degrees. The impulse response was measured at each angle.

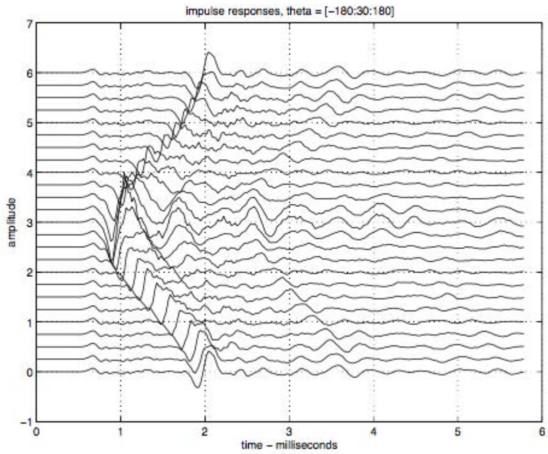

Figure 2.8: *Measured impulse responses of the Leslie rotating-horn at multiples of* 15 *degrees.*

Figure. 2.8 shows the measured impulse responses. Then the power responses were computed and in Fig. 2.9 the average power response of the horn outputs can be seen. Also overlaid in that figure is the average response smoothed, used as $H_0(z)$ in Fig. 2.6. Since Fig. 2.8 indicated the existence of fixed and angle-dependent components in the measured impulse responses, strongly suppressed by baffling in the cabinet enclosure, they were eliminated using an iterative algorithm.

The filters $H_{0L}(z)$ and $H_{0R}(z)$ in Fig. 2.6 were obtained by dividing the smoothed frequency response at each angle by $H_0(z)$ and designing a low-order recursive filter in order to provide equalization dynamically as a function of horn angle. The impulse response arrival times determines where in the delay lines the filter outputs is summed.

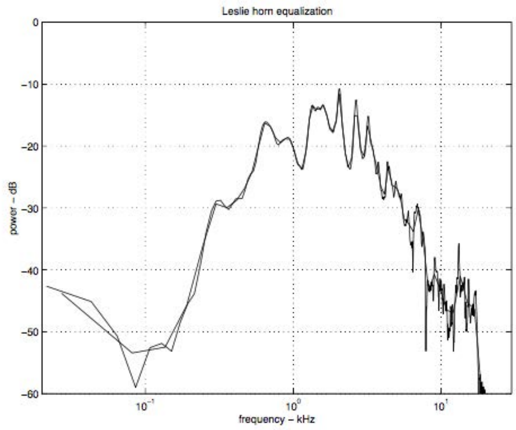

Figure 2.9: *Average angle-dependent amplitude response. The smoothed one used as a fixed equalization applied to the source is also overlaid.*

Leslie Cabinet Simulation

Smith et al., concluded that, without the box (in free field conditions) and far from the rotating source, the Doppler frequency shift and the amplitude modulation are likely to be almost sinusoidal [Smi+02]. They also suggested the inclusion of the reflections, occurring inside the wooden cabinet, in the model. Thus, the authors in [KMV08] continued the Leslie modelling analyzing the cabinet. With this goal, it was placed the cabinet in an anechoic room and driven by a sinusoidal generator. The acoustic pressure was measured using a microphone, at the same height from the floor as the rotating plane of the horns.

Figure 2.10: *The Leslie cabinet used for the measurements: (left) open and (right) closed.*

The cabinet effect was implemented using a real-time moving source model tha simulates the motion of an acoustic source. It processes the input signal corresponding to the acoustic radiation emitted by a fixed source. As can be seen in Fig. 2.11, the model consist of four main components: timbre variation, Doppler shift, intensity variation and reverberation effect. The parameters of the model depend on the relative speed and distance between the listener and the moving source.
A high-shelving second-order IIR filter was used to simulate timbre variations due to the air absorption that mainly affect the high-frequency components. Since the cut-off frequency was found depending weakly on the distance, it was set to 10 kHz. The gain G was related to the distance x in meters by the relation: $G(dB) = -0.5x$ m. The Doppler frequency shift was modelled using the following formula:

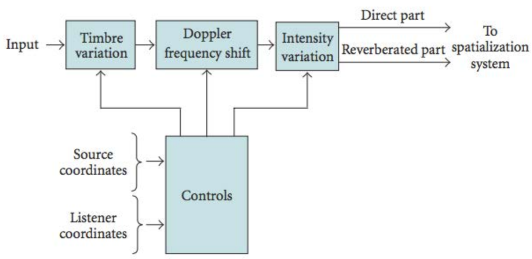

Figure 2.11: *Scheme of the moving source model.*

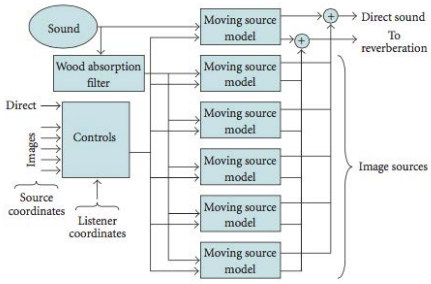

Figure 2.12: *Scheme of the Leslie horn simulator with* 5*-image sources.*

$$\tau(t) = ||L(t) - S(t)||\frac{1}{c}, \tag{2.9}$$

where $L(t)$ and $S(t)$ are the respective positions of the listener and the source at time t. The delay line will be a fractional values of τ. Intensity variations are controlled by the level of the sound. Assuming spherical waves for the sound propagation, the sound level was set varying with respect to $1/x$, where x is the source-to-listener distance. The reverberation was splitted in global and local components. The global reverberation originates from the whole space, whereas the local reverberation originates from the direction of the source. The global reverberation level was defined as $1/(x\sqrt{x})$, and the local on is given by $(1/(\sqrt{x}))(1 - (1/x))$.
From the geometry of the cabinet, the coordinates of the image sources were extrapolated, computing the coordinates of the directly radiating source and those of the reflecting planes and estimating one image source for each reflecting plane as the minimum number to obtain satisfactory perceptual results. The scheme of the simulator can be seen in Fig. 2.12. The output is composed of the sum of the direct sound source and the five image sources and each source is processed using the moving source model. In addition, the input signal is filtered with a FIR digital filter based on the frequency-dependent wood absorption. The same procedure was used for the woofer simulator.

Echoplex Tape Delay

Tape delays were delay effects based on analog tape recording. They used magnetic tape as their recording and playback medium. Electric motors guided a tape loop through a device with mechanisms allowing modification of the effect's parameters. Their signal flow includes a delay and feedback. The feedback set to a value greater than one causes oscillations in the unit, amplifying the input or noise in the system. The feedback loop electronics includes a saturating non-linearity. The sonic character of these devices arises from the tape transport mechanism, which produces quasiperiodic and stochastic component. Tape delay sends audio signal to a tape deck to capture the sound. The delay time is created by the distance of the record head to the playback head. Hence, the delay is a function of the tape speed multiplied by the distance between the two heads. These devices were quickly adopted as an alternative to expensive reverb chambers and plates. The Echoplex was the most famous. It gave the user variable control over the distance of the record. A sliding lever control, labelled Echo Delay, moves the play head up and down a steel track. It allowed a fine control of echo timing and the tape speed, moving at roughly 8 ips, allows large frequency shifts. Moving the record head faster than the tape speed resulted in a "sonic boom". The Echoplex is shown in Fig. 2.13 where can be seen the fixed playback and erase heads, the movable record head and the tape loop. Playback and erase heads are fixed at each side of the Delay Handle travel. The record head writes to the moving tape, which has been newly erased by the erase head. The playback head reads what has been written to the tape, and the delayed signal is played back. The other important parameter is the Echo Sustain/Repeats, which controls the number of repeats.

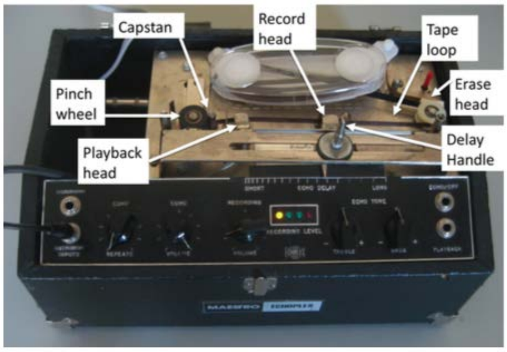

Figure 2.13: *Echoplex EP-4 tape delay unit.*

A simulation using a circular buffer and pointers moving along it was proposed. The Echoplex tape delay was modelled with read, write and erase pointers moving along the buffer. An interpolated write using a time-varying FIR anti-aliasing filter was used to prevent aliasing of this infinite-bandwidth event [AAS08].

Echoplex Simulation

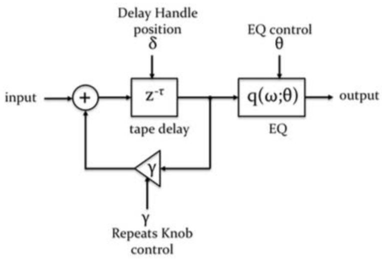

Figure 2.14: *Echoplex signal flow architecture.*

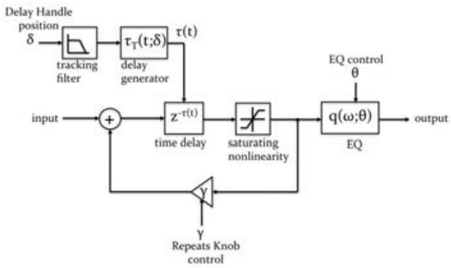

Figure 2.15: *Echoplex model signal flow architecture.*

As seen in Fig. 2.14, the input signal is delayed and fed back to the input. The amount of feedback is controlled by the Echo Repeats Knob and the maximum feedback gain of two. Saturation in the tape and circuitry limits the output level when the feedback gain is greater than one. In this case, tape hiss and 60 Hz harmonics cause the Echoplex to self oscillate. Doppler-shifted sounds can be produced by moving the Delay Handle during self oscillation. Applying a pulse train the time behaviour of the tape delay was measured, capturing also the fluctuating time delay by moving the Delay Handle between the extremes with increments of 0.5 cm.
Fig. 2.16 shows the measured mean time delay, Fig. 2.17 shows its spectrogram, where can be seen periodic components due to the capstan and pinch wheel, evident as is a low-frequency drift. The two vertical lines marked with P, close to 2.5 Hz and 5.0 Hz, correspond to irregularities in the pinch wheel rotation. Similarly, the line marked with C, close to 26 Hz, corresponds to the capstan rotation rate. The drift in the observed time delay appears as a low-pass process in the spectrum. In addition the delay spectra, exhibited a comb filter structure, spectral nulls occur at frequencies

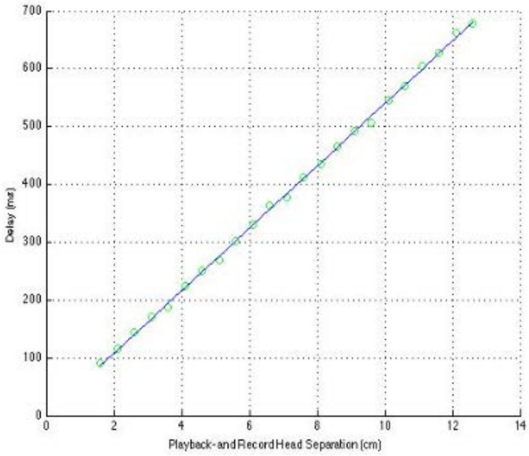

Figure 2.16: *Measured mean time delay as a function of distance between record and playback heads (dotted line) and the least squares fit (dashed line).*

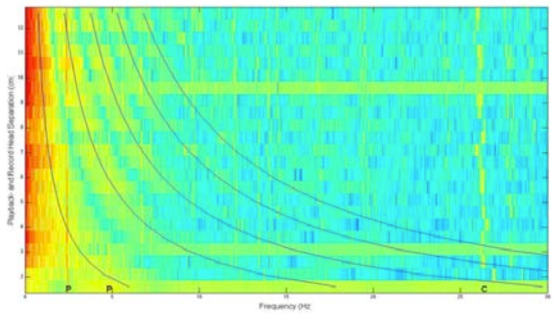

Figure 2.17: *Measured Echoplex Delay Spectrogram over the range of Delay Handle positions.*

proportional to odd integer multiples of the inverse of the distance between the record head and playback one.

The Echoplex model signal flow architecture is shown in Fig. 2.15. The input is applied to a delay line with a time varying delay controlled by the Delay Handle position. The delay line output is applied to a saturating non-linearity and fed back to the input with a gain set by the Repeats Knob. Finally, the output of the non-linearity is equalized to form the Echoplex model output.

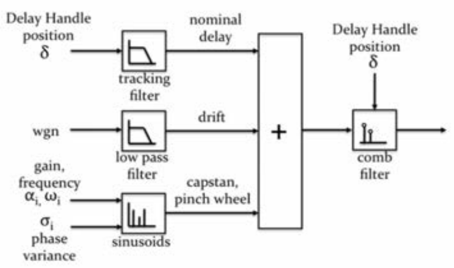

Figure 2.18: *Delay generation signal flow architecture.*

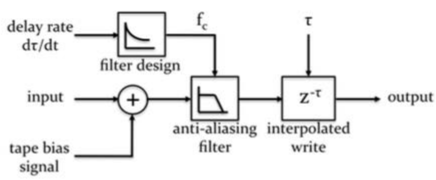

Figure 2.19: *Time delay implementation.*

The fluctuating time delay driving the tape delay model is formed by adding the mean delay, determined by the Delay Handle position, and stochastic processes representing the observed capstan, pinch wheel and low-frequency drift components of the delay. As can be seen in Fig. 2.15, this sum is filtered using a comb filter, also controlled by the Delay Handle position. The pinch wheel and capstan components were generated summing sinusoids having amplitudes equal to those measured. An additive low-pass and zero-mean noise process was introduced according the observed frequency fluctuations. The low-frequency drift component was formed by filtering white Gaussian noise. Since the drift component results from fluctuations in tape speed, its variance was made to be proportional to the distance between the record head and the playback head. Finally, to implement the time varying delay, a circular buffer was used. In order to have the loop splice and other imperfections occurring at the proper intervals, its length was set the same as the modelled tape loop. The read pointer, representing the playback head, moves along the buffer at the rate of one sample per sampling period. The write one, representing the record head, leads the read pointer by the computed tape delay and perform an interpolated write. When the write pointer is moving forward through the circular buffer, the interpolated signal is added into the circular buffer and when the write pointer is moving backward through the circular buffer, the signal written replaces that in the delay line. To avoid aliasing, appearing when the write pointer is advancing through the circular buffer slower than one sample per sampling period, the bandwidth of the interpolated filter used by the interpolated write was adjusted according to the instantaneous tape speed.

Bucket-Brigade Device

The bucket-brigade device (BBD), invented in 1968 [ST69], realizes a time delay using an analog circuit. Hence, it is a discrete-time analogue delay line consisting of a series of capacitors sections which carry the analog signal at a rate determined by the external clock. The input signal is sampled in time and passed into a series of capacitors and MOS transistor switches, giving an output delayed and discrete in time signal. As the charge representing the input signal is transferred from one capacitor to the next, a small amount bleeds to adjacent capacitors, and the output acquires a low-pass characteristic. In addition, while the charge is propagating through the delay line, it decays to the substrate and louder signals are distorted. BBDs were typically found in circuits implementing echo, chorus, vibrato, and flanging effects.

A physical model of the device is presented in [RS10]. Later circuit analysis and measurements, the device was modeled with low-order digital infinite impulse-response (IIR) filters based on the resistance and capacitance values of the filters.

Bucket-Brigade Simulation

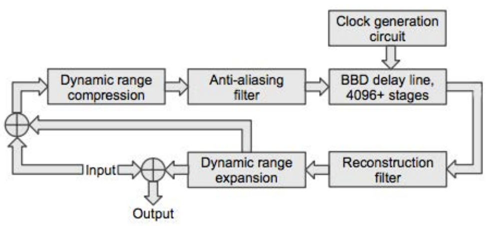

Figure 2.20: *Topology of a BBD-based echo circuit.*

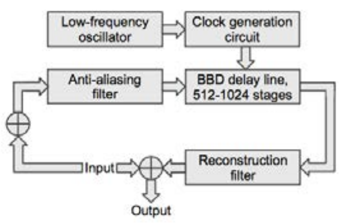

Figure 2.21: *Topology of a BBD-based chorus or flanger circuit.*

Figs. 2.20 and 2.21 show the topology of two BBD-based circuits, the echo and the chorus/flanger effect. The BBDs present non-ideal characteristics, such as transfer inefficiencies, non-linearities and noise. In order to reduce distortion, and noise, they are typically accompanied by low-pass filters and compander circuitry. The preceding and following low-pass filters avoid aliasing. In case of short delay times, as can be found in chorus, flanger, vibrato and reverb circuits, the number of charge-passing stages in the BBD circuit is significantly less reducing these undesirable effects. For this reason, in the latter case, companders are not typically used and the low-pass filters normally have minimal effect in the audio range.

For any BBD, the total time delay is given by

$$D = \frac{N}{2f_{cp}}[s], \tag{2.10}$$

where N is the number of stages in the BBD and f_{cp} is the circuit's clock frequency.

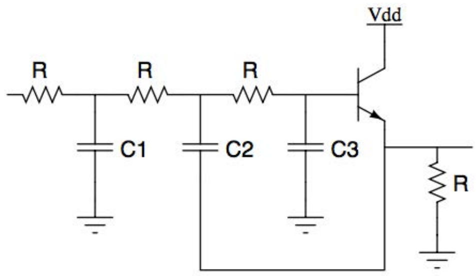

Figure 2.22: *Typical third-order Sallen-Key anti-aliasing filter used in a BBD circuit.*

To obtain a input signal appropriately bandlimited, BBD-based circuits typically use Sallen-Key low-pass filters. A typical Sallen-Key low-pass filter circuit used can be seen in Fig. 2.22. A common implementation is a a third-order filter for anti-aliasing and a third-order filter followed by a second-order (used for a "corner correction") filter for reconstruction. Since the non-linear elements are minimal and do not have drastic effects on the frequency response, the effect of the BBD was ignored and all the filters were treated as in series. In this way, the transfer function was computed as the product of the transfer functions of each filter, resulting in an eighth-order low-pass filter. This simplification method agreed with the measured results, shown in Fig. 2.23.

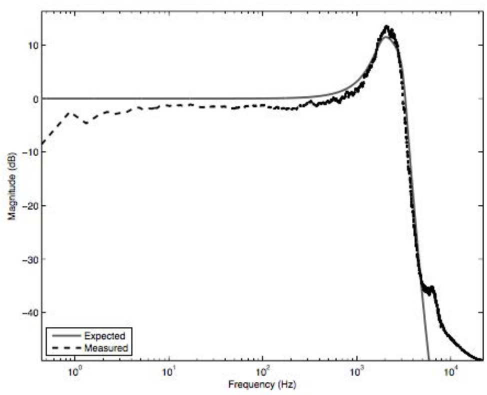

Figure 2.23: *Measured and expected amplitude response curves for the anti-aliasing and reconstruction filters series.*

Using the equation-error method implemented by the `invfreqz()` function included in Matlab's Signal Processing Toolbox [Mat98], an accurate low-order digital IIR filter was found. The filters in any BBD circuit can be, therefore, digitally modeled at low computational cost and based only on the resistance and capacitance values in the circuit.

Companding is used when a signal is sent through a channel with limited dynamic range. A compander circuit consists of a compressor and an expander. The compressor is used before the BBD to lower the dynamic range of the incoming signal, the expander is used in output to retain the original signal's dynamic characteristics. In order to obtain minimal total harmonic distortion with a maximal signal to noise ratio, the compander is typically send a signal with a near-maximum level through the BBD. Its integrated circuit consists of a pair of variable gain amplifiers and signal level averagers. This circuit was modelled determining the gain of a system by using the average signal level. With an output gain directly proportional to the average input level, the system act as an expander, while a the gain inversely proportional to average output signal level will compress the dynamic range. Following this rules, the feedforward expander was modelled by

$$f(x) = avg(|x|)x \tag{2.11}$$

while the feedback compressor by

$$f(x) = \frac{x}{avg(|f(x)|)}, \tag{2.12}$$

where the input x for each equation depends on the architecture of the BBD system.

In case of a system with a large number of stages, non-linearity becomes significant, about 1% of harmonic distortion occurs for every 1024 stages. However, this non-linearity does not vary significantly depending on the signal level, meaning it's not clipping distortion. To characterize the distortion, the spectrum at the output of the reconstruction filters for a pure sine-wave input at various frequencies and amplitudes was measured. The aliasing present before the reconstruction filters resulted in unreliable data and the transistor-based Sallen-Key filters was assumed linear for the low amplitude signals output. As a result, any additional harmonic content present was assumed to be due to the BBD.

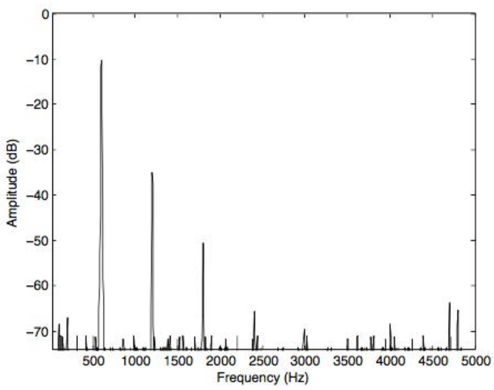

Figure 2.24: *Measured output spectrum of a BBD using a sine wave as input.*

The measured output spectrum is shown in Fig. 2.24, where can be seen the added harmonic components falling off linearly in magnitude, with the fourth and higher harmonics at almost imperceptibly low magnitude. This characteristic holds across all frequencies included in the passbands of the low-pass filtering of the BBD system. This added harmonic components were found decreasing linearly in magnitude.

$$f(x) = \begin{cases} 1 - a - b & \text{for } x > 1 \\ x - ax^2 - bx^3 + a & \text{for } -1 < x < 1 \\ -1 - a + b & \text{for } x < -1 \end{cases} \tag{2.13}$$

To add this feature to the model a third-order polynomial non-linearity, given by (2.13),was used. According the measured BBD output spectrum for a pure sine wave input, a and b parameters were set. The resulting spectrum matched the measured one for high amplitudes but underestimates the level of added harmonics as the amplitude decreases. However, the inaccuracies at low amplitudes were considered less relevant by the presence of the simulated compression and expansion which ensures the input to the non-linearity large enough. This causes the error in the harmonics' amplitudes to be smaller in the modelled system. Finally, the aliasing due to the discrete-time sampling of the input signal, just ideally prevented by anti-aliasing and reconstruction low-pass filters, the frequency-dependent insertion gain due to the constant transfer of charge between the capacitors, resulting in additional unintentional imperfections and the noise not completely removed by the compander, were neglected.

Chapter 3

Sound Propagation in Tubes

Sound wave is the pressure fluctuations above and below the average one that arrive at the ear. The sensation of sound is produced by these pressure variations detected by their mechanical effect on the tympana (ear drums) of the auditory system. The condensations and rarefractions increase and decrease in the number of molecules collisions per second, and in turn this phenomenon causes a change of force on the surface. The fluctuations above and below the ambient pressure are called acoustic pressure and it will be denoted with p_a. The air molecules, mostly diatomic nitrogen and oxygen, collide with each other and lead pressure fluctuations to neighboring molecules. These fluctuations, leading by the collections of molecules, create the waves propagation and the consequently sound perception.

This chapter offers the acoustic background for the understanding of these pressure fluctuations in the narrow tube problem. Sound propagates in a nearly spherical fashion but in a acoustic field far enough from the source, the waves reach can be approximated as progressive plane waves [RF04]. It can be assumed in the case of long tube, in particular with tube length greater then the propagated wavelength. Starting from a discussion on the plane wave and their propagation in air, it will cover the acoustic waves problem in tubes, being of interests for the next chapters.

3.1 Plane Waves in Tubes

Being the acoustic pressure the local deviation from the ambient pressure, taking p the total pressure of the air and the definition of the bulk modulus (measure of how resistant to compressibility is a substance), $K = -V\frac{dP}{dV}$, the sound pressure is given by

$$p_a = dp = -K\frac{dV}{V} = -K\frac{\partial \xi}{\partial x}, \tag{3.1}$$

where ξ is the measure of the air displacement during passage of a sound wave, V the volume of the mass of air and x the direction of the propagation. Fig. 3.1 shows the displacement of the mass of air within the $ABCD$ boundary.

Hence, sound is a vibration generated by the pressure variation through a transmission medium such as a gas, liquid or solid and it is characterized by mechanical energy propagation due to quick compressions and expansions in the elastic mean. The particles of the medium through which the sound moves vibrate in a back and forth motion. This motion has a given frequency, defined as the number of complete back-and-forth vibrations of a particle of the medium per unit of time. In the human case, motions from about 20 Hz to about 20 kHz can be heard. To be noted that

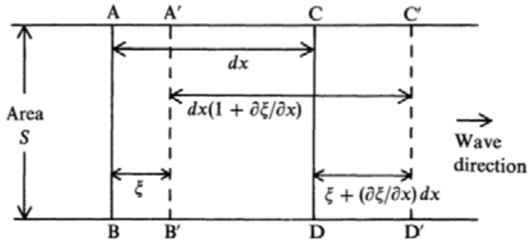

Figure 3.1: *The plane wave of displacement* ξ*.*

air molecules usually don't travel directly along the path of the sound wave. In general, waves can propagate in any medium having mass and elasticity, in a longitudinal and/or transversal fashion. Longitudinal waves have deviations from the equilibrium pressure, causing local regions of compression and rarefaction. In solid materials, however, which have both shear and compressive elasticity, the sound can also propagate as transverse waves, alternating shear stress at right angle to the direction of propagation. Fluids, and in particular gases such as air, have no elastic resistance to shear but a viscous resistance, and, therefore, only longitudinal waves can propagate in them. The local motion of the air is in the same direction as the propagation direction of the wave itself.

As stated, sound waves generated by a small source are spread out in all directions in a spherical fashion. The equation that describe this phenomenon is

$$\frac{\partial^2 p_a}{\partial t^2} = c^2 \nabla^2 p_a, \tag{3.2}$$

where ∇^2 is the Laplace operator and c the speed of sound wave. However, a small section of wave at large distance from the point source can be treated as planes normal to the direction of propagation. The acoustic field of these plane waves only depends on the spatial coordinate, x, in the direction of propagation. In the case of hard-walled tubes, plane waves are also the waves propagating at a frequency lower than a critical value f_t called, cut-off frequency. Although, the exact value depends on the shape of the tube cross section, this frequency is of order of $c/2d$ where d is the diameter tube. If friction can be neglected below the cut-off frequency, the propagation in a tube consists only of plane waves. The condition for the frictionless approximation is valid considering a lower bound for the frequency. At high frequencies, the effect of viscosity is confined to boundary layers of thickness

$$\delta_A = (2\nu/\omega)^{1/2} \tag{3.3}$$

where $\nu = \eta/\rho$ is the kinematic viscosity of the fluid near the walls (η is the dynamic viscosity and ρ the fluid density) and $\omega = 2\pi f$ the angular frequency. In order to have a good plane wave approximation thin viscous boundary layers must be considered, thus

$$\delta_A/d \ll 1 \tag{3.4}$$

Finally, plane wave approximation is valid in a tube with the frequency range within

$$\frac{2\nu}{\pi d^2} \ll f < c/2d. \tag{3.5}$$

where, for air, $\nu = 1.510 - 5\ \mathrm{m}^2/\mathrm{s}$. Hence a plane wave approximation in air is valid for a tube with a diameter $d = O(10^{-2}\ m)$.

Since in acoustic wavelengths no appreciable conduction takes place [RF04], for the case of propagation in air the elastic behaviour is taken adiabatic. Although, for sound waves in tubes or close to solid objects the behaviour becomes isothermal at very low frequencies, for examples below about 0.1 Hz for a 20 mm-tube, it can be neglected. The adiabatic relation is described by

$$pV^{\gamma} = constant \tag{3.6}$$

where $\gamma = C_p/C_v = 1.4$ is the ratio of the specific heats of air at constant pressure and at constant volume and p as before, is the average atmospheric pressure. Under this condition c has the follow relations

$$\frac{K}{\rho} = \frac{\gamma p}{\rho} = c^2, \tag{3.7}$$

where ρ denotes the density of the air, or in general of the medium. Sound speed doesn't depend on atmospheric pressure but on temperature and for air at temperature ΔT degrees Celsius and 50% relative humidity is

$$c \approx 332(1 + 0.00166\Delta T) \quad [m/s]. \tag{3.8}$$

The particle velocity or acoustic fluid velocity, given by

$$u(x) = \frac{\partial \xi}{\partial t} = \frac{p_a}{\rho c}, \tag{3.9}$$

is the speed with which the small cell of molecules moves due to the pressure and in a plane wave it is in phase with the acoustic pressure. For further consideration is also assumed one dimensional. Considering walls rigid enough, transmission of sound through them can also be neglected. Although, this assumption excludes any prediction of environmental noise induced by tube flows. The approximation is also limited to tubes with uniform cross sections A, it means to tubes with slowly varying cross sections and therefore, where

$$dA/dx \ll \sqrt{A} \ll \lambda. \tag{3.10}$$

3.2 Infinite-Length Tubes

An infinitely long or semi-infinite tube will be considered, which is so long that reflections, the accumulation of acoustic energy and phenomena like resonance can be excluded. Considering the axis parallel to the direction of propagation of tube and rigid, perfectly smooth and thermally insulating walls. Pressure, force per unit area on a surface, is given as a function of this distance and time along the tube axis and is taken to be constant on the cross section of the tube. The wave propagating in the x direction can be described by

$$p(x,t) = pe^{j(-kx+wt)}. \tag{3.11}$$

The volume flow, that will be denoted by $U(x,t)$, is the velocity attained by the little cells times the cross-sectional area S of the tube. For a tube where the diameter is small compared to a wavelength, the velocity $u(x,t)$ is essentially uniform across the tube. For this reason, the volume velocity can be defined by $U(x,t) = u(x,t)S$ and given by

$$U(x,t) = \Big(\frac{Sp}{\rho c}\Big)e^{j(-kx+wt)} \tag{3.12}$$

The specific acoustical impedance is

$$Z = \frac{p(x,t)}{u(x,t)}, \quad [Pas^{-1}m] \quad or \quad [kgm^{-2}s^{-1}], \tag{3.13}$$

and, therefore, the acoustic impedance of the tube is defined by

$$Z = \frac{p(x,t)}{U(x,t)} = \frac{\rho c}{S}. \tag{3.14}$$

Thus, the impedance of a tube is inversely proportional to the area of the tube. The specific acoustical impedance is determined at a single point, and determines the impedance mismatch on a sudden change of tube diameter. It is convenient to have a single lumped impedance for tube of given diameter. In the case of no surfaces, the specific acoustical impedance is an intrinsic property of the medium, expressing by

$$Z = \rho c. \tag{3.15}$$

For air at temperature $\Delta T°$C and standard pressure $\rho c \approx 428(1 - 0.0017\Delta T) \quad [kgm^{-2}s^{-1}]$.

Neglecting friction, for a one dimensional flow the conservation laws of mass is given by

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = \frac{\partial \rho\beta}{\partial t} \tag{3.16}$$

and the conservation laws of momentum by

$$\rho\Big(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x}\Big) + \frac{\partial p}{\partial x} = f_x. \tag{3.17}$$

The term $\rho\beta$ corresponds to an external mass injection in the flow and f_x is an external force per unit volume. Being the cause of the perturbation, $\partial\beta/\partial t$ and f_x, must therefore by definition be small. Assuming now the field consisting of a uniform state, ρ_0, p_0, u_0, plus a perturbation, ρ_a, p_a, u_a, small enough, linearization

$$\begin{cases} \rho = \rho_0 + \rho_a \\ p = p_0 + p_a \\ u = u_0 + u_a \end{cases} \tag{3.18}$$

can be allowed. Substituting and neglecting second and higher order terms, the linearized equations become, for Eq. (3.16)

$$\frac{\partial \rho_a}{\partial t} + u_0\frac{\partial \rho_a}{\partial x} + \rho_0\frac{\partial u_a}{\partial x} = \rho_0\frac{\partial \beta}{\partial t} \tag{3.19}$$

and for Eq. (3.17)

$$\rho_0\Big(\frac{\partial u_a}{\partial t} + u_0\rho_0\frac{\partial u_a}{\partial x}\Big) + \frac{\partial p_a}{\partial x} = f_x. \tag{3.20}$$

By assuming a homentropic flow (a flow with uniform and constant entropy) and using the constitutive equation $p_a = c^2\rho_a$, the term ρ_a is replaced. A one-dimensional wave equation is, then, obtained by subtracting the divergence of the momentum conservation law (3.20) from the convected time derivative $(\partial t + u_0\partial x)$ of mass conservation law (3.19), and it is

$$\Big(\frac{\partial}{\partial t} + u_0\frac{\partial}{\partial x}\Big)^2 p_a - c^2\frac{\partial^2 p_a}{\partial x^2} = c^2\Big(\rho_0\frac{\partial^2 \beta}{\partial t^2} - \frac{\partial f_x}{\partial x}\Big). \tag{3.21}$$

In the absence of source terms (the homogeneous problem), the linear perturbation p_a is given as the sum of two waves F^+ and F^- travelling in opposite directions. Hence, p_a can be expressed by

$$p_a = F^+(x - (c + u_0)t) + F^-(x + (c - u_0)t) \tag{3.22}$$

and consequently u_a by

$$u_a = \frac{1}{\rho_0 c}\Big(F^+(x-(c+u_0)t) - F^-(x+(c-u_0)t)\Big). \tag{3.23}$$

The functions F^+ and F^- are determined by the initial and boundary conditions. For a steady harmonic perturbation equation, Eqs. (3.22), the solution of Eq. (3.2) can be expressed in the form

$$p_a = p^+ e^{-\jmath kx} e^{\jmath\omega t} + p^- e^{\jmath kx} e^{\jmath\omega t} = Ae^{-\jmath kx} e^{\jmath\omega t} + Be^{\jmath kx} e^{\jmath\omega t} \tag{3.24}$$

and (3.23)

$$u_a = \frac{1}{\rho_0 c}(p^+ e^{-\jmath kx} e^{\jmath\omega t} + p^- e^{\jmath kx} e^{\jmath\omega t}) = \frac{1}{\rho_0 c}(Ae^{-\jmath kx} e^{\jmath\omega t} + Be^{\jmath kx} e^{\jmath\omega t}) \tag{3.25}$$

where $k = \omega/c$ is the wave number, A and B represent the amplitudes of the waves travelling to the right and the left, respectively, which are functions of ω.

Using now the cylindrical polar coordinates (r, ϕ, x) and taking a is the radius of the tube, the wave equation (3.2) becomes

$$\frac{1}{r}\frac{\partial}{\partial r}\Big(r\frac{\partial p}{\partial r}\Big) + \frac{1}{r^2}\frac{\partial^2 p}{\partial \pi^2} + \frac{\partial^2 p}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}, \tag{3.26}$$

with the solutions of the form

$$P_{mn}(r,\phi,x) = p_{sin}^{cos}(m\phi)J_m\Big(\frac{\pi q_{mn} r}{a}\Big)\exp[\jmath(-k_{mn}x + \omega t)], \tag{3.27}$$

where J_m is a Bessel function and q_{mn} is defined by the boundary condition in order to have the derivative $J'(\pi q_{mn})$ equal to zero.

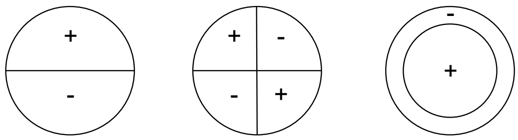

Figure 3.2: *Pressure patterns for the lowest three transverse modes,* $(1,0)$*,* $(2,0)$ *and* $(0,1)$ *respectively, of a cylindrical tube.*

The (m,n) mode, in this case, has an (r,ϕ) pattern for the acoustic pressure p with n nodal circles and m nodal diameters. Fig. 3.2 shows the pressure patterns for the lowest three modes of a tube, omitting the simple plane-wave mode. In the full three-dimensional picture, these become nodal cylinders parallel to the axis and nodal planes through the axis, respectively. The pressure patterns, therefore, have nodal lines, with similar nodal diameters in the transverse flow patterns. Nodal circles for pressure occur for modes of the type $(0,n)$ and the general mode (m,n) has both nodal lines and circles. By substituting Eq. (3.26) into Eq. (3.27), the propagation wave vector k_{mn} for a general mode (m,n) is obtained and it is given by

$$k_{mn}^2 = \Big(\frac{\omega}{c}\Big)^2 - \Big(\frac{\pi q_{mn}}{a}\Big)^2. \tag{3.28}$$

Thus, the plane wave mode with $m = n = 0$ can always propagate with $k = k_{00} = \frac{\omega}{c}$, which is not true for higher modes. The higher mode will be propagated if the frequency exceeds the cut-off value

$$\omega_c = \frac{\pi q_{mn} c}{a}. \tag{3.29}$$

In fact, for frequencies less than ω_c the term k_{mn} is imaginary and the mode is attenuated exponentially with distance. The amplitude falls by a factor e, or about 10 dB, within a distance less than the tube radius. For example, the first higher mode is the antisymmetric $(1,0)$ mode, which has a single nodal plane, and is for above a cutoff frequency $\omega_c = 1.84c/a$. The $(2,0)$ mode, with two nodal planes, is for $\omega_c > 3.05c/a$, and the lowest nonplanar axial mode $(0,1)$ is for $\omega_c > 3.80c/a$. Propagating higher modes are thus possible only when the diameter tube is greater than about two-thirds of the free-space acoustic wavelength. Hence, for modes with q_{mn} enough small, the plane wave component of this combination propagates without any disturbance. In the case of modes with q_{mn} too large, the wave propagates as a low-pass filtered version of the disturbance, while the nonpropagating modes modify the flow in the near neighborhood of the source.

3.3 Wall Losses

A plane sound wave is attenuated as it propagates because of losses. Energy is dissipated by viscous forces and thermal effects. We considered walls rigid enough in order to neglected their mechanical vibrations. More important effects caused by walls are, however, the viscous and thermal ones, since no real walls or real fluids are immune. The walls contribute a viscous drag which the relative magnitude depends on the thickness of the viscous boundary layer, itself depending on the viscosity η, the angular frequency ω and the tube radius a as follow

$$r_v = \left(\frac{\omega\rho}{\eta}\right)^{1/2} a. \tag{3.30}$$

The other lossy factor is given by the thermal exchange between the air and the walls. The relative magnitude of this loss depends on the ratio of the tube radius a to the thermal boundary layer thickness, as expressed by the parameter

$$r_t = \left(\frac{\omega\rho C_p}{\kappa}\right)^{1/2} a \tag{3.31}$$

where C_p is the specific heat of air at constant pressure and κ is its thermal conductivity. Hence, the effect of these loss terms change the characteristic impedance Z of the tube from its ideal real value $\rho c/S$ to a complex quantity. This make, in turn, the wave number k complex and lead to attenuation of the propagating wave as it travel along the tube.

3.4 Reflection and Transmission

Variations in the properties of the medium in which the wave is propagating can cause reflection and refraction. The phenomenon of refraction causes wave speed and propagation direction changing. The reflection includes more abrupt changes, the incident wave is reflected and only a part is transmitted into or through the object. In addition, the part of the energy transmitted can be dissipated by internal losses.
Consider a plane pressure wave $Ae^{(-jkx)}$ moving from a medium of impedance Z_1 to one of impedance Z_2. The reflected wave is denoted by $Be^{(jkx)}$ and the transmitted one by $Ce^{(-jkx)}$. Since the acoustic pressures on either side of the interface must be equal, taking the interface to be at $x = 0$ to simplify the notation, the three waves must be related by $A + B = C$. The reflection and transmission coefficients, refer to pressure amplitudes, are computed by

$$R = \frac{Z_2 - Z_1}{Z_2 - Z_1} \quad T = \frac{2Z_2}{Z_2 - Z_1} \tag{3.32}$$

If $Z_2 > Z_1$, the reflected wave is in phase with the incident one and a pressure maximum is reflected as a maximum. In the case $Z_2 < Z_1$, there is a phase change of 180° between the reflected and the incident wave, then a pressure maximum is reflected as a minimum. Finally, $Z_2 \gg Z_1$ or $Z_2 \ll Z_1$ implies a nearly total reflection. Expressing the formulas in term of intensities and considering $I_0 = A^2/Z_1$ the incident intensity, the formula becomes

$$I_r = \left(\frac{Z_2 - Z_1}{Z_2 - Z_1}\right)^2 \quad I_t = \frac{4Z_2Z_1}{(Z_2 - Z_1)^2}. \tag{3.33}$$

In order to include the the directions of the propagations, the angles between the normal to the interface and the directions of propagation k_i, k_r and k_t, are denoted, respectively, θ_i, θ_r and θ_t. Now the reflection and transmission coefficient are given by

$$I_r = \left(\frac{Z_2 \cos\theta_i - Z_1 \cos\theta_t}{Z_2 \cos\theta_i - Z_1 \cos\theta_t}\right)^2 \quad I_t = \frac{4Z_2 \cos\theta_i Z_1}{(Z_2 \cos\theta_i - Z_1 \cos\theta_t)^2}. \tag{3.34}$$

These expressions include complex quantities impedances, allowing the possibility of wave absorption. Eqs. (3.34) cannot be applied to solids, because the existence of a shear modulus implies longitudinal sound waves partially converted to transverse waves at the boundary. However, Eqs. (3.33) remain valid if the solid is isotropic and the incidence on the interface is normal, since propagation remains longitudinal in the solid. Extending the analysis to a solid medium, characterized by a wave impedance Z_2 and separating two semi-infinite regions of fluid with impedances Z_1 and Z_3, the expression becomes

$$I_r = \frac{4Z_1Z_2^2Z_3}{Z_2^2(Z_1 + Z_3)^2 \cos^2(k_2l) + (Z_2^2Z_1Z_2)^2 \sin(k_2l)} \tag{3.35}$$

where l is the thickness of the solid material.

3.5 Finite-Length Tube

A finite-length tube will be now considered, taking in consideration the reflection due to the end of the tube, whether it is open or closed. Supposing a pressure pulse hitting a rigid end with infinite impedance, the cell of molecules next to the wall will be pushed back on the adjacent cell. It causes the adjacent cell to recoil in the reverse direction and, in turn, pushes on its neighbour. A reversed direction pressure pulse will be created, bouncing the end with no loss of energy. Considering an open end, the pressure region travelling along the tube will find the air outside, at atmospheric pressure. Hence, there will be a sharp drop in impedance and the discontinuity will reflect sound amplitude back with the opposite sign. The low pressure region hitting the end of the tube air, rushes in and creates a compression wave heading back the tube. The opposite happens when a high pressure region hits the end of the tube. Hence, an open tube partly reflects the wave with a change of sign, but reflecting as if the tube longer by about 0.6 times the diameter (for wavelengths that are large compared to the diameter).

The reflection of a wave at the end of a tube is due to an impedance mismatch. Supposing, instead, a pulse through a tube toward an open end with equal pulse propagating outside the tube as well. The pressure exiting the tube finds matched pressure outside but without impedance change. In this latter case, there is no back reflection inside the tube. Summarizing, at an open end the wave inverts: a reflected pressure peak becomes a trough, and a trough becomes a peak. In contrast to the closed end where a pressure peak reflects as a peak. This means the pressure changes are lowest at the open end and highest at the closed end. Reflections happen also in case of diameter tube change. If a positive pressure pulse is travelling from a wider to a narrower one, a positive pressure pulse returns from the junction, reflecting part of the energy. If instead it encounters a wider tube, a negative pressure pulse reflects part of the energy. As can be noted from the Eq. (3.14), the impedance of air in a tube depends on the diameter of the tube. The bigger diameter, the lower the impedance. A high impedance implies that if a small cell of air is pushed, a neighboring cell will push back harder than in a larger tube with lower impedance.

Consider now a wave being pushed to the positive direction at time τ, in free space would be communicated in all directions a distance $x = c\tau$. In a tube, instead, most of those directions lead to the walls, where the pressure pulse created by the push is reflected. Some of the reflected wave returns fast enough to be in phase with the pushing of the cell that was originally disturbed, increasing the impedance. The pressure pulse reflect many times, depending on the diameter of the tube. The wall needs to be within an eighth of a wavelength or so, to have a return in phase. In addition, short wavelength escapes the tube more readily than the longer one. The frequency is higher for the shorter wavelength and a cell of molecules inside the tube may not be in phase. It reinforces reflection from the walls in time to increase its impedance. Hence, stronger reflection is seen in narrow tube, stronger transmission is seen a wider one.

Given this, considering a plane wave $p^+(x,t) = F(t - x/c)$ inside the tube travelling in positive direction, it will reflect into a left-running wave $p^-(x,t)$. Without visco-thermal losses, the boundary condition of vanishing velocity is given by

$$u(0,t) = \frac{p^+(0,t) - p^-(0,t)}{\rho_0 c} = 0. \tag{3.36}$$

This implies a reflected wave $p^-(x,t) = F(t + x/c)$, equal in amplitude and shape to the incident one, and therefore

$$R = \frac{p^-(0,t)}{p^+(0,t)} = 1. \tag{3.37}$$

where R is the reflection coefficient seen in Sec. 3.4. In reality heat transfer at the wall reduces the reflection coefficient. The heat transfer is a result from the difference between the wall temperature T_w, which remains practically constant, and the bulk temperature T of the gas, which varies with the adiabatic pressure fluctuations $p_a = p^+ + p^-$. Supposing a L-long tube terminating at $x = L$ by the impedance Z_L, the pressure in the tube, thus, is a superposition of two waves, moving to the right and left, respectively. Taking A and B the complex amplitudes of these waves at the point x, the pressure is

$$p(x,t) = [Ae^{-jkx} + Be^{jkx}]e^{j\omega t}. \tag{3.38}$$

The acoustic particle velocity is the superposition of the particle velocities associated with these two waves as well, therefore the acoustic flow becomes

$$U(x,t) = \left(\frac{S}{\rho c}\right)[Ae^{-jkx} - Be^{jkx}]e^{j\omega t}. \tag{3.39}$$

At the end $x = L$, pressure and flow are related to the terminating impedance Z_L

$$\frac{p(L,t)}{U(L,t)} = Z_L. \tag{3.40}$$

It gives the way to determine the complex ratio

$$\frac{B}{A} = e^{-2jkL}\left[\frac{Z_L - Z_0}{Z_L + Z_0}\right], \tag{3.41}$$

with $Z_0 = \rho c/S$ the characteristic impedance of the tube. The power reflected from Z_L can be computed as

$$R = \left|\frac{B}{A}\right|^2 = \left|\frac{Z_L - Z_0}{Z_L + Z_0}\right|^2. \tag{3.42}$$

From Eqs. (3.41) and (3.42) can be seen that there is no reflection if $Z_L = Z_0$ and complete reflection if $Z_L = 0$ or ∞. Since Z_0 is real for a lossless tube, there is also perfect reflection if Z_L is purely imaginary. However, if Z_L has a real part that is nonzero, then there will always be some reflection loss. The input impedance Z_{IN} at the point $x = 0$ is

$$Z_{IN} = Z_0\left[\frac{A + B}{A - B}\right] \tag{3.43}$$

and using Eq. (3.41) it becomes

$$Z_{IN} = Z_0\left[\frac{Z_L \cos kL + jZ_0 \sin kL}{jZ_L \sin kL + Z_0 \cos kL}\right]. \tag{3.44}$$

There are two idealized cases, the open and closed end. The first is the case of a rigidly stopped tube at $x = L$, hence with $Z_L = \infty$. For such a tube,

$$Z_{IN}^{closed} = -jZ_0 \cot(kL). \tag{3.45}$$

In opposite, for the case of an ideally open tube with $Z_L = 0$,

$$Z_{IN}^{open} = jZ_0 \tan(kL). \tag{3.46}$$

The resonance frequencies for open and closed tubes are found from applying the condition that the end at $x = 0$ is also open. In this way the resonances occur if $Z_{IN} = 0$. For a closed tube, this requires that $\cot(kL) = 0$, giving

$$\omega^{stopped} = \frac{(2n-1)\pi c}{2L}. \tag{3.47}$$

Eq. (3.47) shows that the resonance frequencies correspond to an odd number of quarter wavelengths in the tube length. For an ideally open tube, with instead $\tan(kL) = 0$ is

$$\omega^{open} = \frac{n\pi c}{L}, \tag{3.48}$$

corresponding to an even number of quarter wavelengths, or any number of half wavelengths, in the tube length. The treatment of a physically open tube is difficult since, while $Z_L \ll Z_0$, it is not a sufficient approximation to set it to zero. In order to calculate the radiation load Z_L on a tube that terminates in a plane flange of size much larger than a wavelength (and therefore effectively infinite), the assumption that the wavefront at the open end is quite planar is made. It gives the follow result:

$$Z^F = R + jX \tag{3.49}$$

where

$$R = Z_0 \left[\frac{(ka)^2}{2} - \frac{(ka)^4}{2^2 3} + \frac{(ka)^6}{2^2 3^2 4} - \dots \right] \tag{3.50}$$

$$X = \frac{Z_0}{\pi k^2 a^2} \left[\frac{(2ka)^3}{3} - \frac{(2ka)^5}{3^2 5} + \frac{(2ka)^7}{3^2 5^2 7} - \dots \right]. \tag{3.51}$$

As usual, a is the radius of the tube. If $ka \ll 1$, then $IZ^F| \ll Z_0$ and most of the wave energy is reflected from the open end. If $ka > 2$ then $Z^F \approx Z_0$ and most of the wave energy is transmitted out of the end of the tube into the surrounding air.

Finally, the behaviour of the tube with physically realistic wall losses is considered. In a tube not unreasonably narrow, with $r_v > 10$, the small change in the characteristic impedance Z_0 can be neglected and allows the possibility that k is complex. Now k is written as $(\omega/v - j\alpha)$ where

$$v = c \left[1 - \frac{1}{r_v \sqrt{2}} - \frac{\gamma - 1}{r_t \sqrt{2}} \right] \tag{3.52}$$

and

$$\alpha = \frac{\omega}{c} \left[\frac{1}{r_v \sqrt{2}} + \frac{\gamma - 1}{r_t \sqrt{2}} \right]. \tag{3.53}$$

Adding this information in Eq. (3.44), with the appropriate expression for Z_L, the behaviour of the input impedance of a real tube can be deduced. For example, in the case of a ideally open tube ($Z_L = 0$) of length L the expression for Z_{IN} is

$$Z_{IN} = Z_0 \left[\frac{\tanh \alpha L + j \tan(\omega L/v)}{1 + j \tanh \alpha L \tan(\omega L/v)} \right]. \tag{3.54}$$

This expression has maxima and minima at the maxima and minima, respectively, of $\tan(\omega L/v)$. The Z_{IN} value at the maxima is $Z_0 \coth \alpha L$, and at the minima is $Z_0 \tanh \alpha L$. The value of α increases with frequency as $\omega^{1/2}$, so these extrema decrease in prominence at higher frequencies, and Z_{IN} converges toward Z_0. For a closed tube at the far end, the factor in square brackets in Eq. (3.54) is simply inverted. For narrow tubes the lower resonances are dominated by this wall-loss mechanism, for wider open tubes, instead, radiation losses from the end become more important and in particular at high frequencies. The low frequency resonances are sharper for wider tube than for the narrower ones because of the reduced relative effect of wall damping, but the high frequency resonances of the wider tube are washed out by the effects of radiation damping. All the impedance maxima and minima have frequencies that are nearly harmonically related, that is as the ratio of two small integers. In fact, because the end correction decreases with increasing frequency, the frequencies of these extrema are all slightly stretched, and this effect is more pronounced for wider than for the narrower tube.

3.6 Sound Radiation From an Open End

Consider now the radiation of sound from a tube. Tubes are used as an impedance matching between a volume source and free space. If the frequency is low enough compared to the tube diameter, the flow near the tube end is incompressible in a region large enough to allow the tube opening to be considered as a monopole sound source. The strength of this monopole is determined by the tube end velocity v. Assuming the end acoustically described for the field inside the pipe by an impedance Z_p, the pressure p inside the tube consists, as already noted, of a right-running incident wave and a left-running reflected wave: $p = p^+ + p^-$. The acoustic velocity is related to the acoustic pressure by

$$v = \hat{v}e^{j\omega t} = \frac{p^+ + p^-}{\rho c}. \tag{3.55}$$

A redistribution of the acoustic mass flow vS through the end into the surface of a compact sphere of radius$r$ and surface $4\pi r^2$ is assumed because the conservation of mass. Knowing that the real part of the radiation impedance of a compact sphere with $ka \ll 1$ is given by

$$\mathrm{Re}\left(\frac{Z}{\rho c}\right) \approx (ka)^2, \tag{3.56}$$

the radiated power for a harmonic field in- and outside the tube can be calculated, by using

$$IS = \frac{1}{2}\hat{v}\hat{v}^*\mathrm{Re}(Z_p)S = \frac{1}{2}\left(\frac{S}{4\pi r^2}\hat{v}\right)\left(\frac{S}{4\pi r^2}\hat{v}^*\right)(k^2r^2\rho c)(4\pi r^2). \tag{3.57}$$

From this conservation of energy relation, the real part of the radiation impedance Z_p of an unflanged tube is found:

$$\mathrm{Re}(Z_p) = \frac{1}{4\pi}k^2S\rho c, \tag{3.58}$$

which is for a tube of radius a, is given by

$$\mathrm{Re}(Z_p) = \frac{1}{4}(ka)^2\rho c. \tag{3.59}$$

The imaginary part $\mathrm{Im}(Z_p)$ takes into account the inertia of the air flow in the compact region just outside the tube. It is equal to $k\delta$, where δ is the so-called "end correction". Just outside the end, in the near field of the monopole, the pressure is a factor ρckr lower than the acoustic velocity, which is much smaller than the ρc of inside the tube. Therefore, the outside field forces the inside pressure to vanish at about the end. Although the exact position of this fictitious point $x = \delta$, where the wave in the tube is assumed to satisfy the condition $p = 0$, depends on geometrical details. It is a property of the tube end and therefore $\delta = O(a)$.

This implies that the end correction amounts to a phase shift of the reflected wave, thus, to a purely imaginary impedance Z_p . Up to order $(ka)^2$ this impedance can be expressed as

$$Z_p = (jk\delta + 1(ka)^2)\rho c, \tag{3.60}$$

where $0.61a \leq \delta \leq 0.85a$ for circular tubes. The lower limit corresponds to an unflanged tube while the upper limit corresponds to a tube end with an infinite baffle (flanged).

3.7 Standing Waves

Lastly, the standing waves problem is considered. This concept directly depends on the reflection of sound. This phenomenon is dependent on the reflection of sound at the two parallel surfaces. Assuming two flat, solid parallel walls separated a given distance, a sound source between them will radiate sound of a specific frequency. As discussed, the wave is reflected back continuously between the two walls. One wave travels to the right, the other one in the opposite direction. The two travelling waves will interact forming a standing wave. Only this interaction will be stationary. This resonant condition between the wavelength and the distance between the two surfaces are established by the frequency of the radiated sound. Hence, inside a tube the plane wave arrived in the end is reflected and coming back it is superimposed with the incident one, producing the standing wave. Standing wave will be a vibrational pattern created within the medium at specific frequencies of vibration. These frequencies are known as harmonic frequencies. At any other frequency, the interference due to the reflected and incident waves results in a irregular and non-repeating disturbance. The natural frequencies of an object are the harmonic frequencies at which standing wave patterns are established within the object. These standing wave patterns are the lowest energy vibrational modes of the object. These natural modes of vibration are representative of the patterns which require the least amount of energy. The wave pattern associated with these natural frequencies is is characterized by points which appear to be standing. These points are referred to as nodal points or nodal positions, occurring as the result of the destructive interference of incident and reflected waves. Each nodal point is surrounded by antinodal points, creating an alternating pattern of nodal and antinodal points.

A standing wave $y_n(x,t)$ can be viewed as a superposition of sinusoidal travelling waves, therefore, is the sum of two sinusoidal waves y^+ and y^-. Using the Werner formulas a standing wave can be written as

$$y_n(x,t) = y_0\{\cos[k_n(ct-x)+\phi_n] - \cos[k_n(ct+x)+\delta_n]\}, \tag{3.61}$$

where y_0 is the arbitrary amplitude of the wave. Using a general form, can been also written

$$y(x,t) = y_0 \cos(k_n x + \phi_n)\cos(\omega_n t + \delta_n). \tag{3.62}$$

Consider now, the open end and the closed end cases. Open end, as shown in the previous sections, means an uncovered end such that the air at the end of the tube can freely vibrate when the sound wave reaches it. Hence, the wave has a back-and-forth longitudinal motion and assumes the antinodal position of the standing wave pattern. At the closed end, instead, the wave is not free to vibrate and is forced into assuming the nodal position. It means that vibrational antinodes will be present at any open end and vibrational nodes will be present at any closed end.
Three different cases can happens: tube open at both ends, tube closed at both ends, tube closed at one end and open at the other one. In the first case, the pattern for the fundamental frequency (the lowest frequency and longest wavelength pattern) has antinodes at the open ends and a single node in between. The distance between antinodes will be equivalent to one-half of a wavelength. Thus, the length of the air column is equal to one-half of the wavelength for the first harmonic. It follows that the wavelength length of the standing wave depends on the tube length. The standing wave pattern for the other harmonics can be obtained adding a node and an antinode. Assume a L-long tube, it will have a node in $L/2$ and two antinodes in $x = 0$ and $x = L$. Being

$$y(0,t) = y(L,t) = 0, \tag{3.63}$$

Eq. (3.62) can be written

$$y(x,t) = y_0 \cos(k_n x)\cos(\omega_n t). \tag{3.64}$$

It means

$$\cos(k_n L) = \pm 1 \quad k_n L = n\pi, \tag{3.65}$$

and, thus, the equation relating the length of a wave and the length of the tube is given by

$$\lambda_n = \frac{2}{n} L \tag{3.66}$$

where n is the number of the harmonic and using $k = \frac{2\pi}{\lambda}$. Tab. 3.1 summarizes the relationships between the standing wave pattern and the length-wavelength for the first five harmonics.

Table 3.1: *Relationships between the standing wave pattern and the length-wavelength for the first five harmonics, in the case of two open end.*

Harmonic	N∘ of Wave in the Tube	N∘ of Nodes	N∘ of Antinodes	Wavelength
1	1/2	1	2	$\lambda = (2/1)L$
2	1	2	3	$\lambda = (2/2)L$
3	3/2	3	4	$\lambda = (2/3)L$
4	2	4	5	$\lambda = (2/4)L$
5	5/2	4	5	$\lambda = (2/5)L$

In the latter case, an open end and an closed one, the pattern for the fundamental frequency will have a node at the closed end and an antinode at the open end. The distance between adjacent antinodes is equivalent to one-half of a wavelength. Since nodes the distance between an antinode and a node is equivalent to one-fourth of a wavelength. For the first harmonic, therefore, the length of the air column is equal to one-fourth of the wavelength. The next harmonic will have one more node and antinode. Unlike the other case, there is no even harmonic. The next frequency above the fundamental frequency is the third harmonic. Only odd-numbered harmonics are produced. This phenomenon is due to the constraints for the closed end (a node at the closed end and an antinode at the open end) not met by the even harmonics. In this case, assuming the closed end in $x = 0$ and the open end in $x = L$, there will be a node in $x = 0$ and an antinodes in $x = L$. Hence,

$$y(x,t) = y_0 \sin(k_n x)\cos(\omega_n t), \tag{3.67}$$

$$\sin(k_n L) = \pm 1, \quad k_n L = \frac{2n-2}{2}\pi. \tag{3.68}$$

Finding this time

$$\lambda_n = \frac{2}{2n-1}L. \tag{3.69}$$

Tab. 3.2 summarizes the relationships in this latter case.

Table 3.2: *Relationships between the standing wave pattern and the length-wavelength for the first five harmonics, in the case of an open end and a closed end.*

Harmonic	N∘ of Wave in the Tube	N∘ of Nodes	N∘ of Antinodes	Wavelength
1	1/4	1	1	$\lambda = (4/1)L$
3	3/4	2	2	$\lambda = (4/3)L$
5	5/4	3	3	$\lambda = (4/5)L$
7	7/4	4	4	$\lambda = (4/7)L$
9	9/4	5	5	$\lambda = (4/9)L$

The tube closed at both ends case is not needed for this work but after discussed the others two cases, it's easy to derive. The nodes are in $x = 0$ and $x = L$, thus

$$y(x,t) = y_0 \sin(k_n x)\cos(\omega_n t), \tag{3.70}$$

$$\sin(k_n L) = 0 \quad k_n L = n\pi. \tag{3.71}$$

Finally, the condition is given by

$$\lambda_n = \frac{2}{n}L. \tag{3.72}$$

Chapter 4

Acoustic Measurements

Acoustic measurements are an important tool for the analysis of acoustical problems and for perform acoustic investigations. In order to reach good accuracies, they need certain requirements concerning the instrumentation and acoustical conditions [MM12]. Acoustic measurements are often difficult to perform and there is not expectations about the absolutely reproducibility of the results. The measurement system can be separated into source and receiver components. The source component is the sound source utilized for play the stimulus necessary to excite the system under analysis. The receiver component consists of a sound level meter or sound analyser, displaying the sound level in decibels or any other frequency-dependent data. In addition, the measurements are usually performed in special environment.

In this chapter, a description of the components of acoustic measurements will be proposed, including the difficulties that may arise and the typical sources of errors. The measurement techniques and the signal processing usually used follow. The relative properties and issues of the different techniques will be also considered and explained. A focus on the technique used in this work will be given.

4.1 Microphones

Microphones normally convert sound pressures into electrical signals, which in turn can be displayed, stored, and analysed by analogue or digital techniques. In a wider sense, electroacoustical or electromechanical transducers can be called microphones [MM12]. Microphones used for capture wave in air contain a very thin and flexible diaphragm, which follows the air movement of the local sound field. The vibration of the diaphragm is converted by the electromechanical force interaction into an electrical signal, process that, in the optimal case, is linear and frequency independent. The electrical signals produced by the electroacoustic conversion are proportional to one of the specific sound field quantities, sound pressure or to sound velocity. If the sound pressure excites only one side of the diaphragm, the electromechanical force on it is proportional to the sound pressure. In the case of both sides excitement, it is proportional to the sound pressure gradient. To take in account that, the microphone presence distorts the sound field and this distortion increase by increasing its size in comparison to the wavelengths of sound.
The sensitivity of a pressure microphone is due to the open-circuit voltage with reference to the sound pressure on the diaphragm. To be distinguished the free-field or diffuse-field sensitivity, that is related to the sound pressure in the sound field, without the microphone in place. The frequency responses of these differ only slightly. At high frequencies (> 10 kHz) free-field responses

are significantly higher by several decibels due to directivity effects.

A condenser microphone is a passive electrostatic transducer consisting of a mobile diaphragm and a rigid backplate. Between the mechanical force and the voltage on the condenser the relation is non-linear since two charged plates interact with a quadratic law of force and voltage. For this reason a constant polarisation voltage U_0 (typically 200 V) is applied over a very large resistance R (> 10 GΩ), creating a constant charge on the condenser. A sound-induced modulation of the distance between diaphragm and backplate results in a change in capacity with constant charge, and thus in a sound-induced AC voltage signal U added to the constant polarisation voltage [MM12]. If the amplitudes are not too high (up to 140 dB) the relation between sound pressure and voltage can be considered linear. Typically material of the diaphragm is pure nickel foil of few micrometres thick. The capsule of the microphone has extremely high impedance (10-100 GΩ) and for this reason a preamplifier for impedance transformation is placed near the capsule with the possible use of long cables. The microphone is considered the whole arrangement of capsule and preamplifier. The significant components of a condenser microphone are the resistance and capacity, as electrical components, and the compliance of both the diaphragm connected to the housing and of the air cavity, as mechanical components. The frequency range covered is limited by electrical and mechanical high-pass effects and by capillary tubes for quasi-static pressure equalisation at low frequencies. The mechanical resonance of diaphragm mass and total stiffness also limit the range at high frequencies. The sensitivity is constant over all the range covered at approximately

$$\frac{U_{I=0}}{p} = nS\frac{U_0}{d} \tag{4.1}$$

where $U_{I=0}$ is the open-circuit receiving voltage, p the sound pressure, n the total compliance (diaphragm stiffness and air cavity), U_0 the polarisation voltage, S the diaphragm surface and d the distance from the diaphragm to the backplate. The displacement of the diaphragm is given by the integral over the surface elements excited by the incident sound wave with locally varying sound pressures in amplitude and phase. The sensitivity is, then, dependent on the direction. The polarisation U_0 voltage can be ignored if a dielectric (called electret) material with permanent polarisation is placed between the condenser electrodes.

For the measurements described in the following chapters, particular microphone with dimensions of a few millimetres was used. Miniature microphones such that can be built using electret foils in order to eliminates the need for a polarizing power supply by using a permanently charged material. This microphones are called electret microphone and they are a type of electrostatic capacitor-based microphone.

4.2 Loudspeakers

The dynamic loudspeaker is the most frequently used in the measurements, because of their relatively small size. An electrodynamic loudspeaker consists of a conical membrane, which in its centre is connected to a cylindrical voice coil [MM12]. The coil, with a resistance typically of several Ohms, is placed in the air gap of a pot magnet of hard magnetic ferrite or of Alnico alloy, in which a radially homogeneous magnetic field is established by a magnetic flux density of some Vs/m^2. At higher frequencies the inductance is significant. The latter can be reduced by copper rings in the air gap, which at the same time add damping to the mechanical resonance. The movable parts of the loudspeaker (the membrane and voice coil) are supported by springs. This is realised by a spider to keep the voice coil in the nominal position and by a soft mounting of the membrane at its outer perimeter. The membrane is made of a material with high internal damping and low density in order to suppress bending waves. The common material is paper, but PVC or light metals are also used. The relatively large mass of the vibrating parts and the mechanical system's resonance limit the possibility of generating short pulses and it is usually not acceptable. It can be improved with low impedance of the connected power amplifier or using digital signal processing and inverse filtering. With these methods a linear sound reproduction in a wide frequency range can be obtained. Non-linear stiffness of the membrane support can produces non-linear distortions of dynamic loudspeakers if the signals are broadband signals.

In many acoustic measurements, plane waves are desired. Plane waves can be produced by common loudspeakers if the measurement area is small and on the main radiation axis. Improvements are given if the loudspeaker membrane area S is small and with coaxial multiple loudspeakers, which have a common axis of radiation. The first point is motivated by

$$r_F \approx \frac{S}{\lambda} \tag{4.2}$$

where r_F is the far-field distance. Eq. (4.2) shows as r_F is proportional to S. Concentrating the radiation on the main axis the wavefront in far field is symmetrical and approximately plane. In situations where special directivity are necessary, different loudspeakers can be used. Since the loudspeaker dimensions are not very small compared to the wavelength, an omnidirectional radiation can at least be approximated by loudspeakers in spherical symmetry. This can be done by means of housings based on regular polyhedra. The dodecahedron is most commonly used.

4.3 Sources of Errors in Digital Measurement

In practice, measurements are always affected by non-idealities, such noise, non-linearity and time variance [MM12]. Background noise (acoustical, electrical or thermal), is usually not correlated with the input signal. Excitation signals like impulses, pure tone or broadband stochastic signals are spread out by the cross-correlation over the measurement and are noticeable in the impulse response obtained only with their mean power. If some part of the impulse response results dominated by noise, can be simply deleted with the windowing technique.

Non-linearities, like distortion, appear in the measured impulse response as an apparent noise floor. For this reason it can be hard to perform the separation, and they are treated like background noise by using the window technique. Non-linearities can be noted by observing the effective signal-to-noise ratio improvement by coherent averaging and by checking if the gain in dynamic range is asymptotically limited. Weak non-linearities can often be tolerated and their effect can be reduced by reducing the input amplitude. Signal-to-noise ratio is limited by non-linearities in the components of the instrumentation, depending on the choice of the excitation signal, on the system to be measured, on power amplifiers, loudspeakers, Sample&Hold devices and A/D converters. From this point of view, signals with a low crest factor, defined by

$$C = \frac{|x_{peak}|}{x_{rms}}, \tag{4.3}$$

are better choices. As it will be seen soon, maximum-length sequences are superior since they have a crest factor of 0 dB than sweeps and chirps that have a crest factor of 3 dB. However, under extreme conditions (SNR $<$ 70 dB) maximum-length sequences are not the best choice. Another point to be considered is the choice of periodic or aperiodic processing. The FFT and MLS technique are both related to periodic signal processing, swept-sinusoidal signals instead could be performed by periodic or aperiodic algorithms. The choice of aperiodic signal processing, at the cost of heavier computational load, may result in several practical advantages. The amplitude of the excitation signal is, often, a compromise between increasing distortions at high levels and decreasing the signal-to-noise ratio at low levels. Direct aperiodic processing requires de-convolution in the time domain, a sequence of length N need N^2 multiplications. Using a sweep with increasing frequency, the response to harmonic components appear before the main excitation at that frequency and the harmonic distortion products in the excitation, appearing at negative time can easily be removed by using a time window. Another aspect, concerning the noise floor, is that periodic processing results in an impulse response with a noise floor that is approximately constant up to the time where the first distortion products appear. Aperiodic direct de-convolution, instead, produces a decaying noise tail that is increasingly low-pass filtered towards its end because the last part of the impulse response derives from steady-state noise convolved with the excitation sweep in reverse order. This decreasing noise floor can be confused as a reverberant tail of the impulse response.

Lastly, time variance can be produced by temperature changes and they, in turn, produce changing in the signal shape hardly to detected. Phase distortion in sequences and between sequences is the reason for measurement errors. To avoid the effects of time variances the maximum temperature drift in a room during the measurement must to be not larger than

$$\Delta\theta < \frac{300}{fT}, \tag{4.4}$$

where $\Delta\theta$ is the temperature drift in degrees Celsius, T is the reverberation time and f one-third octave band or octave midband frequency.

4.4 Anechoic Chamber

Performing acoustic measurements, test arrangements or rooms are implemented to create well-defined acoustic environments [MM12]. To obtain waves propagation undisturbed by reflections and diffraction, particular rooms called "anechoic chambers" are built. These rooms have walls absorbing sound by 99.9%. They provide a level reduction of reflections by 30 dB. This requirement is met by mounting wedge-like porous material on the walls and ceiling, either with or without airspace behind. The reached absorption coefficient is above 50% and depends on the dimensions of the wedges. The floor can be treated in the same way using a net to allow the walking on it. In addition, anechoic chambers should have good insulation against background noise, reached using a vibration-isolated foundation. In this case the room stands on springs, and the low resonance frequency hinders vibrations of the surrounding building from outside to propagate into the room.

4.5 Impulse Response Measurement

The impulse response and their associated transfer function measurement is one of the most important task in the analysis of an acoustical space. The accuracy of this measurement has an impact on the acoustical parameters that derive from it. When spatial information is neglected and both source and receivers are considered as a point and omnidirectional, the whole information about the system is contained in its impulse response. The acoustics of the acoustical space is assumed a linear and time-invariant system, including both time-domain effects (echoes, discrete reflections, statistical reverberant tail) and frequency-domain effects (frequency response, frequency-dependent reverberation). In order to obtain the impulse response, a computer generates a special test signal, which passes through an audio power amplifier and is emitted through a loudspeaker placed inside the space under analysis. The resulted signal is captured by a microphone and digitalized after preamplification by the same computer used for the test signal. Basically, the measurements are based on feeding an input to a system and computing the response of the system using the relative output.

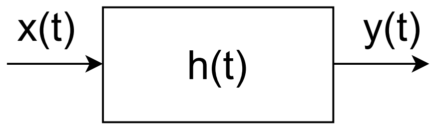

Figure 4.1: *Diagram of measurement processing.*

As can be seen in Fig. 4.1, the system is fed with an excitation signal $x(t)$, containing energy on all frequencies of interest, and the output $y(t)$ of the system is collected in order to compare it with the input. The response of the system is computed measuring the input and the output signals, the aim is to understand how the system modify the input signal. The linear dependency on two signals can be described by a correlation function

$$R_{xy}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_T x(t)y(t+\tau)\,\mathrm{d}t \tag{4.5}$$

or its Fourier transform

$$K_{xy}(f) = \int_{-\infty}^{+\infty} R_{xy}(\tau)e^{-j2\pi f\tau}\,\mathrm{d}\tau. \tag{4.6}$$

The cross correlation is a measure of similarity between two signals and in practice, it is computed using the Fourier transforms of input $x(t)$ and output $y(t)$:

$$K_{xy}(f) = X^*(f)Y(f). \tag{4.7}$$

The characteristics of the transmission path can be computed by

$$H(f) = \frac{Y(f)}{X(f)} = \frac{Y(f)X^*(f)}{X(f)X^*(f)} = \frac{K_{xy}(f)}{K_{xx}(f)} \tag{4.8}$$

and the impulse response by the inverse Fourier transform.

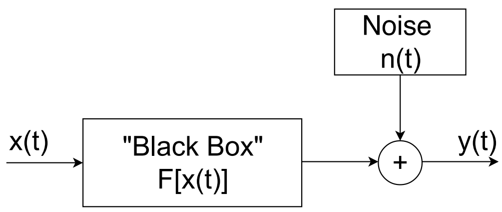

Figure 4.2: *A basic input/output system.*

As discussed before, the first approximation of an acoustical space is a “black box”, conceptually described as a Linear Time Invariant System, with added some noise to the output. This model is shown in Fig. 4.2. However, considering the loudspeaker often subjected to not-linear phenomena and the wave propagation not always perfectly time-invariant, a more accurate model can be built. Fig. 4.3 shows the model of the global system, including the loudspeaker (considered as a non linear element) and the acoustical space (considered as a linear system), target of the measurements. Measuring the impulse response of the linear system $h(t)$ the artefacts caused by noise, not-linear behaviour of the loudspeaker and time-variance must be removed.

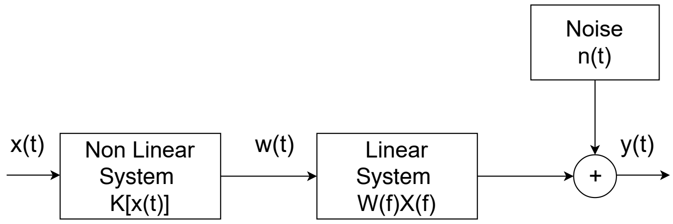

Figure 4.3: *Modelization of the system including the loudspeaker and the acoustical space.*

All the method for measuring the impulse response is based on applying a known input signal and to measure the system's output. The choice concerning the excitation signal and the deconvolution technique, used to obtain the impulse response from the measured output, is of essential importance. In particular, the emitted signal and the deconvolution technique have to maximize the SNR of the deconvolved impulse response and eliminate all the non linear artefacts in the process. Different ways to measure transfer functions have evolved, common to all of them is the use of an excitation signal containing all the frequencies of interest. Therefore, it is desirable to use excitation signals with high energy in order to achieve a sufficient SNR in the whole frequency range of interest.

The method used to excite the system can be divided in

- Nondeterministic stimulus methods (2-channel FFT)
- Deterministic stimulus methods

In the first category, as suggest the name, nondeterministic input signals are used. In the second one, instead, the input stimulus can be impulse, noise/pseudorandom noise (FFT, MLS), sinusoidal (stepped tone, multitone) or frequency sweeps (level recorder, TDS, Farina's method [Far00]).

2-channel FFT Method

The method called 2-channel FFT is suitable for systems where deterministic stimuli cannot be applied [MM12]. It usually uses random noise like white or pink noise. Measurement and the signal processing are performed in the frequency domain. Input and output signals are measured simultaneously using a 2-channel-FFT analyser and then FFT transformed in order to be processed by complex spectrum division(see Eq. (4.8)). The signal used should be a sufficiently broad bandwidth and not contain zeros in the spectrum to avoid problems in the division. The dual channel analyzer must always average over several individual measurements to obtain a reliable result. In any of those single measurements, the SNR may be insufficient at some frequencies. These should then be excluded from the averaging process in order to avoid errors in the displayed frequency response, which might occur due to the division of the spectra of both channels. After performing the spectrum division, the impulse response is computed using the inverse Fourier transformation.

Because the necessity to average many blocks of data to achieve a consistent result, the responsiveness of dual-channel analysis is very poor. In addition, the methods is slow, imprecise and cannot remove unwanted reflections from impulse response by prior windowing. It also requires two channels. Another drawbacks to take in account is that the precise propagation delay of the acoustical transmission path must be known and the direct signal must be delayed to analyze the same parts of the excitation signal on both channels. Moreover, in order to avoid leakage effects, the signal blocks submitted to the FFT analysis must be windowed. This is a considerable source of error as delayed components are attenuated more than the direct sound. The dual channel analysis could be improved considerably by generating the impulse response of every single measurement by inverse FFT. Windowing the impulse response offers to control the amount of reflections entering into the result and to mute the noise outside the windowed interval. In addition the SNR may vary between data blocks, noise-contaminated data blocks should be excluded to avoid errors in the frequency response.

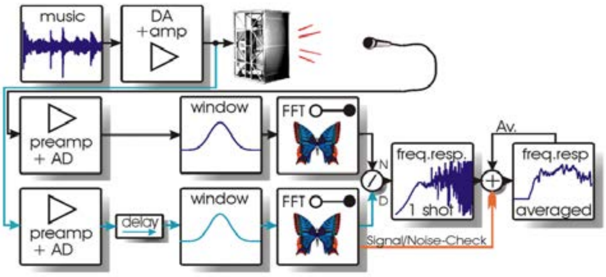

Figure 4.4: *Signal processing steps for 2-channel FFT analysis.*

Deterministic stimulus methods

The deterministic stimulus methods category, instead, includes the use of different stimulus:

- Stepped sine: using as input pure sine tones the transfer function is measured step by step at individual frequencies. It's very slow but very accurate. The spectral resolution is limited by the total time available for measurements but it can completely suppresses surrounding frequencies obtaining a excellent SNR. In addition reflections can be removed but the analysis time window must be shorter than the reflections' relative propagation time delays.

- Multitone: differs from the previous one only by use two or more sinusoids with different frequency. It's faster and arbitrary spectrum can be created for different purpose but there's the intermodulation distortion between sines problem.

- Impulse: an impulse is used and no additional processing is required. The stimulus is ideally a Dirac delta function but in practice it's an approximation. Due to its very little energy can results in low SNR. A solution can be using impulse trains, but there can be time aliasing because the impulse response obtained is periodic. It's a very fast and simple method, immune to time variance but the level of the noise and the distortion components cannot be separated from responses. It's suitable for measurements without acoustic path, like loudspeaker testing.

- Maximum Length Sequence (MLS): it uses a signal consisting of a binary sequences built by a shift register. A binary sequences of length $L = 2^n - 1$ contains 2^n times more energy than a single impulse as the excitation is stretched out over the whole measurement period. The signal is generated from a deterministic and reproducible process, so the inverse MLS^{-1} can be computed. The impulse response can be recovered by periodic cross correlation with only addition and subtraction involved and giving the possibility to compute the convolution in real time. Disadvantages are that the MLS measurements are very sensitive to distortion and time variance.

- Sweep: linear or logarithmic sweep is used. The logarithmic sweep increases the frequencies in a logarithmic way and provide more energy in the low frequency, a critical zone, going more rapidly towards the high frequency. The inverse signal is fast to compute because is the same signal with the time axis inverted. The FFT spectrum of such a logarithmic sweep declines by 3 dB/octave. Every octave shares the same energy, but this energy spreads out over an increasing bandwidth. Therefore, the magnitude of each frequency component decreases. One of main properties is that the spectral distribution is often quite well adapted to the ambient noise, resulting also in a good SNR at the critical low end of the frequency scale.

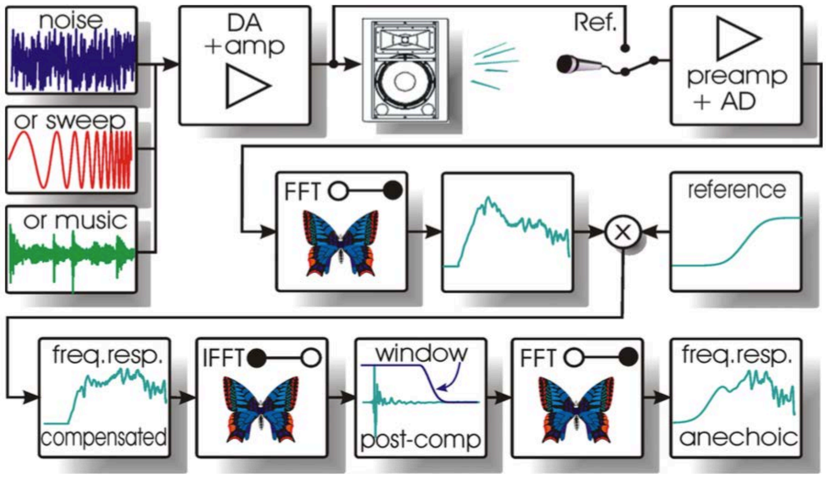

Figure 4.5: *Signal processing steps for measurements with any deterministic signal.*

Fig. 4.5 shows the processing stages in the case of deterministic stimulus. A more complete discussion about deterministic stimulus methods follows with also some consideration about the different techniques. The chapter is concluded offering a deeper presentation about the Farina's method.

Level Recorder technique The Level Recorder is one of the oldest methods. It involved a logarithmic sweep as excitation signal, generated by an analogue generator [MM12]. The system's response is rectified and smoothed by a low-pass filter. The resulting voltage and the voltage derived from a precision potentiometer, which is linked mechanically to the writing pen, become the inputs to a differential amplifier. The differential amplifier's output controls the writing pen, which is swept over a sheet of paper with the appropriate scale printed on it. The potentiometer may be either linear or logarithmic to produce amplitude or dB readings on the paper. As the paper is moved with constant speed under the writing pen, the frequency scale on the paper is correspondingly logarithmic. This method cannot suppress neither noise or reflections. The ripple in a frequency response caused by a reflection as well as any irregular movement induced by noise can be smoothed by reducing the velocity of the writing pen. In addition, if the spectral details to be revealed are too blurred by the reduced responsiveness of the writing pen, reducing the sweep rate helps to re-establish the desired spectral resolution. The shortcomings are that it does not show phase information and the produced spectra reside on a sheet of paper instead of being written to a hard disk for further processing. Lastly, evidently the accuracy and the precision can't match the ones offered by digital solutions.

Time Delay Spectrometry (TDS) technique TDS is another method concerning sweeps, devised by Heyser [Hey67] especially for the measurement of loudspeakers.

A generator produces both a swept sine and, simultaneously, a phase-locked swept cosine. The sine is fed to the loudspeaker under test and its captured response is multiplied separately by both the original sine, to get the transfer function's real part, and the 90° phase-shifted cosine, to get

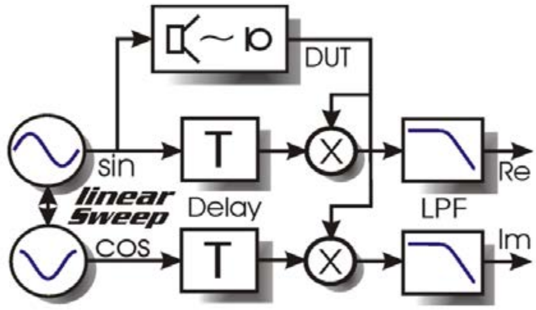

Figure 4.6: *Signal processing steps for TDS.*

the imaginary part. The multiplier outputs are filtered by a low-pass with fixed cut-off frequency. The multipliers produce the sums and differences of the input frequencies. The sum terms of both multiplier outputs must be rejected by the low-pass filters, whereas the difference terms may pass, depending on their frequency. If both the generated and the captured frequencies are almost equal, the output difference frequency will be very low and thus not be attenuated by the low-pass filters. As the sound to the microphone arrives with a delay, its momentary frequency will be lower than the current generator signal. This causes a higher output difference frequency that, depending on the cut-off frequency, will be attenuated by the low-pass filters. For this reason, the generated signal must be time-delayed, by an amount equivalent to the distance between loudspeaker and microphone, before being multiplied with the response. Thus, the difference frequency will be near DC. In contrast, reflections always take a longer way than the direct sound and thus arrive with a lower instantaneous frequency, causing higher frequency components in the multiplier outputs, which will be attenuated by the low-pass filters. With proper selection of the sweep rate and the low-pass cut-off frequency, a quasi-free-field measurements simulation can be performed. In addition, distortion products arrive with a higher instantaneous frequency and thus cause high output frequencies and they will be strongly attenuated by the filters. Likewise, extraneous noise in the wide band above the filter cut-off frequency will be rejected. Having the linear sweep, $df/dt =$ constant, the frequency difference between incoming direct sound and reflection is constant over the whole sweep range, keeping the attenuation of each reflection frequency-independent. Using a logarithmic sweep, the low-pass filters would have to increase their cut-off frequency by a constant factor per time to avoid a narrowing of the equivalent time window. Since the higher frequency components of a typical loudspeaker impulse response decay faster than the lower ones, a narrowing of the window at higher frequencies is desirable. In this way, the SNR at high frequencies can be increased without corrupting the impulse response.

The main drawback associated with this method is that it uses linear sweeps and hence a white excitation spectrum. In most measurement setups, this can lead to poor SNR at low frequencies. If the whole audio range from 20 Hz to 20 kHz is swept through in 1 s, then the subwoofer range up to 100 Hz will only receive energy within 4 ms. This most often is insufficient in a frequency region where the output of a loudspeaker decreases while ambient noise increases. To overcome the poor spectral energy distribution, the sweep must be made very long or the measurement split into two ranges (for example one below and one above 500 Hz). Another problem is the ripple, which occurs at low frequencies. The multipliers produce sum and difference terms of the time-delayed excitation signal and the incoming response. At higher instantaneous frequencies, the sum is sufficiently high to be attenuated by the output low-pass filter. But at the low end of the sweep range, when the sum is close to or lower than the low-pass cut-off frequency, "beating" appears in the recovered magnitude response. A way to remedy is using a very long sweep and reduce the low-pass cut-off frequency by the same factor. A better solution, however, is to repeat the measurement exciting the system with a cosine instead of a sine and add the real part of the complex result of this second measurement to the real part obtained by the previous one, while the imaginary part is subtracted. The effect of this operation is that the sum terms of the output will be cancelled. As a consequence, the absence of the interfering sum terms over the whole sweep range, the low-pass filters following the multiplier stages may be omitted. In fact, they have to be omitted if a full impulse response is to be recovered and the attenuation of reflections by the low-pass filter is not desired. Measurements of room acoustics, which involves acquiring lengthy impulse responses, is only feasible with this method. If, however, a loudspeaker is the object of interest, it is worth keeping the low-pass filters inserted in order to reject reflections, noise and harmonics. Even with the sum-term-cancelling double excitation method, some ripple might still appear at the very beginning and at the end of the sweep frequency range because of the sudden onset of the linear sweep. The switched sine produces a corrugated spectrum near the initial frequency, corresponding to multiplying a continuous time signal with a rectangular window. A common way to solve this problem is to let the excitation sweep start below the lowest frequency of interest. A better possibility would be to formulate the excitation sweep in the spectral domain to create a signal that does not suffer spectral leakage.

Stepped Sine technique This method acquires a transfer function exciting the system step by step with pure tones of increasing frequency. The response can be either analyzed by filtering and rectifying the fundamental, or by performing an FFT and retrieving the fundamental from the spectrum. The latter method requires the use of a sine that is exactly periodic within the bounds of one FFT block-length to avoid spectral leakage. The FFT method allows for the complete suppression of all other frequencies and thus is the preferable method over analysis in the time domain, which involves band-pass filters with restricted selectivity and precision. After each single measurement, the excitation sine's frequency is raised by a value according to the desired spectral resolution.

The spectral resolution in stepped sine measurements is much lower at high frequencies compared to what could be achieved using a broad-band excitation signal with FFT analysis. But this is not necessary a disadvantage as the frequency-linear resolution of FFT-spectra often yields unnecessary fine frequency steps in the high frequency region while sometimes lacking information in the low frequency region, which occurs when the time interval used for the FFT is too small. The biggest advantage of the stepped-sine method is the enormous SNR that can be realized in a single measurement. All energy is concentrated at a single frequency and the feeding sine wave has a low crest factor of only 3 dB. The measurement certainty and repeatability for a single frequency can be very high compared to broad-band excitation. Thus, despite of the considerable amount of time needed for the complete evaluation of a transfer function, this method offers precision measurement. A disadvantage is that gating out reflections is only possible when the difference in the time-of-flight between direct sound and reflection is longer than the analysis interval.

Impulse technique Using an impulse as excitation signal an impulse response can be directly obtained. It is the most straightforward approach to performing FFT-based transfer function measurements. The captured response, after feeding the system, already is the desired impulse response. To increase the SNR, the pulse can be repeated periodically and the responses of each period added, leading in a periodic response. The synchronous averaging leads to a reduction in uncorrelated noise by 3 dB for each doubling of the number of averages.

The response may optionally be shifted to the left to compensate for the delay introduced by the propagation time between loudspeaker and microphone. Muting with the windowing the unwanted reflections can increases the SNR and multiplying the result by a reference spectrum

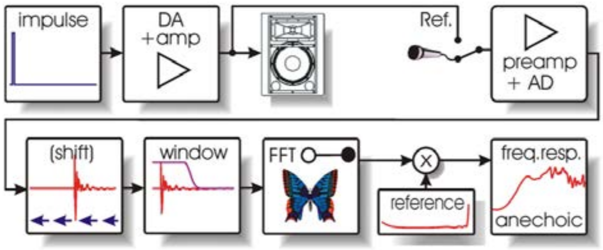

Figure 4.7: *Signal processing steps for measurement with impulses.*

increases the precision of the measurement considerably. This reference spectrum is obtained by linking the output and the input of the measurement system and inverting the measured transfer function. Applying this technique, independently of the kind of excitation signal, pre-emphasizes the excitation signal to adapt it to the spectral contribution of background noise. This pre-emphasis is automatically removed from the resulting transfer function by applying the obtained reference spectrum in all subsequent measurements.

Impulses are a simple choice when the measurement is purely electrical and there is no acoustic path in the measurement chain. However, they require a low noise floor and usually the method offers SNR performance far from optimal. In addition, impulses can even be useful in acoustics. In an anechoic chamber where ambient noise is typically very low at high frequencies, tweeters can be measured with reasonable SNR. Because of their short duration, pulses can be fed with very high voltage without the risk of overheating the voice coil. Care must be taken, however, not to cause excursion into the non-linear range of the speaker. In general, all distortion in a pulse measurement occurs simultaneously with the impulse response and, hence, cannot be separated from it. To increase the SNR of a tweeter measurement, the impulses can be repeated and averaged. Lastly, impulse testing is pretty immune to the detrimental effects of time variance.

Maximum length sequences (MLS) technique MLS technique were first proposed by Schroeder in 1979 [Sch79]. It is based on a periodic pseudo-random signal having almost the same stochastic properties as a pure white noise. MLS signals are binary sequences that can be generated very easily with an N-staged shift register and an XOR-gate connected with feedback in such a way that all possible 2^N states, minus the case "all 0", are run through. For all orders of shift registers, a bipolar excitation signal of 1 corresponds to a positive amplitude $+U_0$ and of 0 to a negative value $-U_0$.

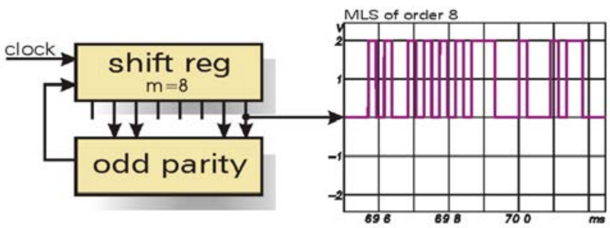

Figure 4.8: *Generation of MLS signal, with shift register fed back over odd parity generator.*

As the case "all 0" is excluded from the sequence, the length of an MLS signal is $2^N - 1$. MLS signals have some unique properties that make them suited for transfer function measurements. Their auto-correlation comes close to a Dirac pulse, indicating a white spectrum. Repeated periodically as a pulse train, all frequency components have indeed exactly the same amplitude, meaning their spectrum is perfectly white. Compared to a pulse of same amplitude, much more energy can be fed to the system as the excitation signal is stretched out over the whole measurement period and this increases SNR. A MLS signal is not output as a pulse train, it is usually kept constant between two clock pulses. This first-order hold function leads to a $sinc(x)$ aperture loss, which reaches almost 4 dB at $f_S/2$ and therefore must be compensated. In contrast, when the MLS is output by an oversampling audio DA converter the spectrum will be flat up to the digital filter's cut-off frequency. Excitation signals with white spectrum allow the use of the cross-correlation. The periodic impulse response obtained in the record $h'[n]$ is related to the linear impulse response by the equation

$$h'[n] = \sum_{l=-\infty}^{+\infty} h[n + lL]. \tag{4.9}$$

Equation (4.9) reflects the problem of the time-aliasing error. This error is significant if the length L of one period is shorter than the length of the impulse response to be measured. The order m of the sequence must be high enough in order to overcome this error. The phase spectrum of the sequence is strongly erratic, with a uniform density of probability in the $[-\pi, +\pi]$ interval. Because this property, this technique is able to randomize the phase spectrum of any component of the output signal which is not correlated with the MLS input sequence. This implies any disturbing signal, such white or impulsive noise, leading to a uniform repartition of the disturbing effects along the deconvolved impulse response, instead of localized noise contributions along the time axis. A post-averaging method can be used in order to reduce this phenomenon.

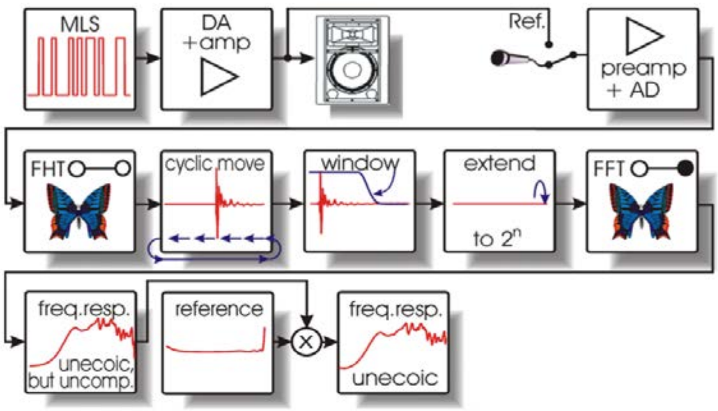

Figure 4.9: *Signal processing steps for measurement with MLS signals.*

While normally a cross correlation is most efficiently performed in the spectral domain by complex-conjugate multiplication, the fast Hadamard transform (FHT) [Alr83] can perform this task without leaving the time domain. The butterfly algorithm employed in the FHT only uses additions and subtractions and can operate on the integer data delivered by the AD converter. This means calculation times much shorter than that of an FFT of similar length. However, the processing times are no longer of concern, as the transformations with today's more powerful processors can be performed much faster than real time. In an MLS based measurement, the FHT is the first signal-processing step after digitization by the AD converter. The resulting impulse response can be shifted in a cyclic fashion and windowed. If the transfer function is the objective, an additional FFT must be performed. But as MLSs have a length of $2^N - 1$, one sample must be inserted in order to have 2^N length. It must be placed in a region where the impulse response has decayed to near zero to avoid errors. Using a window, the sample can be placed in the muted area. The acquired transfer function should be corrected by multiplication with a reference spectrum obtained previously by a self-response measurement.

MLS measurements have several drawbacks. The main problem of the MLS method is caused by the appearance of distortion artefacts. These artefacts are uniformly distributed along the deconvolved impulse response. The origin of the distortion is in the non linearities inherent to the measurement system and especially to the loudspeaker. It suffer also of a high vulnerability to time variance. Another undesired property of MLSs is their white spectrum, a non-white spectrum is desirable for almost all acoustical measurements. This requirement can be achieved by coloring the MLS with an appropriate emphasis, losing its binary character by pre-filtering. Creating an emphasized MLS can be done most efficiently by means of the inverse fast Hadamard transform (IFHT) [MM95]. The IFHT consists of time inverting the impulse response of the desired emphasis filter, applying a normal FHT on the inverted impulse response and then time inverting the result again. This yield an MLS periodically convolved with the emphasis filter. Due to the periodicity, every discrete frequency component of the former MLS can be influenced independently in both amplitude and phase. When an emphasized instead of a pure MLS signal is used as stimulus, the response obtained by FHT consists of the impulse response of the system convolved with the impulse response of the emphasis filter. For acoustical measurements, it is meaningful to give to the excitation signal a strong bass boost (20 or 30 dB), making the recovered impulse response much broader than the one of the system alone. This broadening often constraints windowing, especially when unwanted reflections are in close proximity of the main peak. In these cases, applying a window to the non-equalized impulse response attenuates the low frequency energy spread out in time. Thus, it is better to perform the windowing after transforming the uncorrected impulse response into the spectral domain, then multiplying it with the inverse emphasis frequency response, and eventually back-transforming it into the time domain. This yields the impulse response of the system alone, which can be windowed with lesser low frequency energy loss.

In conclusion, a different signal, called Inverse Repeated Sequence (IRS), was proposed as an alternative to the MLS technique allowing a theoretical reduction of the distortion artefacts [Rea70]. The IRS $x[n]$ of samples period $2L$ is defined from the corresponding MLS signal $mls[n]$ of period L by

$$x[n] = \begin{cases} mls[n], & \text{if } n \text{ is even, } 0 \leq n < 2L \\ -mls[n], & \text{if } n \text{ is odd, } 0 < n < 2L \end{cases} \tag{4.10}$$

Being twice the length of the corresponding MLS sequence it improves also the SNR. The deconvolution process used with this signal is still a circular cross-correlation.

Logarithmic SineSweep technique The logarithmic SineSweep technique overcomes most of the limitations encountered in the other techniques. It uses an exponential time-growing frequency sweep being able to deconvolve the linear impulse response and to separate each impulse response corresponding to the harmonic distortions. In this way, a impulse response exempting from any non linearity is obtained. In addition, the measurement of the harmonic distortion at various orders can be performed. The deconvolution process is realized by linear convolution of the measured output with the inverse filter preprocessed from the excitation signal. It avoids time-aliasing problems. If the time analysis window has the same length as the emitted signal and is shorter than the impulse response measured, the tail of the system response may be lost, but this will not introduce time aliasing. A silence of sufficient duration is added at the end of the sweep signal in order to recover the tail. The inverse filter $f(t)$ is generated by the temporally reversing of the logarithmic sweep. Then it is delayed in order to obtain a causal signal, since the reversed signal is in the negative region of the time axis. Because the time reversal causes a sign inversion in the phase spectrum, the convolution with the input signal results in a linear phase signal. Finally, since this operation introduces a squaring of the magnitude spectrum, it is divided by the square of the magnitude spectrum of the initial signal. In addition, in order to minimize the influence of the transients, appearing at the beginning and at the end of the emission of the excitation signal, introduced by the measurement system, the ends of the inverse sweep are exponentially attenuated with a exponential growth at the beginning and with a exponential decrease at the end. If the aim is to perform acoustical measurements, the signal must extend all the audible range. In practice, a sweep with initial sweep frequency $f_1 = 10$ Hz and final sweep frequency $f_2 = 22$ kHz is used, in the way to include out of this range the transients. A deeper description of this method will be done in the last section of this chapter.

Considerations

Some final considerations connected with the properties of the different stimulus and method will be now proposed.

Measurement Duration In the case of a non-periodic pulse, to avoid errors, the capture period must to be at least as long as the impulse response target. With non-periodic sweep must be a little bit longer at the low frequencies. Since the largest signal delays occur there, while sweeping through the high frequencies there should be sufficient time to catch the delayed low-frequency components. In the case of periodic excitation, the period length and the capture time must be no longer than the impulse response length. Using a shorter length would lead to time-aliasing, folding back the tail of the response that crosses the end of the period and adding it to the beginning. In addition, in room acoustic measurements the gap of silence following the emission of the sweep usually must be as long as the reverberation at the highest frequencies.

Crest factor The crest factor, as introduced before, is the relation of peak to RMS voltage of a signal. If either the measurement system or the system under analysis is limited by a distinct voltage level, the peak value of any considered excitation signal must be normalized to this value in order to extract the maximum possible energy in a measurement. The RMS level will then be lower according to the crest factor. Thus, the crest factor indicates how much energy is lost when employing a certain excitation signal, compared to the ideal case of a stimulus whose RMS voltage equals the peak value (crest = 0 dB).

Noise Rejection The difference between the various methods lies in the way that the noise is distributed over the period of the recovered impulse response. Using the same spectral distribution in an excitation signal requires the same inverted coloration in the deconvolution process. That is why the magnitude of an interfering noise source does not vary when changing the stimulus type. The phases, however, turn out to be very different. Some kinds of noise sources will appear similarly in all measurements, as their general character is not altered by manipulating the phases. Monofrequency noise, such as hum, is an example. Likewise, uncorrelated noise, for instance, air conditioning, still appear as noise. Any other disturbance, however, is reproduced quite differently, depending on the type of stimulus. Short, impulsive noise sources, such as clicks and pops, are transformed into noise when using noise as a stimulus.

Time Variance Periodic noise sequences in general and MLS in particular are extremely vulnerable to even slight time variances [SN99]. A noise sequence and a linear sweep, both with white flat spectrum, result in disturbance of the base band spectra negligible at low frequencies, but then increases dramatically for the jittered noise spectrum. In contrast, the jittered sweep spectrum only displays a minor corrugation at the high end that can be removed by applying gentle smoothing.

Distortion In any measurement using noise as excitation signal, the distortion products are distributed as noise over the entire period of the impulse response. The reason is that the distortion products of a stimulus with pseudo-random phases also have more or less random phases, and the deconvolution process again involves random phases that produces an error spectrum with random phases, corresponding to a randomly distributed noise signal. As this error signal is correlated with the excitation signal, synchronous averaging does not improve the situation. Increasing the length reduce the noise level but increase the vulnerability to time variance, becoming predominant. Using sweeps they can be excluded totally from the recovered impulse response.

Equalizing loudspeakers for room acoustical measurements Examining the obtained transfer function can reveal disturbing room modes or tonal misbalance of a sound reinforcement system. When using a loudspeaker without any further precautions, the acquired room transfer function is colored by the loudspeaker's frequency response. Coloration by the loudspeaker's frequency response is highly undesirable for auralization purposes. In these cases, it is necessary to use a pre-emphasis to remove this coloration. This equalization could also be done by post-processing the transfer function with the inverse of the speaker's response, but this would not improve the poor SNR at frequency regions where the acoustical output of the loudspeaker is low. That is why it is advantageous to pre-filter the excitation signal in order to allow for a frequency-independent power output.

Periodicity of Signals Using a periodic signal of length 2^N samples the system response can be transformed directly to the spectral domain, omitting the FHT. The FHT, being a cross-correlation algorithm, is able to merely reshuffle only the phases of MLS signal. In contrast, the FFT approach compensates the phase and the magnitude of any excitation signal. The only restriction is that the excitation signal must have enough signal energy over the whole frequency range of interest to avoid noisy parts in the transfer function obtained. Deterministic signal in contrast to the uncorrelated noise sources, normally used in dual-channel FFT analyzers, performs using a single channel FFT analyzer avoiding any problems given by differences in the frequency response of the two input channels that are reflected in the system's measured frequency response. In addition, a dual-channel analyzer must always average over many single measurements before being able to present a reliable result. Deterministic stimulus can be custom tailored by defining an arbitrary magnitude spectrum adapted to the prevailing noise floor, to accomplish a frequency-independent SNR.

For several reasons, sweeps are a better choice for measurements than noise sequences. First, the spectrum of a non-repeated single sweep is almost identical to that of its periodic repetition. This means that it is not necessary to emit the excitation signal twice to establish the expected spectrum. Thus the measurement duration is cut in half, maintaining the same spectral resolution and SNR as in a measurement with the stimulus periodically repeated. The other enormous advantage of a sweep measurement is the fact that the harmonic distortion components can be isolated. In contrast, measurements using noise as stimulus lead to the distribution of the distortion products over the whole period. In order to actually place the distortion products at negative times of the acquired impulse response, a linear deconvolution suited for non-periodic signals would be necessary. In order to capture better impulse responses, the sweeps used must be longer than the impulse response. In this way SNR increases and the influence of time variance decreases.

Lastly, there is an important difference concerning the noise floor in the responses obtained by linear and circular deconvolution. A circular deconvolution results in a noise floor which is basically constant in both amplitude and frequency distribution, up to the point where the first distortion products appear. The linear deconvolution, however, yields a decaying noise tail which is increasingly low-pass filtered towards its end. This stems from the fact that this last part of the deconvolution result originates from steady noise convoluted with a sweep in reverse order. The data-acquiring period in non-periodic measurements must be made sufficiently large so as to capture all delayed components. This means that the sweep always must be shorter than the capturing period and the subsequent FFT length.

Sweep Synthesis Sweeps can be created either directly in the time domain or indirectly in the frequency domain. In the latter case, their magnitude and group delay are synthesized and the sweep is obtained by IFFT. The two most commonly known types are the linear and the logarithmic sweep. The linear sweep has a white spectrum and increases with fixed rate per time unit and therefore

$$\frac{f_2 - f_1}{T_2 - T_1} = const \tag{4.11}$$

where f_2, T_2 denote the final frequency and time values and f_1, T_1 the initial ones. The logarithmic sweep has, instead, a pink spectrum and hence, its amplitude decreases with 3 dB/octave. This means that every octave contains the same energy. The frequency increases with a fixed fraction of an octave per time unit. It can be expressed by

$$\frac{\log(f_2/f_1)}{T_2 - T_1} = const \tag{4.12}$$

While sweeps generated in the time domain have a perfect envelope and thus the same ideal crest factor as a sine wave (3 dB), their spectrum is not exactly what is expected. The sudden switch-on at the beginning and switch-off at the end create unwanted ripple at the extremities of the excitation spectrum. Normally, these irregularities have no effect on the recovered frequency response when correcting them with a reference spectrum derived by inversion of the excitation one. If, however, the deconvolution is done with the time-inverted and amplitude-shaped stimulus, as proposed in [Far00], the errors can be expected near the start and end frequency of the sweep.

Constructing the sweep in the spectral domain avoids these problems. The synthesis can be done by defining the magnitude and the group delay of an FFT-spectrum, calculating real and imaginary parts from them, and finally transforming the sweep spectrum into the time domain by IFFT. The associated group delay for the linear sweep can be set by:

$$\tau_G(f) = \tau_G(0) + fk, \tag{4.13}$$

where

$$k = \frac{\tau_G(f_S/2) - \tau_G(0)}{N}. \tag{4.14}$$

The group delay of a logarithmic sweep is given by:

$$\tau_G(f) = A + B\log_2(f), \tag{4.15}$$

where B is given by

$$B = \frac{\tau_G(f_{final}) - \tau_G(f_{initial})}{\log_2(f_{final}/f_{initial})} \tag{4.16}$$

and A by

$$A = \tau_G(f_{initial}) - B\log_2(f_{initial}). \tag{4.17}$$

Synthesizing a sweep in the spectral domain causes some oddities in the resulting time signal. It is important that the resulted phase reaches exactly 0° or 180° at $f_s/2$. Even satisfying this condition, the sweep is not confined exactly to the values given by $\tau_G(f_{initial})$ and $\tau_G(f_{final})$, but spread out further in both directions. This is a consequence of the desired magnitude spectrum in which the oscillations that would occur with abrupt sweep start and stop points are precisely not present. Because of the broadening, the group delay for the lowest frequency bin should not be set to zero, but instead be a little higher. In this way, the sweep's first half-wave has more time to evolve. However, it always starts with a value greater than zero, while the remaining part left of the starting point folds back to negative times at the end of the period. It can "smear" into the high-frequency tail of the sweep if the group delay chosen for $f_s/2$ is too close to the length of the FFT time interval. A way to avoid contaminating the late decay of the tail by low-frequency components is to choose an FFT block length that is at least double the desired sweep length. To force the sweep's desired start and end point to zero, fading operations are a solution to avoid switching noise. A deviation from the desired magnitude spectrum occurs but it can be kept insignificant by choosing sufficiently narrow parts at the very beginning and end of the sweep. The ripple introduced by the fading operations (half-cosine windows are most suitable) can easily be kept under 0.1 dB and should not cause any concern, because it is cancelled by performing and applying the reference measurement.

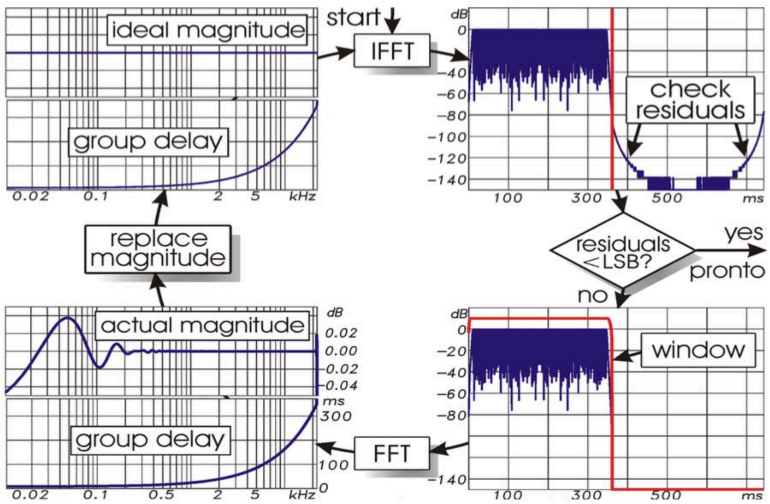

Figure 4.10: *Iterative method to construct broad-band sweeps with perfect magnitude response.*

The iterative method, shown in Fig. 4.10, permits to reject the ripple completely, establishing exactly the desired magnitude spectrum while maintaining the sweep confined to the desired length. The iteration consists of consecutively performing the fade in/out-operation, transforming the time signal to the spectral domain, replacing the corrugated magnitude spectrum with the target magnitude while maintaining the phases, and eventually back-transforming the resulted spectrum into the time domain. Before imposing the fade-in/out windows another time, the residuals outside the sweep interval are examined. If their peak value is below the LSB of the sweep's intended final quantization, the windowing is omitted and the iteration ends. Usually, the process converges rapidly. However, the perfect magnitude response is traded off by a very light distortion of the phase spectrum.

4.6 Farina's Method

Farina's method [Far07] employs sine sweeps and allows improvements dealing with the problem of distortion and non linear time variant systems. It is based on an exponential sweep test signal with aperiodic deconvolution and achieved a good noise rejection, not-linear effects perfect separation and avoids any trouble in case the system has some time variance. The mathematical definition of the test signal is

$$x(t) = \sin\left[\frac{\omega_1 T}{\ln\left(\frac{\omega_2}{\omega_1}\right)}\left(e^{\frac{t}{T}\ln\left(\frac{\omega_2}{\omega_1}\right)} - 1\right)\right], \tag{4.18}$$

where ω_1 and ω_2 are the initial and final radian frequency of the sweep of duration T. This signal has constant amplitude and some seconds of silence are added to it. Playing through the loudspeaker, the recorded response exhibit the effects of the space, of the noise and of the not-linear distortion. The distorted harmonic components appear as straight lines, above the main line, which corresponds with the linear response of the system. In order to extract the linear system's impulse response $h(t)$, the output signal $y(t)$ must be filtered with a proper filter $f(t)$, defined mathematically by

$$h(t) = y(t) \otimes f(t). \tag{4.19}$$

In order to avoid that the resulting impulse response folds back from the end to the beginning of the time frame, which would cause the harmonic distortion products to contaminate the linear response, the convolution aperiodically is implemented and the Time Reversal Mirror approach is used to create the inverse filter $f(t)$. In this way $f(t)$ is the time-reversal of the test signal $x(t)$. The sweep signal has not a flat spectrum, due to the fact that the instantaneous frequency sweeps slowly at low frequencies and much faster at high frequencies. For this reason the resulting spectrum is pink and falls down by -3 dB/octave. Taking in account that, the inverse filter compensates for this an amplitude modulation is applied to the reversed sweep signal having its amplitude increasing by $+3$ dB/octave. Convolving the output signal $y(t)$ with the inverse filter $f(t)$, the linear response is an almost perfect impulse response, with a delay equal to the length of the test signal. The harmonic distortion responses appear at precise time delay, occurring earlier than the linear response. At this point, applying a time window it is possible to extract the portion desired, containing only the linear response and discarding the distortion products.

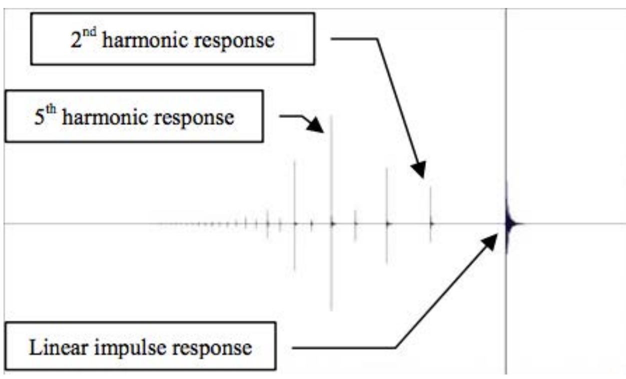

Figure 4.11: *Output signal $y(t)$ convolved with the inverse filter $f(t)$.*

As discussed above, the output signal can be written as the sum of the generated noise and a deterministic function of the input signal,

$$y(t) = n(t) + F[x(t)]. \tag{4.20}$$

Assuming the system linear and time-invariant, the function F has the form of the convolution between the input signal and the system's impulse response $h(t)$,

$$y(t) = n(t) + x(t) \otimes h(t). \tag{4.21}$$

Releasing the linear constraint for the system, non-linearities must to be take in account. Typically the distortion occurs in the electro-mechanical transducer, it means that as the sound is radiated into air, it passes through a linear propagation process. The non-linearities of the system happen usually at beginning and are memoryless. At this point, the input signal is assumed passing through a memoryless not linear device, characterized by a N^{th} order Volterra kernel $k_N(t)$, and the result of such a distortion process, $w(t)$, is reverberated through the linear filter $h'(t)$. A memory-less harmonic distortion process can be represented by

$$w(t) = x(t) \otimes k_1(t) + x^2(t) \otimes k_2(t) + \ldots + x^N(t) \otimes k_N(t). \tag{4.22}$$

As the convolution of $w(t)$ with the following linear process $h'(t)$ possesses the distributive property, the measured output signal can be represented as

$$y(t) = n(t) + x(t) \otimes k_1(t) \otimes h'(t) + x^2(t) \otimes k_2(t) \otimes h'(t) + \ldots + x^N(t) \otimes k_N(t) \otimes h'(t). \tag{4.23}$$

Considering the deterministic part of the transfer function is described by a set of impulse responses $h_i(t)$, each of them being convolved by a different power of the input signal, Eq. (4.23) can be written as

$$y(t) = n(t) + x(t) \otimes h_1(t) + x^2(t) \otimes h_2(t) + \ldots + x^N(t) \otimes h_N(t), \tag{4.24}$$

where $h_i(t) = k_i(t) \otimes h(t)$. As already discussed, a common practice for measuring the transfer function in linear and time invariant systems, is to apply a known signal as input, and to measure the system's response. The SNR is improved by taking multiple synchronous averages of the output signal. Calling $\hat{y}(t)$ the averaged output signal and since a circular convolution process relates the periodic input and output, in order to deconvolving $h(t)$, FFTs and IFFTs transforms can be employed:

$$h(t) = \mathrm{IFFT}\left[\frac{\mathrm{FFT}(\hat{y}(t))}{\mathrm{FFT}(x(t))}\right]. \tag{4.25}$$

Due to the continuous repetition of the test signal and the fact that a circular deconvolution is performed, there is the risk of the time aliasing error. This happens if the period of the repeated input signal is shorter than the duration of the system's impulse response $h(t)$. With sine sweeps is easy to add a segment of silence after the signal, for avoiding this problem. Not-linear behaviour of the system appears, at various positions of the deconvolved impulse response, as that they resemble scaled-down copies of the principal impulse response. Using sine sweeps in which the instantaneous frequency is made to vary linearly with time, the distortion products cause a sort of noise to appear everywhere in the deconvolved $h(t)$. This noise is correlated with the input signal and it does not disappear by averaging. Using, instead, a sine sweep generated with instantaneous frequency varying exponentially with time, a logarithmic sweep, the spurious distortion peaks appear with their typical impulsive sound. In order to remove these unwanted distortion peaks, a linear deconvolution substituted the circular deconvolution, directly implemented in the time domain. A proper inverse filter $f(t)$ can be generated in order to be able to transform the input signal $x(t)$ into a delayed Dirac's delta function $\delta(t)$. The deconvolution of the system's impulse response can then be obtained convolving the measured output signal $y(t)$ with the inverse filter $f(t)$. With this approach, any distortion products caused by harmonics produce output signals at frequencies higher than the instantaneous input frequency.

The inverse filter is the input signal itself, reversed along the time axis, hence, the instantaneous frequency decrease with time. In the case of exponentially-swept sine, an amplitude modulation is added, for compensating the different energy generated at low and high frequencies. The inverse filter has the effect to delay the signal which is convolved with and with an amount of time which varies with frequency. This delay is linearly proportional to frequency for linear sweeps, and proportional to the logarithm of frequency for the logarithmic sweep. The harmonic distortions appear as straight lines, parallel to the linear response in the case of the log sweep and with a increasing slope in the case of linear sweep. The linear deconvolution, instead of the circular one, thus enables the measurement of the system's linear impulse response also if the loudspeaker is working in a not-linear region. The log sweep has the only advantage over the linear sweep of producing a better SNR ratio at low frequencies. In the case of the log sweep, if it is slow enough, each harmonic distortion appear into a separate impulse response, without overlap with the preceding one. It is possible thus to window out each of them. Each of these impulse responses corresponds exactly to the rows of the Volterra kernel, and thus to the terms $h_n(t)$ of Eq. (4.24).

Lastly, considering a not-time-invariant system, the impulse responses $h_N(t)$ change slowly in time. The variation is usually slow enough for avoiding audible effects such as tremolo or other form of modulation and in most cases they can be neglected and there are not significant differences in the objective acoustical parameters or in the subjective effects. The continuous variation affects the measurements, as it pones problems removing the unwanted extraneous noise $n(t)$. Avoiding the technique of multiple averages but taking a logarithmic sinesweep very long, produces a distortion-free linear response with separated harmonic distortion up to very high orders. The estimated response is, in this way, not affected by the time variation, as a single measure is taken. In addition, a lot of energy is diluted over a long time and then packed back to a short response obtaining usually a SNR improvement of 60 dB or more in comparison with the generation of a single impulse having the same maximum amplitude [Far00].

Problems with Farina's Method

This technique, as seen, presents important improvements in the impulse response measurements, in comparison with all the other previously employed methods. However, despite the significant advantages, it is not free from problems and in particular it can suffer from

- pre-ringing at low frequency before the arrival of the direct sound pulse
- sensitivity to abrupt impulsive noises during the measurement
- skewing of the measured impulse response when the playback and recording digital clocks were mismatched
- cancellation of the high frequencies in the late part of the tail when performing synchronous averaging
- time-smearing of the impulse response when amplitude-based pre-equalization of the test signal was employed

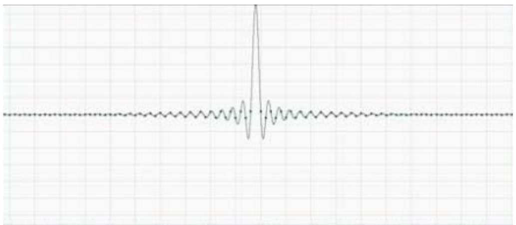

Figure 4.12: *Pre-ringing artifact with fade-out.*

Figure 4.13: *Pre-ringing artifact without fade-out.*

Pre-ringing

The obtained impulse response often is affected by some significant pre-ringing before the arrival of the direct sound. As shown in Fig. 4.12, the peak has a Sync function shape and it shows a number of damped oscillations both before and after the main peak. This is due to the limited bandwidth of the signal and to the presence of some fade-in and fade-out on the envelope of the test signal. These two factors define a trapezoidal window in the frequency-domain, which becomes the Sync-like function in time domain.

Removing the fade-out the effect is attenuated. Fig. 4.13 shows this improvement. The waveform still looks like a Sync function but it is very close to a theoretical Dirac's Delta function and no pre-ringing or post-ringing are anymore significantly present. However, since the final value computed could be not-zero with the consequence of spreading a lot of energy all along the spectrum because the excitation of a step function, it's better not remove completely the fade-out. An alternative solution can be to continue the sweep up to the Nyquist frequency and cutting it manually at the latest zero-crossing before its abrupt termination. In this way no impulsive sound is generated at the end, and the full-bandwidth of the sweep removes almost completely the high-frequency pre-ringing. In some cases, also low frequencies can cause a significant pre-ringing, also without fade-out in the envelope. The way of controlling this type of pre-ringing, due to the analog equipment, is to create a proper time packing filter, and to apply it to the measured impulse response.

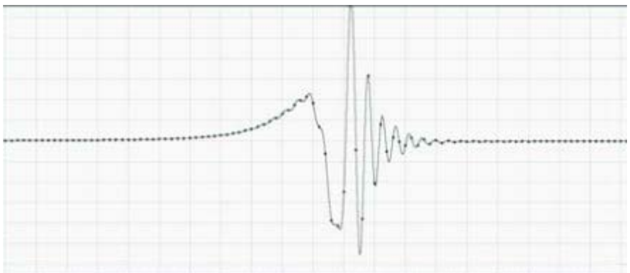

Figure 4.14: *Low-frequency pre-ringing artefact.*

Fig. 4.14 shows the case of low-frequency pre-ringing. A packing filter is a filter capable of compacting the time-signature of the impulse response and removing the effect of power amplifier, loudspeaker and microphones. An example method is using the Kirkeby algorithm [KNH98]:

1. The FFT transform of the impulse response is computed:

$$H(f) = \mathrm{FFT}[h(t)] \tag{4.26}$$

2. The inverse filter in frequency domain, having the form

$$C(f) = \frac{Conj[H(f)]}{Conj[H(f)]H(f) + \epsilon(f)}, \tag{4.27}$$

is computed, where $\epsilon(f)$ is the regularization parameter, which can be frequency-dependent. Usually chosen with a very small value inside the frequency range covered by the sine sweep, and a much larger value outside.

3. Finally, using the IFFT the inverse filter is found:

$$c(t) = \mathrm{IFFT}[C(f)] \tag{4.28}$$

In conclusion, pre-ringing artefacts can be avoided by combining the usage of a wide-band sweep running up to the Nyquist frequency, without any fade-out, and the usage of a “compacting” inverse filter, computed with the Kirkeby method.

Equalization of the equipment

When the goal of the measurement is to analyze the acoustical transfer function between an ideal sound source and an ideal receiver, the effect of the electroacoustical devices should be removed. In this case, taking a ”reference” measurement, such a complete anechoic measurement, the Kirkeby inverse filter can be built to remove any time-domain and frequency-domain artifact caused by the whole measurement system. Although the inverse filter doesn’t provide a perfect result, it provide a transfer function to closely approach the ideal one. Hence, the electroacoustical sound system can be employed for measurements without any significant biasing effect.

A question can be if it is better to apply this equalizing filter to the test signal before playing it through the system, or to the recorded signal. Both approaches have some advantages and disadvantages. Applying the filter to the test signal usually results in a weaker test signals being radiated by the loudspeaker. It can also be in clipping at extreme frequencies, where the boost provided by the equalizing filter is greater. On the other hand, the filter after the measurement results in “colouring” the spectrum of the background noise, which can, in some case, become audible and disturbing. In practice, often the better strategy revealed to be hybrid, thus, the test signal is first roughly equalized, then, a reference anechoic measurement is performed, employing the pre-equalized test signal, and the inverse filter computed.

This inverse filter is applied as a post-filter, to the measured data, ensuring that the total transfer function of the measurement system is made perfectly flat. The first equalization limits the boost at extreme frequencies and the gain loss at medium ones, providing an already almost flat radiated sound. The latter equalization, instead, removes the residual colouring of the measurement system.

Impulsive Noises During the Measurement

A drawback using long sweeps for improving the SNR is that some impulsive noise can occur during the measurement. Sources of impulsive noise can be objects falling on the floor, movements of seats or “cracks” caused by steps over wooden floors. After convolution with the inverse filter, this impulsive event causes an artefact on the deconvolved impulse response. Figs. 4.15 and 4.16 show a impulsive noise in the measurement and its effect in the computed response.
The artefact appears as a sort of frequency-decreasing sweep, starting before the beginning of the linear impulse response and continuing after it. The part of this spurious sweep occurring in the late part of the measurement causes problems since it cannot be deleted with windowing. Solutions can be silencing the recording in correspondence of the impulsive event or using a Click/Pop Eliminator

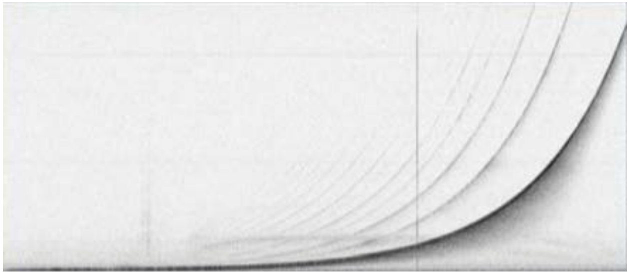

Figure 4.15: *Sonogram showing a pulsive event (the vertical line) contaminating a measurement.*

algorithm. In this way the artefact can be reduced but it can be non enough. A better way can be finding the frequency of the sine sweep at the moment in which the impulsive occurred. Using a narrow-passband filter at that frequency the wide-band noise can be removed, restoring a clean sinusoidal waveform. Fig. 4.17 shows the result of this latter solution. As it can be seen in the figure, the noise is been strongly reduced.

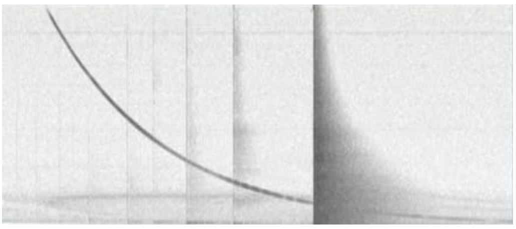

Figure 4.16: *Sonogram showing an artifact caused by a pulsive event.*

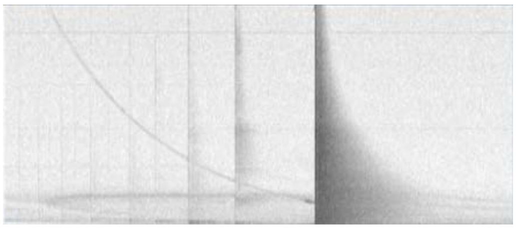

Figure 4.17: *Sonogram showing the obtained IR after removed the pulsive event by filtering.*

Clock Mismatch

With this method the synchronization between the playback clock and the recording clock is not required. However, when the mismatch between the two clocks is significant, the deconvolved impulse response starts to be "skewed" in the frequency-time plane.

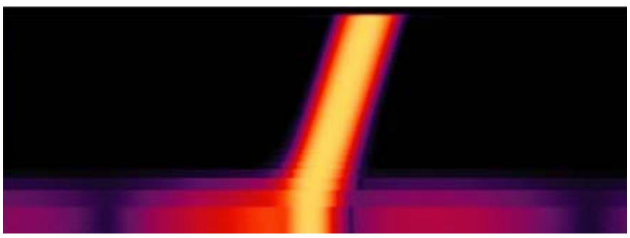

Figure 4.18: *Sonograph of a "skewed" IR caused by clock mismatch (in logarithmic frequency scale).*

Fig. 4.18 shows an example of "skewed" impulse response. The waveform shows that low frequencies are starting earlier than high frequencies. Can be seen clearly that the response does not has a vertical (synchronous) appearance, but a sloped (skewed) one. A method for re-aligning the clocks is using a "reference" measurement and a Kirkeby inverse filter for fixing the mismatch. Fig. 4.19 shows the result using this solution. It shows a quite good result, correcting the magnitude of the frequency response of the system and the frequency-dependent delay. However, this approach requires the availability of a clean reference measurement.

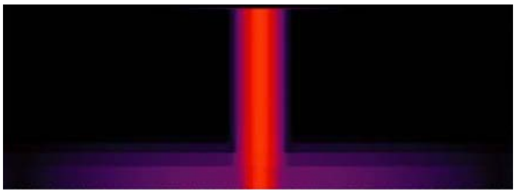

Figure 4.19: *Sonograph of the correction of a "skewed" IR employing a Kirkeby inverse filter.*

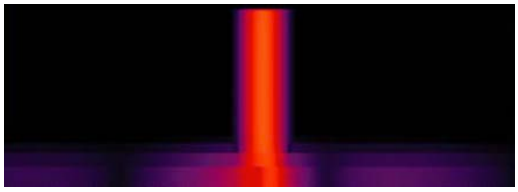

Figure 4.20: *Sonograph of the correction of a "skewed" IR employing deconvolution with a longer inverse sweep.*

Another possible solution is using a pre-strecthed inverse filter for performing the impulse response deconvolution, generating a new sine sweep longer than the original one. For example, if the skewness is 4 ms long, the new sine sweep will be 4 ms longer than the original one. In this way, the inverse sweep filter slightly longer is computed and convolved with the recorded signal. This result is shown in Fig. 4.20 and, as it can be see, it's not clean as the one obtained with the Kirkeby inversion. However, a quite good clock re-alignment without the need of a reference measurement can be obtained.

Time Averaging

In the case of exponential sine sweep, synchronous time averaging works only if the whole system is perfectly time-invariant, never the case when the system involves propagation of the sound in air. It is because the air movement and change of the air temperature. For improving SNR, thus, instead to average a number of distinct measurements is better perform a single very long sweep measurement. However, in some cases, the usage of long sweeps is not allowed, for example for computation and memory reason. In these cases time-synchronous averaging is the only way for getting results in a noisy environment. Even a very slight time-variance of the system can produce significant artefacts in the late part of the impulse response and at higher frequencies. This happens because the sound arriving after a longer path is more subject to the variability of the time-of flight due to unstable atmospheric conditions. Furthermore, a given differential time delay translates in a phase error which increases with frequency [Far07]. Fig. 4.21 compares the sonographs of two impulse response, the first comes from a single and long sweep of 50 s, the second from the average of a series of 50 short sweeps of 1 s each. As can be noted, in the second measurement the energy of the reverberant tail is at high frequency significantly underestimated.

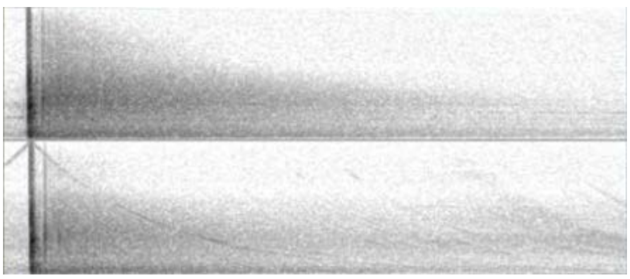

Figure 4.21: *Sonograph of a single sweep of* 50 *s (above) versus* 50 *sweeps of* 1 *s (below).*

Chapter 5

Acoustic Tube Delay Measurement

In order to perform the measurements, a computer connected to a full duplex audio interface, able to record from a channel while it is reproducing the sound in another one, an amplifier, a loudspeaker and a microphone were used. The software MATLAB® was used to play the excitation sound though the audio interface and in turn to the loudspeaker, receive the signal recorded by the microphone connected in the audio interface input and elaborate it, performing all the signal processing needed, in order to obtain the impulse response. Figure 5.1 shows the diagram flow of the measurement setup and summarizes this process.

This chapter will present the measurements performed on the tube, giving the information on the used equipment and on the modalities of the experiments. Finally, the obtained tube response will be analyzed and discussed.

5.1 Equipment

As presented in the beginning of this work, garden hoses was chosen for perform the measurements and analysis. Garden hoses are typically made of extruded synthetic rubber or soft plastic, often reinforced with an internal web of fibres. As a result of these materials, garden hoses are flexible but with walls enough hard, allowing a good sound absorption and a constant cross-section S. Summarizing, the used equipment were:

- Motu-UltraLite-mk3 Hybrid audio interface[1]
- Yamaha amplifier
- Miniature microphone
- Custom full range loudspeaker
- Three PVC (polyvinyl chloride, $1,40 - 1,45$ g/cm^3) garden hoses
- MATLAB®

Three different tubes were used with an internal diameter of 1.2, 1.9 and 2.5 cm, respectively. The first tube was 8.8 m long and the other ones 25 m. The tube responses were measured with a logarithmic sine sweep that was played back to the tube with a full range loudspeaker. Figure 5.3 shows the equipment and the setup of the measurements. The logarithmic sine sweep made for

[1] `http://motu.com/products/motuaudio/ultralite-mk3/summary.html`

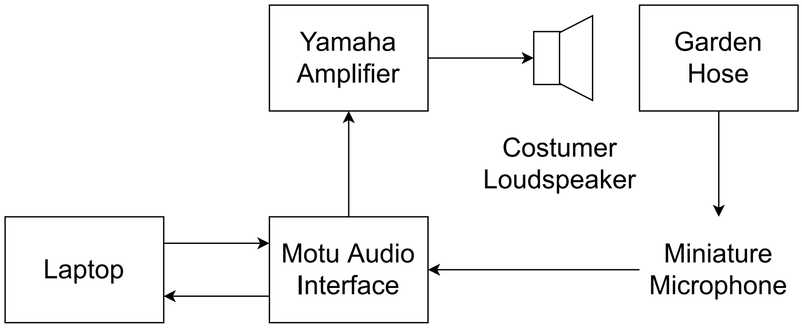

Figure 5.1: *Diagram of the measurement setup.*

excite the system can be seen in Fig. 5.2 and it was computed according Eq.(4.18). As can be seen in the figure, the sweep presents 1 s in the beginning and 3.6 s in the end for the reason explained in Sec. 4.6.

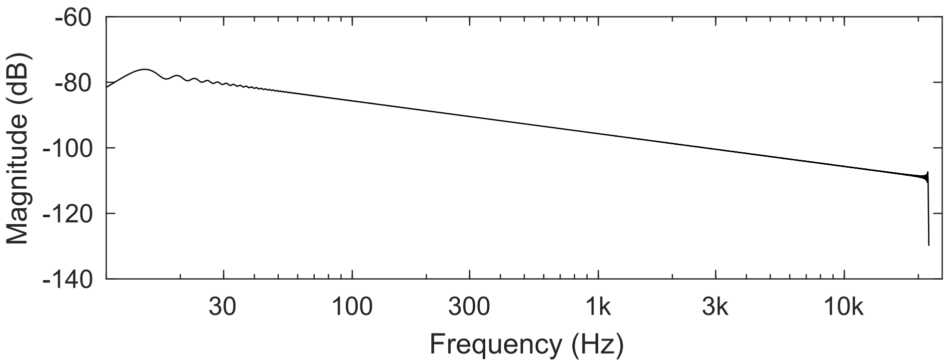

Figure 5.2: *Logarithmic sine sweep, spaced in the range* [10 22050] *Hz and* 3 *s long, used for the measurements.*

The measurements were conducted in an anechoic chamber and in two modalities: closed end and open end. The goal of the first modality was to obtain a clean impulse response caused by propagation and losses without any reflections. Polyurethane and a metal plate were used to absorb and block reflections from the tube end. The polyurethane and the metal plate were chosen based on initial experiments to minimize the reflections, where the shape of the polyurethane aiming to attenuate the sound and material aiming to block the reflection were tested. The tubes were coiled and separating each coil in order to avoid cross-talking.

Different holes were drilled along the tube with a distance of 1 m from each other, starting from 2.5 cm from the loudspeaker side of the tube. Multiple measurements were made, recording the response of one hole at a time with a miniature microphone while blocking the others with moldable plastic material in order to avoid a “flute finger-hole effect” in the recordings. Nine holes were made in the 1.2-cm tube, ten in the 1.9-cm tube and five in the 2.5-cm tube. The latter tube was the last one considered for the measurements and five holes only was decided enough for the purpose. In addition, in order to record the cleanest possible impulse responses, the measurements were made up to 10 m. The aim was to separate the end of the tube from the microphone location with a large distance. Thus, the reflected waves had to travel at least 30 m before meeting the microphone again, and in this way, they were both attenuated in level and separated in time. An exception was made with the 1.2 cm diameter tube, since it was only 8.8 m long.

(a) Microphone inside the tube with gray moldable plastic to attach it to the hole.

(b) Loudspeaker attached to the end of the tube with a conical adaptor.

(c) Short narrow tube (length 8.8 m, inner diameter 1.2 cm).

(d) Long medium-sized tube (25 m, 1.9 cm).

Figure 5.3: *Measurement setup in the anechoic chamber.*

The impulse response of the system was computed using Farina's method, convolving the recorded signal with the time-inverted logarithmic sweep. The convolution was performed in the frequency domain. The obtained response was FFT transformed and then divide by FFT transform of the input sweep. Finally, the result was IFFT transformed in order to perform the analysis in the time domain. The FFT operations were done removing the initial silence portion.

The input signal was pre-amplified using the Yamaha amplifier resulting in -41 dB of amplitude. The gain of the amplifier was chosen after several experiments in order to find a trade-off between the signal-to-noise ratio and the harmonic distortion. Finally, the measurements were performed with a sample rate of 44.1 kHz.

5.2 Tube Delay Analysis

Recordings for each hole were collected and stored. Figs. 5.4, 5.5 and 5.6 show a impulse response, with its magnitude spectrum, example for each tube in the closed mode measurements. Figs. 5.7, 5.8 and 5.9 show the respective ones for the open end modality. In the responses are present some reflections, especially in the 1.2-cm tube, 8.8 m long, where there is not enough distance from the end in order to attenuate them. The main spike of the impulse response followed by some ripple and reflections can be seen. The ripple is due to the holes along the tube. The holes could not be filled completely, and the resulting cavities created small reflections. The phenomena is less visible in the case of the 2.5-cm tube where less holes were drilled along. In addition, the magnitude spectra show noise and in particular in the high frequency.

Regard the magnitude responses, it can be seen the presence of noise, especially in the high side of the spectrum, and in addition, the presence of some spikes (see Figs. 5.5 and 5.8). Comparing the spectra in the two different modalities, it can be noted the presence of ripple in the open end mode measurements. It is clearly visible in the low frequencies and consists of the main difference between the two modes. The ripple was given by the reflections, in the open the reflections were not attenuated and it resulted in stronger interference effects between the input signal and its reflections.

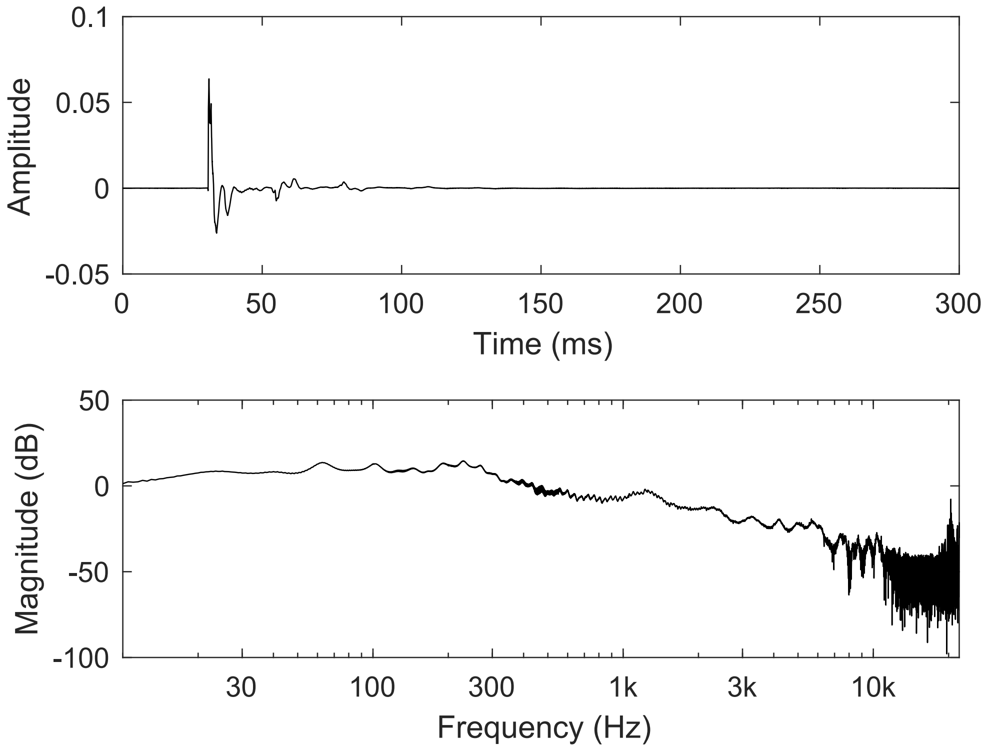

Figure 5.4: *Impulse response measured in the closed* 1.2-*cm tube at* 4.5 *m (top) and its magnitude spectrum (bottom).*

The harmonic distortion are in the end of the measurements, separated from the main lobe. Figures 5.10, 5.11, 5.12 and 5.13 show the response recorded in the closed end modality, with the respective harmonic distortion. It can be seen how the distortion increase by increasing the tube diameter. The distortions are bounded in the $[-0.01\ 0.01]$ range of amplitude, while main lobes in $[-0.2\ 0.2]$. The average SNR in the measurements ranged from 50 dB (for the narrowest tube) up to 40 dB (in the largest one). Finally, the harmonic distortion appears greater in the open end case and in the measurements recorded at smaller distance, it means when the recorded signal has more energy.

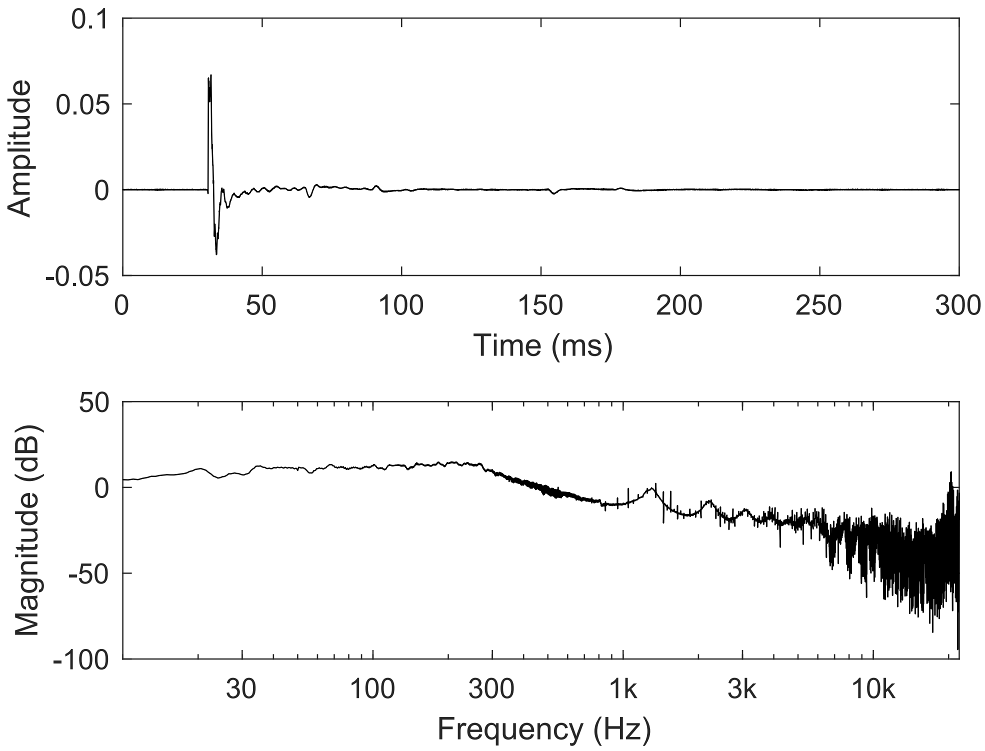

Figure 5.5: *Impulse response measured in the closed* 1.9*-cm tube at* 4.5 *m (top) and its magnitude spectrum (bottom).*

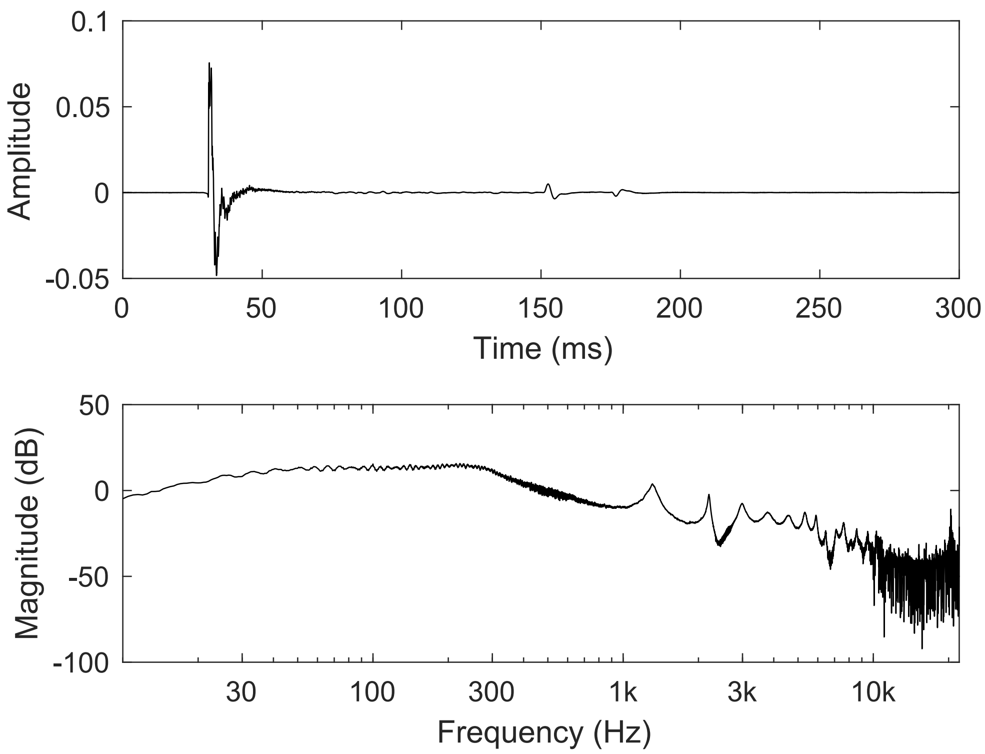

Figure 5.6: *Impulse response measured in the closed* 2.5*-cm tube at* 4.5 *m (top) and its magnitude spectrum (bottom).*

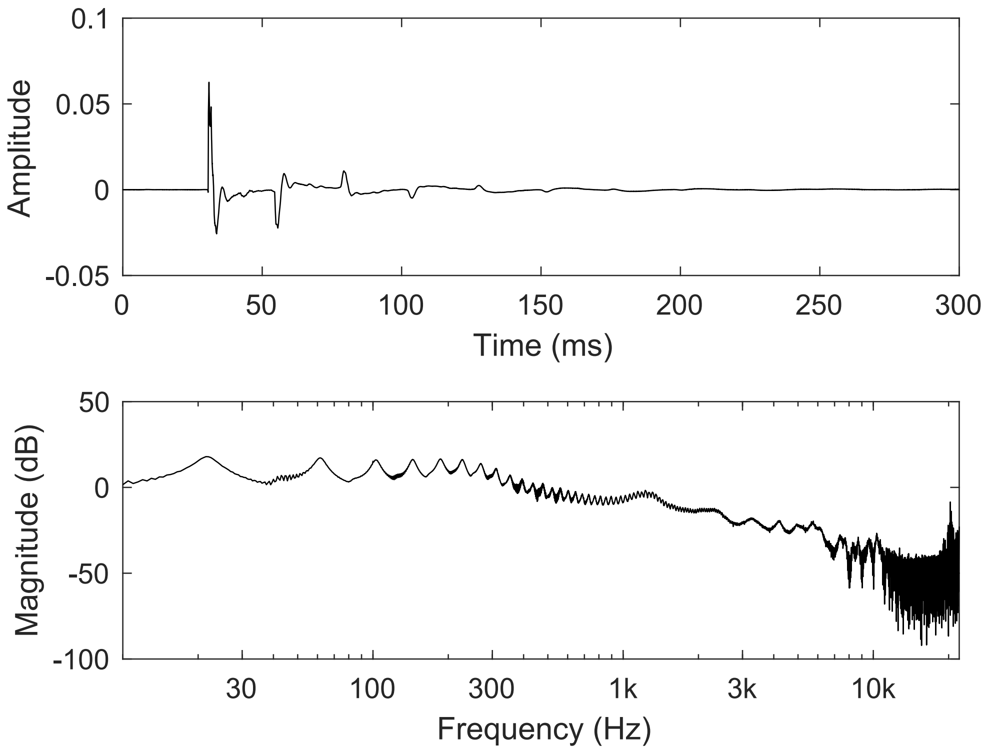

Figure 5.7: *Impulse response measured in the open* 1.2-*cm tube at* 4.5 *m (top) and its magnitude spectrum (bottom).*

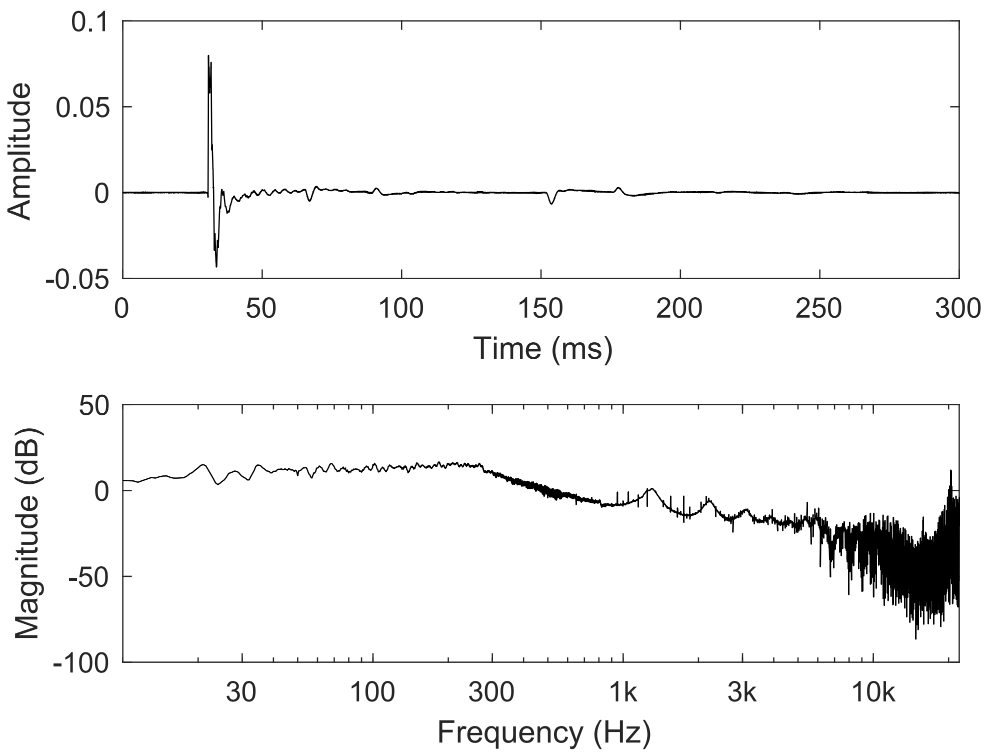

Figure 5.8: *Impulse response measured in the open* 1.9-*cm tube at* 4.5 *m (top) and its magnitude spectrum (bottom).*

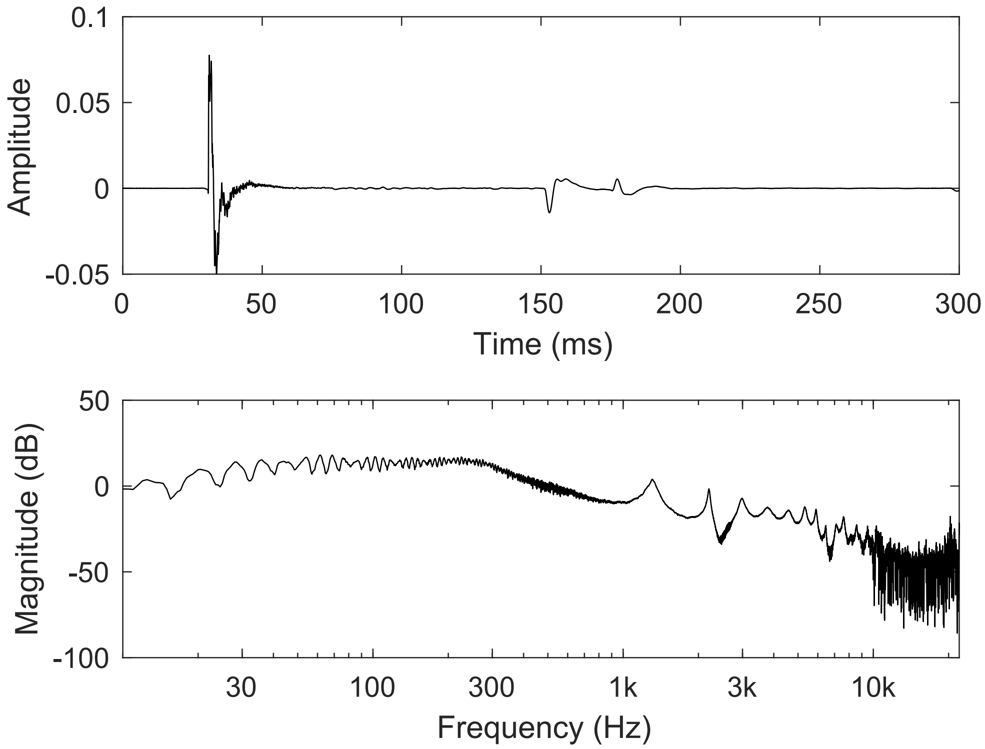

Figure 5.9: *Impulse response measured in the open* 2.5*-cm tube at* 4.5 *m (top) and its magnitude spectrum (bottom).*

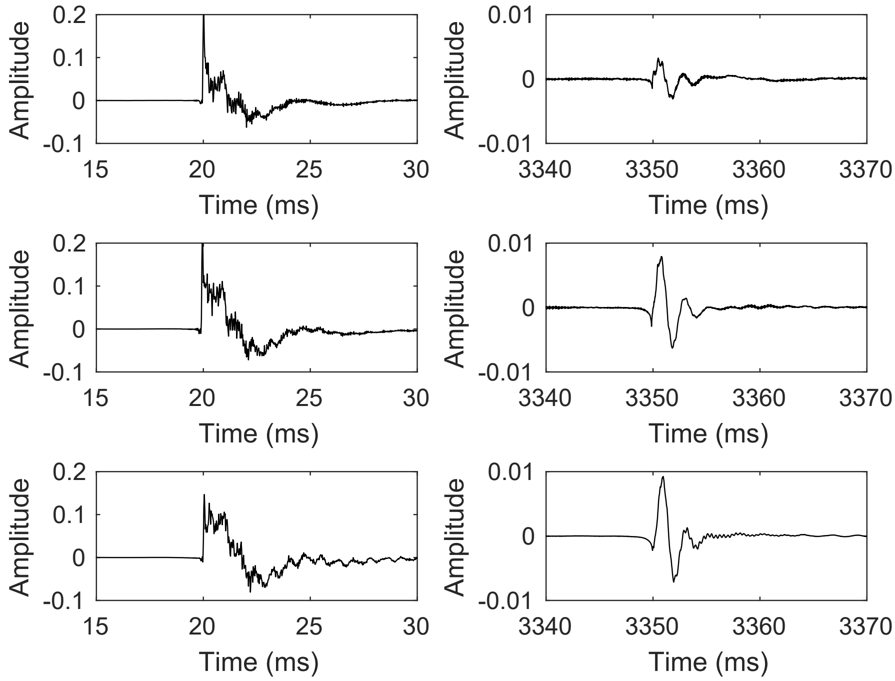

Figure 5.10: *Measured responses (left) and respective harmonic distortion (right) in the* 1.2*-cm (top),* 1.9*-cm (middle) and* 2.5*-cm (bottom) tube at the hole closest to the speaker and with closed end.*

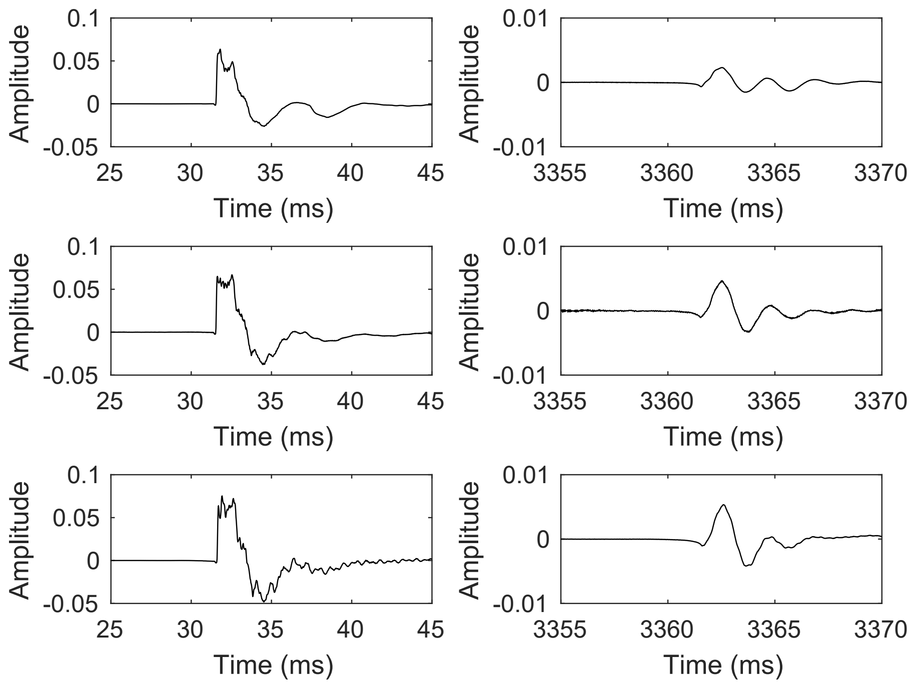

Figure 5.11: *Measured responses (left) and respective harmonic distortion (right) in the* 1.2*-cm (top),* 1.9*-cm (middle) and* 2.5*-cm (bottom) tube at* 4 *m from the speaker and with closed end.*

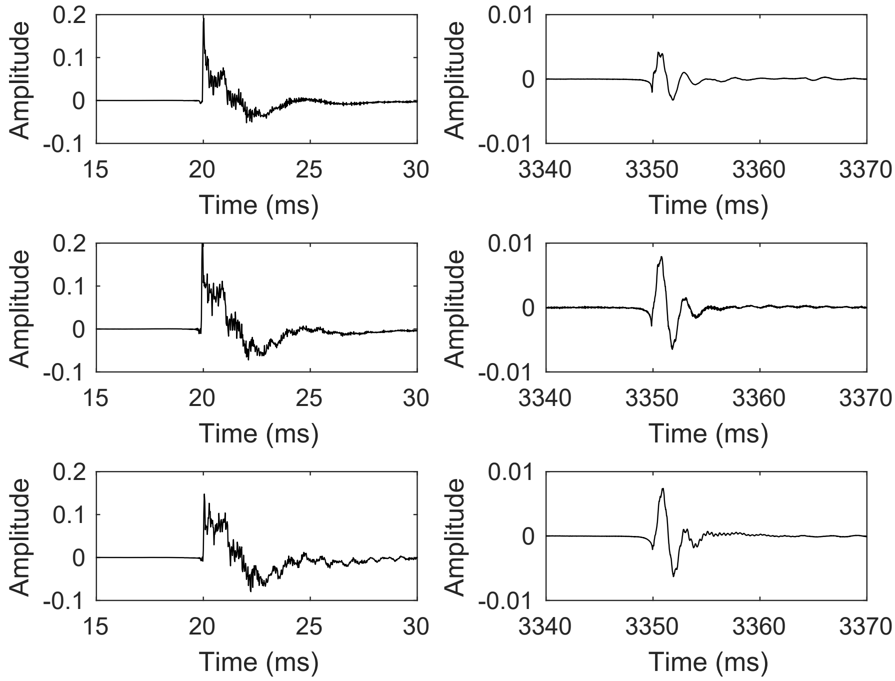

Figure 5.12: *Measured responses (left) and respective harmonic distortion (right) in the* 1.2*-cm (top),* 1.9*-cm (middle) and* 2.5*-cm (bottom) tube at the hole closest to the speaker and with open end.*

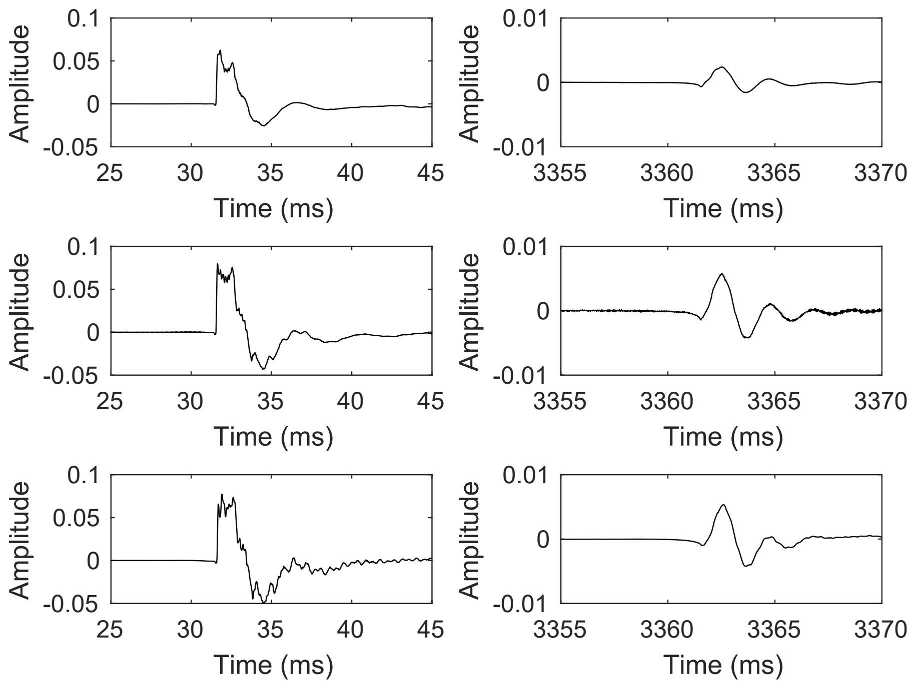

Figure 5.13: *Measured responses (left) and respective harmonic distortion (right) in the* 1.2*-cm (top),* 1.9*-cm (middle) and* 2.5*-cm (bottom) tube at* 4 *m from the speaker and with open end.*

The spectra of the measurements were noisy, especially in the high frequencies. The measured responses were low-pass filtered in order to remove any unwanted frequencies, higher than 20 kHz. Then, they were windowed in time to remove harmonic distortion components and unwanted reflections and finally smoothed. In order to suppress noise and analyze the frequency spectra, a third-octave filtering was performed. The choice of the filter was because its banks have the bandwidths approximating the bandwidths of the auditory filter and it is often used in audio analysis. The algorithm used in order to perform the smoothing can be seen in Appendix C.

In the windowing, an L-point Tukey window was used with L the number of samples needed for windowing the main spike. A Tukey window is a rectangular window with the first and last $r/2$ percent of the samples equal to parts of a cosine and it is defined as

$$w(x) = \begin{cases} \frac{1}{2}\left[1+\cos(\frac{2\pi}{r}x - \frac{r}{2})\right], & \text{for } 0 \leq x < \frac{r}{2} \\ 1, & \text{for } \frac{r}{2} \leq x < 1 - \frac{r}{2} \\ \frac{1}{2}\left[1+\cos(\frac{2\pi}{r}x - 1 + \frac{r}{2})\right], & \text{for } 1 - \frac{r}{2} \leq x \leq 1 \end{cases} \tag{5.1}$$

where the parameter r is the ratio of cosine-tapered section length to the entire window length with $0 \leq r \leq 1$. Setting $r = 0$, an L-point rectangular window is returned, instead setting $r = 1$, an L-point Hann window is returned. The parameter r was setted to 0.05. Fig. 5.14 shows the window used, with its respective magnitude response.

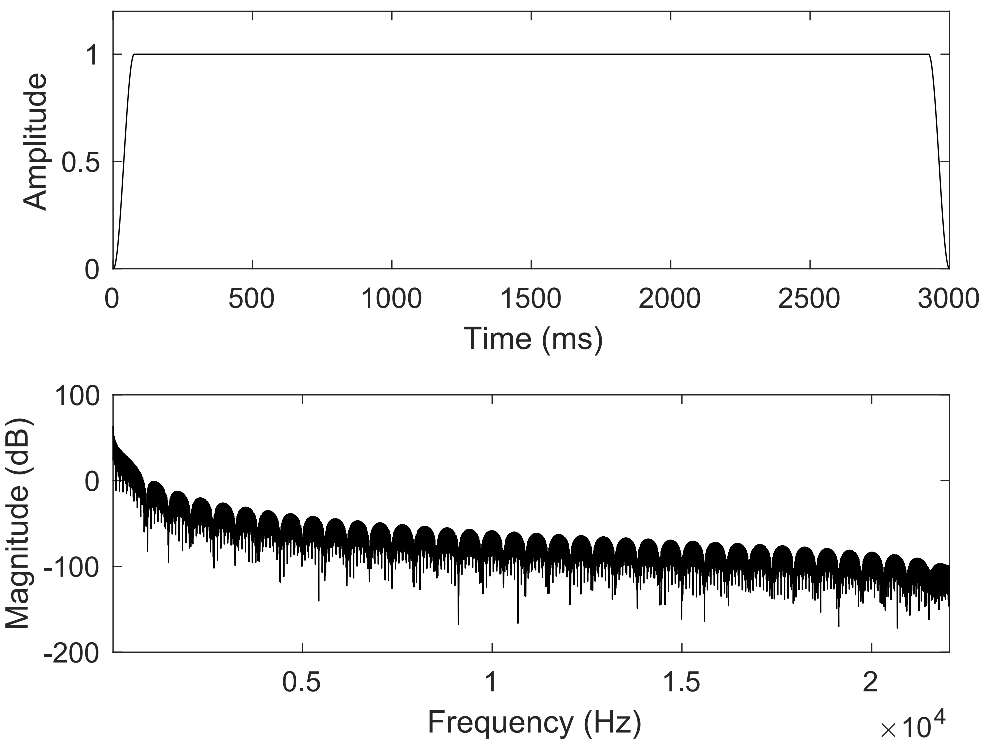

Figure 5.14: *Tukey window used in the responses analysis.*

The result of windowing can be seen in Fig. 5.16 and 5.17. Fig. 5.16 shows the original measurement and the windowed one.

The frame duration of the temporal window was defined by selecting the part of the signal of interests while ensuring to leave out the reflections. In order to avoid any significant differences and sharp step change in the start and end of the signal producing noise, the parameter r was set in way to mitigate this phenomenon and reduce the magnitude down to zero at the edges of the window. In the same time, because the presence of the reflections and other unwanted effects very close to the main spike, the shape at the edges was set sufficiently steep. In addition, the steepness at the edges affects the low frequency causing some losses. Therefore, these losses depend on the parameter r, Fig. 5.15 shows these losses depending on the choice of the parameter. It can be seen how the different value of r affects the losses in the low frequencies, decreasing the parameter r and, in turn, the steepness of the window, the losses are reduced. The selected window was able to reduce the amplitude of the discontinuities at the boundaries, preserve the shape of the part of the response of interests and cut off the unwanted part of the measurement. Zero padding before and after the window was also used.

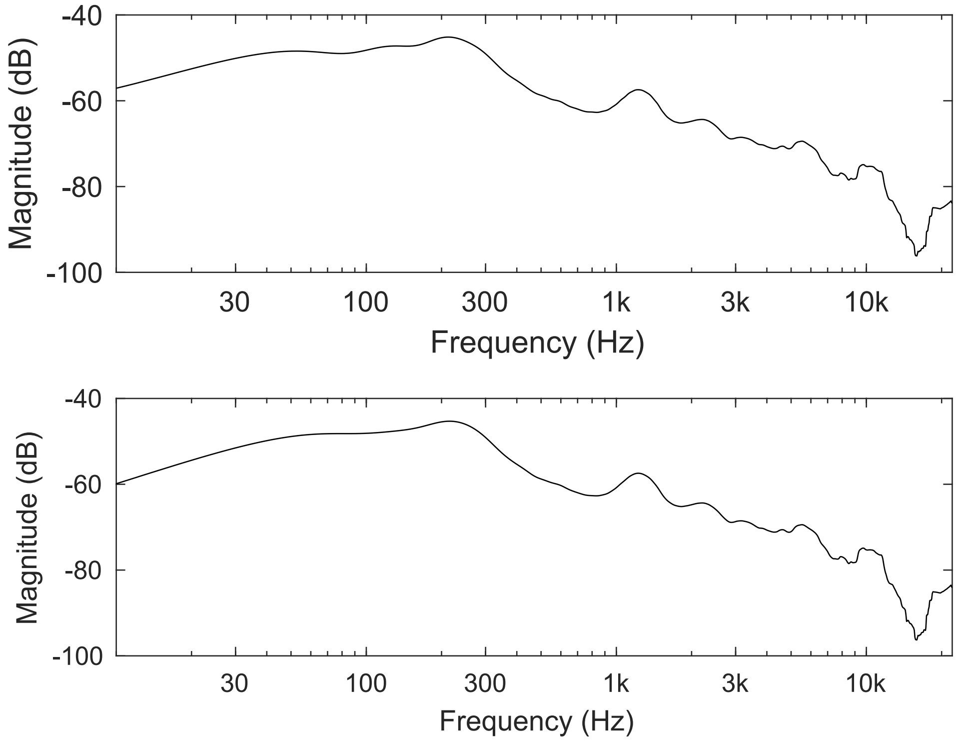

Figure 5.15: *Magnitude spectrum resulted by using different values of* r*,* 0.05 *(top) and* 0.5 *(bottom). Losses in the low frequencies are more present in the* $r = 0.5$ *case. Example obtained from the* 1.2-*cm tube at the distance of* 1.25 *m.*

Figure. 5.17 shows the magnitude response of the signal shown in Fig. 5.16. It can be seen that the windowing removed the spikes and attenuated ripple and noise. Finally, in Fig. 5.18 can be seen the effect of the smoothing. The filter removed the noise taking the underlying structure.

Impulse Response Analysis

A processed impulse response is presented in Fig. 5.19. The frequency response exhibits losses in the high end of the spectrum caused by propagation losses through the tube. There are also some losses in the low frequencies caused by the windowing. Significant attenuations of 20 dB or more appear above about 300 Hz.

As expected, spectral analysis of the windowed responses exhibits highly attenuated behaviour at very high frequencies, as seen in the example in Fig. 5.19. This can be caused in part by the effect of non-planar wave propagation above the cut-off of planar waves. The behaviour of the spectrum in the extreme high end is very noisy and, thus, unreliable.

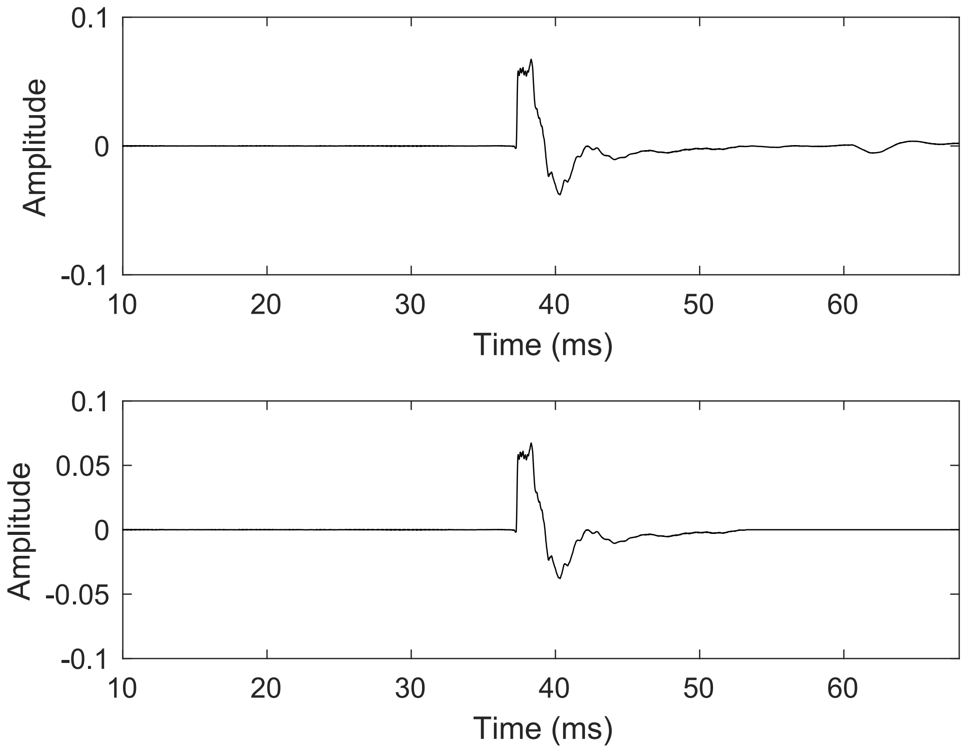

Figure 5.16: *Comparison with the original response and that one resulted of the windowing. Example obtained from the* 1.9*-cm tube at the distance of* 6.25 *m.*

Figure 5.20 shows the impulse responses recorded at three different holes. The time delay and the propagation loss can be observed here. Their corresponding frequency contents are shown in Fig. 5.21, and they reveal an increase of the attenuation with the increasing distance travelled and more significant losses at high frequencies when compared to low frequencies.

In addition, the measurements show that energy losses at high frequencies depend on the diameter of the tube. This behaviour can be observed in Fig. 5.22, where the windowed responses captured at 4.25 m from the beginning of the tube for the three different tube diameters 1.2, 1.9, and 2.5 cm. Their corresponding frequency spectra, are shown in Fig. 5.23. The attenuation is seen to increase with decreasing diameter, showing more losses especially at high frequencies.

The group delay was also computed and it showed an approximately flat response, indicating no time delay between the various sinusoidal components of the signal. This suggested that a delay line was suitable for simulating the propagation delay. The algorithm used to compute the group delay is showed in Appendix B.

Reflection Analysis

The measurements with the open end required further analysis of the reflection behaviour, as the impulse responses contained clearly observable repeating reflections due the both end. Figure 5.24 shows two example impulse responses collected in the open-end mode. The ripple is identified with circles and reflections can be seen.

As already noted, due to the finite length of the tube, the microphone recordings contain reflections from both ends. The waves propagating through the tube are reflected at the open end and, coming back, they are reflected again from the loudspeaker. Reflections appear in pairs repeated in time and progressively attenuated along the response. The location of the impulses can

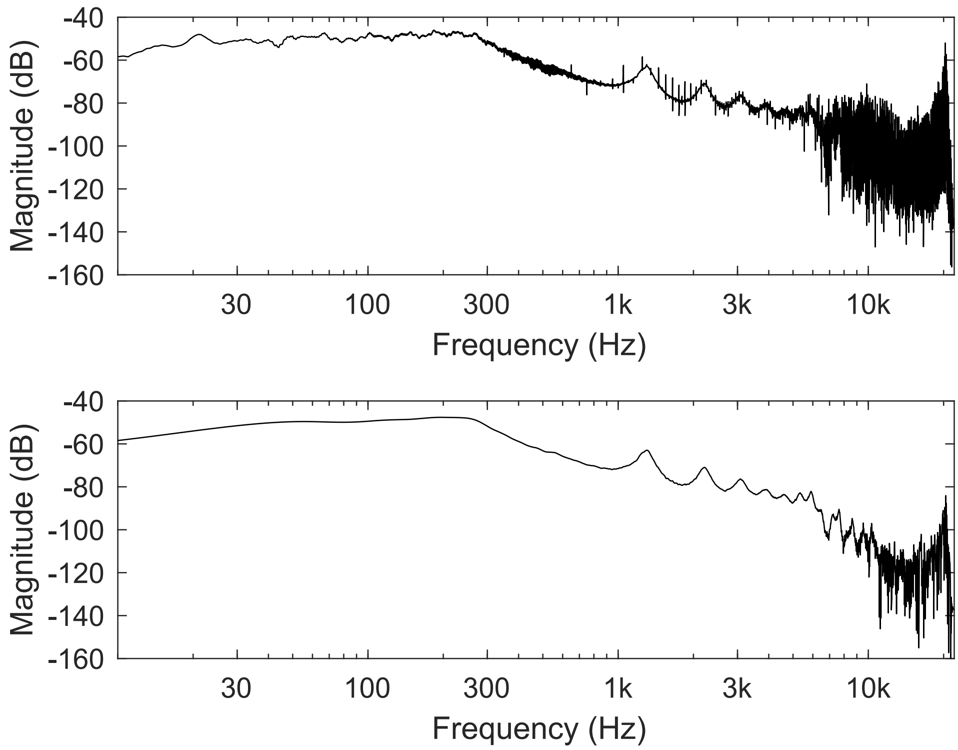

Figure 5.17: *Comparison with the original frequency response and that one resulted of the windowing. Example obtained from the* 1.9-*cm tube at the distance of* 6.25 *m.*

also be seen to differ between the two measurements in Fig. 5.24 due to the increased distance of the microphone from the loudspeaker.

Figure 5.24 shows the behaviour of the reflections at the distance of 4.25 m and 9.25 m. The negative reflection and the positive one can be clearly seen. The gap between reflections depends on the position of the microphone which recorded them. The farthest hole is 9.25 m from the loudspeaker and 15.75 m from the open end, which means a longer distance for the reflections to meet the microphone.

The reflections were windowed and then filtered with the third-octave filter as well. A measurement for each tube, having a enough separation between the reflections and the main spike, were chosen in order to window the negative reflection and analyze it. Since the presence of more than one reflection in a measurement, that one with more energy, it means the closest one to the main spike, was selected.

The analysis shows that energy exhibits losses in the high end of the spectrum and, instead, it is concentrated in the low frequencies. Figures 5.25, 5.26 and 5.27 show the windowed reflections result at the tube end together with its spectrum for the different tubes. From 300 Hz up to 1.5 kHz the spectrum exhibits a steep slope and above that extreme low energy values. The inverted pressure pulse due the open end can also be noticed.

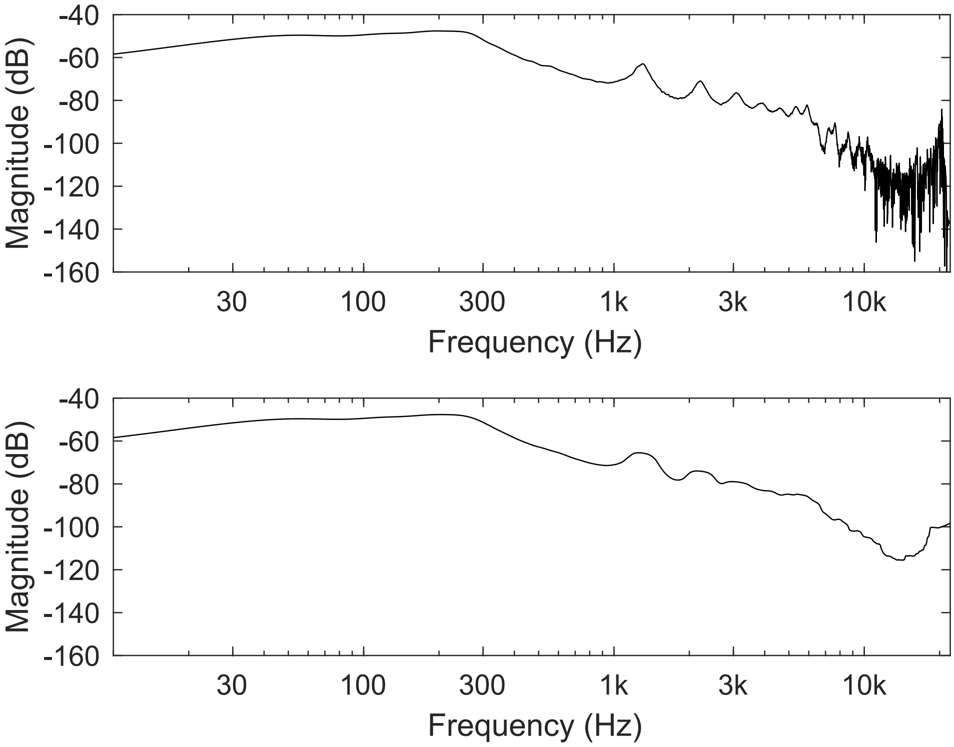

Figure 5.18: *Comparison with the frequency response of windowed signal and the smoothed one. Example obtained from the* 1.9*-cm tube at the distance of* 6.25 *m.*

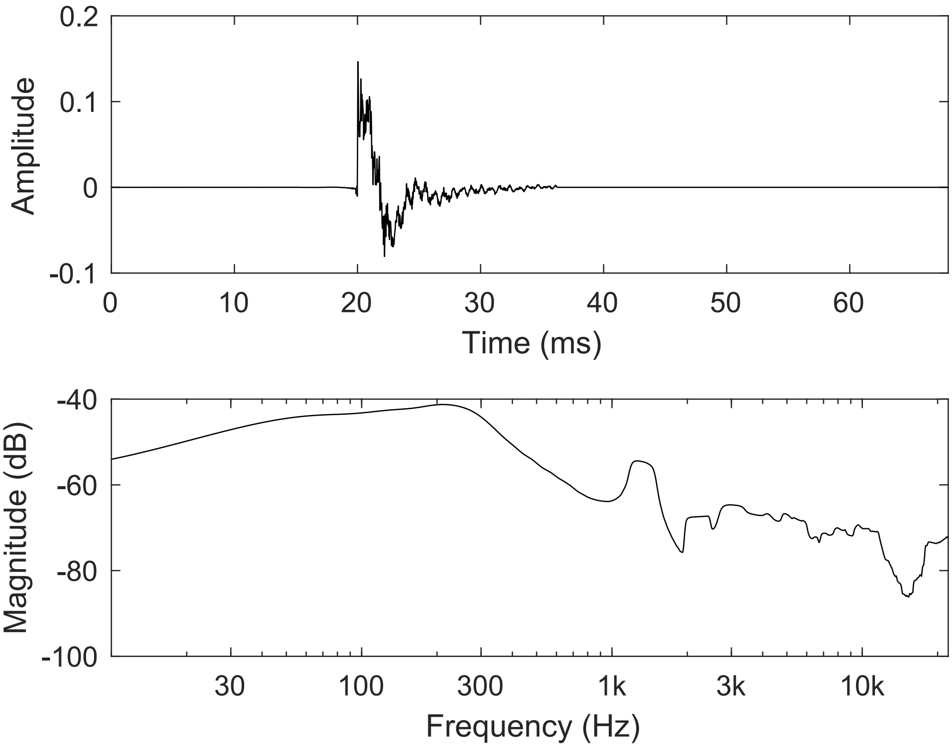

Figure 5.19: *Example of windowed impulse response (top) obtained from the* 2.5*-cm tube and its magnitude spectrum (bottom).*

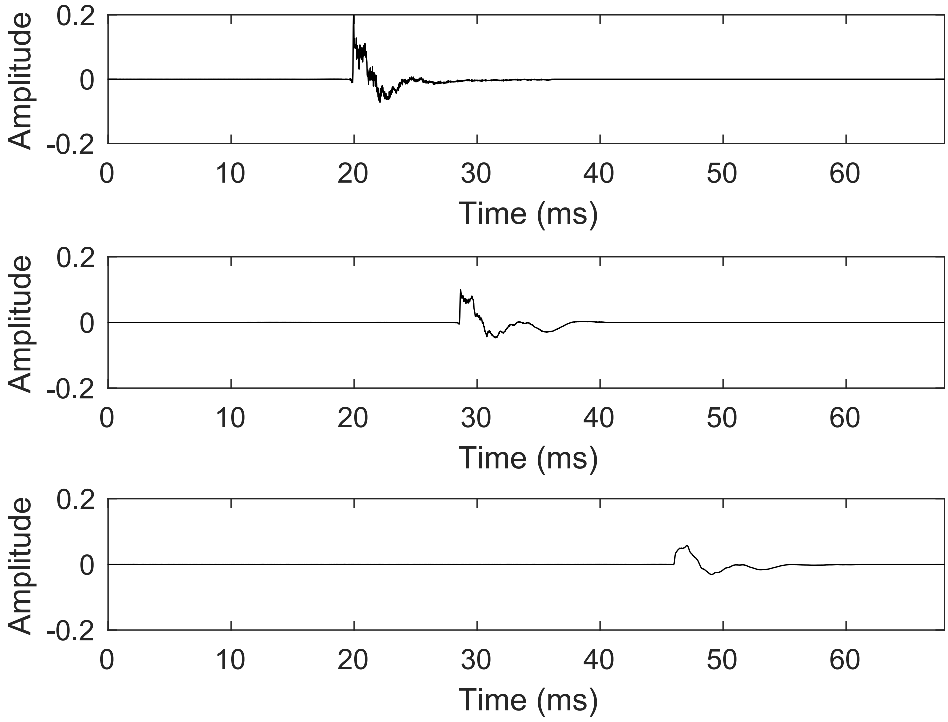

Figure 5.20: *Impulse response measured in the* 1.9*-cm garden hose at the distance of* 2.5 *cm (top),* 3.25 *m (middle), and* 9.25 *m (bottom) from the loudspeaker.*

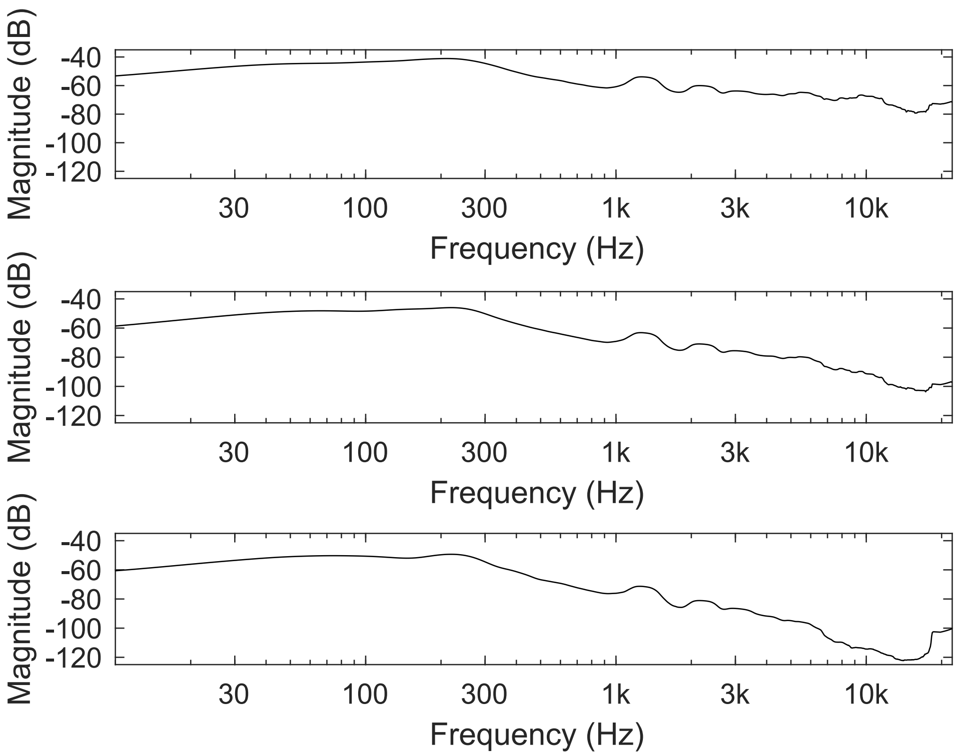

Figure 5.21: *Magnitude response measured in the* 1.9*-cm garden hose at the distance of* 2.5 *cm (top),* 3.25 *m (middle), and* 9.25 *m (bottom) from the loudspeaker.*

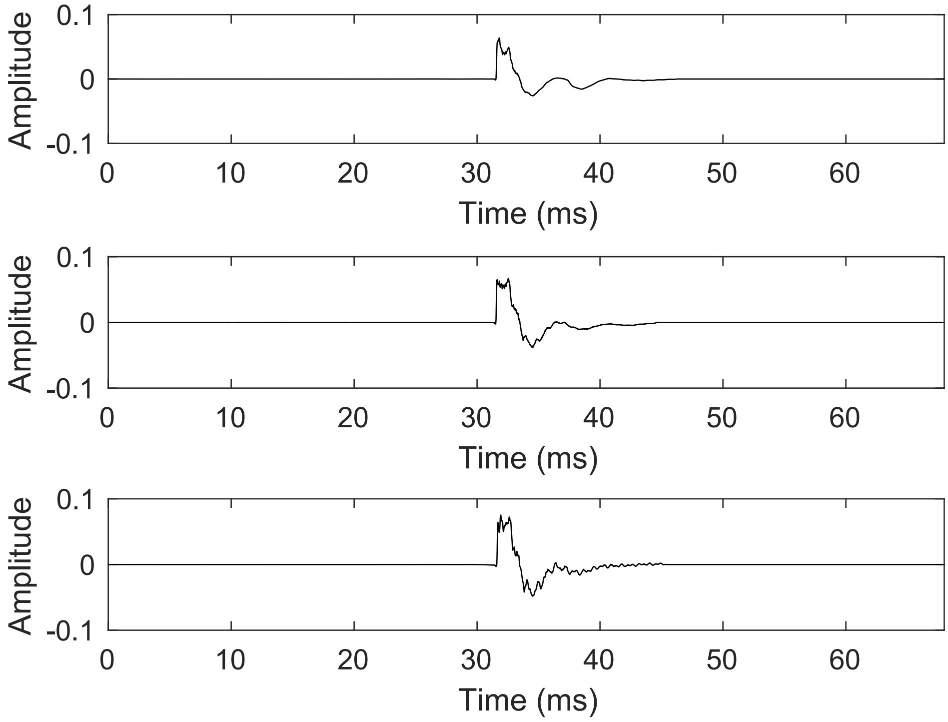

Figure 5.22: *Impulse responses measured at the distance of* 4.25 *m in the* 1.2*-cm (top),* 1.9*-cm (middle), and* 2.5*-cm (bottom) tube.*

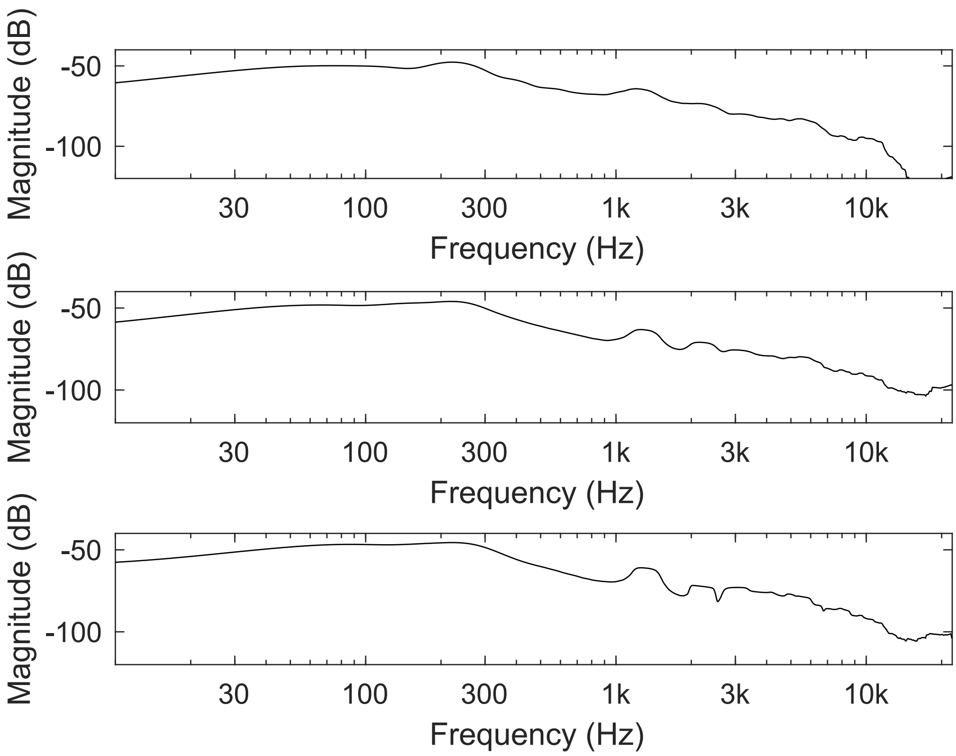

Figure 5.23: *Magnitude responses measured at the distance of* 4.25 *m in the* 1.2*-cm (top),* 1.9*-cm (middle), and* 2.5*-cm (bottom) tube.*

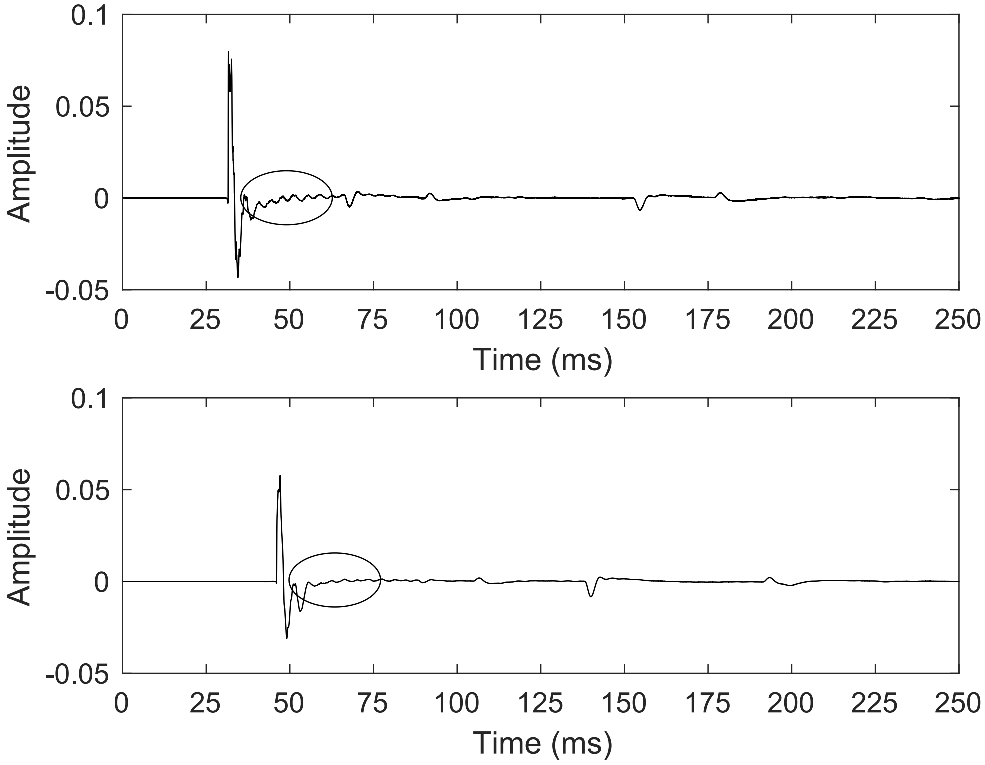

Figure 5.24: *Impulse responses measured in the* 1.9-*cm tube in the open end case, at the distance of* 4.25 *m (top) and* 9.25 *m (bottom).*

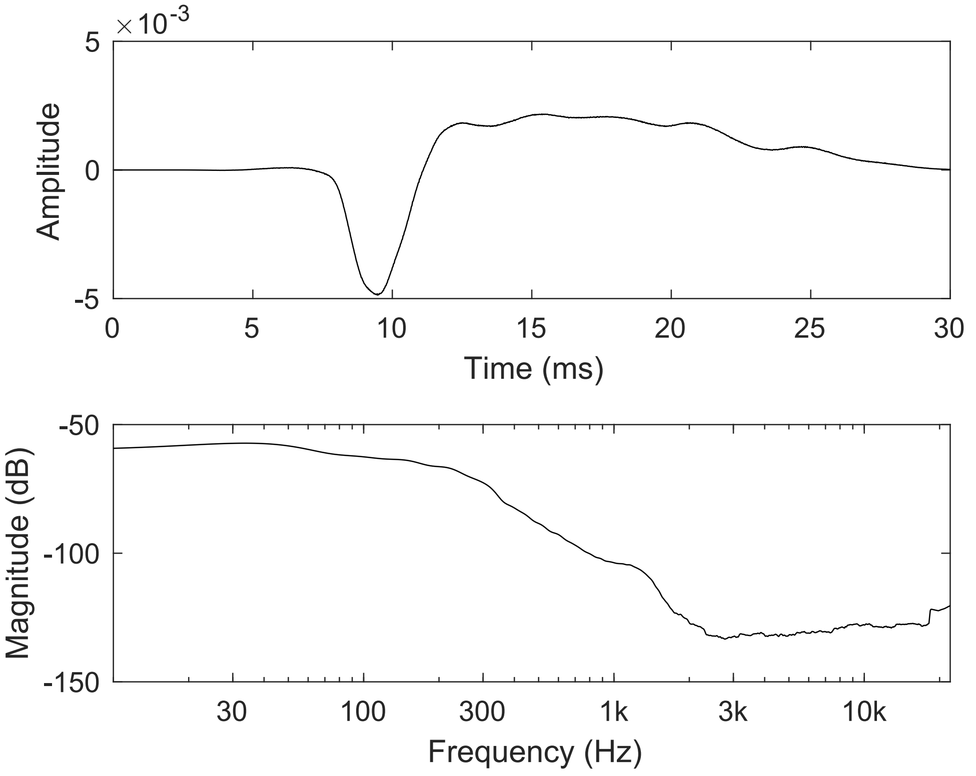

Figure 5.25: *A windowed reflection (top) and its magnitude spectrum (bottom) recorded with the* 1.2-*cm tube.*

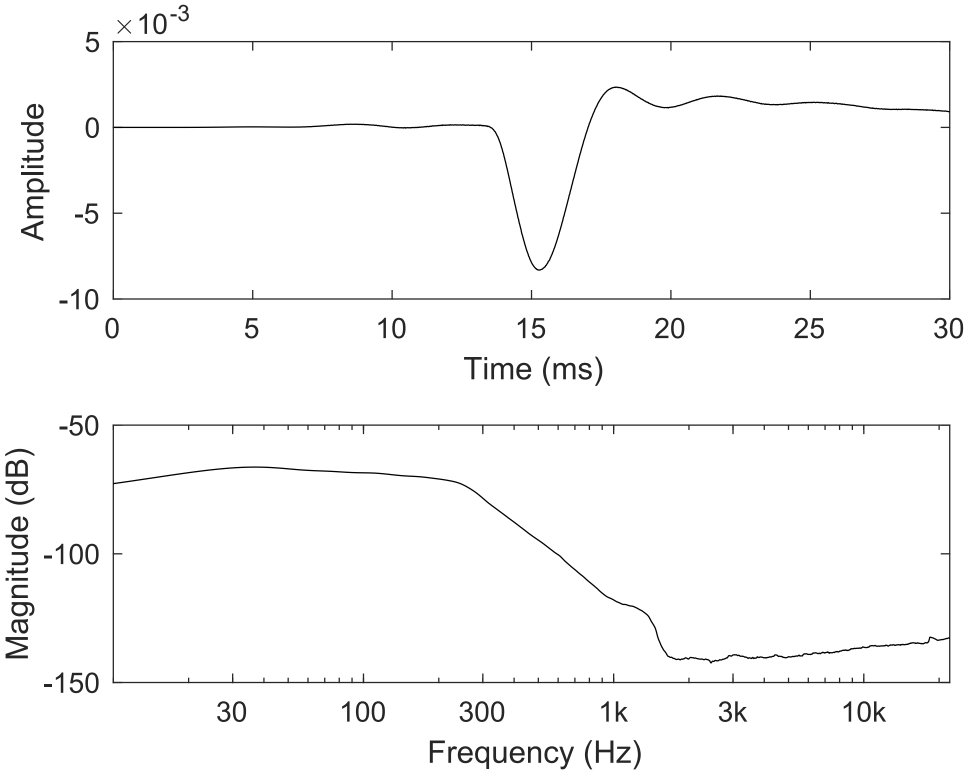

Figure 5.26: *A windowed reflection (top) and its magnitude spectrum (bottom) recorded with the 1.9-cm tube.*

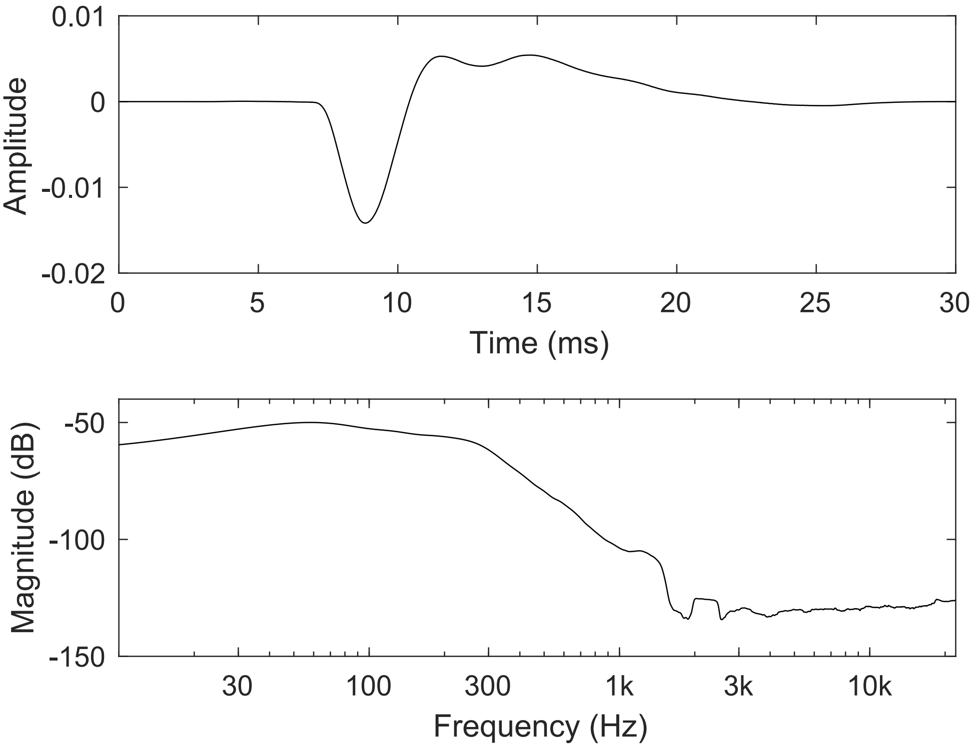

Figure 5.27: *A windowed reflection (top) and its magnitude spectrum (bottom) recorded with the 2.5-cm tube.*

Chapter 6

Virtual Tube Model

In order to model the system, the spectra of all the windowed signals were analyzed collectively. More specifically, in order to analyze the spectral changes associated with each meter travelled through the tube, the differences in the spectra of the respective signals were computed. From this analysis a two different filters were computed and a model including digital delay lines was built. The resulting system consists of a digital waveguide model.

In this chapter is presented the modelling for a virtual tube system and the computation performed to obtain the filter describing the propagation of the sound in it. Two different filter will be discussed, the Propagation Filter controlling the energy losses based on the tube length and the Reflection Filter controlling the tube-end effect. In the end, the comparison of the filters computed with the results of measurements performed will be propose .

6.1 Propagation Filter

As already introduced, the spectral changes associated with each meter travelled through the tube was computed for each tube size. The differences in the spectra of the respective signals were computed with the following equation:

$$\frac{H^i_{\mathrm{dB}}(f) - H^j_{\mathrm{dB}}(f)}{d_{ij}} \quad \forall i, j \ , \tag{6.1}$$

where $H_{\mathrm{dB}}(f)$ is the spectrum magnitude of the signal in decibels, smoothed with a third-octave filter, and d_{ij} the distance in meters between the i-th and j-th holes, where the signals were recorded. These differences were computed for each tube. Then, the arithmetic mean of the results obtained was computed for each tube. In this way, an average behaviour for a 1 m segment of each tube was obtained. The results are shown together in Fig. 6.1.

It can be noticed that the attenuation increases towards the high end of the spectrum and that it depends on the tube diameter. Increasing diameters result in a steeper shape, but with smaller attenuation. The responses below 300 Hz, despite some oscillations, are very similar to each other near 0 dB. Attenuation is noticeable above 300 Hz and becomes more significant around 1 kHz.

Since the first modes of the tubes are at 8054 Hz, 10598, and 16780, the results above these frequencies are unreliable. For this reason, the responses above these frequencies were not considered, and a continuous slope for the frequencies larger than 10 kHz in the design of the filters was taken.

Based on the above considerations, the spectrum can be assumed to have a low-pass shape. Increasing the tube diameter decreases the spectral slope and increases the cut-off frequency.

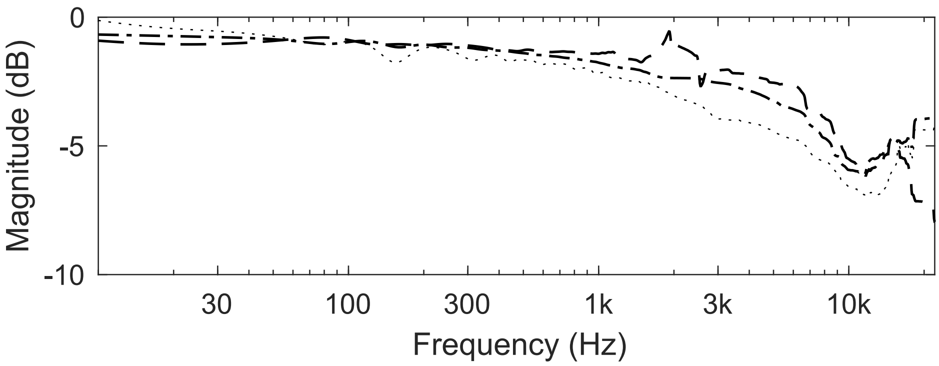

Figure 6.1: *Average "difference filters" for a* 1*-m segment (see Eq. (6.1)) of a* 1.2*-cm (dotted line), a* 1.9*-cm (dash-dot line), and a* 2.5*-cm (dashed line) tube.*

Reflections from the End of the Tube

In order to understand the effect of the open end on the responses, a different approach was chosen. Using the acquired information, the impulse response measured at the farthest hole from the loudspeaker was filtered with the filter approximating an appropriate power of the 1 m segment shape of Fig. 6.1. The aim here was to simulate the losses of the same distance that the reflected pulse had travelled. This simulation could be compared with the reflection, separating the reflection effect of the open end. The distance travelled by the reflection was computed and used to build the filter, accounting for the approximation error which becomes significant for long distances.

Figure 6.2 shows the spectrum of the reflection captured by the microphone and the simulated spectrum as it should be without the open-end effect. A slight attenuation can be seen below 100 Hz, and a stronger one up to 1 kHz. Since the impulse travels along the whole tube before reaching the open end, it has very low energy above 3 kHz and the recorded reflection is superimposed by the noise. When the impulse crosses the boundary at open end, the pressure wave hits the outside air, at atmospheric pressure, creating a compression wave heading back down the tube with some energy left.

Using the filter designed for the tube model, the effect of the reflection R due the open end was obtained:

$$R = \frac{H^{i}_{\mathrm{ref}}(f)}{H^{i}_{\mathrm{sim}}(f)}, \tag{6.2}$$

where $H_{\mathrm{sim}}(f)$ is the spectrum of the response without the open end effect simulated with the approach described above using the same distance travelled by the corresponding windowed reflection $H_{\mathrm{ref}}(f)$. This allows for the estimation of how the reflection affects the spectrum. Equation (6.2) was estimated for each measure where the reflections were isolated enough and could be windowed. Finally, the average for each tube size was computed. The shapes shown in Fig. 6.3 summarize the results.

The results show that the attenuation depends on the diameter of the tube, starting with a low value increasing above 100 Hz. The attenuation becomes smaller at higher frequencies because of the noise level.

6.2 Filter Design

Using the results found, the design of the filters simulating the sound propagation through the tube and the reflection effect by the open end was done. For each of these effects, the average

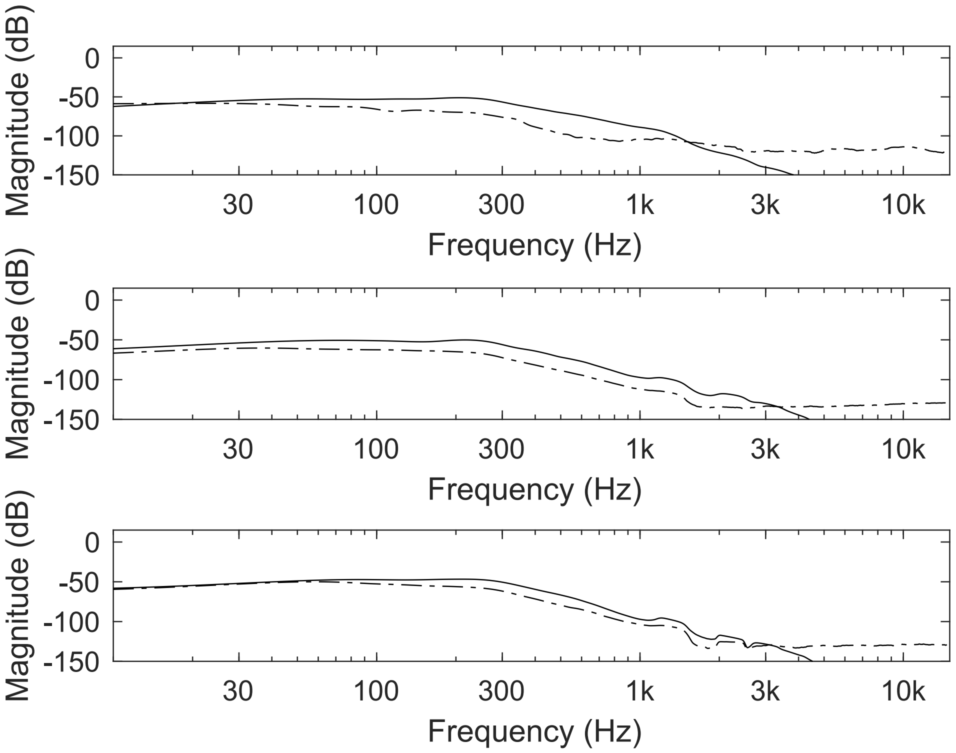

Figure 6.2: *Comparison between the spectrum of the reflection captured by the microphone (dash-dot line), and the simulated spectrum as it should be without the open end effect (solid line):* 1.2-*cm (top)* , 1.9-*cm (middle) and* 2.5-*cm (bottom) diameter tubes.*

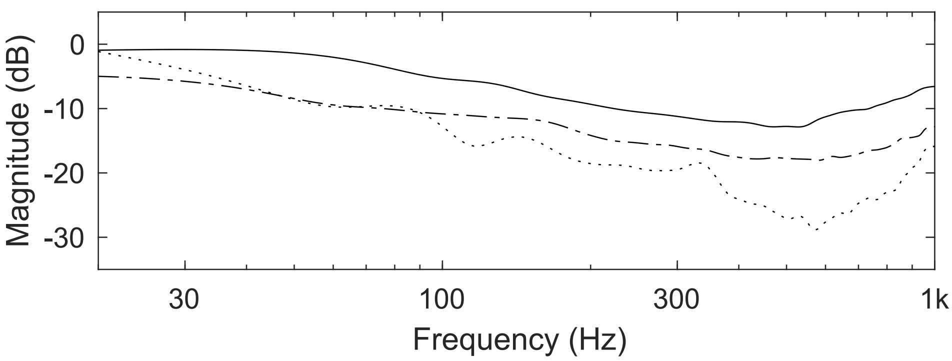

Figure 6.3: *Average filters estimating the open end effect (see Eq. (6.2)) of a* 1.2-*cm (dotted line),* 1.9-*cm (dash-dot line), and* 2.5-*cm (dashed line) tube.*

filters previously computed and summarized in Figs. 6.1 and 6.3 were used as target shapes to be approximated with low-order filters. Then, a unique form to interpolate between the different diameters values was found.

Propagation Filter

The filters simulating the sound propagation through the tube was called Propagation Filter. Given the simple shapes of these filters (see Fig. 6.1), attempts were made to find a low-order filter simulating their behaviour. Keeping the three averages as targets, three parametric filters were computed, approximating the shape in order to minimize audible errors.

A cascade of two high-shelving filters and one low-pass filter was built, resulting in a 5th-order parametric filter. The high-shelving filters were used to approximate the shape from 300 Hz to 3 kHz, while the low-pass filter was needed to cut the high end of the spectrum.

Since the three target shapes behave very similarly at low frequencies, the filters have the same behaviour until 300 Hz with a slight attenuation depending on the diameter of the tube. The significant variations are in the range above 1 kHz, where different attenuations and cut-offs can be seen. The cut-off frequencies for the three target shapes are 4062, 5950, and 7015 Hz, respectively.

Figure 6.4 shows the different filters designed for the three diameter tubes to be compared with those in Fig. 6.1. With these low-order filters, a tube with arbitrary length can be simulated. Moreover, interpolating between the three filters allows to simulate different diameters sizes.

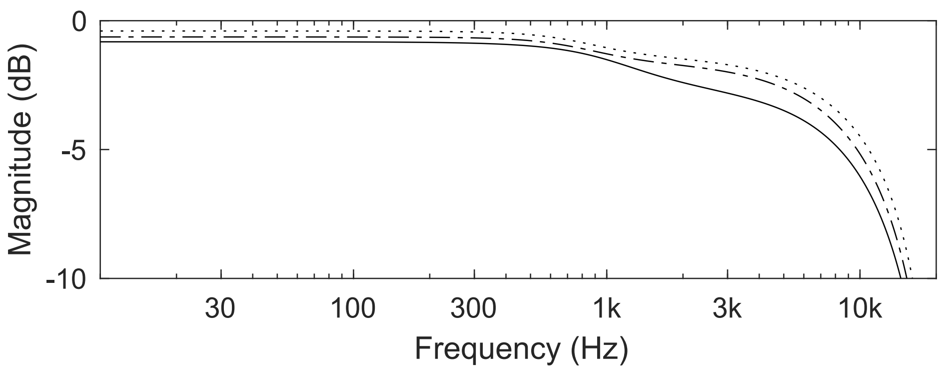

Figure 6.4: *Low-order approximations of the average "difference filters" for a* 1*-m segment (see Eq. (6.1)):* 1.2*-cm (solid line),* 1.9*-cm (dash-dot line), and* 2.5*-cm (dotted line) diameter tubes.*

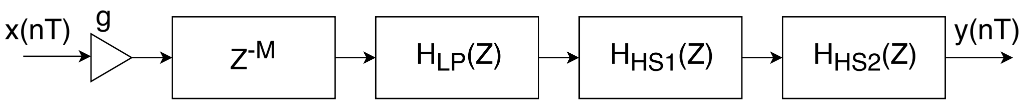

Figure 6.5: *Modeling the sound propagation using a delay line and three filters.*

Figure 6.5 shows the three parametric filters in cascade and the delay line composing the system. The system can be described mathematically as follows:

$$H_{\mathrm{tube}}(z) = gz^{-M} H_{\mathrm{HS1}}(z) H_{\mathrm{HS2}}(z) H_{\mathrm{LP}}(z), \tag{6.3}$$

where g is a gain factor, z^{-M} is the delay line of M samples, $H_{\mathrm{HS1}}(z)$ and $H_{\mathrm{HS2}}(z)$ are 2nd-order IIR high-shelving filters, and $H_{\mathrm{LP}}(z)$ is a 1st-order IIR low-pass filter.

The coefficients of the high-shelving and low-pass filters were computed with the usual formulas of the 1st- and 2nd-order filters [Dut+11]. The low-pass filter can be written as

$$H_{\mathrm{LP}}(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}, \tag{6.4}$$

where the numerator coefficients are given by

$$b_0 = b_1 = \frac{K}{K+1}, \tag{6.5}$$

and the denominator one is given by

$$a_1 = \frac{K-1}{K+1}. \tag{6.6}$$

The high-shelving filter, instead, can be written as

$$H_{HS}(z) = \frac{d_0 + d_1 z^{-1} + d_2 z^{-2}}{1 + c_1 z^{-1} + c_2 z^{-2}}. \tag{6.7}$$

where the numerator coefficients are given by

$$c_1 = \frac{(2(V_0K^2 - 1))}{(1 + \frac{1}{Q}\sqrt{V_0}K + VK^2)}, \quad c_2 = \frac{(1 - \frac{1}{Q}\sqrt{V_0}K + V_0K^2)}{(1 + \frac{1}{Q}\sqrt{V_0}K + V_0K^2)}, \tag{6.8}$$

and the denominator ones are given by

$$d_0 = \frac{(V_0(1 + \frac{1}{Q}K + K^2)}{(1 + \frac{1}{Q}\sqrt{V_0}K + V_0K^2))}, \quad d_1 = \frac{(2V_0(K^2 - 1))}{(1 + \frac{1}{Q}\sqrt{V_0}K + V_0K^2)}, \quad d_2 = \frac{V_0(1 - \frac{1}{Q}K + K^2)}{(1 + \frac{1}{Q}\sqrt{V_0}K + V_0K^2)}. \tag{6.9}$$

In order to design the filters, the cut-off frequency f_c, the bandwidth f_b and the gain G were set in order to compute the parameters $K = \tan(\pi f_c/f_s)$, $V_0 = 10^{G/20}$ and the quality factor $Q = \frac{f_b}{f_c}$, finally, in turn, the coefficients.

Three different IIR filters, in this way, were designed, one for each tube diameter (1.2, 1.9, 2.5 cm), giving the possibility to approximate the different behaviours by controlling the shape with the cut-off frequencies of the designed IIR digital filter. In order to control the filter behaviour as a function of the diameter of the simulated tube, the cut-off frequencies of all the filters and the gain factor g are linearly varied while the gains (dB) and the quality factors of the two high-shelving filters are kept fixed. Table 6.1 reports these latter values while Table 6.2 summarizes the filter cut-off frequencies and the gain factor for each tube diameter.

Starting from these values, an interpolation was made with a granularity of 1 mm. The values of f_c and g were varied in 13 step obtaining a different filter for each size. Fig. 6.6 show all the steps obtained varying the tube size. It can be seen the interpolation from 1.2 to 1.9 cm (top) and 1.9 to 2.5 cm (bottom).

Since a cascade is an inefficient approach to produce tubes longer than 1 m, an approximation was found. Starting from the filter computed for the 1.2 cm tube, all the parameters of the three basic filters composing it were gradually varied in a linear way to achieve an approximated filter for longer lengths. A cascade of two 1$^{\text{st}}$-order low-pass filter replaced the simple 1$^{\text{st}}$-order one, resulting in a 6$^{\text{th}}$-order parametric filter. A good approximation up to 30 m (which is sufficient for the purpose of the audio effect) was obtained with an error smaller than 0.6 dB. In addition, with this method a better accuracy creating the tube can be achieved. Instead of 1 m as the incremental step, a finer control, like 1 cm, can be implemented. Figure 6.7 shows the approximation for 30 m. The designed filter follows accurately the general shape except for a critical range between 300 Hz and 1 kHz. In the case of 30 m tube, the maximum error is 0.57 dB. Fig. 6.8 , instead, presents the filter approximating 1, 5, 10, 15, 20, 25 and 30 m

In order to compute an accurate approximation, an empirical investigation on the values was done.The range of interests was splitted in [1 15] and [16 30] m. For each range a incremental or decremental value was found. Being an approximation of a cascade of filters, the value g, representing the gain factor (see (6.3)), was computed by

$$g = 0.85^i \tag{6.10}$$

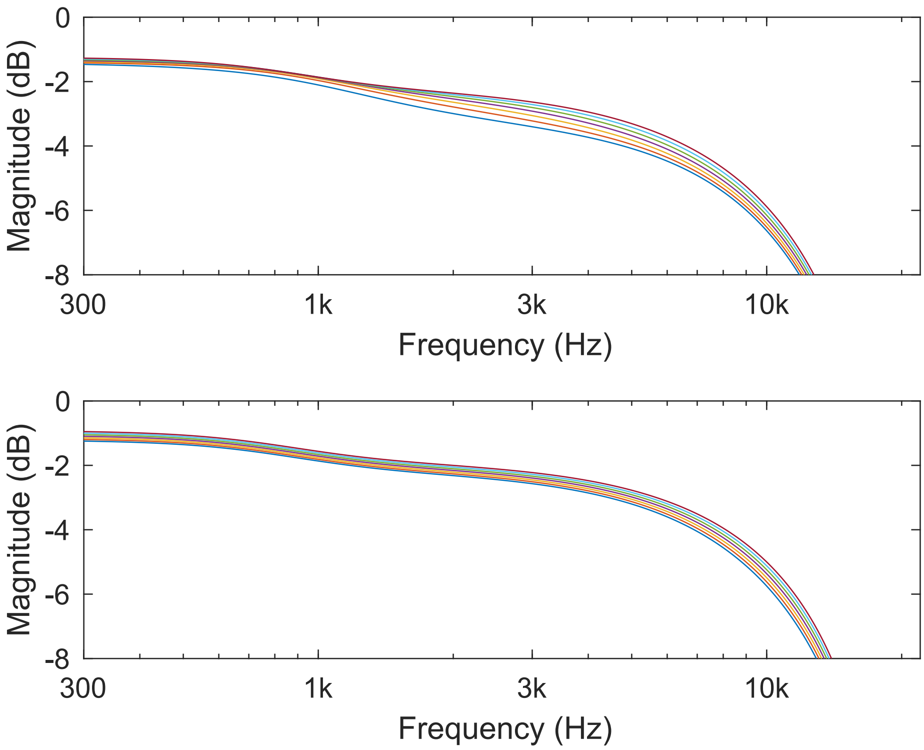

Figure 6.6: *Interpolation of the low-order approximations of the average "difference filters" for a 1-m segment,* $1.2 - 1.9$ *cm (top) and* $1.9 - 2.5$ *cm (bottom).*

where i denotes the distance in meters. Indicating with the apexes 1 and 2 the values for, respectively, the first and second high-shelving filters, the parameters were given by the follow expression.

The cut-off frequencies fc, the gain G and the quality factor Q of the high-shelving filters were given by

$$f_c^1 = \begin{cases} 1200 + (i-1)30 & \text{for } 1 \leq i \leq 15 \\ 1620 + (i-15)20 & \text{for } 16 \leq i \leq 30 \end{cases} \qquad f_c^2 = \begin{cases} 1500 + (i-1)50 & \text{for } 1 \leq i \leq 15 \\ 2200 & \text{for } 16 \leq i \leq 30 \end{cases} \tag{6.11}$$

$$G^1 = \begin{cases} -1 - 0.85(i-1) & \text{for } 1 \leq i \leq 15 \\ -12.9 - 0.6(i-15) & \text{for } 16 \leq i \leq 30 \end{cases} \qquad G^2 = \begin{cases} -0.9 - (i-1) & \text{for } 1 \leq i \leq 15 \\ -14.9 - (i-15)0.1 & \text{for } 16 \leq i \leq 30 \end{cases} \tag{6.12}$$

and

$$Q^1 = 0.65 \qquad Q^2 = \begin{cases} 0.5 & \text{for } 1 \leq i \leq 15 \\ 0.5 + (i-15) * 0.02 & \text{for } 16 \leq i \leq 30 \end{cases} \tag{6.13}$$

In the case of the low-pass filter, the cut-off frequency was decremented, starting from 9500 Hz, according the distance selected. Since the 1$^{\text{st}}$ order was not able to fit in an accurate way the target, with distance greater than 2 m a 2$^{\text{nd}}$ order low-pass filter was used. In addition, since the difficulties to find an accurate interpolation for the low-pass cut-off, they were selected empirically and stored in a 30-long vector.

After obtaining an accurate approximation of frequency attenuations due to propagation in the tube, the final filter was obtained by using a delay line that simulates the propagation delay and is connected in series with the previously discussed filter.

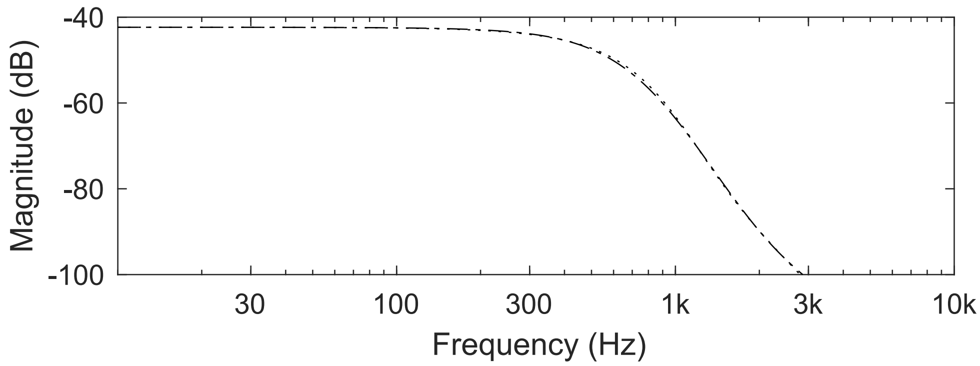

Figure 6.7: *Example of a parametric filter designed to approximate* 30 *m long tube: target filter (dotted line), and approximation (dash-dot line).*

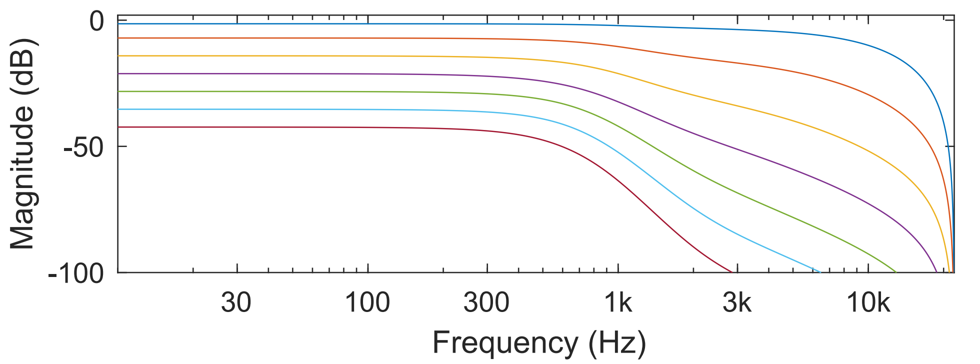

Figure 6.8: *Example of a parametric filter designed to approximate* 1, 5, 10, 15, 20, 25 *and* 30 *m long tube.*

The value of M, the number of samples, is found with a simple computation depending on the length of the tube:

$$M = \frac{l}{c} f_s \tag{6.14}$$

where l is the length of the tube, c the speed of sound, and f_s the sampling rate.

Table 6.1: *Propagation filter: gain and quality factor values for the two high-shelving filters.*

Type of filter	$G[dB]$	Q
HS1	-1	0.65
HS2	-0.9	0.5

Reflection Filter

The block scheme in Fig. 6.9 shows the approach used to simulate the reflection. The delayed input is first filtered with the filter $H_{\text{ref}}(z)$ that approximates the losses given by the open end reflection, and the output is fed to the filter $H_{\text{tube}}(z)$ that simulates the losses caused by sound

Table 6.2: *Propagation filter: cut-off frequencies of low-pass and high-shelving filters and overall gain for the three tube diameters.*

Type of filter	f_{HS1} [Hz]	f_{HS2} [Hz]	f_{LP} [Hz]	g
1.2 cm	1200	1500	9500	0.85
1.9 cm	900	7000	10200	0.87
2.5 cm	900	7000	11000	0.90

propagation in the tube. The computed reflection is finally added to the delayed sound resulting from unperturbed propagation in the tube.

The measured reflections have extremely low values in the high end of the spectrum (above 3 kHz) because of the long distance travelled. The simulation produces lower values in the high frequency region than the measured values. The extremely low values superimposed by noise produce unreliable results in this region of the spectrum. Since a steeper shape in the high frequency side due to high frequencies losses were expected, an approximation of the differences found with a continuous slope was done.

In order to approximate $H_{\text{ref}}(z)$, a cascade of a 2^{nd}-order high-shelving filter and a 1^{st}-order low-pass was chosen. Similarly to the propagation filter, by controlling the quality factors, the gains, and the cut-off frequencies, we were able to perform a linear interpolation between different diameters. An additional gain factor g_{ref} was introduced to control the scale for the different sizes. Table 6.3 summarizes the parameters values of the different filters.

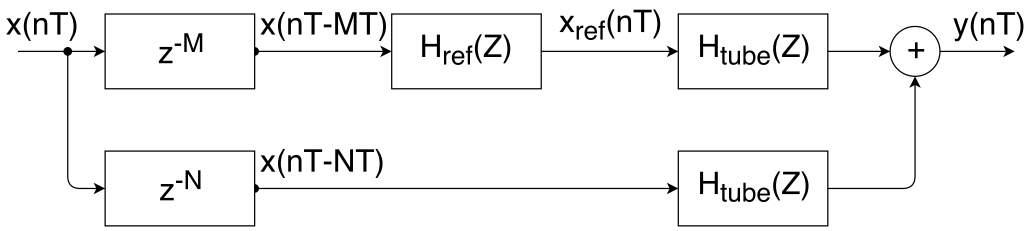

Figure 6.9: *Block diagram of the reflection simulation.*

Table 6.3: *Reflection filter: parameters of the low-pass and high-shelving filter and overall gain for the three tube diameters.*

Type of filter	1.2 cm	1.9 cm	2.5 cm
f_{LP} [Hz]	100	250	600
f_{HS} [Hz]	500	225	160
G_{HS} [dB]	-14	-14	-10
Q_{HS}	0.35	0.4	0.60
g_{ref}	0.4225	0.5180	0.9

Figure 6.10 shows the different filters designed for the three diameter tubes to be compared with those in Fig. 6.3. Fig. 6.11 summarizes the interpolation between the three filters allows to simulate different diameters sizes, from 1.2 to 1.9 cm (top) and 1.9 to 2.5 cm (bottom)

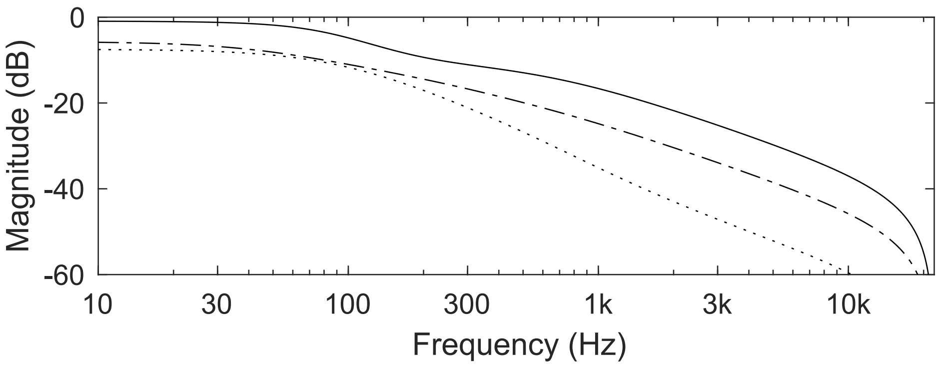

Figure 6.10: *Low-order approximations of the average "difference filters" for the tube end effect (see Eq. (6.2)):* 1.2-*cm (solid line),* 1.9-*cm (dash-dot line), and* 2.5-*cm (dotted line) diameter tubes.*

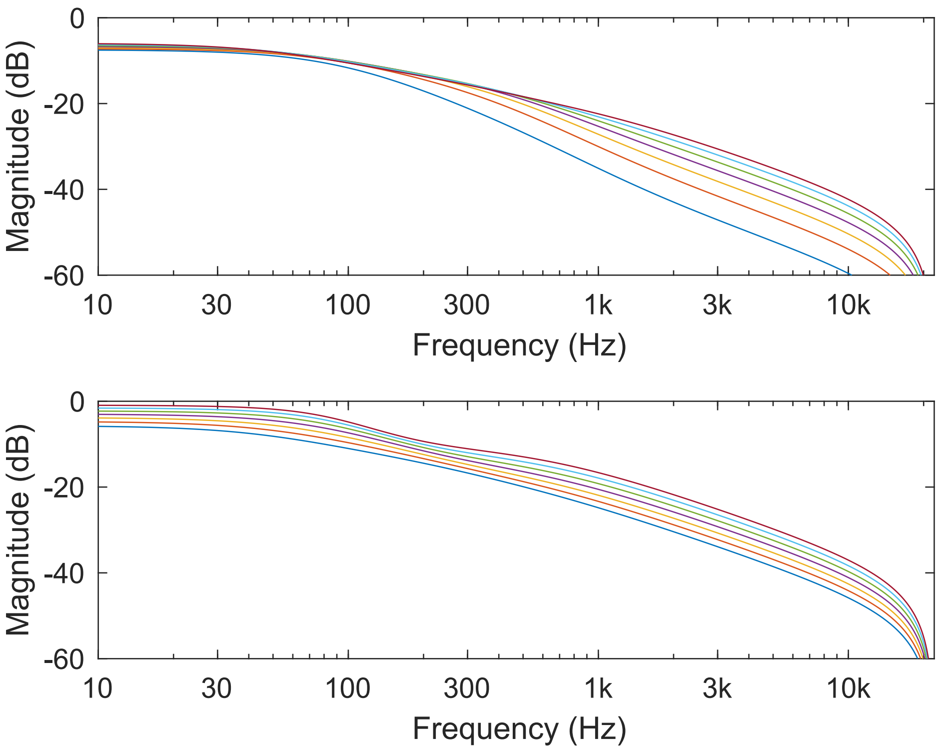

Figure 6.11: *Interpolation of the low-order approximations of the average "difference filters" for the tube end effect,* $1.2 - 1.9$ *cm (top) and* $1.9 - 2.5$ *cm (bottom).*

6.3 Filter Evaluation and Comparison

In this section, a comparison between the designed filters and the measurements is performed. The accuracy of the design is discussed, presenting the maximum approximation error in the frequency range of interest. Considering that the frequencies above 10 kHz are unreliable, as discussed in Subsec. 6.2, the comparison refers the range between 20 Hz and 10 kHz.

Propagation Filter

Figure 6.12 shows the three designed propagation filters compared with the results obtained from the measurements. The filter approximating the 1.2-cm tube has a maximum error of 0.97 dB, which is mainly due to the shelf filter having a flat magnitude response at low frequencies instead of the declining slope of the measured response as shown in the top of Fig. 6.12. This way, a good approximation at high frequencies is obtained, which is considered to be more important that the response below 100 Hz.

The 1.9-cm filter presents a maximum error of 0.5 dB in the lowest part of the frequency range. The fit becomes very accurate at higher frequencies as seen in Fig. 6.12 (middle). The error is 0.31 dB at 60 Hz and decreases close to zero at frequencies above 100 Hz.

The third filter is shown in Fig. 6.12 (bottom) that, with the exception of an anomaly at about 1900 Hz, also fits the target shape with good accuracy. It has a maximum error of 0.5 dB at 6184 Hz, and an error smaller than 0.3 dB in the rest of the frequency range.

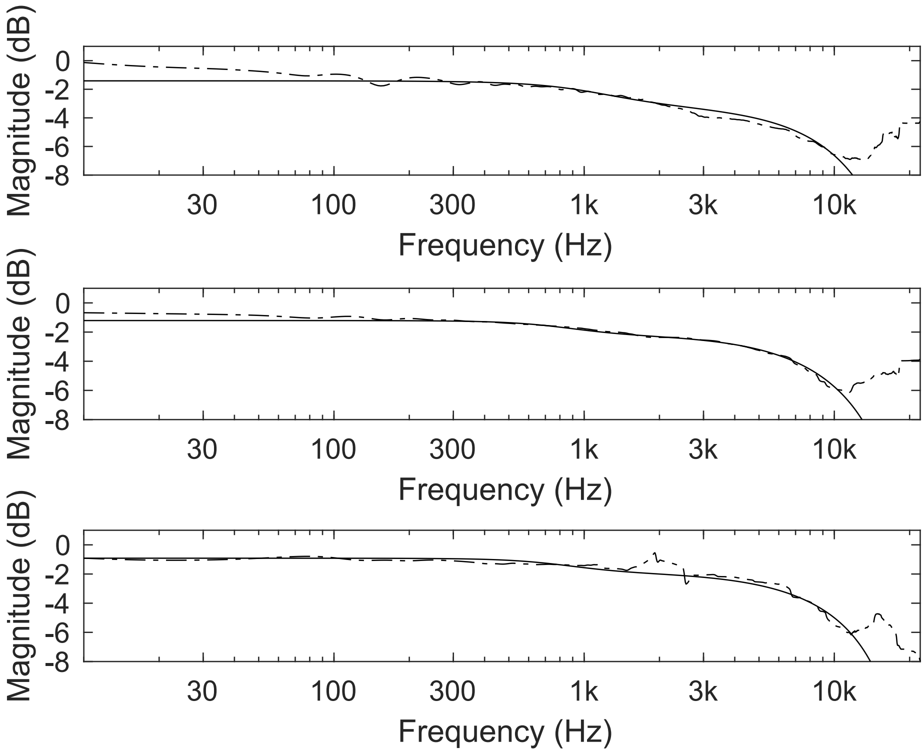

Figure 6.12: *Filters designed (solid line) and their corresponding targets (dash-dot line) for the* 1.2-*cm (top),* 1.9-*cm (middle) and* 2.5-*cm (bottom) tube.*

Reflection Filter

Figure 6.13 shows the difference between the three designed reflection filters and the simulation results. In this case, the range between 20 and 500 Hz is significant for the comparison as discussed in Sec. 6.1.

The filter for the 1.2-cm diameter tube, shown in the top of Fig. 6.12, presents the same initial behaviour of the one compared in the previous section. Because of the high variability in the magnitude target, it is difficult to approximate accurately the shape, and the maximum error is 4.57 dB. The error becomes smaller than 1 dB after 60 Hz except for a deviation at 330 Hz where the error is 3.57 dB. Also in this design, a better approximation for frequencies higher than 60 Hz at the expense of the frequencies below was done.

The reflection filter for the 1.9-cm tube can be seen in the middle of Fig. 6.13. In the beginning of the spectrum, it has a maximum error of 1.26 dB. The error becomes smaller than 1.2 dB above 30 Hz, thus providing a good fit in the remaining range.

The third filter, as seen in Fig. 6.13 (bottom), is the most accurate with a maximum error of 0.52 dB at 40 Hz and close to zero above 100 Hz.

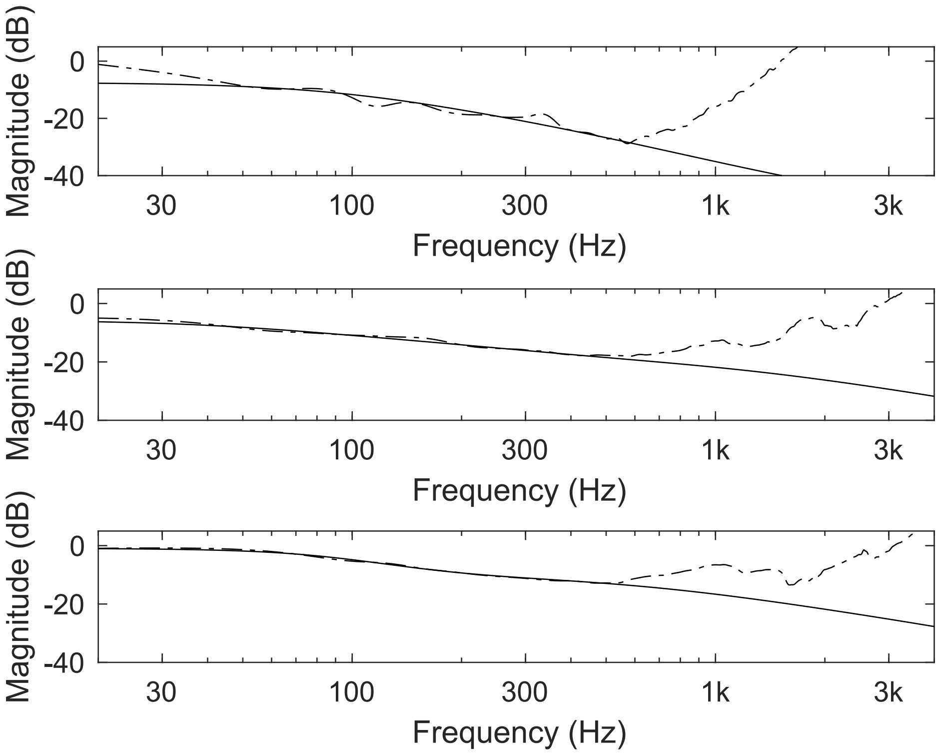

Figure 6.13: *Filters designed for the reflection (solid line) and their corresponding targets (dash-dot line) of the* 1.2*-cm (top),* 1.9*-cm (middle) and* 2.5*-cm (bottom) size tube.*

Chapter 7

The Virtual Delay Tube Effect Plugin

The implementation was written in C++ as an external library for PURE DATA, an open-source real-time environment for audio processing. The complete code can be found in the Sec. D.5 of the Appendix D, where it is also described the architectural details for writing PURE DATA externals.

In this chapter, the plugin is initially presented, describing its features and functions. Then the performed digital signal processing is showed and commented.

7.1 The Virtual Delay Tube Effect$\sim$

The stereo plugin, working at sample rate 44.1 kHz, simulates the wave propagation in a narrow tube and produces associated audio effects. It creates two virtual tubes, one for each channel. The diameter of the two tubes is always the same. The length of each tube can be set by the user and determines the desired delay in milliseconds.

The speed of sound is assumed to be 345 m/s corresponding to a temperature of 23°C. In addition, it is possible to control the volume of the delayed sound and the ratio of the dry and the wet signals in the output. The filter simulates the tube length for each 1 cm added. However, the size parameter gives the possibility to change the virtual tube diameter with a granularity of 1 mm by changing the filter parameters.

To enrich the system, the possibility of summing a reflection in the output was also implemented. This option simulates the wave reflection due the open end of the tube. A reflection, whose frequency content depends on the distance chosen for the "virtual open end," can be created for each virtual tube. This way, the length of the virtual tube becomes the sum of the length chosen for the delay effect and the length chosen in the reflection options. The sound is captured at a virtual microphone at the distance selected by combining the delayed part of the sound and the reflection coming from the end of the tube. Since the reflection captured this way is too soft to be clearly audible, a gain control was added.

Including the reflection option, the system computes three filters: the filter simulating the length desired for the main delay, the filter simulating the open end, and the one simulating the residual length travelled by the sound to reach the end of the tube and come back to meet the virtual microphone. The block scheme shown in Fig. 7.1 summarizes the system. The residual length is represented by $G_{\mathrm{tube}}(z)$ and is twice the length chosen in the reflection options. In order to decrease the complexity of the computation, the different coefficients of the reflection filters were pre-computed and stored.

The plugin offers the possibility to create virtual tubes up to 30 m long in default mode, and

40 m long tubes in the reflection mode. These maximums correspond to a delay of 87 ms and a reflection coming after 29 ms. Figure 7.2 shows a screenshot of the plugin implemented in Pure Data.

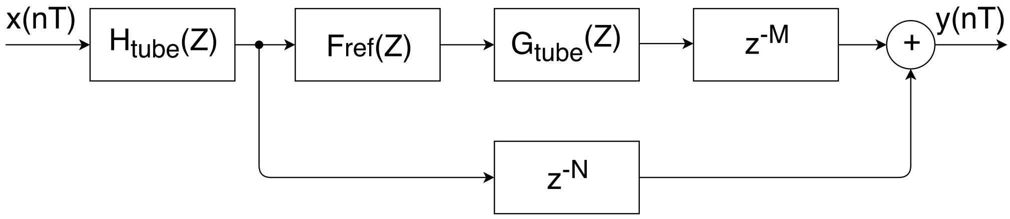

Figure 7.1: *Block scheme for the audio flow in the plugin.*

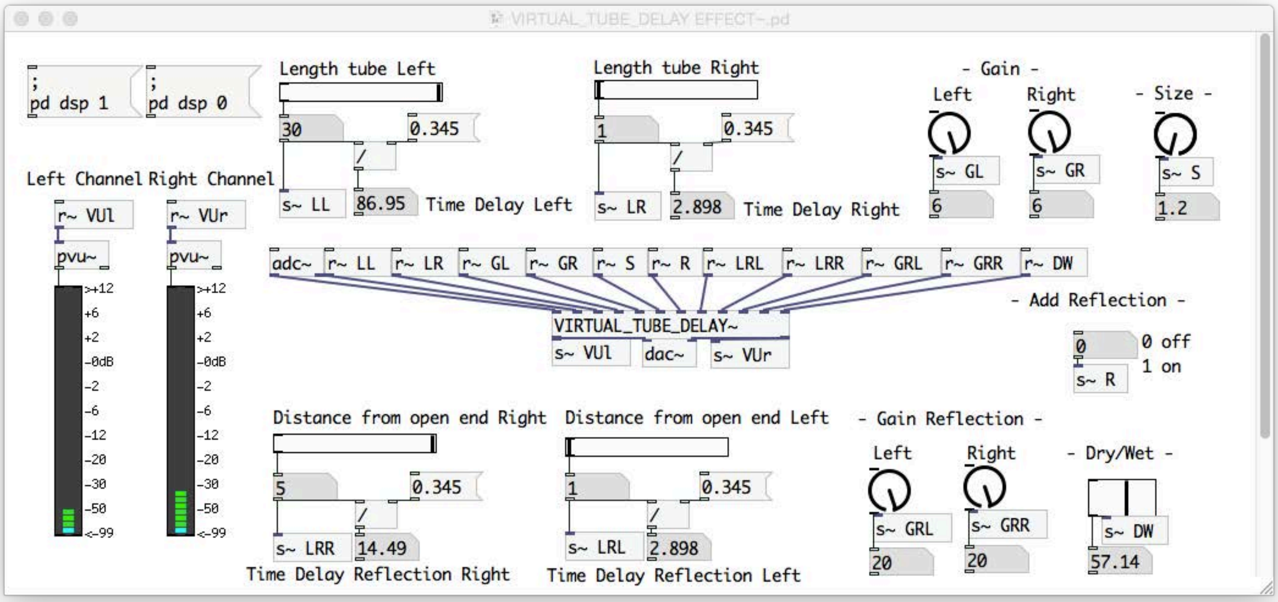

Figure 7.2: *The virtual tube delay effect plugin in* Pure Data.

7.2 Implementation

The method `void process(...)`, called in the perform routine, performs the digital signal processing and the main functions of the plugin. The arguments of the method are the input signal, the input signal length, the others inlet and outlet values. Inside this method, the inlet values are manipulated in order to compute the filters, delays and in turn, the output signals. In order to simplify, the code will be discussed for a mono channel case. To implement a stereo plugin needs only to perform the follows steps for both the channel output.

```
void process(const float* input, float* lengthL, float* lengthR, float* gainL,
    float* gainR, float* size, float* ref, float* lengthRefL, float* lengthRefR,
    float* gainRefL, float* gainRefR, float* dry_wet, float* outputR, float*
    outputL, int n)
  {
    ...

    return;
  } //end process cycle
```

From the input parameters, as the selected lengths for the virtual tube, including the values for both right and left channel, the different delays in samples are computed. Taking the floor of lengths (in centimeters), the delays in milliseconds are computed by dividing them with the sound velocity (345 m/s). Finally, the values in samples are obtained by multiplying the computed times with the sampling period (44.1 kHz) and then casting them in an integer. In a variable (called `rad`) is stored the information about the tube size. Firstly, the input range [1.2 2.5] is transformed in the range [0 13]. The following pieces of code implements the discussed computation, described mathematically by

$$delaySamples = \frac{\lfloor (length * 100)) \rfloor}{100} \frac{F_s}{c}, \qquad sizeLevel = \lfloor (sizeTube * 10 - 12) \rfloor \tag{7.1}$$

where c is the sound velocity and F_s the sampling period. In the case of the length chosen in the reflection options, the delay is doubled since it has to be considered both the time to reach the end of the tube and the time to come back.

```

  //delay time = length/c
  int lengt = floor(length[n-1]*100);

  double leng = lengt/100;

  int lengtRef = floor(lengthRef[n-1]*100);

  double lengRef = lengtRef/100;

  double length_tot = 2*lengRef;

  int stepLength = floor(leng);

  int stepLengthRef = floor(length_tot);

  int delayMilliseconds = leng/0.345;  // time delay

  int delayMillisecondsRef = lengRef/0.345;

  //number of samples = delay time * f
  //sample rate = 44100 Hz

  int delaySamples = (int)((float)delayMilliseconds * 44.1f);
```

```

int delaySamplesRef = (int)((float)delayMillisecondsRef * 44.1f);

int sizeLevel = floor(size[n-1]*10 - 12);

...
```

Subsequently, using the length and size values, the different filters are computed. The system computes the coefficients for one filter: the filter simulating the length desired for the main delay, and the one simulating the residual length travelled by the sound to reach the end of the tube and come back. Since the coefficients for the filter simulating the open end are pre-computed and already stored, the system simply retrieved these stored coefficients according the size value.

Using the coefficients for the low-pass and high-shelving filters, found using the formulas in [Dut+11], the coefficients for the propagating filter, described in Sec. 6.1, are computed. Considering the basic filters in Eqs. (6.4) and (6.7), the coefficients for the propagation filter are the followings:

$$h_{d,0} = 1 \qquad h_{d,1} = a_1 + c_1 + e_1 \qquad h_{d,2} = a_2 + c_1a_1 + c_2 + e_1a_1 + e_1c_1 + e_2 \tag{7.2}$$

$$h_{d,3} = c_1a_2 + c_2a_1 + e_1a_2 + e_1c_1a_1 + e_1c_2 + e_2a_1 + e_2c_1 \tag{7.3}$$

$$h_{d,4} = c_2a_2 + e_1c_1a_2 + e_1c_2a_1 + e_2a_2 + e_2c_1a_1 + e_2c_2 \tag{7.4}$$

$$h_{d,5} = e_1c_2a_2 + e_2c_1a_2 + e_2c_2a_1 \qquad h_{d,6} = e_2a_2c_2 \tag{7.5}$$

$$h_{n,0} = f_0d_0b_0 \qquad h_{n,1} = f_0d_0b_1 + f_0d_1b_0 + f_1d_0b_0 \tag{7.6}$$

$$h_{n,2} = f_0d_0b_2 + f_0d_1b_1 + f_0d_2b_0 + f_1d_0b_1 + f_1d_1b_0 + d_0b_0f_2 \tag{7.7}$$

$$h_{n,3} = f_0d_1b_2 + f_0d_2b_1 + f_1d_0b_2 + f_1d_1b_1 + f_1d_2b_0 + f_2d_1b_0 + f_2d_0b_1 \tag{7.8}$$

$$h_{n,4} = d_2b_2f_0 + f_1d_1b_2 + f_1d_2b_1 + d_2b_0f_2 + f_2d_1b_1 + f_2d_0b_2 \tag{7.9}$$

$$h_{n,5} = f_1d_2b_2 + f_2d_2b_1 + f_2d_1b_2 \qquad h_{n,6} = f_2d_2b_2 \tag{7.10}$$

where the apices b_i/a_i, d_i/c_i and f_i/e_i denote the coefficients of the low-pass and high-shelving filters, $h_{d,i}$ and $h_{n,i}$ are the denominator and numerator of the 6^{th} order IIR filter.

The next step consists of the filtering of the input with the filters made. In the no reflections mode (it is indicated by the vector $ref[n-1]$, $0 =$ no reflections mode and $1 =$ reflections mode), only two propagation filter is needed (considering the left and right channel). The filtering is performed in the for-loop, for each input sample the correspondingly output sample is computed. With this aim, two vector are used as buffer: $x[n]$ and $y[n]$. $x[n]$ stored the values passed from the input signal to the plugin and $y[n]$ stored the output given by filter $x[n]$ with the propagation filter. Since the filter order, only the first 6 past values of the $x[n]$ and $y[n]$ are needed. This gives the length of the vectors. In $y[0]$ and $x[0]$ are stored the values for the current time and in $y[i]$, $x[i]$ with $i = 1 \ldots 6$ there are the past samples. The filtering step can be described by

$$\begin{aligned} y[0] = b_0x[0] + b_1x[1] + b_2x[2] + b_3x[3] + b_4x[4]b_5x[5] + b_6x[6] \\ - a_1y[1] - a_2y[2] - a_3y[3] - a_4y[4] - a_5y[5] - a_6y[6] \end{aligned} \tag{7.11}$$

where a_i, b_i the denominator and numerator coefficients of the propagation filter.

The output of this operation, $y[0]$, is stored in another intermediate buffer, called $buffer[n]$, that is necessary to implement the delay operations. The buffer length needed must be greater or equal to the maximum delay in samples allowed by the plugin. In $y[0]$ is stored in the position given by the delay samples required and previously computed. The filtered output will be $buffer[0]$. More specifically, the plugin output is given by a percentage of the input dry sample and the value in $buffer[0]$ multiplied by a gain value g.

$$\begin{aligned} buffer[delaySamples] &= y[0] \\ output[i] &= dry * input[i] + wet * g * buffer[0] \end{aligned} \tag{7.12}$$

where $output[i]$ is the output buffer, $input[i]$ the input one. dry/wet are two parameters with values in [01] and such as $dry + wet = 1$. Once passed the value in the plugin output buffer, $x[n]$, $y[n]$ and $buffer[n]$ are updated translating their values of one position.

```
//DSP cycle
for (int i = 0; i < n; ++i)
{

  in[0] = (t_sample)input[i];

  if (ref[n-1] == 0) {

    out[0] = b0f*in[0] + b1f*in[1] + b2f*in[2] + b3f*in[3] + b4f*in[4] + b5f*in[5]
    + b6f*in[6] - a1f*out[1] - a2f*out[2] - a3f*out[3] - a4f*out[4] - a5f*out[5] -
    a6f*out[6];

    buffer[delaySamples] = out[0];

    output[i] = dry*(t_sample)input[i] + wet*g*buffer[0];

    for (int k = 5; k >= 0; k--) {

      out[k+1] = out[k];
      in[k+1] = in[k];

    }

    in[0] = 0;
    out[0] = 0;

    for (int k = 0; k < len; k++) {

      buffer[k] = buffer[k+1];
    }
```

```

    buffer[len] = 0;
```

When $ref[n-1]$ is equal to 1, the plugin will include the reflections in the produced output sound. Thus, the input samples have to be filtered by three different filters. The main delay is computed as already discussed. The reflection, instead, is given by further filtering, and in addition, it has a own different delay. Other two vectors are used, $u[n]$ and $w[n]$. The first collects the intermediate values after the filtering to simulate the total distance travelled by the reflection, and the latter one represents the signal also consisting of the open end effect. The vectors $u[n]$ and $w[n]$ are given by

$$\begin{aligned} u[0] = b_0 y[0] + b_1 y[1] + b_2 y[2] + b_3 y[3] + b_4 y[4] b_5 y[5] + b_6 y[6] \\ - u_1 y[1] - u_2 y[2] - u_3 y[3] - u_4 y[4] - u_5 y[5] - u_6 y[6] \end{aligned} \tag{7.13}$$

and

$$w[0] = br_0 u[0] + br_1 u[1] + br_2 u[2] + br_3 u[3] - ar_1 w[1] - ar_2 w[2] - ar_3 w[3]; \tag{7.14}$$

where br_i and ar_i are the reflection filter coefficients.

Then, $y[0]$ and $w[0]$ are passed in $buffer[n]$ according their delay:

$$\begin{aligned} buffer[delaySamples] = buffer[delaySamples] + g * y[0] \\ buffer[delaySamplesTot] = buffer[delaySamplesTot] + g_{Ref} * w[0] \end{aligned} \tag{7.15}$$

where $delaySamplesTot$ and g_{Ref} denote the delay in samples and gain valuer for the reflection. Finally, the plugin output has the same form of the no reflection mode and all the buffers are updated.

```
}else if (ref[n-1] == 1) {

  out[0] = b0f*in[0] + b1f*in[1] + b2f*in[2] + b3f*in[3] + b4f*in[4] + b5f*in[5] +
    b6f*in[6] - a1f*out[1] - a2f*out[2] - a3f*out[3] - a4f*out[4] - a5f*out[5] -
    a6f*out[6];

  out2[0] = b0fRef*out[0] + b1fRef*out[1] + b2fRef*out[2] + b3fRef*out[3] + b4fRef*
    out[4] + b5fRef*out[5] + b6fRef*out[6] - a1fRef*out2[1] - a2fRef*out2[2] -
    a3fRef*out2[3] - a4fRef*out2[4] - a5fRef*out2[5] - a6fRef*out2[6];

  out3[0] = br[0]*out2[0] + br[1]*out2[1] + br[2]*out2[2] + br[3]*out2[3] - ar[0]*
    out3[1] - ar[1]*out3[2] - ar[2]*out3[3];

  buffer[delaySamples] = buffer[delaySamples] + g*out[0];

  buffer[delaySamples_tot] = buffer[delaySamples_tot] + gRef*out3[0];

  output[i] = dry*(t_sample)input[i] + wet*(buffer[0]);

  for (int k = 5; k >= 0; k--) {

    out[k+1] = out[k];
    in[k+1] = in[k];
    out2[k+1] = out2[k];

  }

  for (int k = 2; k >= 0; k--) {

    out3[k+1] = out3[k];
```

```

    }

  for (int k = 0; k < len; k++) {

    buffer[k] = buffer[k+1];

  }

  buffer[len] = 0;

  }// end if ref

}//end dsp cycle
```

Chapter 8

Conclusion

Delay effect is experienced everyday and in any acoustical spaces. It has an important contribution in the spatial image of the sound. Long tubes are popular tools used to produce this kind of audio effect. Since the presence of the walls, the propagation in a tube is affected by energy losses that, with the time delay given by the distance travelled by the waves, creates a particular coloration in the sound. As presented in the beginning of this work, these particular sonic features were exploited in designing analog spatial audio effect. Plastic tubes were the most used for this purpose. The digital imitation of this analog audio system, inspired by old analog effects, was the aim of these studies. The virtual analog modelling carried out, could be employed in the ongoing digitization trend of all equipment used in music production. In addition, the results of this thesis could be also employed for the distance simulation in a virtual environment and in the non-linear system modelling.

Taking in account these motivation, by performing several measurements and analysis with MATLAB®, a simulation of the audio effect produced by these kind of media was done. Therefore, this work proposed a tube delay model and plugin. Acoustic wave propagation in garden hoses of three different diameter was analyzed and the delay effect caused by these long narrow tubes was reproduced. The investigation consisted of studying and elaborating the recorded tube responses, obtained by measuring the different tubes with the Farina's Method. A virtual tube model, in this way, was developed and digital IIR filters were designed in order to simulate the energy losses due to the propagation through the tube and the reflection caused by the tube-end. A 6^{th} IIR filter controlling frequency attenuations according the length and the diameter of the virtual tube was estimated with a negligible error. The parametric filter was designed in which the tube diameter and length can be continuously varied. Because of the simplicity of the magnitude response shapes, a cascade of two high shelving filters and a low-pass filter was sufficient for approximating the behaviour correctly. The filter is able to approximate in an accurate way distance up to 30 m and sizes in the range [1.2 2.5] cm. Hence, the connection in series of the parametric filter with a delay line is able to simulate the propagation delay of a long narrow tube. In addition, an analysis on the reflection due to the open end of the tube was conducted and added to the model. A filter approximating the open end effect of a tube was computed, resulting in a 3^{th} IIR parametric filter controlling the frequency content pushed back, by hitting the atmospheric pressure at the tube-end, according the diameter of the virtual tube. Then, besides the filter simulating the length desired for the main delay, the filter simulating the open end was built and connected in series with a delay line as well. The final result was a digital waveguide model.

Finally, the model was implemented using the C++ language and in PURE DATA enviroment. A stereo delay effect plugin simulating the studied system was the final product of this thesis work.

Three filters, extrapolated from the measurements, are needed: a filter simulates the length desired for the main delay, filter simulates the open end, and another one the residual length travelled by the sound to reach the end of the tube and come back to meet the listening point. The delay lines, instead, are necessary to simulate the propagation time delays.

Appendix A

Waveguide Modelling

Digital waveguide models are computational physical models which consist of delay lines, digital filters, and/or non-linear elements [Ava01]. They models sampled acoustic travelling waves and follow geometry and physical properties of the acoustic system. In addition, losses and dispersion effects of the real system are consolidated at sparse points along each waveguide. Waveguide Modelling is mainly used for synthesis of string and wind musical instruments and artificial reverberation.

A digital delay line is used to model the acoustic propagation delay. It is a functional unit, which allows the input to be delayed by a number of samples. It represents a fundamental building block of delay effects and digital waveguide models. The function of a delay line is to introduce a time delay between its input and output. As it can be seen in Fig. A.1, the delay by N samples is denoted by z^{-N}.

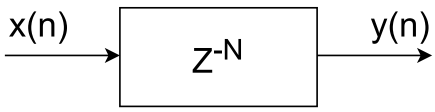

Figure A.1: *Block diagram of the N-sample delay line.*

Taking the input signal $x(n)$, $n = 0, 1, 2, \ldots$, and the delay line length of N samples, then the output signal $y(n)$ is specified by the following relation

$$y(n) = x(n - N), \quad n = 0, 1, 2, \ldots, \tag{A.1}$$

where $x(n) = 0$ for $n < 0$.

A lossless digital waveguide is a bidirectional delay line, where each delay line unit contains a sampled travelling wave component.

Waveguide models exploit the analytical solution to the D'Alembert wave equation, which can be seen as a superposition of travelling waves and which is simulated in the discrete space-temporal domain using delay lines.

A.1 Discretization

Considering the acoustic pressure p for a cylindrical bore, the analytical solution has the form

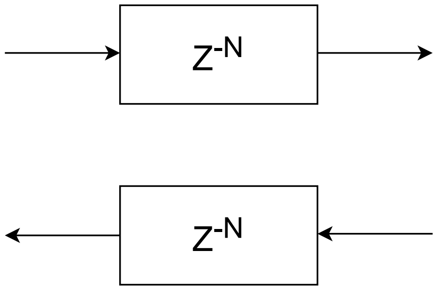

Figure A.2: *Block diagram of a sampled travelling-wave simulation for an ideal string or acoustic tube.*

$$p(x,t) = -(\rho_{air}/S)\frac{\partial p}{\partial x}(x,t) = -(\rho_{air}c/S)\left[\frac{\partial p^+}{\partial x}(ct-x)+\frac{\partial p^-}{\partial x}(ct+x)\right] = \frac{\rho_{air}c}{S}\left[\dot{p}^+(ct-x)+\dot{p}^-(ct+x)\right]. \tag{A.2}$$

where S the constant cross-sectional area of the bore and the equation pressure waves $p^{\pm}$ can be defined as $p^{\pm} = \mp\frac{\rho_{air}c}{S}\dot{p}^{\pm}$. The flow u (volume velocity) is given by

$$u(x,t) = \frac{\partial p^+}{\partial x}(ct-x) + \frac{\partial p^-}{\partial x}(ct+x). \tag{A.3}$$

From the previous equations, defining $Z_0 = \rho_{air}c/S$ the wave impedance, it follows that

$$p^{\pm}(ct \mp x) = \pm Z_0 u^{\pm}(ct \mp x). \tag{A.4}$$

and, in turn, the following relations hold:

$$p = p^+ + p^-, \quad u = \frac{1}{Z_0}[p^+ - p^-], \tag{A.5}$$

$$p^+ = \frac{p + Z_0 u}{2}, \quad p^- = \frac{p - Z_0 u}{2}, \tag{A.6}$$

that transform the pair (p, u) into the pair (p^+, p^-), and vice versa.

Finally, the discretization can be performed considering a pressure distribution $p = p^+ + p^-$, inside an ideal lossless cylindrical bore and the sampling period T_s. Taking the spatial sampling step as $X_s = cT_s$ and the time sampling T_s itself, the discretized version of p is obtained through the variable substitution $x \mapsto mX_s$ and $t \mapsto nT_s$ (with $m, n \in \mathbb{N}$). It leads to

$$p(mX_s, nT_s) = p^+(ncT_s - mX_s) + p^-(ncT_s + mX_s) = p^+((n-m)cT_s) + p^-((n+m)cT_s), \tag{A.7}$$

that removing the constant sampling steps yields

$$p[m,n] = p^+[n-m] + p^-[n+m]. \tag{A.8}$$

The term $p^+[n-m]$, thus, will be the output from a digital delay line of length m, whose input is $p^+[n]$. Analogously for the the term $p^-[n+m]$. As introduced above, this leads to the definition of a waveguide as a bidirectional delay line, where the horizontal direction of the structure corresponds to the position x along the axis of the cylindrical bore.

Reflection Conditions

Considering a finite-length cylindrical bore, the reflection conditions have be taken in account. The reflection conditions are derived by formulating boundary conditions. Assuming a cylindrical bore of length L, with a closed end at $x = 0$ and an open end at $x = L$, the first condition implies $u = u^+ + u^- = [p^+ - p^-]/Z_0 = 0$ at $x = 0$. It means no flow through a closed end and implies the reflection conditions $u^+ = -u^-$ and $p^+ = p^-$. The second condition, instead, implies $p = p^+ + p^- = 0$ at $x = L$. In this case, p matches the atmospheric pressure at the open boundary, which implies the reflection conditions $p^- = -p^+$ and $u^+ = u^-$.

Real Systems

Dealing with real systems, more complex behaviours have to be described. Particularly relevant for sound production, there are two phenomena: dissipation and dispersion. Both can be accounted for by adding time, space or time-space derivatives of different orders to the ideal wave equation. Correspondingly the waveguide model is modified by inserting appropriate loss and dispersion filters in the loop. Therefore, physical phenomena such as frequency dependent losses and dispersion can be included in the models by incorporating filters describing the phenomena in the delay line scheme.

Appendix B

Phase and Group Delay

The phase response of an LTI filter, identified as $\Theta(\omega)$, denotes the radian phase shift added to the phase of each sinusoidal component of the input signal [Smi07]. The phase delay, often used because more intuitive, is defined as

$$P(\omega) = -\frac{\Theta(\omega)}{\omega}. \tag{B.1}$$

The phase delay gives the time delay experienced by each sinusoidal component of the input signal. Considering an input $x(n) = \cos(\omega nT)$ and a frequency response $H(e^{j\omega T}) = G(\omega)e^{j\Theta(\omega)}$, the output will be given by

$$\begin{aligned} y(n) &= G(\omega)\cos[\omega nT + \Theta(\omega)] \\ &= G(\omega)\cos\{\omega[nT - P(\omega)]\}, \end{aligned}$$

where can be seen that the phase delay expresses the phase response as a time delay.

The phase response is usually "unwrapped", that means to include all multiples of 2π in $\Theta(\omega)$. To be noted that $\Theta(\omega)$ is the complex angle of the frequency response $H(e^{j\omega T})$, thus, it is not sufficient for obtaining a phase response which can be converted to time delay. Discarding the multiples of 2π, the phase delay is modified by multiples of the sinusoidal period. In order to perform a filter analysis, it is often useful to define the filter phase response as a continuous function with $\Theta(0) = 0$ or π. When the amplitude response goes to zero or infinity at some frequency, unwrapping the phase response a limit from below and above that frequency is taken. A numerical algorithm for phase unwrapping is implemented in Matlab with a function called *unwrap()*.

A common representation of filter phase response is called the group delay and it is defined by

$$D(\omega) = -\frac{d}{d\omega}\Theta(\omega) = -\frac{d}{d\omega}\angle H(e^{j\omega T}) \tag{B.2}$$

In the case of linear phase responses, $\Theta(\omega) = -\alpha\omega$ with α a constant, the group delay is equal to the phase delay, and it may be interpreted as time delay. The time delay will be equal to α samples when $\omega \in [-\pi, \pi]$.

For non-linear phase response, instead, the relative phases of the sinusoidal signal components are generally altered by the filter, causing phase distortion. This type of phase distortion is called phase dispersion and normally "smear" attack transients, such as in percussive sounds.

The group delay $D(\omega)$ may be interpreted as the time delay of the amplitude envelope of a sinusoid at frequency ω. The bandwidth of the amplitude envelope is restricted to a frequency interval over which the phase response is approximately linear. Therefore, $D(\omega)$ specifies the delay experienced by a narrow-band group of sinusoidal components which have frequencies within a narrow frequency interval about ω. The width of the interval is limited to that over which $D(\omega)$ is approximately constant.

B.1 Numerical Computation

In order to compute the group delay, the frequency response $H(e^{j\omega T})$ is expressed in polar form

$$H(e^{j\omega T}) = G(\omega)e^{j\Theta(\omega)}, \tag{B.3}$$

and then, decomposed in the logarithmic of magnitude and phase:

$$\ln H(e^{j\omega T}) = \ln G(\omega) + j\Theta(\omega). \tag{B.4}$$

The real part of the logarithm of the frequency response equals the log amplitude response, while the imaginary part equals the phase response. Since differentiation is linear, the logarithmic derivative becomes

$$\frac{d}{d\omega}\ln H(e^{j\omega T}) = \frac{G'(\omega)}{G(\omega)} + j\Theta'(\omega), \tag{B.5}$$

where $G'(\omega)$ and $\Theta'(\omega)$ denote the derivatives of $G(\omega)$ and $\Theta(\omega)$, respectively, with respect to ω. Finally, the group delay can be express as

$$\begin{aligned} D(\omega) &= -\frac{d}{d\omega}\Theta(\omega) \\ &= -im\left\{\frac{d}{d\omega}\ln H(e^{j\omega T})\right\} \\ &= -im\left\{\frac{H'(e^{j\omega T})}{H(e^{j\omega T})}\right\}. \end{aligned}$$

Considering, the FIR case in which $H(z) = B(z)$, with $B(z) = b_0 + b_1 z^{-1} + b_2 z^{-2} + \cdots + b_M z^{-M}$, the derivative is given by

$$\begin{aligned} B'(e^{j\omega T}) &= \frac{d}{d\omega}\left[b_0 + b_1 e^{-j\omega T} + b_2 e^{-j2\omega T} + b_M e^{-jM\omega T}\right] \\ &= -jT\left[b_1 e^{-j\omega T} + 2b_2 e^{-j2\omega T} + Mb_M e^{-jM\omega T}\right] \\ &= -jT\, B_r(e^{j\omega T}), \end{aligned}$$

where $B_r(z)$ denotes B ramped, i.e., the ith coefficient of the polynomial B_r is $i\,b_i$, for $i = 0, 1, 2, \ldots, M$. In conclusion, the group delay of an FIR filter $B(z)$ can be expressed by

$$\begin{aligned} D(\omega) &= -im\left\{\frac{B'(e^{j\omega T})}{B(e^{j\omega T})}\right\} \\ &= -im\left\{-jT\frac{B_r(e^{j\omega T})}{B(e^{j\omega T})}\right\} \\ &= T\,re\left\{\frac{B_r(e^{j\omega T})}{B(e^{j\omega T})}\right\} \end{aligned}$$

In the case of IIR filter, instead, there are both poles and zeros

$$H(e^{j\omega T}) = \frac{B(e^{j\omega T})}{A(e^{j\omega T})}, \tag{B.6}$$

where

$$A(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_N z^{-N}. \tag{B.7}$$

This time, the differentiation yields

$$\frac{H'}{H} = \frac{(B/A)'}{(B/A)} = \frac{B'A - BA'}{BA}, \tag{B.8}$$

and a faster algorithm results from converting the IIR case to the FIR case:

$$C(z) = B(z)\left[z^{-N}\overline{A}(1/z)\right] = B(z)\tilde{A}(z) \tag{B.9}$$

where

$$\tilde{A}(z) = z^{-N}\overline{A}(1/z) \tag{B.10}$$

Finally, knowing that

$$\angle\tilde{A}(e^{j\omega T}) = -\angle A(e^{j\omega T}) - N\omega T, \tag{B.11}$$

the phase of the IIR filter can be computed by

$$\begin{aligned}
\angle H(e^{j\omega T}) &= \angle B(e^{j\omega T}) - \angle A(e^{j\omega T}) \\
&= \angle B(e^{j\omega T}) - \angle\tilde{A}(e^{j\omega T}) + N\omega T \\
&= \angle C(e^{j\omega T}) + N\omega T,
\end{aligned}$$

and the group delay by

$$D(\omega) = -\frac{d}{d\omega}\angle C(e^{j\omega T}) - NT = T\,re\left\{\frac{C_r(e^{j\omega T})}{C(e^{j\omega T})}\right\} - NT.$$

This method is implemented in the Signal Processing Toolbox of Matlab with the function `grpdelay`.

```

function [gd,w] = grpdelay(b,a,nfft,whole,Fs)

  if (nargin<1 || nargin>5)
    usage("[g,w]=grpdelay(b [, a [, n [,'whole'[,Fs]]]])");
  end
  if nargin<5
    Fs=0; % return w in radians per sample
    if nargin<4, whole='';
    elseif ~isstr(whole)
      Fs = whole;
      whole = '';
    end
    if nargin<3, nfft=512; end
    if nargin<2, a=1; end
  end

  if strcmp(whole,'whole')==0, nfft = 2*nfft; end
```

```
w = 2*pi*[0:nfft-1]/nfft;
if Fs>0, w = Fs*w/(2*pi); end

oa = length(a)-1;             % order of a(z)
oc = oa + length(b)-1;        % order of c(z)
c = conv(b,fliplr(a));        % c(z) = b(z)*a(1/z)*z^(-oa)
cr = c.*[0:oc];               % derivative of c wrt 1/z
num = fft(cr,nfft);
den = fft(c,nfft);
minmag = 10*eps;
polebins = find(abs(den)<minmag);
for b=polebins
  disp('*** grpdelay: group delay singular! setting to 0')
  num(b) = 0;
  den(b) = 1;
end
gd = real(num ./ den) - oa;

if strcmp(whole,'whole')==0
  ns = nfft/2; % Matlab convention - should be nfft/2 + 1
  gd = gd(1:ns);
  w = w(1:ns);
end

w = w'; % Matlab returns column vectors
gd = gd';
```

Appendix C

Third-Octave Filter

The third-octave filter consists of banks having the bandwidths approximating the measured bandwidths of the auditory filters. Third-octave banks have been internationally standardized for use in audio analysis.

In a third-octave filter bank, the center frequencies of the various bands f_k are defined relative to a bandpass filter centered at $f_0 = 1000$ Hz, by the following formula:

$$f_k = 2^{k/3} f_0 \tag{C.1}$$

The filter is formed by dividing octave band, the upper and lower band edges in the kth band are given by the geometric means

$$f_{kh} = \sqrt{f_k f_{k+1}}, \tag{C.2}$$

and

$$f_{kl} = \sqrt{f_{k-1} f_k}. \tag{C.3}$$

From the above equations, it may be found that the bandwidth of the k-th band is given by

$$B_k = f_k \frac{2^{1/3} - 1}{2^{1/6}}. \tag{C.4}$$

As it can be seen, the bandwidth is proportional to center frequency. The bandwidth B_k is 23.1% of the center frequency f_k. The quality factor of each third-octave band filter is independent of k and the third-octave bank are referred to as constant Q filter banks.

The follow Matlab script shows the function used in order to implements the 1/3 octave smoothing in Ch. 5.

```
function H = mag_smoothing(B,n)

%%
%
% This is a Matlab function that implements the 1/n octave smoothing of the
% input magnitude response B.
```

```
smooth_factor=1/n; % 1/n octave bandwidth
Q=sqrt(2^(smooth_factor))/(2^(smooth_factor)-1); % Q-factor
B1=abs(B); % Only magnitude
H=B1(1:end); % Initialize output vector
q=fix(Q)+1; % pointer to first frequency to be smoothed
for i=q:length(H) % scan each frequency bin

    N=fix(i/Q); % ammount of frequency bins in the smoothing window
    if mod(N,2)==0 % detect if it is even
        N=N+1;
    end;

    fc=(N+1)/2; % center frequency
    fh=N-fc;

    if i+fh>length(H) % detect if there is not enough bandwidth
        f=N-(i+fh-length(H));
    else
        f=N;
    end;

    H(i)=sqrt(sum(B1(i-fh:i+fh-(N-f)).^2)/f); % smoothing
end;
```

Appendix D

Pure Data Externals

Pure Data is an open source visual programming language for multimedia. It is written in the programming language C and due to its graphical nature, Pure Data is a object-oriented system.

Several functions are already built into Pure Data and, in addition, it can be extended with self made primitives (objects) using complex programming-languages, like C/C++. In this way, it is possible to create a patch with a certain desired functionality. These new objects are called externals and, thus, they are classes that are not built into Pure Data but are loaded at runtime.

Pure Data also provides many predefined types, generally starting with `t_`, and summarized in Tab. D.1.

Table D.1: *Pure Data types.*

Pure Data type	Description
t_atom	atom
t_float	floating point value
t_symbol	symbol
t_int	pointer-sized integer value (do not use this for integers)
t_signal	structure of a signal
t_sample	audio signal-value (floating point)
t_outlet	outlet of an object
t_inlet	inlet of an object
t_object	object-interna
t_class	a Pd-class
t_method	class-method
t_newmethod	pointer to a constructor

In order to write an external, a C-compiler that supports the ANSI-C-Standard, like the Gnu C-compiler (gcc) on linux-systems or Visual-C++ on windows-platforms, is necessary. Basic concepts for developing an external can be found in `http://iem.at/pd/externals-HOWTO/` and some examples are also available in `https://github.com/pure-data/externals-howto#table-of-contents`. The guide for Xcode Configuration can be also found in `http://puredata.info/docs/developer/PdExternalsInXcode`.

D.1 Externals

Firstly, a defined interface, provided in the header-file `m_pd.h`, is needed. Then, a new class and the data space for this class has to be defined.

```
typedef struct VIRTUAL_TUBE_DELAY_tilde
{
    t_object     x_ob;
    VIRTUAL_TUBE_DELAYObj *dd;
    float       default_input;
} t_VIRTUAL_TUBE_DELAY_tilde;

...

t_class *VIRTUAL_TUBE_DELAY_tilde_class;
```

In this example, referring to the external proposed in this work, `VIRTUAL_TUBE_DELAY_tilde_class` is the pointer to the new class and the structure `t_VIRTUAL_TUBE_DELAY_tilde`, of the type `VIRTUAL_TUBE_DELAY_tilde`, is the data space of the class. The variable of the type `t_object` is a necessary element of the data space, since it is used to store internal object-properties like the graphical presentation of the object or data about inlets and outlets. This variable has to be the first entry in the structure.

Then, in order to generate the new class, information of the data space and the method space of this class, have to be passed to Pure Data when the library is loaded. During the process of loading a new library (`VIRTUAL_TUBE_DELAY_tilde`), Pure Data tries to call the setup function (`VIRTUAL_TUBE_DELAY_tilde_setup()`), that declares the new classes and their properties. It is only called when the library is loaded and in the case of function-call fails, no external of the library will be loaded. The setup function VIRTUAL_TUBE_DELAY~ external is

```
void VIRTUAL_TUBE_DELAY_tilde_setup(void)

{
  post("VIRTUAL_TUBE_DELAY_tilde_setup");
  // creation of the Virtual_Tube_Delay~ instance
  VIRTUAL_TUBE_DELAY_tilde_class = class_new(gensym("VIRTUAL_TUBE_DELAY~"),
        (t_newmethod)VIRTUAL_TUBE_DELAY_tilde_new,
        (t_method)VIRTUAL_TUBE_DELAY_tilde_delete,
        sizeof(t_VIRTUAL_TUBE_DELAY_tilde),
        0,
        (t_atomtype) 0);

  //sound processing
  class_addmethod(VIRTUAL_TUBE_DELAY_tilde_class,
        (t_method)VIRTUAL_TUBE_DELAY_tilde_dsp,
        gensym("dsp"), A_CANT, (t_atomtype) 0);

  //signal input
  CLASS_MAINSIGNALIN(VIRTUAL_TUBE_DELAY_tilde_class, t_VIRTUAL_TUBE_DELAY_tilde,
        default_input);
}
```

The function `class_new` creates a new class and returns a pointer to it. The first argument of the function is the symbolic name of the class. The following two arguments are the constructor and destructor of the class: `newmethod` is the constructor that creates an instance of the class and returns a pointer to this instance; `freemethod` needs when the memory is reserved dynamically and has to be freed when the object is destroyed. The constructor in the external is `(t_newmethod)VIRTUAL_TUBE_DELAY_tilde_new` and `(t_method)VIRTUAL_TUBE_DELAY_tilde_delete` is the destructor. The allocated memory for the static data space is automatically reserved and freed. The fourth argument is the size of the data structure and enables Pure Data to reserve and free enough memory for the static data space. It is returned by `sizeof(t_VIRTUAL_TUBE_DELAY_tilde)`. The fifth argument is related to the graphical representation of the class objects. The default value is `CLASS_DEFAULT` or simply 0, that defines a normal object with one inlet. Using `CLASS_PD` is defined a object without graphical presentation. The remaining arguments define the types of object-arguments passed at the creation of a class-object. They can be up to six numeric and symbolic object-arguments. The list of arguments is terminated by 0. If more arguments are needed, `A_GIMME` can be used for passing an arbitrary list of arguments. In the example no object-arguments are defined. Summarizing, the function `class_new` is defined as

```
t_class *class_new(t_symbol *name, t_newmethod newmethod, t_method freemethod,
                size_t size, int flags, t_atomtype arg1, ...);
```

In order to manipulate the data, a set of methods have to defined. When a message is sent to an instance of the class, a method is called. Methods are the interfaces to the message system of Pure Data. The function `class_addmethod` adds a method to the class. The first argument defines the class, the second one defines the method. The next argument is the selector. The selector is a symbol that defines the type of a message. The remaining ones define the types of the list of atoms that follow the selector. As for `class_new`, a maximum of six arguments can be passed and if more `A_GIMME` can be used. The list of arguments is terminated by 0. The function is defined as

```
void class_addmethod(t_class *c, t_method fn, t_symbol *sel, t_atomtype arg1, ...);
```

Representing `VIRTUAL_TUBE_DELAY_tilde` a signal class, a class offering methods for signals, it has to declare that it uses signal inlets and for this purpose the `CLASS_MAINSIGNALIN` macro has to be added.

```
CLASS_MAINSIGNALIN(<class_name>, <class_data>, <f>);
```

This macro enables signals at the first inlet, the default one. In the case of more than one signal inlet, they have to be created explicitly in the constructor method. The first argument is a pointer to the class, the second one is the type of the class-data space and the last argument is a dummy floating point variable of the data space. The latter one is necessary in order to automatically convert float messages into signals if no signal is present at the signal inlet.

Whenever Pure Data's audio engine is started, a message with the selector "dsp" is sent to each object. Each class that has a method for the "dsp" message is recognised as signal class. The arguments following the "dsp" selector is usually set `A_CANT` because this make it impossible to manually send an illegal dsp message to the object, triggering a crash.

The function `gensym()` checks if the `C-string *s` has already been inserted into the symbol table and a pointer to the symbol is returned. If no entry exists, it is created.

```
t_symbol *gensym(char *s);
```

As already mentioned, the constructor is defined with the `class_new` function. It has to be of type `void*`. The arguments of the constructor depend on the arguments defined with `class_new`. The constructor has to return a pointer to the instantiated data space. The command `pd_new` is needed to reserves memory for the data space, initialises the variables that are internal to the object and returns a pointer to the data space. The constructor, in this case is defined by

```
void *VIRTUAL_TUBE_DELAY_tilde_new(void)
{
    post("VIRTUAL_TUBE_DELAY_new");

    t_VIRTUAL_TUBE_DELAY_tilde *x = (t_VIRTUAL_TUBE_DELAY_tilde *)pd_new(
    VIRTUAL_TUBE_DELAY_tilde_class);
```

```
    x->default_input = 0;
    x->dd = NULL;

    try
    {
        // call to the constructor
        x->dd = new VIRTUAL_TUBE_DELAYObj();
    }
    catch (int n)
    {
        x->dd = 0;
        post("VIRTUAL_TUBE_DELAY_new catched error ");
        return (void*) 0;
    }

    // delaytime inlet left
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // delaytime inlet right
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gain inlet left
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gain inlet right
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // param inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // size inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // ref inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // lengthRefL inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // lengthRefR inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gainRefL inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gainRefR inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);

    // stereo outlet
    outlet_new(&x->x_ob, &s_signal);
    outlet_new(&x->x_ob, &s_signal);

    return (void *)x;
}
```

The command `pd_new` that generates a new instance of a class and returns a pointer to this instance, is defined as

```
t_pd *pd_new(t_class *cls);
```

In addition, inlets and outlets are added using the `inlet_new`/`outlet_new` command. The routines for inlets and outlets need a pointer to the object-interna of the class instance, that is passed with the first argument. The second argument is a symbolic description of the inlet/outlet type. These functions return a pointer to the new inlet/outlet and saves this pointer in the `t_object` variable `x_obj.ob_inlet`/`x_obj.ob_outlet`. In the case of only one inlet/outlet, the pointer need not to be stored in the data space. Otherwise the pointers have to be stored in the data space, because the `t_object` variable can only hold one inlet/outlet pointer. The `inlet_new` function is the following

```
t_inlet *inlet_new(t_object *owner, t_pd *dest, t_symbol *s1, t_symbol *s2);
```

It generates an additional inlet for the object that is pointed at by owner. Generally, `dest` points at `owner.ob_pd`. The selector `s1` at the new inlet is substituted by the selector `s2` and thus, when a message with selector `s1` appears at the new inlet, the class method for the selector `s2` is called. Therefore, the substituting selector has to be declared by `class_addmethod` in the setup routine. The `outlet_new` function is instead

```
t_outlet *outlet_new(t_object *owner, t_symbol *s);
```

It generates a new outlet for the object that is pointed at by owner. The symbol `s` indicates the type of the outlet.

D.2 DSP Methods

The class `VIRTUAL_TUBE_DELAY~`, as discussed above, is a signal class and this is indicated with the tilde (~) at the end of the symbolic names, that usually identifies the signal classes. Since a signal class, it provide methods for digital signal processing and in order to be able for that, a method for the selector "dsp", followed by no atoms, has to be added. Whenever the audio engine is started, all objects that provide a DSP method are identified as instances of signal classes. A DSP method has the form

```
void my_dsp_method(t_mydata *x, t_signal **sp)
```

where the first argument is the data space of the object and the second one is an array of signals. The signals in the array are arranged from left to right, first the inlets, then the outlets. Tab D.2 shows the array organization in case there are both two in- and out- signals. The structure `t_signal` contains a pointer to the its signal vector `().s_vec`, an array of samples of type `t_sample`, and the length of this signal vector `().s_n`.

Table D.2: *Signals array in case there are both two in- and out- signals.*

Pointer	To signal
$sp[0]$	left in-signal
$sp[1]$	right in-signal
$sp[2]$	left out-signal
$sp[3]$	right out-signal

Table D.3: *Signal vector structure.*

Structure element	Description
s_n	length of the signal vector
s_vec	pointer to the signal vector

In the DSP method, a class method for signal processing is added to the DSP tree by the function `dsp_add`:

```
void dsp_add(t_perfroutine f, int n, ...);
```

It adds the perform routine, passed as first argument, to the DSP tree. The second argument defines the number of following pointer arguments. Pointers to the data space of the object and to the signal vectors are not limited. The length of the signal vectors is also passed to manipulate signals effectively. Since all signal vectors of a patch have the same length, it is sufficient to get the length of one of these vectors.

The perform routine is called at each DSP cycle. A pointer to an integer array is passed to it. This array contains the pointers, that were passed via `dsp_add`, which must be cast back to their real type. The first pointer is stored in $w[1]$.

```
t_int *my_perform_routine(t_int *w)
```

The perform routine has to return a pointer to integer, that points directly behind the memory, where the object's pointers are stored. This means, that the return argument equals the routine's argument plus the number of used pointers, defined in the second argument of `dsp_add`, plus one. When the audio engine is turned on, all signal objects declare their perform routines that are to be added to the DSP tree. Finally, in the `VIRTUAL_TUBE_DELAY`~ class the perform routine is defined as

```
void VIRTUAL_TUBE_DELAY_tilde_dsp(t_VIRTUAL_TUBE_DELAY_tilde *x, t_signal **sp)
{
    dsp_add(VIRTUAL_TUBE_DELAY_tilde_perform,
            16,
            x,
            sp[0]->s_vec, //inlet1
            sp[1]->s_vec, //inlet2
            sp[2]->s_vec, //inlet3
            sp[3]->s_vec, //inlet4
            sp[4]->s_vec, //inlet5
            sp[5]->s_vec, //inlet6
            sp[6]->s_vec, //inlet7
            sp[7]->s_vec, //inlet8
            sp[8]->s_vec, //inlet9
            sp[9]->s_vec, //inlet10
            sp[10]->s_vec, //inlet11
            sp[11]->s_vec, //inlet12
            sp[12]->s_vec, //outlet1
            sp[13]->s_vec, //outlet2
            sp[0]->s_n); //vector size
}
```

In this case, $sp[0] - sp[11]$ represent the in-signals and $sp[12] - [13]$ points to the out-signal.

The perform routine is the main part of the signal class. Each sample of the signal vectors is read and manipulated in the while-loop. Optimisation of the DSP tree tries to avoid unnecessary copy operations. Therefore it is possible, that in- and out-signal are located at the same address in the memory. In this case, in order to avoid overwriting data that is not yet saved, the writing operation into the out-signal has to be after having read the in-signal.

```
t_int *VIRTUAL_TUBE_DELAY_tilde_perform(t_int *w)
{
    t_VIRTUAL_TUBE_DELAY_tilde *x = (t_VIRTUAL_TUBE_DELAY_tilde *)(w[1]);   //obj ref

    // 4 signal input: signal, time, gain, dry/wet and changeF
    t_sample *input = (t_sample *)(w[2]);  //input samples mono
    t_sample *lengthL = (t_sample *)(w[3]);  //time left
    t_sample *lengthR = (t_sample *)(w[4]);  //time right
    t_sample *gainL = (t_sample *)(w[5]);  //gain left
    t_sample *gainR = (t_sample *)(w[6]);  //gain right
    t_sample *size = (t_sample *)(w[7]);   //size
    t_sample *ref = (t_sample *)(w[8]);   //ref
    t_sample *lengthRefL = (t_sample *)(w[9]); //lenght ref left
    t_sample *lengthRefR = (t_sample *)(w[10]); //lenght ref right
    t_sample *gainRefL = (t_sample *)(w[11]);   //gain ref left
    t_sample *gainRefR = (t_sample *)(w[12]);   //gain ref right
```

```
    t_sample   *dry_wet = (t_sample *)(w[13]);   //dry/wet

    // 2 signal output: filtered stereo signal
    t_sample *outputL = (t_sample *)(w[14]); //output samples left
    t_sample *outputR = (t_sample *)(w[15]); //output samples right

    int n = (int)(w[16]);

    x->dd->process(input, lengthL, lengthR, gainL, gainR, size, ref, lengthRefL,
    lengthRefR, gainRefL, gainRefR, dry_wet, outputR, outputL, n);

    return (w+17);
}
```

D.3 Extern "C"

Using the C++ library in Pure Data, the critical aspect is the symbol table. In a mixed C/C++ project is used a C++ compiler, which also compiles the C code. In the object file (.o file) and in built product (`VIRTUAL_DELAY_TUBE~.pd_darwin` in the OSX case), the function symbols appear in the so-called mangled form. This is a problem, except for the case of the setup function in the Pure Data class. The setup function is the entry point for Pure Data setting up the class and that the own symbols have to follow a conventional syntax, otherwise it will be not found. In order to obtain a setup function read as regular symbols of C, it has to be declared as extern "C".

D.4 Hidden Symbols and -fvisibility Flag

In Pure Data classes, most of the function are static, it means they can be called in the file only. Function symbols that have to called from other executable files should be visible for those files. This holds for the setup function. Functions not static are visible in the default case, from other files via static linking, and from executable files via dynamic linking. Others names defining visibility are: `exported`, `global`, `external` and the opposite ones `hidden`, `invisible`, `local`. In the case of multiple file dynamic libraries, symbols visibility cannot be organized at the language level (C or C++). The Compiler/linker options and attributes have to help. The 4.0 version, GCC has a `-fvisibility` flag: `-fvisibility=default` makes visible all the symbols in the target, if not already defined in other way; `-fvisibility=hidden`, instead, hides them, still if not already defined in other way. The flag and attribute syntax are compiler/platform-specific. Hidden symbols are recommended because in the case of few exported symbols, the dynamic linker load the programme faster. In addition, the possibility of collision between symbols is reduced.

D.5 Pure Data External Code

Follow the completed code for the external.

```
//
//  VIRTUAL_TUBE_DELAY~.cpp
//  VIRTUAL_TUBE_DELAY~
//
//  Created by Riccardo Simionato on 10/03/18.
//  Copyright 2018 Riccardo Simionato. All rights reserved.
//

```

```c
#include "m_pd.h"
#include <math.h>
#include <tgmath.h>

const double PI = 3.14159265358979;

//propagation filter parameters values
const double f12 = -42.8571428571429;
const double f12LP = 100;
const double g12 = 0.00285714285714286;
const double f212 = 785.714285714286;

const double g23 = 0.00500000000000000;
const double f23LP = 133.333333333333;

const double fc1_0 = 1200;
const double fcLP_0 = 9500;
const double fc2_0 = 1500;

const double fc1_1 = 900;
const double fcLP_1 = 10200;
const double fc2_1 = 7000;

//reflection filters coefficients values

//step_0 - 1.2 cm
const double B0r_0 = 0.00148875526585464;
const double B1r_0 = -0.00120708383241042;
const double B2r_0 = -0.00148190029224148;
const double B3r_0 = 0.00121393880602358;

const double A1r_0 = -2.89790633258181;
const double A2r_0 = 2.79802595323123;
const double A3r_0 = -0.900105910702194;

//step_1 - 1.3 cm
const double B0r_1 = 0.00335592831460436;
const double B1r_1 = -0.00277817344018852;
const double B2r_1 = -0.00334268955647620;
const double B3r_1 = 0.00279141219831668;

const double A1r_1 = -2.88825538503306;
const double A2r_1 = 2.77989104675777;
const double A3r_1 = -0.891609184208459;

//step_2 - 1.4 cm
const double B0r_2 = 0.00517270575773444;
const double B1r_2 = -0.00436824166171923;
const double B2r_2 = -0.00515548132738072;
const double B3r_2 = 0.00438546609207296;

const double A1r_2 = -2.87865489770775;
const double A2r_2 = 2.76156314738057;
const double A3r_2 = -0.882873800812105;

//step_3 - 1.5 cm
const double B0r_3 = 0.00694154483627482;
const double B1r_3 = -0.00597476665661756;
const double B2r_3 = -0.00692237672582804;
const double B3r_3 = 0.00599393476706433;
```

```
const double A1r_3 = -2.86910528623180;
const double A2r_3 = 2.74306469622661;
const double A3r_3 = -0.873921073773921;

//step_4 - 1.6 cm
const double B0r_4 = 0.00866475675339795;
const double B1r_4 = -0.00759543715065878;
const double B2r_4 = -0.00864533731394742;
const double B3r_4 = 0.00761485659010931;

const double A1r_4 = -2.85960675039235;
const double A2r_4 = 2.72441611016413;
const double A3r_4 = -0.864770520892871;

//step_5 - 1.7 cm
const double B0r_5 = 0.0103445175450547;
const double B1r_5 = -0.00922813353428962;
const double B2r_5 = -0.0103261959767865;
const double B3r_5 = 0.00924645510255783;

const double A1r_5 = -2.85015929946656;
const double A2r_5 = 2.70563596673260;
const double A3r_5 = -0.855440024129502;

//step_6 - 1.8 cm
const double B0r_6 = 0.0119828779968571;
const double B1r_6 = -0.0108709108283402;
const double B2r_6 = -0.0119666668108299;
const double B3r_6 = 0.0108871220143674;

const double A1r_6 = -2.84076277470616;
const double A2r_6 = 2.68674117057348;
const double A3r_6 = -0.845945973495265;

//step_7 - 1.9 cm
const double B0r_7 = 0.00836341467664481;
const double B1r_7 = -0.00771081517160211;
const double B2r_7 = -0.00835515175240575;
const double B3r_7 = 0.00771907809584116;

const double A1r_7 = -2.88259911416098;
const double A2r_7 = 2.76830119673105;
const double A3r_7 = -0.885685556721597;

//step_8 - 2.0 cm
const double B0r_8 = 0.00900195734508397;
const double B1r_8 = -0.00838163049001607;
const double B2r_8 = -0.00899386146161446;
const double B3r_8 = 0.00838972637348558;

const double A1r_8 = -2.88561377593316;
const double A2r_8 = 2.77407915692424;
const double A3r_8 = -0.888449189224143;

//step_9 - 2.1 cm
const double B0r_9 = 0.00969383015351482;
const double B1r_9 = -0.00910235464057830;
const double B2r_9 = -0.00968593989333203;
const double B3r_9 = 0.00911024490076109;
```

```
const double A1r_9 = -2.88827763754024;
const double A2r_9 = 2.77918335198401;
const double A3r_9 = -0.890889933923404;

//step_10 - 2.2 cm
const double B0r_10 = 0.0104428831033720;
const double B1r_10 = -0.00987777161476080;
const double B2r_10 = -0.0104352379207295;
const double B3r_10 = 0.00988541679740335;

const double A1r_10 = -2.89066491723660;
const double A2r_10 = 2.78375610909223;
const double A3r_10 = -0.893075901490345;

//step_11 - 2.3 cm
const double B0r_11 = 0.0112533756990923;
const double B1r_11 = -0.0107128762780108;
const double B2r_11 = -0.0112460156575885;
const double B3r_11 = 0.0107202363195145;

const double A1r_11 = -2.89283178576968;
const double A2r_11 = 2.78790514723905;
const double A3r_11 = -0.895058641386365;

//step_12 - 2.4 cm
const double B0r_12 = 0.0121299824661228;
const double B1r_12 = -0.0116129408030305;
const double B2r_12 = -0.0121229478695646;
const double B3r_12 = 0.0116199753995886;

const double A1r_12 = -2.89482161761424;
const double A2r_12 = 2.79171365153887;
const double A3r_12 = -0.896877964731514;

//step_13 - 2.5 cm
const double B0r_13 = 0.0130778078453958;
const double B1r_13 = -0.0125835707077660;
const double B2r_13 = -0.0130711387557259;
const double B3r_13 = 0.0125902397974359;

const double A1r_13 = -2.89666851198821;
const double A2r_13 = 2.79524702918906;
const double A3r_13 = -0.898565179021510;

class VIRTUAL_TUBE_DELAYObj{

public:
    VIRTUAL_TUBE_DELAYObj() // constructor
    {
        //input buffers
        inR = new float[6];
        inL = new float[6];

        //propagation filter output buffers
        outR = new float[6];
        outL = new float[6];

        //reflection filter output buffers
        out2R = new float[6];
```

```
        out2L = new float[6];

        //propagation filter output buffers (residual distance in reflection mode)
        out3R = new float[3];
        out3L = new float[3];

        //output buffers
        len = 4490; //[46.7755102040816] m
        outBufferL = new float[len];
        outBufferR = new float[len];

        //decrement steps for LP filters
        fLPValues[0] = 0;
        fLPValues[1] = 0;
        fLPValues[2] = 850;
        fLPValues[3] = 1700;
        fLPValues[4] = 2700;
        fLPValues[5] = 3500;
        fLPValues[6] = 4000;
        fLPValues[7] = 4500;
        fLPValues[8] = 4900;
        fLPValues[9] = 5200;
        fLPValues[10] = 5400;
        fLPValues[11] = 5600;
        fLPValues[12] = 5800;
        fLPValues[13] = 6000;
        fLPValues[14] = 6200;
        fLPValues[15] = 6400;
        fLPValues[16] = 6700;
        fLPValues[17] = 7000;
        fLPValues[18] = 7300;
        fLPValues[19] = 7500;
        fLPValues[20] = 7700;
        fLPValues[21] = 7800;
        fLPValues[22] = 7950;
        fLPValues[23] = 8050;
        fLPValues[24] = 8150;
        fLPValues[25] = 8250;
        fLPValues[26] = 8350;
        fLPValues[27] = 8400;
        fLPValues[28] = 8500;
        fLPValues[29] = 8550;
        fLPValues[30] = 8600;
        fLPValues[31] = 8650;
        fLPValues[32] = 8690;
        fLPValues[33] = 8750;
        fLPValues[34] = 8770;
        fLPValues[35] = 8900;

        incValues[0] = 8.5;
        incValues[1] = 8.5;
        incValues[2] = 8.5;
        incValues[3] = 10;
        incValues[4] = 8;
        incValues[5] = 5;
        incValues[6] = 5;
        incValues[7] = 4;
        incValues[8] = 3;
        incValues[9] = 2;
        incValues[10] = 2;
```

```
        incValues[11] = 2;
        incValues[12] = 2;
        incValues[13] = 2;
        incValues[14] = 2;
        incValues[15] = 2;
        incValues[16] = 3;
        incValues[17] = 3;
        incValues[18] = 3;
        incValues[19] = 2;
        incValues[20] = 2;
        incValues[21] = 1;
        incValues[22] = 1.5;
        incValues[23] = 0.5;
        incValues[24] = 1;
        incValues[25] = 1;
        incValues[26] = 1;
        incValues[27] = 0.5;
        incValues[28] = 1;
        incValues[29] = 1.5;
        incValues[30] = 0.5;
        incValues[31] = 0.5;
        incValues[32] = 0.4;
        incValues[33] = 0.6;
        incValues[34] = 1.3;
        incValues[35] = 0;

        //reflection filter coefficients matrix
        a_Ref = new float*[14];
        for (int i = 0; i < 14; ++i) {
            a_Ref[i] = new float[4];
        }
        b_Ref = new float*[14];
        for (int i = 0; i < 14; ++i) {
            b_Ref[i] = new float[4];
        }

        //step_0 - 1.2 cm
        b_Ref[0][0] = B0r_0;
        b_Ref[0][1] = B1r_0;
        b_Ref[0][2] = B2r_0;
        b_Ref[0][3] = B3r_0;

        a_Ref[0][0] = A1r_0;
        a_Ref[0][1] = A2r_0;
        a_Ref[0][2] = A3r_0;

        //step_1 - 1.3 cm
        b_Ref[1][0] = B0r_1;
        b_Ref[1][1] = B1r_1;
        b_Ref[1][2] = B2r_1;
        b_Ref[1][3] = B3r_1;

        a_Ref[1][0] = A1r_1;
        a_Ref[1][1] = A2r_1;
        a_Ref[1][2] = A3r_1;

        //step_2 - 1.4 cm
        b_Ref[2][0] = B0r_2;
        b_Ref[2][1] = B1r_2;
        b_Ref[2][2] = B2r_2;
```

```
        b_Ref[2][3] = B3r_2;

        a_Ref[2][0] = A1r_2;
        a_Ref[2][1] = A2r_2;
        a_Ref[2][2] = A3r_2;

        //step_3 - 1.5 cm
        b_Ref[3][0] = B0r_3;
        b_Ref[3][1] = B1r_3;
        b_Ref[3][2] = B2r_3;
        b_Ref[3][3] = B3r_3;

        a_Ref[3][0] = A1r_3;
        a_Ref[3][1] = A2r_3;
        a_Ref[3][2] = A3r_3;

        //step_4 - 1.6 cm
        b_Ref[4][0] = B0r_4;
        b_Ref[4][1] = B1r_4;
        b_Ref[4][2] = B2r_4;
        b_Ref[4][3] = B3r_4;

        a_Ref[4][0] = A1r_4;
        a_Ref[4][1] = A2r_4;
        a_Ref[4][2] = A3r_4;

        //step_5 - 1.7 cm
        b_Ref[5][0] = B0r_5;
        b_Ref[5][1] = B1r_5;
        b_Ref[5][2] = B2r_5;
        b_Ref[5][3] = B3r_5;

        a_Ref[5][0] = A1r_5;
        a_Ref[5][1] = A2r_5;
        a_Ref[5][2] = A3r_5;

        //step_6 - 1.8 cm
        b_Ref[6][0] = B0r_6;
        b_Ref[6][1] = B1r_6;
        b_Ref[6][2] = B2r_6;
        b_Ref[6][3] = B3r_6;

        a_Ref[6][0] = A1r_6;
        a_Ref[6][1] = A2r_6;
        a_Ref[6][2] = A3r_6;

        //step_7 - 1.9 cm
        b_Ref[7][0] = B0r_7;
        b_Ref[7][1] = B1r_7;
        b_Ref[7][2] = B2r_7;
        b_Ref[7][3] = B3r_7;

        a_Ref[7][0] = A1r_7;
        a_Ref[7][1] = A2r_7;
        a_Ref[7][2] = A3r_7;

        //step_8 - 2.0 cm
        b_Ref[8][0] = B0r_8;
        b_Ref[8][1] = B1r_8;
        b_Ref[8][2] = B2r_8;
```

```
        b_Ref[8][3] = B3r_8;

        a_Ref[8][0] = A1r_8;
        a_Ref[8][1] = A2r_8;
        a_Ref[8][2] = A3r_8;

        //step_9 - 2.1 cm
        b_Ref[9][0] = B0r_9;
        b_Ref[9][1] = B1r_9;
        b_Ref[9][2] = B2r_9;
        b_Ref[9][3] = B3r_9;

        a_Ref[9][0] = A1r_9;
        a_Ref[9][1] = A2r_9;
        a_Ref[9][2] = A3r_9;

        //step_10 - 2.2 cm
        b_Ref[10][0] = B0r_10;
        b_Ref[10][1] = B1r_10;
        b_Ref[10][2] = B2r_10;
        b_Ref[10][3] = B3r_10;

        a_Ref[10][0] = A1r_10;
        a_Ref[10][1] = A2r_10;
        a_Ref[10][2] = A3r_10;

        //step_11 - 2.3 cm
        b_Ref[11][0] = B0r_11;
        b_Ref[11][1] = B1r_11;
        b_Ref[11][2] = B2r_11;
        b_Ref[11][3] = B3r_11;

        a_Ref[11][0] = A1r_11;
        a_Ref[11][1] = A2r_11;
        a_Ref[11][2] = A3r_11;

        //step_11 - 2.4 cm
        b_Ref[12][0] = B0r_12;
        b_Ref[12][1] = B1r_12;
        b_Ref[12][2] = B2r_12;
        b_Ref[12][3] = B3r_12;

        a_Ref[12][0] = A1r_12;
        a_Ref[12][1] = A2r_12;
        a_Ref[12][2] = A3r_12;

        //step_12 - 2.5 cm
        b_Ref[13][0] = B0r_13;
        b_Ref[13][1] = B1r_13;
        b_Ref[13][2] = B2r_13;
        b_Ref[13][3] = B3r_13;

        a_Ref[13][0] = A1r_13;
        a_Ref[13][1] = A2r_13;
        a_Ref[13][2] = A3r_13;

    };

    ~VIRTUAL_TUBE_DELAYObj() // destructors
    {
```

```
        delete [] outBufferL;
        delete [] outBufferR;
        delete [] inL;
        delete [] inR;
        delete [] outR;
        delete [] outL;
        delete [] out2R;
        delete [] out2L;
        delete [] out3R;
        delete [] out3L;

        for (int i=0; i<3; ++i){

            delete a_Ref[i];
        }

        for (int i=0; i<3; ++i){

            delete b_Ref[i];
        }
    }

    void process(const float* input, float* lengthL, float* lengthR, float* gainL,
    float* gainR, float* size, float* ref, float* lengthRefL, float* lengthRefR,
    float* gainRefL, float* gainRefR, float* dry_wet, float* outputR, float*
    outputL, int n)
    {

        //virtual tube length (propagation only)
        int lengtL = floor(lengthL[n-1]*100);
        int lengtR = floor(lengthR[n-1]*100);

        double lengL = lengtL/100;
        double lengR = lengtR/100;

        if (lengL < 1)
            lengL = 1;

        if (lengR < 1)
            lengR = 1;

        //virtual tube additional length
        int lengtRefL = floor(lengthRefL[n-1]*100);
        int lengtRefR = floor(lengthRefR[n-1]*100);

        double lengRefL = lengtRefL/100;
        double lengRefR = lengtRefR/100;

        if (lengRefL < 1)
            lengRefL = 1;

        if (lengRefR < 1)
            lengRefR = 1;


        //virtual tube total length (reflection only)
        double length_totL = 2*lengRefL;
        double length_totR = 2*lengRefR;

        int stepLengthL = floor(lengL);
```

```
        int stepLengthR = floor(lengR);

        int stepLengthRefL = floor(length_totL);
        int stepLengthRefR = floor(length_totR);


        //delay time = length/c
        //(propagation only)
        int delayMillisecondsL = lengL/0.345;
        int delayMillisecondsR = lengR/0.345;

        //total delay time
        int delayMillisecondsRefL = (2*lengRefL)/0.345;
        int delayMillisecondsRefR = (2*lengRefR)/0.345;

        //number of samples = delay time * f
        //sample rate = 44100 Hz
        //(propagation only)
        int delaySamplesL = (int)((float)delayMillisecondsL * 44.1f);
        int delaySamplesR = (int)((float)delayMillisecondsR * 44.1f);

        //(reflection only)
        int delaySamplesRefL = (int)((float)delayMillisecondsRefL * 44.1f); //
    assumes 44100 Hz sample rate
        int delaySamplesRefR = (int)((float)delayMillisecondsRefR * 44.1f); //
    assumes 44100 Hz sample rate

        //(total)
        int delaySamplesL_tot = delaySamplesL + delaySamplesRefL;
        int delaySamplesR_tot = delaySamplesR + delaySamplesRefR;

        //gains
        float gL = gainL[n-1];
        float gR = gainR[n-1];
        float gRef_R = gainRefR[n-1];
        float gRef_L = gainRefL[n-1];

        //dry-wet parameter
        float wet = dry_wet[n-1]/100.00f;
        float dry = 1 - dry_wet[n-1]/100.00f;

        //virtual tube diameter
        int rad = floor(size[n-1]*10 - 12);

        if (rad == -12)
            rad = 0;

        br[0] = b_Ref[rad][0];
        br[1] = b_Ref[rad][1];
        br[2] = b_Ref[rad][2];
        br[3] = b_Ref[rad][3];

        ar[0] = a_Ref[rad][0];
        ar[1] = a_Ref[rad][1];
        ar[2] = a_Ref[rad][2];

        if (rad <= 6){

            inc_R = (pow(0.85 + rad*g12,(lengR)));
            inc_L = (pow(0.85 + rad*g12,(lengL)));
```

```
        Q1_R = 0.65;
        fcLP_R = fcLP_0 + rad*f12LP - (fLPValues[stepLengthR] + incValues[
    stepLengthR]*(lengR - floor(lengR)));

        Q1_L = 0.65;
        fcLP_L = fcLP_0 + rad*f12LP - (fLPValues[stepLengthL] + incValues[
    stepLengthL]*(lengL - floor(lengL)));

        //ref
        incRef_R = (pow(0.85 + rad*g12,(length_totR)));
        incRef_L = (pow(0.85 + rad*g12,(length_totL)));
        Q1Ref_R = 0.65;
        fcLPRef_R = fcLP_0 + rad*f12LP - (fLPValues[stepLengthRefR] + incValues
    [stepLengthRefR]*(length_totR - floor(length_totR)));

        Q1Ref_L = 0.65;
        fcLPRef_L = fcLP_0 + rad*f12LP - (fLPValues[stepLengthRefL] + incValues
    [stepLengthRefL]*(length_totL - floor(length_totL)));


        //right
        if (lengR <= 15) {

            G1_R = -1 - 0.85*(lengR-1);
            V1_R = pow(10,(G1_R/20));

            fc1_R = fc1_0 + rad*f12 +(lengR-1)*30;

            G2_R = -0.9-(lengR-1)*1;
            V2_R = pow(10,(G2_R/20));

            fc2_R = fc2_0 + rad*f212 + (lengR-1)*50;

            Q2_R = 0.5;

        }else{

            G1_R = -12.9000-0.6*(lengR-15);
            V1_R = pow(10,(G1_R/20));

            fc1_R = 1620 + rad*f12 + (lengR-15)*20;

            G2_R = -14.9000-(lengR-15)*0.1;
            V2_R = pow(10,(G2_R/20));

            fc2_R = 2200 + rad*f212;

            Q2_R = 0.5 + (lengR-15)*0.02;

        }

        //ref right

        if (length_totR <= 15) {

            G1Ref_R = -1 - 0.85*(length_totR-1);
            V1Ref_R = pow(10,(G1Ref_R/20));

            fc1Ref_R = fc1_0 + rad*f12 +(length_totR-1)*30;

```

```
            G2Ref_R = -0.9 - (length_totR -1)*1;
            V2Ref_R = pow(10,(G2Ref_R/20));

            fc2Ref_R = fc2_0 + rad*f212 + (length_totR -1)*50;

            Q2Ref_R = 0.5;

        }else{

            G1Ref_R = -12.9000 - 0.6*(length_totR -15);
            V1Ref_R = pow(10,(G1Ref_R/20));

            fc1Ref_R = 1620 + rad*f12 + (length_totR -15)*20;

            G2Ref_R = -14.9000-(length_totR -15)*0.1;
            V2Ref_R = pow(10,(G2Ref_R/20));

            fc2Ref_R = 2200 + rad*f212;

            Q2Ref_R = 0.5+(length_totR -15)*0.02;

        }

        // left

        if (lengL <= 15) {

            G1_L = -1 - 0.85*(lengL -1);
            V1_L = pow(10,(G1_L/20));

            fc1_L = fc1_0 + rad*f12 + (lengL -1)*30;

            G2_L = -0.9 - (lengL -1)*1;
            V2_L = pow(10,(G2_L/20));

            fc2_L = fc2_0 + rad*f212 + (lengL -1)*50;

            Q2_L = 0.5;

        }else{

            G1_L = -12.9000 - 0.6*(lengL -15);
            V1_L = pow(10,(G1_L/20));

            fc1_L = 1620 + rad*f12 + (lengL -15)*20;

            G2_L = -14.9000-(lengL -15)*0.1;
            V2_L = pow(10,(G2_L/20));

            fc2_L = 2200 + rad*f212;

            Q2_L = 0.5 + (lengL -15)*0.02;

        }

        // ref left

        if (length_totL <= 15) {

            G1Ref_L = -1 - 0.85*(length_totL -1);
```

```
                V1Ref_L = pow(10,(G1Ref_L/20));

                fc1Ref_L = fc1_0 + rad*f12 +(length_totL -1)*30;

                G2Ref_L = -0.9 - (length_totL -1)*1;
                V2Ref_L = pow(10,(G2Ref_L/20));

                fc2Ref_L = fc2_0 + rad*f212 + (length_totL -1)*50;

                Q2Ref_L = 0.5;

            }else{

                G1Ref_L = -12.9000-0.6*(length_totL -15);
                V1Ref_L = pow(10,(G1Ref_L/20));

                fc1Ref_L = 1620 + rad*f12 + (length_totL -15)*20;

                G2Ref_L = -14.9000 - (length_totL -15)*0.1;
                V2Ref_L = pow(10,(G2Ref_L/20));

                fc2Ref_L = 2200 + rad*f212;

                Q2Ref_L = 0.5+(length_totL -15)*0.02;

            }



        }else if(rad >= 7){

            inc_R = (pow(0.87 + (rad-7)*g23,(lengR)));
            inc_L = (pow(0.87 + (rad-7)*g23,(lengL)));
            Q1_R = 0.65;
            fcLP_R = fcLP_1 + (rad-7)*f23LP - (fLPValues[stepLengthR] + incValues[stepLengthR]*(lengR - floor(lengR)));

            Q1_L = 0.65;
            fcLP_L = fcLP_1 + (rad-7)*f23LP - (fLPValues[stepLengthL] + incValues[stepLengthL]*(lengL - floor(lengL)));

            //ref
            incRef_R = (pow(0.87 + (rad-7)*g23,(length_totR)));
            incRef_L = (pow(0.87 + (rad-7)*g23,(length_totL)));
            Q1Ref_R = 0.65;
            fcLPRef_R = fcLP_1 + (rad-7)*f23LP - (fLPValues[stepLengthRefR] + incValues[stepLengthRefR]*(length_totR - floor(length_totR)));

            Q1Ref_L = 0.65;
            fcLPRef_L = fcLP_1 + (rad-7)*f23LP - (fLPValues[stepLengthRefL] + incValues[stepLengthRefL]*(length_totL - floor(length_totL)));

            //right

            if (lengR <= 15) {

                G1_R = -1 - 0.85*(lengR-1);
                V1_R = pow(10,(G1_R/20));

                fc1_R = fc1_1 +(lengR-1)*30;
```

```

            G2_R = -0.9 - (lengR-1)*1;
            V2_R = pow(10,(G2_R/20));

            fc2_R = fc2_1 + (lengR-1)*50;

            Q2_R = 0.5;

        }else{

            G1_R = -12.9000 - 0.6*(lengR-15);
            V1_R = pow(10,(G1_R/20));

            fc1_R = 1320 + (lengR-15)*20;

            G2_R = -14.9000 - (lengR-15)*0.1;
            V2_R = pow(10,(G2_R/20));

            fc2_R = 7700;

            Q2_R = 0.5 + (lengR-15)*0.02;

        }

        // ref right

        if (length_totR <= 15) {

            G1Ref_R = -1-0.85*(length_totR-1);
            V1Ref_R = pow(10,(G1Ref_R/20));

            fc1Ref_R = fc1_1 + (length_totR-1)*30;

            G2Ref_R = -0.9 - (length_totR-1)*1;
            V2Ref_R = pow(10,(G2Ref_R/20));

            fc2Ref_R = fc2_1 + (length_totR-1)*50;

            Q2Ref_R = 0.5;

        }else{

            G1Ref_R = -12.9000 - 0.6*(length_totR-15);
            V1Ref_R = pow(10,(G1Ref_R/20));

            fc1Ref_R = 1320 + (length_totR-15)*20;

            G2Ref_R = -14.9000 - (length_totR-15)*0.1;
            V2Ref_R = pow(10,(G2Ref_R/20));

            fc2Ref_R = 7700;

            Q2Ref_R = 0.5 + (length_totR-15)*0.02;

        }

        //left

        if (lengL <= 15) {

```

```
            G1_L = -1 - 0.85*(lengL-1);
            V1_L = pow(10,(G1_L/20));

            fc1_L = fc1_1 +(lengL-1)*30;

            G2_L = -0.9 - (lengL-1)*1;
            V2_L = pow(10,(G2_L/20));

            fc2_L = fc2_1 + (lengL-1)*50;

            Q2_L = 0.5;

        }else{

            G1_L = -12.9000 - 0.6*(lengL-15);
            V1_L = pow(10,(G1_L/20));

            fc1_L = 1320 + (lengL-15)*20;

            G2_L = -14.9000 - (lengL-15)*0.1;
            V2_L = pow(10,(G2_L/20));

            fc2_L = 7700;

            Q2_L = 0.5 + (lengL-15)*0.02;

        }
        // ref left

        if (length_totL <= 15) {

            G1Ref_L = -1 - 0.85*(length_totL-1);
            V1Ref_L = pow(10,(G1Ref_L/20));

            fc1Ref_L = fc1_1 +(length_totL-1)*30;

            G2Ref_L = -0.9 - (length_totL-1)*1;
            V2Ref_L = pow(10,(G2Ref_L/20));

            fc2Ref_L = fc2_1 + (length_totL-1)*50;

            Q2Ref_L = 0.5;

        }else{

            G1Ref_L = -12.9000-0.6*(length_totL-15);
            V1Ref_L = pow(10,(G1Ref_L/20));

            fc1Ref_L = 1320 + (length_totL-15)*20;

            G2Ref_L = -14.9000 - (length_totL-15)*0.1;
            V2Ref_L = pow(10,(G2Ref_L/20));

            fc2Ref_L = 7700;

            Q2Ref_L = 0.5+(length_totL-15)*0.02;

        }
```

```
        }

        //1st filter

        K1_R = tan(PI*fc1_R/44100);

        a1_R = (2*(V1_R*K1_R*K1_R - 1))/(1 + (sqrt(V1_R)*K1_R)/Q1_R + V1_R*K1_R*K1_R);

        a2_R = (1 - (sqrt(V1_R)*K1_R)/Q1_R + V1_R*K1_R*K1_R)/(1 + (sqrt(V1_R)*K1_R)/Q1_R + V1_R*K1_R*K1_R);

        b0_R = V1_R*(1 + (K1_R)/Q1_R + K1_R*K1_R) / (1 + (sqrt(V1_R)*K1_R)/Q1_R + V1_R*K1_R*K1_R);

        b1_R = (2*V1_R*(K1_R*K1_R - 1)) / (1 + (sqrt(V1_R)*K1_R)/Q1_R + V1_R*K1_R*K1_R);

        b2_R = V1_R*(1 - (K1_R)/Q1_R + K1_R*K1_R) / (1 + (sqrt(V1_R)*K1_R)/Q1_R + V1_R*K1_R*K1_R);

        //2nd filter

        K2_R = tan(PI*fc2_R/44100);

        a11_R = (2*(V2_R*K2_R*K2_R - 1))/ (1 + (sqrt(V2_R)*K2_R)/Q2_R + V2_R*K2_R*K2_R);

        a22_R = (1 - (sqrt(V2_R)*K2_R)/Q2_R + V2_R*K2_R*K2_R) / (1 + (sqrt(V2_R)*K2_R)/Q2_R + V2_R*K2_R*K2_R);

        b00_R =  V2_R*(1 + (K2_R)/Q2_R + K2_R*K2_R) / (1 + (sqrt(V2_R)*K2_R)/Q2_R + V2_R*K2_R*K2_R);

        b11_R = (2*V2_R*(K2_R*K2_R - 1)) / (1 + (sqrt(V2_R)*K2_R)/Q2_R + V2_R*K2_R*K2_R);

        b22_R = V2_R*(1 - (K2_R)/Q2_R + K2_R*K2_R)/ (1 + (sqrt(V2_R)*K2_R)/Q2_R + V2_R*K2_R*K2_R);

        //LP filter
        if (lengR < 2){

            KLP_R = tan(PI*fcLP_R/44100);

            a2LP_R = 0;

            a1LP_R = (KLP_R-1)/(KLP_R+1);

            b0LP_R = KLP_R/(KLP_R+1);

            b1LP_R = KLP_R/(KLP_R+1);

            b2LP_R = 0;

        }else{

            KLP_R = tan(PI*fcLP_R/44100);

            a2LP_R = pow(((KLP_R-1)/(KLP_R+1)),2);
```

```
            a1LP_R = 2*((KLP_R-1)/(KLP_R+1));

            b0LP_R = pow((KLP_R/(KLP_R+1)),2);

            b1LP_R = 2*(KLP_R/(KLP_R+1))*(KLP_R/(KLP_R+1));

            b2LP_R = pow((KLP_R/(KLP_R+1)),2);

        }

        //final filter

        a1f_R = a1LP_R  + a1_R + a11_R;

        a2f_R = a2LP_R + a1_R * a1LP_R + a2_R + a11_R * a1LP_R + a11_R * a1_R +
    a22_R;

        a3f_R = a1_R * a2LP_R + a2_R * a1LP_R + a11_R * a2LP_R + a11_R * a1_R *
    a1LP_R + a11_R * a2_R + a22_R * a1LP_R + a22_R * a1_R;

        a4f_R = a2_R * a2LP_R + a11_R * a1_R * a2LP_R + a11_R * a2_R * a1LP_R +
    a22_R * a2LP_R + a22_R * a1_R * a1LP_R + a22_R * a2_R;

        a5f_R = a11_R * a2_R * a2LP_R + a22_R * a1_R * a2LP_R + a22_R * a2_R *
    a1LP_R;

        a6f_R = a22_R * a2LP_R * a2_R;

        b0f_R = inc_R*(b00_R * b0_R * b0LP_R);

        b1f_R = inc_R*(b00_R * b0_R * b1LP_R + b00_R * b1_R * b0LP_R + b11_R * b0_R
     * b0LP_R);

        b2f_R = inc_R*(b00_R * b0_R * b2LP_R + b00_R * b1_R * b1LP_R + b00_R * b2_R
     * b0LP_R + b11_R * b0_R * b1LP_R + b11_R * b1_R * b0LP_R + b0_R * b0LP_R *
    b22_R);

        b3f_R = inc_R*(b00_R * b1_R * b2LP_R + b00_R * b2_R * b1LP_R + b11_R * b0_R
     * b2LP_R + b11_R * b1_R * b1LP_R + b11_R * b2_R * b0LP_R + b22_R * b1_R *
    b0LP_R + b22_R * b0_R * b1LP_R);

        b4f_R = inc_R*(b2_R * b2LP_R * b00_R + b11_R * b1_R * b2LP_R + b11_R * b2_R
     * b1LP_R + b2_R * b0LP_R * b22_R + b22_R * b1_R * b1LP_R + b22_R * b0_R *
    b2LP_R);
        b5f_R = inc_R*(b11_R * b2_R * b2LP_R + b22_R * b2_R * b1LP_R + b22_R * b1_R
     * b2LP_R);

        b6f_R = inc_R*(b22_R * b2_R * b2LP_R);

        //left

        //1st filter

        K1_L = tan(PI*fc1_L/44100);

        a1_L = (2*(V1_L*K1_L*K1_L - 1))/(1 + (sqrt(V1_L)*K1_L)/Q1_L + V1_L*K1_L*
    K1_L);

```

```
    a2_L = (1 - (sqrt(V1_L)*K1_L)/Q1_L + V1_L*K1_L*K1_L)/(1 + (sqrt(V1_L)*K1_L)
/Q1_L + V1_L*K1_L*K1_L);

    b0_L = V1_L*(1 + (K1_L)/Q1_L + K1_L*K1_L) / (1 + (sqrt(V1_L)*K1_L)/Q1_L +
V1_L*K1_L*K1_L);

    b1_L = (2*V1_L*(K1_L*K1_L - 1)) / (1 + (sqrt(V1_L)*K1_L)/Q1_L + V1_L*K1_L*
K1_L);

    b2_L = V1_L*(1 - (K1_L)/Q1_L + K1_L*K1_L) / (1 + (sqrt(V1_L)*K1_L)/Q1_L +
V1_L*K1_L*K1_L);

    //2nd filter

    K2_L = tan(PI*fc2_L/44100);

    a11_L = (2*(V2_L*K2_L*K2_L - 1))/ (1 + (sqrt(V2_L)*K2_L)/Q2_L + V2_L*K2_L*
K2_L);

    a22_L = (1 - (sqrt(V2_L)*K2_L)/Q2_L + V2_L*K2_L*K2_L) / (1 + (sqrt(V2_L)*
K2_L)/Q2_L + V2_L*K2_L*K2_L);

    b00_L =  V2_L*(1 + (K2_L)/Q2_L + K2_L*K2_L) / (1 + (sqrt(V2_L)*K2_L)/Q2_L +
 V2_L*K2_L*K2_L);

    b11_L = (2*V2_L*(K2_L*K2_L - 1)) / (1 + (sqrt(V2_L)*K2_L)/Q2_L + V2_L*K2_L*
K2_L);

    b22_L = V2_L*(1 - (K2_L)/Q2_L + K2_L*K2_L)/ (1 + (sqrt(V2_L)*K2_L)/Q2_L +
V2_L*K2_L*K2_L);

    //LP filter
    if (lengL < 2){

        KLP_L = tan(PI*fcLP_L/44100);

        a2LP_L = 0;

        a1LP_L = (KLP_L-1)/(KLP_L+1);

        b0LP_L = KLP_L/(KLP_L+1);

        b1LP_L = KLP_L/(KLP_L+1);

        b2LP_L = 0;

    }else{

        KLP_L = tan(PI*fcLP_L/44100);

        a2LP_L = pow(((KLP_L-1)/(KLP_L+1)),2);

        a1LP_L = 2*((KLP_L-1)/(KLP_L+1));

        b0LP_L = pow((KLP_L/(KLP_L+1)),2);

        b1LP_L = 2*(KLP_L/(KLP_L+1))*(KLP_L/(KLP_L+1));

        b2LP_L = pow((KLP_L/(KLP_L+1)),2);

```

```
    }


    //final filter

    a1f_L = a1LP_L  + a1_L + a11_L;

    a2f_L = a2LP_L + a1_L * a1LP_L + a2_L + a11_L * a1LP_L + a11_L * a1_L + a22_L;

    a3f_L = a1_L * a2LP_L + a2_L * a1LP_L + a11_L * a2LP_L + a11_L * a1_L * a1LP_L + a11_L * a2_L + a22_L * a1LP_L + a22_L * a1_L;

    a4f_L = a2_L * a2LP_L + a11_L * a1_L * a2LP_L + a11_L * a2_L * a1LP_L + a22_L * a2LP_L + a22_L * a1_L * a1LP_L + a22_L * a2_L;

    a5f_L = a11_L * a2_L * a2LP_L + a22_L * a1_L * a2LP_L + a22_L * a2_L * a1LP_L;

    a6f_L = a22_L * a2LP_L * a2_L;

    b0f_L = inc_L*(b00_L * b0_L * b0LP_L);

    b1f_L = inc_L*(b00_L * b0_L * b1LP_L + b00_L * b1_L * b0LP_L + b11_L * b0_L * b0LP_L);

    b2f_L = inc_L*(b00_L * b0_L * b2LP_L + b00_L * b1_L * b1LP_L + b00_L * b2_L * b0LP_L + b11_L * b0_L * b1LP_L + b11_L * b1_L * b0LP_L + b0_L * b0LP_L * b22_L);

    b3f_L = inc_L*(b00_L * b1_L * b2LP_L + b00_L * b2_L * b1LP_L + b11_L * b0_L * b2LP_L + b11_L * b1_L * b1LP_L + b11_L * b2_L * b0LP_L + b22_L * b1_L * b0LP_L + b22_L * b0_L * b1LP_L);

    b4f_L = inc_L*(b2_L * b2LP_L * b00_L + b11_L * b1_L * b2LP_L + b11_L * b2_L * b1LP_L + b2_L * b0LP_L * b22_L + b22_L * b1_L * b1LP_L + b22_L * b0_L * b2LP_L);
    b5f_L = inc_L*(b11_L * b2_L * b2LP_L + b22_L * b2_L * b1LP_L + b22_L * b1_L * b2LP_L);

    b6f_L = inc_L*(b22_L * b2_L * b2LP_L);

    //ref

    //1st filter

    K1Ref_R = tan(PI*fc1Ref_R/44100);

    a1Ref_R = (2*(V1Ref_R*K1Ref_R*K1Ref_R - 1))/(1 + (sqrt(V1Ref_R)*K1Ref_R)/Q1Ref_R + V1Ref_R*K1Ref_R*K1Ref_R);

    a2Ref_R = (1 - (sqrt(V1Ref_R)*K1Ref_R)/Q1Ref_R + V1Ref_R*K1Ref_R*K1Ref_R)/(1 + (sqrt(V1Ref_R)*K1Ref_R)/Q1Ref_R + V1Ref_R*K1Ref_R*K1Ref_R);

    b0Ref_R = V1Ref_R*(1 + (K1Ref_R)/Q1Ref_R + K1Ref_R*K1Ref_R) / (1 + (sqrt(V1Ref_R)*K1Ref_R)/Q1Ref_R + V1Ref_R*K1Ref_R*K1Ref_R);

    b1Ref_R = (2*V1Ref_R*(K1Ref_R*K1Ref_R - 1)) / (1 + (sqrt(V1Ref_R)*K1Ref_R)/Q1Ref_R + V1Ref_R*K1Ref_R*K1Ref_R);

```

```
    b2Ref_R = V1Ref_R*(1 - (K1Ref_R)/Q1Ref_R + K1Ref_R*K1Ref_R) / (1 + (sqrt(V1Ref_R)*K1Ref_R)/Q1Ref_R + V1Ref_R*K1Ref_R*K1Ref_R);

    //2nd filter

    K2Ref_R = tan(PI*fc2Ref_R/44100);

    a11Ref_R = (2*(V2Ref_R*K2Ref_R*K2Ref_R - 1))/ (1 + (sqrt(V2Ref_R)*K2Ref_R)/Q2Ref_R + V2Ref_R*K2Ref_R*K2Ref_R);

    a22Ref_R = (1 - (sqrt(V2Ref_R)*K2Ref_R)/Q2Ref_R + V2Ref_R*K2Ref_R*K2Ref_R) / (1 + (sqrt(V2Ref_R)*K2Ref_R)/Q2Ref_R + V2Ref_R*K2Ref_R*K2Ref_R);

    b00Ref_R =  V2Ref_R*(1 + (K2Ref_R)/Q2Ref_R + K2Ref_R*K2Ref_R) / (1 + (sqrt(V2Ref_R)*K2Ref_R)/Q2Ref_R + V2Ref_R*K2Ref_R*K2Ref_R);

    b11Ref_R = (2*V2Ref_R*(K2Ref_R*K2Ref_R - 1)) / (1 + (sqrt(V2Ref_R)*K2Ref_R)/Q2Ref_R + V2Ref_R*K2Ref_R*K2Ref_R);

    b22Ref_R = V2Ref_R*(1 - (K2Ref_R)/Q2Ref_R + K2Ref_R*K2Ref_R)/ (1 + (sqrt(V2Ref_R)*K2Ref_R)/Q2Ref_R + V2Ref_R*K2Ref_R*K2Ref_R);

    //LP filter
    if (length_totR < 2){

        KLPRef_R = tan(PI*fcLPRef_R/44100);

        a2LPRef_R = 0;

        a1LPRef_R = (KLPRef_R-1)/(KLPRef_R+1);

        b0LPRef_R = KLPRef_R/(KLPRef_R+1);

        b1LPRef_R = KLPRef_R/(KLPRef_R+1);

        b2LPRef_R = 0;

    }else{

        KLPRef_R = tan(PI*fcLPRef_R/44100);

        a2LPRef_R = pow(((KLPRef_R-1)/(KLPRef_R+1)),2);

        a1LPRef_R = 2*((KLPRef_R-1)/(KLPRef_R+1));

        b0LPRef_R = pow((KLPRef_R/(KLPRef_R+1)),2);

        b1LPRef_R = 2*(KLPRef_R/(KLPRef_R+1))*(KLPRef_R/(KLPRef_R+1));

        b2LPRef_R = pow((KLPRef_R/(KLPRef_R+1)),2);

    }


    //final filter

    a1fRef_R = a1LPRef_R  + a1Ref_R + a11Ref_R;

    a2fRef_R = a2LPRef_R + a1Ref_R * a1LPRef_R + a2Ref_R + a11Ref_R * a1LPRef_R + a11Ref_R * a1Ref_R + a22Ref_R;
```

```
    a3fRef_R = a1Ref_R * a2LPRef_R + a2Ref_R * a1LPRef_R + a11Ref_R * a2LPRef_R
 + a11Ref_R * a1Ref_R * a1LPRef_R + a11Ref_R * a2Ref_R + a22Ref_R * a1LPRef_R +
 a22Ref_R * a1Ref_R;

    a4fRef_R = a2Ref_R * a2LPRef_R + a11Ref_R * a1Ref_R * a2LPRef_R + a11Ref_R
* a2Ref_R * a1LPRef_R + a22Ref_R * a2LPRef_R + a22Ref_R * a1Ref_R * a1LPRef_R +
 a22Ref_R * a2Ref_R;

    a5fRef_R = a11Ref_R * a2Ref_R * a2LPRef_R + a22Ref_R * a1Ref_R * a2LPRef_R
+ a22Ref_R * a2Ref_R * a1LPRef_R;

    a6fRef_R = a22Ref_R * a2LPRef_R * a2Ref_R;

    b0fRef_R = incRef_R*(b00Ref_R * b0Ref_R * b0LPRef_R);

    b1fRef_R = incRef_R*(b00Ref_R * b0Ref_R * b1LPRef_R + b00Ref_R * b1Ref_R *
b0LPRef_R + b11Ref_R * b0Ref_R * b0LPRef_R);

    b2fRef_R = incRef_R*(b00Ref_R * b0Ref_R * b2LPRef_R + b00Ref_R * b1Ref_R *
b1LPRef_R + b00Ref_R * b2Ref_R * b0LPRef_R + b11Ref_R * b0Ref_R * b1LPRef_R +
b11Ref_R * b1Ref_R * b0LPRef_R + b0Ref_R * b0LPRef_R * b22Ref_R);

    b3fRef_R = incRef_R*(b00Ref_R * b1Ref_R * b2LPRef_R + b00Ref_R * b2Ref_R *
b1LPRef_R + b11Ref_R * b0Ref_R * b2LPRef_R + b11Ref_R * b1Ref_R * b1LPRef_R +
b11Ref_R * b2Ref_R * b0LPRef_R + b22Ref_R * b1Ref_R * b0LPRef_R + b22Ref_R *
b0Ref_R * b1LPRef_R);

    b4fRef_R = incRef_R*(b2Ref_R * b2LPRef_R * b00Ref_R + b11Ref_R * b1Ref_R *
b2LPRef_R + b11Ref_R * b2Ref_R * b1LPRef_R + b2Ref_R * b0LPRef_R * b22Ref_R +
b22Ref_R * b1Ref_R * b1LPRef_R + b22Ref_R * b0Ref_R * b2LPRef_R);
    b5fRef_R = incRef_R*(b11Ref_R * b2Ref_R * b2LPRef_R + b22Ref_R * b2Ref_R *
b1LPRef_R + b22Ref_R * b1Ref_R * b2LPRef_R);

    b6fRef_R = incRef_R*(b22Ref_R * b2Ref_R * b2LPRef_R);


    //left

    //1st filter

    K1Ref_L = tan(PI*fc1Ref_L/44100);

    a1Ref_L = (2*(V1Ref_L*K1Ref_L*K1Ref_L - 1))/(1 + (sqrt(V1Ref_L)*K1Ref_L)/
Q1Ref_L + V1Ref_L*K1Ref_L*K1Ref_L);

    a2Ref_L = (1 - (sqrt(V1Ref_L)*K1Ref_L)/Q1Ref_L + V1Ref_L*K1Ref_L*K1Ref_L)
/(1 + (sqrt(V1Ref_L)*K1Ref_L)/Q1Ref_L + V1Ref_L*K1Ref_L*K1Ref_L);

    b0Ref_L = V1Ref_L*(1 + (K1Ref_L)/Q1Ref_L + K1Ref_L*K1Ref_L) / (1 + (sqrt(
V1Ref_L)*K1Ref_L)/Q1Ref_L + V1Ref_L*K1Ref_L*K1Ref_L);

    b1Ref_L = (2*V1Ref_L*(K1Ref_L*K1Ref_L - 1)) / (1 + (sqrt(V1Ref_L)*K1Ref_L)/
Q1Ref_L + V1Ref_L*K1Ref_L*K1Ref_L);

    b2Ref_L = V1Ref_L*(1 - (K1Ref_L)/Q1Ref_L + K1Ref_L*K1Ref_L) / (1 + (sqrt(
V1Ref_L)*K1Ref_L)/Q1Ref_L + V1Ref_L*K1Ref_L*K1Ref_L);

    //2nd filter
```

```
        K2Ref_L = tan(PI*fc2Ref_L/44100);

        a11Ref_L = (2*(V2Ref_L*K2Ref_L*K2Ref_L - 1))/ (1 + (sqrt(V2Ref_L)*K2Ref_L)/
    Q2Ref_L + V2Ref_L*K2Ref_L*K2Ref_L);

        a22Ref_L = (1 - (sqrt(V2Ref_L)*K2Ref_L)/Q2Ref_L + V2Ref_L*K2Ref_L*K2Ref_L)
    / (1 + (sqrt(V2Ref_L)*K2Ref_L)/Q2Ref_L + V2Ref_L*K2Ref_L*K2Ref_L);

        b00Ref_L =  V2Ref_L*(1 + (K2Ref_L)/Q2Ref_L + K2Ref_L*K2Ref_L) / (1 + (sqrt(
    V2Ref_L)*K2Ref_L)/Q2Ref_L + V2Ref_L*K2Ref_L*K2Ref_L);

        b11Ref_L = (2*V2Ref_L*(K2Ref_L*K2Ref_L - 1)) / (1 + (sqrt(V2Ref_L)*K2Ref_L)
    /Q2Ref_L + V2Ref_L*K2Ref_L*K2Ref_L);

        b22Ref_L = V2Ref_L*(1 - (K2Ref_L)/Q2Ref_L + K2Ref_L*K2Ref_L)/ (1 + (sqrt(
    V2Ref_L)*K2Ref_L)/Q2Ref_L + V2Ref_L*K2Ref_L*K2Ref_L);

        //LP filter
        if (length_totL < 2){

            KLPRef_L = tan(PI*fcLPRef_L/44100);

            a2LPRef_L = 0;

            a1LPRef_L = (KLPRef_L-1)/(KLPRef_L+1);

            b0LPRef_L = KLPRef_L/(KLPRef_L+1);

            b1LPRef_L = KLPRef_L/(KLPRef_L+1);

            b2LPRef_L = 0;

        }else{

            KLPRef_L = tan(PI*fcLPRef_L/44100);

            a2LPRef_L = pow(((KLPRef_L-1)/(KLPRef_L+1)),2);

            a1LPRef_L = 2*((KLPRef_L-1)/(KLPRef_L+1));

            b0LPRef_L = pow((KLPRef_L/(KLPRef_L+1)),2);

            b1LPRef_L = 2*(KLPRef_L/(KLPRef_L+1))*(KLPRef_L/(KLPRef_L+1));

            b2LPRef_L = pow((KLPRef_L/(KLPRef_L+1)),2);

        }


        //final filter

        a1fRef_L = a1LPRef_L  + a1Ref_L + a11Ref_L;

        a2fRef_L = a2LPRef_L + a1Ref_L * a1LPRef_L + a2Ref_L + a11Ref_L * a1LPRef_L
     + a11Ref_L * a1Ref_L + a22Ref_L;

        a3fRef_L = a1Ref_L * a2LPRef_L + a2Ref_L * a1LPRef_L + a11Ref_L * a2LPRef_L
     + a11Ref_L * a1Ref_L * a1LPRef_L + a11Ref_L * a2Ref_L + a22Ref_L * a1LPRef_L +
     a22Ref_L * a1Ref_L;

```

```
    a4fRef_L = a2Ref_L * a2LPRef_L + a11Ref_L * a1Ref_L * a2LPRef_L + a11Ref_L * a2Ref_L * a1LPRef_L + a22Ref_L * a2LPRef_L + a22Ref_L * a1Ref_L * a1LPRef_L + a22Ref_L * a2Ref_L;

    a5fRef_L = a11Ref_L * a2Ref_L * a2LPRef_L + a22Ref_L * a1Ref_L * a2LPRef_L + a22Ref_L * a2Ref_L * a1LPRef_L;

    a6fRef_L = a22Ref_L * a2LPRef_L * a2Ref_L;

    b0fRef_L = incRef_L*(b00Ref_L * b0Ref_L * b0LPRef_L);

    b1fRef_L = incRef_L*(b00Ref_L * b0Ref_L * b1LPRef_L + b00Ref_L * b1Ref_L * b0LPRef_L + b11Ref_L * b0Ref_L * b0LPRef_L);

    b2fRef_L = incRef_L*(b00Ref_L * b0Ref_L * b2LPRef_L + b00Ref_L * b1Ref_L * b1LPRef_L + b00Ref_L * b2Ref_L * b0LPRef_L + b11Ref_L * b0Ref_L * b1LPRef_L + b11Ref_L * b1Ref_L * b0LPRef_L + b0Ref_L * b0LPRef_L * b22Ref_L);

    b3fRef_L = incRef_L*(b00Ref_L * b1Ref_L * b2LPRef_L + b00Ref_L * b2Ref_L * b1LPRef_L + b11Ref_L * b0Ref_L * b2LPRef_L + b11Ref_L * b1Ref_L * b1LPRef_L + b11Ref_L * b2Ref_L * b0LPRef_L + b22Ref_L * b1Ref_L * b0LPRef_L + b22Ref_L * b0Ref_L * b1LPRef_L);

    b4fRef_L = incRef_L*(b2Ref_L * b2LPRef_L * b00Ref_L + b11Ref_L * b1Ref_L * b2LPRef_L + b11Ref_L * b2Ref_L * b1LPRef_L + b2Ref_L * b0LPRef_L * b22Ref_L + b22Ref_L * b1Ref_L * b1LPRef_L + b22Ref_L * b0Ref_L * b2LPRef_L);
    b5fRef_L = incRef_L*(b11Ref_L * b2Ref_L * b2LPRef_L + b22Ref_L * b2Ref_L * b1LPRef_L + b22Ref_L * b1Ref_L * b2LPRef_L);

    b6fRef_L = incRef_L*(b22Ref_L * b2Ref_L * b2LPRef_L);


    // DSP cycle
    for (int i = 0; i < n; ++i)
    {

        inR[0] = (t_sample)input[i];
        inL[0] = (t_sample)input[i];

        if (ref[n-1] == 0) {


            outR[0] = b0f_R*inR[0] + b1f_R*inR[1] + b2f_R*inR[2] + b3f_R*inR[3] + b4f_R*inR[4] + b5f_R*inR[5] + b6f_R*inR[6] - a1f_R*outR[1] - a2f_R*outR[2] - a3f_R*outR[3] - a4f_R*outR[4] - a5f_R*outR[5] - a6f_R*outR[6];

            outL[0] = b0f_L*inL[0] + b1f_L*inL[1] + b2f_L*inL[2] + b3f_L*inL[3] + b4f_L*inL[4] + b5f_L*inL[5] + b6f_L*inL[6] - a1f_L*outL[1] - a2f_L*outL[2] - a3f_L*outL[3] - a4f_L*outL[4] - a5f_L*outL[5] - a6f_L*outL[6];


            outBufferL[delaySamplesL] = outL[0];
            outBufferR[delaySamplesR] = outR[0];


            outputR[i] = dry*(t_sample)input[i] + wet*gR*outBufferR[0];
            outputL[i] = dry*(t_sample)input[i] + wet*gL*outBufferL[0];


            for (int k = 5; k >= 0; k--) {
```

```
                    outL[k+1] = outL[k];
                    outR[k+1] = outR[k];
                    inL[k+1] = inL[k];
                    inR[k+1] = inR[k];

                }

                inL[0] = 0;
                inR[0] = 0;
                outL[0] = 0;
                outR[0] = 0;

                for (int k = 0; k < len; k++) {

                    outBufferR[k] = outBufferR[k+1];
                    outBufferL[k] = outBufferL[k+1];
                }

                outBufferL[len] = 0;
                outBufferR[len] = 0;

            }else if (ref[n-1] == 1) {


                outR[0] = b0f_R*inR[0] + b1f_R*inR[1] + b2f_R*inR[2] + b3f_R*inR[3]
 + b4f_R*inR[4] + b5f_R*inR[5] + b6f_R*inR[6] - a1f_R*outR[1] - a2f_R*outR[2] -
 a3f_R*outR[3] - a4f_R*outR[4] - a5f_R*outR[5] - a6f_R*outR[6];

                outL[0] = b0f_L*inL[0] + b1f_L*inL[1] + b2f_L*inL[2] + b3f_L*inL[3]
 + b4f_L*inL[4] + b5f_L*inL[5] + b6f_L*inL[6] - a1f_L*outL[1] - a2f_L*outL[2] -
 a3f_L*outL[3] - a4f_L*outL[4] - a5f_L*outL[5] - a6f_L*outL[6];


                //
                out2R[0] = b0fRef_R*outR[0] + b1fRef_R*outR[1] + b2fRef_R*outR[2] +
 b3fRef_R*outR[3] + b4fRef_R*outR[4] + b5fRef_R*outR[5] + b6fRef_R*outR[6] -
a1fRef_R*out2R[1] - a2fRef_R*out2R[2] - a3fRef_R*out2R[3] - a4fRef_R*out2R[4] -
 a5fRef_R*out2R[5] - a6fRef_R*out2R[6];

                out2L[0] = b0fRef_L*outL[0] + b1fRef_L*outL[1] + b2fRef_L*outL[2] +
 b3fRef_L*outL[3] + b4fRef_L*outL[4] + b5fRef_L*outL[5] + b6fRef_L*outL[6] -
a1fRef_L*out2L[1] - a2fRef_L*out2L[2] - a3fRef_L*out2L[3] - a4fRef_L*out2L[4] -
 a5fRef_L*out2L[5] - a6fRef_L*out2L[6];


                out3R[0] = br[0]*out2R[0] + br[1]*out2R[1] + br[2]*out2R[2] + br
[3]*out2R[3] - ar[0]*out3R[1] - ar[1]*out3R[2] - ar[2]*out3R[3];

                out3L[0] = br[0]*out2L[0] + br[1]*out2L[1] + br[2]*out2L[2] + br
[3]*out2L[3] - ar[0]*out3L[1] - ar[1]*out3L[2] - ar[2]*out3L[3];


                outBufferL[delaySamplesL] = outBufferL[delaySamplesL] + gL*outL[0];
                outBufferR[delaySamplesR] = outBufferR[delaySamplesR] + gR*outR[0];

                outBufferL[delaySamplesL_tot] = outBufferL[delaySamplesL_tot] +
gRef_L*out3L[0];
                outBufferR[delaySamplesR_tot] = outBufferR[delaySamplesR_tot] +
gRef_R*out3R[0];
```

```


                outputR[i] = dry*(t_sample)input[i] + wet*(outBufferR[0]);
                outputL[i] = dry*(t_sample)input[i] + wet*(outBufferL[0]);


                for (int k = 5; k >= 0; k--) {

                    outL[k+1] = outL[k];
                    outR[k+1] = outR[k];
                    inL[k+1] = inL[k];
                    inR[k+1] = inR[k];
                    out2L[k+1] = out2L[k];
                    out2R[k+1] = out2R[k];

                }

                for (int k = 2; k >= 0; k--) {

                    out3L[k+1] = out3L[k];
                    out3R[k+1] = out3R[k];

                }


                for (int k = 0; k < len; k++) {

                    outBufferR[k] = outBufferR[k+1];
                    outBufferL[k] = outBufferL[k+1];
                }

                outBufferL[len] = 0;
                outBufferR[len] = 0;


            }// end if ref

        }//end dsp cycle

        return;
    } //end process cycle

    //global variables
private:

    int len;

    float *outR;
    float *outL;
    float *out2R;
    float *out2L;
    float *out3R;
    float *out3L;

    float *inR;
    float *inL;

    float *outBufferR;
    float *outBufferL;

```

```
    float **a_Ref;
    float **b_Ref;

    float ar[3];
    float br[4];

    int fLPValues[36];
    int incValues[36];

    float fc1_R, fc1Ref_R;
    float fc2_R, fc2Ref_R;
    float inc_R, incRef_R;

    float fc1_L, fc1Ref_L;
    float fc2_L, fc2Ref_L;
    float inc_L, incRef_L;

    float fcLP_R, fcLPRef_R;
    float fcLP_0_R, fcLPRef_0_R;
    float fcLP_1_R, fcLPRef_1_R;

    float fcLP_L, fcLPRef_L;
    float fcLP_0_L, fcLPRef_0_L;
    float fcLP_1_L, fcLPRef_1_L;

    float Q1_R, Q1Ref_R;
    float Q2_R, Q2Ref_R;
    float G1_R, G1Ref_R;
    float G2_R, G2Ref_R;
    float V1_R, V1Ref_R;
    float V2_R, V2Ref_R;

    float Q1_L, Q1Ref_L;
    float Q2_L, Q2Ref_L;
    float G1_L, G1Ref_L;
    float G2_L, G2Ref_L;
    float V1_L, V1Ref_L;
    float V2_L, V2Ref_L;

    float K1_R, K1Ref_R;
    float K1_L, K1Ref_L;

    float a1_R, a1Ref_R;
    float a1_L, a1Ref_L;

    float a2_R, a2Ref_R;
    float a2_L, a2Ref_L;

    float b0_R, b0Ref_R;
    float b0_L, b0Ref_L;

    float b1_R, b1Ref_R;
    float b1_L, b1Ref_L;

    float b2_R, b2Ref_R;
    float b2_L, b2Ref_L;

    float K2_R, K2Ref_R;
    float K2_L, K2Ref_L;

```

```
    float a11_R, a11Ref_R;
    float a11_L, a11Ref_L;

    float a22_R, a22Ref_R;
    float a22_L, a22Ref_L;

    float b00_R, b00Ref_R;
    float b00_L, b00Ref_L;

    float b11_R, b11Ref_R;
    float b11_L, b11Ref_L;

    float b22_R, b22Ref_R;
    float b22_L, b22Ref_L;

    float a1f_R, a1fRef_R;
    float a1f_L, a1fRef_L;

    float a2f_R, a2fRef_R;
    float a2f_L, a2fRef_L;

    float a3f_R, a3fRef_R;
    float a3f_L, a3fRef_L;

    float a4f_R, a4fRef_R;
    float a4f_L, a4fRef_L;

    float a5f_R, a5fRef_R;
    float a5f_L, a5fRef_L;

    float a6f_R, a6fRef_R;
    float a6f_L, a6fRef_L;

    float b0f_R, b0fRef_R;
    float b0f_L, b0fRef_L;

    float b1f_R, b1fRef_R;
    float b1f_L, b1fRef_L;

    float b2f_R, b2fRef_R;
    float b2f_L, b2fRef_L;

    float b3f_R, b3fRef_R;
    float b3f_L, b3fRef_L;

    float b4f_R, b4fRef_R;
    float b4f_L, b4fRef_L;

    float b5f_R, b5fRef_R;
    float b5f_L, b5fRef_L;

    float b6f_R, b6fRef_R;
    float b6f_L, b6fRef_L;

    float KLP_R, KLPRef_R;
    float KLP_L, KLPRef_L;

    float a2LP_R, a2LPRef_R;
    float a2LP_L, a2LPRef_L;

```

```
    float a1LP_R, a1LPRef_R;
    float a1LP_L, a1LPRef_L;

    float b0LP_R, b0LPRef_R;
    float b0LP_L, b0LPRef_L;

    float b1LP_R, b1LPRef_R;
    float b1LP_L, b1LPRef_L;

    float b2LP_R, b2LPRef_R;
    float b2LP_L, b2LPRef_L;

};

/* the data structure for each copy of "virtual_tube_delay_tilde".  In this case we
 only need pd's obligatory header (of type t_object). */
typedef struct VIRTUAL_TUBE_DELAY_tilde
{
    t_object    x_ob;
    VIRTUAL_TUBE_DELAYObj *dd;
    float      default_input;
} t_VIRTUAL_TUBE_DELAY_tilde;

/* this is a pointer to the class for "virtual_tube_delay_tilde", which is created in the
 "setup" routine below and used to create new ones in the "new" routine. */
t_class *VIRTUAL_TUBE_DELAY_tilde_class;

/* ------------------------------------------------------------------------------ */
/* ------------------------------------------------------------------------------ */

/* this is called when a new "virtual_tube_delay_tilde" object is created. */
void *VIRTUAL_TUBE_DELAY_tilde_new(void)
{
    post("VIRTUAL_TUBE_DELAY_new");

    t_VIRTUAL_TUBE_DELAY_tilde *x = (t_VIRTUAL_TUBE_DELAY_tilde *)pd_new(
    VIRTUAL_TUBE_DELAY_tilde_class);
    x->default_input = 0;
    x->dd = NULL;

    try
    {
        // call to the constructor
        x->dd = new VIRTUAL_TUBE_DELAYObj();

    }
    catch (int n)
    {
        x->dd = 0;
        post("VIRTUAL_TUBE_DELAY_new catched error ");
        return (void*) 0;
    }

    // delaytime inlet left
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // delaytime inlet right
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gain inlet left
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
```

```
    // gain inlet right
    inlet_new(&x->x_ob, &x->x_ob.ob_pd, &s_signal, &s_signal);
    // param inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // size inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // ref inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // lengthRefL inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // lengthRefR inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gainRefL inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);
    // gainRefR inlet
    inlet_new(&x->x_ob , &x->x_ob.ob_pd, &s_signal, &s_signal);

    // stereo outlet
    outlet_new(&x->x_ob, &s_signal);
    outlet_new(&x->x_ob, &s_signal);

    return (void *)x;
}

static void VIRTUAL_TUBE_DELAY_tilde_delete(t_VIRTUAL_TUBE_DELAY_tilde *x)
{
    post("VIRTUAL_TUBE_DELAY_delete");

    if (x->dd) {
        delete x->dd; x->dd = NULL;
    }


}

t_int *VIRTUAL_TUBE_DELAY_tilde_perform(t_int *w)
{
    t_VIRTUAL_TUBE_DELAY_tilde *x = (t_VIRTUAL_TUBE_DELAY_tilde *)(w[1]);   //obj
    ref

    // 4 signal input: signal, time, gain, dry/wet and changeF
    t_sample  *input = (t_sample *)(w[2]);  //input samples mono
    t_sample  *lengthL = (t_sample *)(w[3]);  //time left
    t_sample  *lengthR = (t_sample *)(w[4]);  //time right
    t_sample  *gainL = (t_sample *)(w[5]);  //gain left
    t_sample  *gainR = (t_sample *)(w[6]);  //gain right
    t_sample  *size = (t_sample *)(w[7]);   //size
    t_sample  *ref = (t_sample *)(w[8]);   //ref
    t_sample  *lengthRefL = (t_sample *)(w[9]); //lenght ref left
    t_sample  *lengthRefR = (t_sample *)(w[10]); //lenght ref right
    t_sample  *gainRefL = (t_sample *)(w[11]);   //gain ref left
    t_sample  *gainRefR = (t_sample *)(w[12]);   //gain ref right
    t_sample  *dry_wet = (t_sample *)(w[13]);  //dry/wet


    // 2 signal output: filtered stereo signal
    t_sample *outputL = (t_sample *)(w[14]); //output samples left
    t_sample *outputR = (t_sample *)(w[15]); //output samples right

    int n = (int)(w[16]);
```

```

    x->dd->process(input, lengthL, lengthR, gainL, gainR, size, ref, lengthRefL, lengthRefR, gainRefL, gainRefR, dry_wet, outputR, outputL, n);

    return (w+17);
}

void VIRTUAL_TUBE_DELAY_tilde_dsp(t_VIRTUAL_TUBE_DELAY_tilde *x, t_signal **sp)
{
    dsp_add(VIRTUAL_TUBE_DELAY_tilde_perform,
            16,
            x,
            sp[0]->s_vec, //inlet1
            sp[1]->s_vec, //inlet2
            sp[2]->s_vec, //inlet3
            sp[3]->s_vec, //inlet4
            sp[4]->s_vec, //inlet5
            sp[5]->s_vec, //inlet6
            sp[6]->s_vec, //inlet7
            sp[7]->s_vec, //inlet8
            sp[8]->s_vec, //inlet9
            sp[9]->s_vec, //inlet10
            sp[10]->s_vec, //inlet11
            sp[11]->s_vec, //inlet12
            sp[12]->s_vec, //outlet1
            sp[13]->s_vec, //outlet2
            sp[0]->s_n); //vector size
}

/* this is called once at setup time, when this code is loaded into Pd. */
extern "C" {
#ifdef WIN32
    __declspec(dllexport) void VIRTUAL_TUBE_DELAY_tilde_setup(void)
#else
    void VIRTUAL_TUBE_DELAY_tilde_setup(void)
#endif
    {
        post("VIRTUAL_TUBE_DELAY_tilde_setup");
        // creation of the virtual_tube_delay~ instance
        VIRTUAL_TUBE_DELAY_tilde_class = class_new(gensym("VIRTUAL_TUBE_DELAY~"),
                                                   (t_newmethod) VIRTUAL_TUBE_DELAY_tilde_new,
                                                   (t_method) VIRTUAL_TUBE_DELAY_tilde_delete, sizeof(t_VIRTUAL_TUBE_DELAY_tilde),
                                                   0,
                                                   (t_atomtype) 0);

        // sound processing
        class_addmethod(VIRTUAL_TUBE_DELAY_tilde_class,
                        (t_method)VIRTUAL_TUBE_DELAY_tilde_dsp, gensym("dsp"), (t_atomtype) 0);

        // signal input
        CLASS_MAINSIGNALIN(VIRTUAL_TUBE_DELAY_tilde_class, t_VIRTUAL_TUBE_DELAY_tilde, default_input);
    }
}
```

Bibliography

[Zöl11] U. Zölzer, ed. *DAFX – Digital Audio Effects*. Second. John Wiley & Sons, 2011.

[OB60] H. F. Olson and J. C. Bleazey. "Synthetic Reverberator". In: *J. Audio Eng. Soc.* 8.1 (1960), pp. 37–41.

[FR08] F. Fontana and D. Rocchesso. "Auditory distance perception in an acoustic pipe". In: *ACM Trans. Appl. Percpt.* 5.3 (2008). Article 16.

[GAF16] M. Geronazzo, F. Avanzini, and F. Fontana. "Auditory navigation with a tubular acoustic model for interactive distance cues and personalized head-related transfer functions". In: *J. Multimodal User Interfaces* 10.3 (2016), pp. 273–284.

[Väl95] Vesa Välimäki. *Discrete-time modeling of acoustic tubes using fractional delay filters*. Helsinki University of Technology, 1995.

[Roc00] Davide Rocchesso. "Fractionally addressed delay lines". In: *IEEE Transactions on Speech and Audio Processing* 8.6 (2000), pp. 717–727.

[Mat98] Inc MathWorks. *Signal Processing Toolbox: For Use with MATLAB : Computation, Visualization, Programming*. User's guide, Version 5. MathWorks Incorporated, 1998. URL: `https://books.google.it/books?id=pprwswEACAAJ`.

[MK06] Sanjit Kumar Mitra and Yonghong Kuo. *Digital signal processing: a computer-based approach*. Vol. 2. McGraw-Hill Higher Education New York, 2006.

[Smi+02] J. Smith et al. "Doppler simulation and the Leslie". In: *Proc. Workshop Digital Audio Effects (DAFx-02)*. Hamburg, Germany, 2002, pp. 188–191.

[KMV08] R. Kronland-Martinet and T. Voinier. "Real-time perceptual simulation of moving sources: application to the Leslie cabinet and 3D sound immersion". In: *EURASIP J. Audio, Speech, and Music Processing* 2008.849696 (2008).

[HHA09] J. Herrera, C. Hanson, and J. S. Abel. "Discrete time emulation of the Leslie speaker". In: *Proc. Audio Eng. Soc. 127th Conv.* New York, NY, USA, 2009.

[AAS08] S. Arnardottir, J.S. Abel, and J.O. Smith. "A digital model of the Echoplex tape delay". In: *Proc. Audio Eng. Soc. 125th Conv.* San Francisco, CA, USA, 2008.

[Väl+11] V. Välimäki et al. "Virtual analog effects". In: *DAFX: Digital Audio Effects, Second Edition*. Ed. by U. Zölzer. Wiley, 2011, pp. 473–522.

[Dut+11] P. Dutilleux et al. "Filters and delays". In: *DAFX: Digital Audio Effects, Second Edition*. Ed. by U. Zölzer. Wiley, 2011, pp. 47–81.

[RS10] C. Raffel and J. Smith. "Practical modeling of bucket-brigade device circuits". In: *Proc. 13th Int. Conf. Digital Audio Effects (DAFx'10)*. Graz, Austria, 2010.

[Far00] A. Farina. “Simultaneous measurement of impulse response and distortion with a swept-sine technique”. In: *Proc. Audio Eng. Soc. 108th Conv.* Paris, France, 2000.

[Hen81] C. A. Henricksen. “Unearthing the mysteries of the Leslie cabinet”. In: *Recording Engineer/Producer Magazine* (1981), pp. 130–134.

[ST69] FLJ Sangster and K Teer. “Bucket-brigade electronics: new possibilities for delay, time-axis conversion, and scanning”. In: *IEEE Journal of Solid-State Circuits* 4.3 (1969), pp. 131–136.

[Val+10] Vesa Valimaki et al. “Introduction to the special issue on virtual analog audio effects and musical instruments”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.4 (2010), pp. 713–714.

[Pak+11] Jyri Pakarinen et al. “Recent advances in real-time musical effects, synthesis, and virtual analog models”. In: *EURASIP Journal on Advances in Signal Processing* 2011.1 (2011), p. 940784.

[RF04] Thomas D Rossing and Neville H Fletcher. *Principles of vibration and sound.* 2004.

[Sch79] Manfred R Schroeder. “Integrated-impulse method measuring sound decay without using impulses”. In: *The Journal of the Acoustical Society of America* 66.2 (1979), pp. 497–500.

[Hey67] Richard C Heyser. “Acoustical measurements by time delay spectrometry”. In: *Journal of the Audio Engineering Society* 15.4 (1967), pp. 370–382.

[Alr83] H Alrutz. “A fast transform method for the evaluation of measurements using pseudo-random test signal”. In: *Proc. 11th ICA (Paris)* 6 (1983), pp. 235–238.

[MM95] Eckard Mommertz and Swen Müller. “Measuring impulse responses with digitally pre-emphasized pseudorandom noise derived from maximum-length sequences”. In: *Applied Acoustics* 44.3 (1995), pp. 195–214.

[SN99] Peter Svensson and Johan L Nielsen. “Errors in MLS measurements caused by time variance in acoustic systems”. In: *Journal of the Audio Engineering Society* 47.11 (1999), pp. 907–927.

[Aos81] Nobuharu Aoshima. “Computer-generated pulse signal applied for sound measurement”. In: *The Journal of the Acoustical Society of America* 69.5 (1981), pp. 1484–1488.

[Rea70] N Ream. “Nonlinear identification using inverse-repeatm sequences”. In: *Proceedings of the Institution of Electrical Engineers.* Vol. 117. 1. IET. 1970, pp. 213–218.

[SEA02] Guy-Bart Stan, Jean-Jacques Embrechts, and Dominique Archambeau. “Comparison of different impulse response measurement techniques”. In: *Journal of the Audio Engineering Society* 50.4 (2002), pp. 249–262.

[MM12] Gerhard Müller and Michael Möser. *Handbook of engineering acoustics.* Springer Science & Business Media, 2012.

[MM01] Swen Müller and Paulo Massarani. “Transfer-function measurement with sweeps”. In: *Journal of the Audio Engineering Society* 49.6 (2001), pp. 443–471.

[Far07] Angelo Farina. “Advancements in impulse response measurements by sine sweeps”. In: *Audio Engineering Society Convention 122.* Audio Engineering Society. 2007.

[KNH98] Ole Kirkeby, Philip A Nelson, and Hareo Hamada. “The’Stereo Dipole’: A virtual source imaging system using two closely spaced loudspeakers”. In: *Journal of the Audio Engineering Society* 46.5 (1998), pp. 387–395.

[RH04] Sjoerd W Rienstra and Avraham Hirschberg. "An introduction to acoustics". In: *Eindhoven University of Technology* 18 (2004), p. 19.

[Smi07] Julius Orion Smith. *Introduction to digital filters: with audio applications*. Vol. 2. Julius Smith, 2007.

[Ava01] Federico Avanzini. "Computational issues in physically-based sound models". In: (2001).

Acknowledgements

It has been a long way to get here, all that remains for me now is to express my thanks to the people around me who have helped to achieve this result.

I would first like to thank my parents and my sister for the unconditional love, continuous encouragement and unfailing support throughout my life. Without them and their sacrifice, I wouldn't be here and this accomplishment would not have been possible. I could not have been luckier.

I would also like to express my profound gratitude to Prof. Federico Avanzini of the Dept. of Computer Science at University of Milan. The support of my study and the opportunities he offered me. It has been very important and has enabled me to grow both as a person and as a student. His whose knowledge and encouragement have guided me through these past years in the right direction.

My sincere thanks also goes to Prof. Antonio Rodà of the Dept. of Information Engineering at University of Padova. His support and advice have been essential to complete my studies. In these years, he has contributed to my achievement encouraging and stimulating me.

I would also like to acknowledge Prof. Vesa Välimäki and Phd. stud. Juho Liski, my advisor, of the Dept. of Signal Processing and Acoustics at Aalto University as my guides during the wonderful time spent at Aalto University. I would like to express my sincere gratitude for the amazing opportunity that I received. Their patience, enthusiasm, and immense knowledge have given me the proper support for my master thesis. I also thank everyone in Otakaari 5G for the stimulating discussions, the football table matches and hospitality that made me feel like home.

Finally, I would like to thank all my dearest friends for the support and the patience throughout these years.

I have learnt a lot from all of you. I sincerely thank you all.