Simulating Hearing Loss

Using a Transmission Line Cochlea Model

L. Koop

Simulating Hearing Loss

Using a Transmission Line Cochlea Model

by

L. Koop

to obtain the degree of Master of Applied Mathematics
in the faculty of EEMCS at the Delft University of Technology,
to be defended publicly on Thursday October 29, 2015 at 11:00 AM.

Student number:	4424387	
Project duration:	September 24, 2014 – October 29, 2015	
Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft, supervisor
	Dr. ir. P. W. J. van Hengel,	INCAS^3, supervisor
	Dr. J. L. A. Dubbeldam,	TU Delft

Preface

On the afternoon of June 26, 2001 I lived through a traumatic experience which profoundly impacted the direction my life would take. In the thirty minutes, during which time the life of my sister seemed to hang in the balance and the memory of which is distorted by adrenaline, my life went from quiet, to exceedingly chaotic, and then back to a promise of normality. That day taught me the value of life and I also saw quality of life in a new light. The events of that day impacted my decision to go through a rigorous Emergency Medical Technician (EMT) training, volunteer time to Emergency Medical Services, and when given the opportunity, to conduct research in medically related fields.

While enrolled at the Full-time Training in Anaheim, a graduate level bible college, I worked on their audio visual team. Therefore, armed with medical training and an interest in audio engineering the topic of "Simulating Hearing Loss" piqued my interest.

This topic of research was proposed by the research institute INCAS^3 based in Assen. It is therefore:

An INCAS^3 project.

The human ear is a very complex organ and there are many phenomena like the cochlear amplifier and various otoaucustic emissions that we are just beginning to understand. It is only logical to use models of the inner ear to simulate hearing loss in order to incorporate as many of the interdependent phenomena as possible. In this ground breaking research a transmission line cochlea model was used to simulate a damaged cochlea and a backward transformation method was developed to allow for sound to be reconstructed from the model of a damaged cochlea. Even early results show that this method for simulating hearing loss allows for the affects of a damaged region to be propagated to an area in the cochlea that is not damaged. This is a significant advantage over pure signal processing based hearing loss simulation.

Because this type of simulation had not been successful previously, some unforeseen complications also arose with the mathematical model that was believed to be tried and tested. For the back transformation of the sound to be possible, an energy handling flaw in the model first needed to be corrected.

My work on this project would not have succeeded if not for the help from many people. Most importantly I would like to thank Peter van Hengel for answering countless questions, pointing out helpful study material, and providing valuable feedback. Kees Vuik also helped in many ways, was always available to answer questions, and most importantly asked good questions in return.

Henk and Tamara, if it were not for your home, Delft would have been a lonelier place. To Pieter and Lilian Rodenburg, being able to come and help with the farm work was a great and needed study break. Jacob and Priscilla, as well as Roger, thank you for visiting me all the way from Belize. I shall always remember our time together in Europe fondly. To the Nieuwelink families in Vlaardingen, thank you for all the times that you had me in your homes. Without all of you this project would not have been successful.

L. Koop
Delft, October 2015

Contents

List of Acronyms and Definitions

0.1. Acronyms

CM	Cochlear model or model of the cochlea
CP	Cochlear Partition
CT	Combination Tone
DCP	Digital signal processing
DPOAE	Distortion Product Otoacoustic Emission
FFT	Fast Fourier transform
FIR	Or FIR filter corresponds to Finite impulse response filter
FT	Fourier transform
IFFT	Inverse fast Fourier transform
IFT	Inverse Fourier transform
ODE	Ordinary Differential Equation
RK4	4 step Runge-Kutta method
SOAE	Spontaneous Otoacoustic Emission

0.2. Definitions

Cochlea	The cochlea is in the inner ear and is the organ that transduces sound coming into the ear into nerve signals
Cochlear partition	The term used when the Basilar membrane, Reissner's membrane, and the scala media of the cochlea are all together viewed simply as the partition between the scala vestibuli and scala tympani

1 Introduction

Hearing loss is a significant disability and can greatly decrease the quality of life. In February of this year the World Health Organization published a news release with this stark title “1.1 Billion People at Risk of Hearing Loss”[11]. The main reason for this is the use of personal audio devices. Of course hearing loss is not something new, but due to the recent increase in noise exposure because of the increased use of music in popular culture, the relevance of caring for the hearing disabled will be catapulted to new importance. This in turn increases the importance of proper modeling of hearing and hearing damage. Accurate simulation of hearing loss could be used to aid in prevention and education, as well as faster development of improved hearing aids. Fortunately, with the advent of the computer, attempts have been made to simulate hearing loss, but most fall short of truly simulating what a hearing disabled person hears.

The human ear is a remarkable and also an exceedingly complex organ. Trying to simulate hearing loss is certainly not a new endeavor. However, the current typical simulator focuses mainly on a digital signal processing approach without the mechanics of the ear being taken into account. In such an approach the sound signal will be edited for certain frequency ranges. This helps to give an impression of what hearing loss may sound like. But such an approach will not be able to really capture the interdependent nature of damage in the hearing system.

To really be able to capture hearing loss with realism, a new approach to simulating hearing loss is necessary, an approach that simulates the physical mechanics of the ear as closely as possible. Of the three parts of the ear (to be introduced in Chapter 2) this report will focus largely on the inner ear, also known as the cochlea. A transmission line type mathematical model of the cochlea is central to the method of simulating hearing loss in the approach taken here. Because sound has not before been successfully resynthesized from this type of model, the new techniques developed will facilitate an entirely new approach to simulating hearing loss.

1.1. Research Question

The goal of the research project “Simulating Hearing Loss” is to resynthesize sound from a transmission line cochlea model of a damaged cochlea. Specific attention will be given to the minimization of sound artifacts.

In the case of the linear model there are a number of things that will need to be considered. One approach to getting the sound from the model is to use a so called inverse filter method (see Section 4.3). For this method the CM is viewed as a sound filter. Then in a very similar way as described in Section 4.1 an inverse filter can be found and the original sound is reconstructed. With this method some things that need to be considered are the effects that the following have on the quality of sound reconstruction

- Choice of oscillator(s)
- Choice of window
- Breadth of window
- Length of impulse response
- Number of oscillators in the CM

- How to combine the results from the different oscillators effectively
- The best way to reduce sound artifacts
- If there is an effect of using a repeated signal

1.2. Report Map

This report is divided into sections as follows. In Chapter 2 the reader is guided through the anatomy of the human ear. It explains the functions of the different components of the auditory system. Common physiological damages that lead to hearing loss are also presented. Because the models of the cochlea play such a central role in hearing loss simulation, Chapter 3 examines both linear and nonlinear models of the cochlea. (An original goal of this project was to also include nonlinear cochlea models.) Chapter 4 gives a broad picture of the solution method as well as the algorithms used. Both frequency domain variants and also time domain variants of the algorithms are considered. The differences between the two is also explained. Chapter 5 then examines the changes that were made to the cochlea model in order for sound reconstruction to work. In Chapter 6 the results of the sound reconstructions and hearing loss simulations are presented. The benefits of using this simulation technique is also pointed out. A final discussion, recommendations for future work, and conclusions are found in Chapter 7. Finally, for the reader interested in the MatLab source code, it is provided in appendices A and B.

2 Hearing and Hearing Loss

Of the five senses hearing is perhaps the one that contributes the most to one's situational awareness as well as our complex social interaction skills. It is therefore no surprise that any deterioration of hearing directly impacts the quality of life.

2.1. Components of the Hearing System

The main components of the auditory system are the outer ear, the middle ear, and the inner ear. The outer ear functions to funnel sound into the ear and is composed of the pinna and the ear canal. The middle ear transfers sound vibrations from the eardrum to the oval window of the inner ear or cochlea. Within the cochlea sound is neurally encoded to be transmitted to the brain via the cochlear nerve. See Figure 2.1.

2.1.1. The Outer and Middle Ear

The outer ear, consisting of the pinna, which we see, and the ear canal directs sound into our ear. The outer ear ends at the ear drum. Internal to the eardrum is the middle ear. Within the hollow cavity of the middle ear are three bones the malleus, the incus, and the stapes which are also known as the hammer, the anvil and the stirrup respectively. Together these bones are known as the ossicles. Of these bones the malleus is connected to the eardrum and the stapes is connected to, or rests on, the oval window of the inner ear. These bones allow for accurate sound wave transfer from the air-filled outer ear to the fluid-filled inner ear. If it were not for this mechanism then much of the sound would bounce away from the ear without making the air to fluid transition. The one remaining thing that we should point out here is that there are two muscles connected to these bones, the stapedius muscle and the tympani muscle. These muscles allow loud sound transfer to the inner ear to be damped when they contract in response to loud sounds.

2.1.2. The Cochlea

The inner ear consists of the vestibular system and the cochlea. The vestibular system enhances our balancing ability and is not related to hearing. Because of this, little attention will be given to it. The cochlea on the other hand will take center stage for much of this report. The cochlea is a coiled tube that resembles the shape of a snail. Within the cochlea are three chambers. The superior (superior versus inferior is medical terminology for a part of the body that is above or below another part of the body when the body is in a natural standing position) chamber is the scala vestibule and the inferior chamber is the scala tympani. Between these two is the scala media in which is the organ of Corti which contains the hair cells that sense the sound vibrations coming into the cochlea. The hair cells in the organ of Corti then transduce these pressure waves to action potentials. The action potentials travel to the brain as electrical stimulus via the cochlear nerve. The scala vestibuli and the scala tympani are connected at the apex. All three chambers of the cochlea are fluid filled. Both the scala vestibuli and the scala tympani are filled with perilymph and the scala media is filled with endolymph. These fluids are mechanically equivalent to water. They differ however in chemical composition, which is key to creating a potential difference over the hair cells, which is essential in the creation of nerve signals from the motion of the organ of Corti.

The two ends of the cochlea are referred to as the base, near the middle ear, and the apex, at the tip of the spiral. At the base are two membranes, the oval window (or fenestra ovalis) and the round window. The stapes

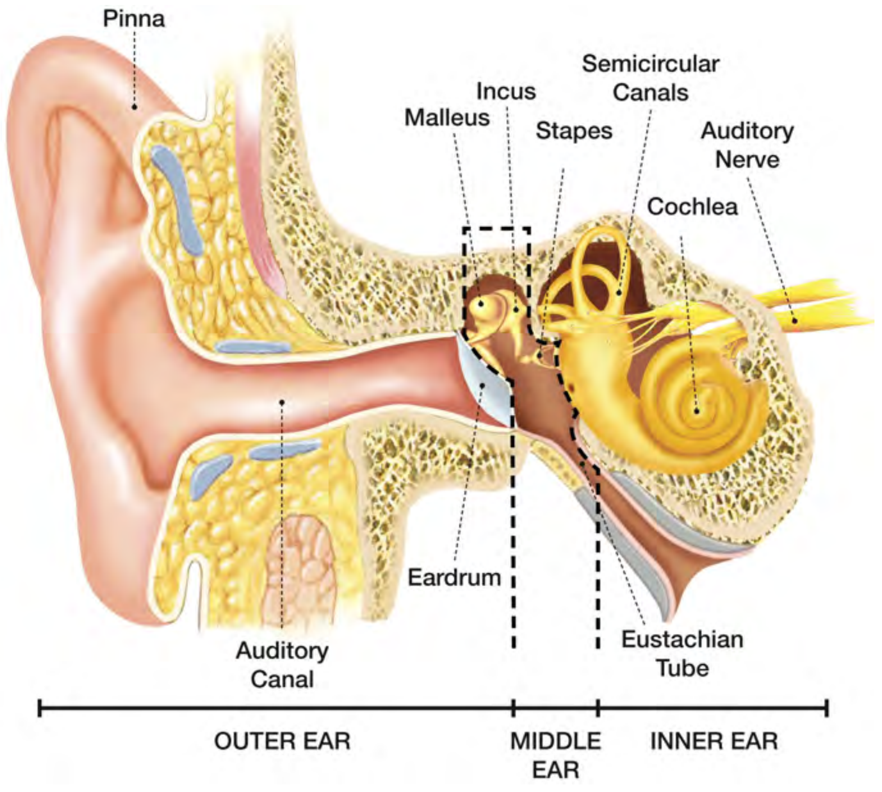

Figure 2.1: The human ear. Image source: [13]

bone from the middle ear rests on the oval window which is how sound comes to the cochlea. The apex of the cochlea is also called the helicotrema. In the mathematical model of the cochlea there was a fault in how energy was treated in this region. Therefore the helicotrema or apex will be frequently referred to, especially in Chapter 5.

The scala media is separated from the scala vestibule by the Reissner's membrane. And the basilar membrane separates the scala media from the scala tympani. These two membranes and the scala media are often referred to simply as the cochlear partition in models of the cochlea. The basilar membrane is also more frequently mentioned in the literature for two reasons. Firstly, the basilar membrane is the main contributor to the stiffness of the cochlear partition. Because of this it is the mechanically important membrane. Reissner's membrane, on the other hand, is acoustically transparent. Secondly the basilar membrane is easier to access. The cochlea is well protected in the very hard temporal bone. The geometry of the cochlea is also such that the scala tympani is easier to access, especially in rodents. Access to the cochlear partition via the scala tympani brings the basilar membrane into view. Therefore, when the literature cites measurements taken of the vibrations of the cochlear partition the basilar membrane is likely to be what was measured.

One should note that in the organ of Corti there are a few different rows of hair cells. The row of inner hair cells are responsible for the actual electrical encoding. Three rows of outer hair cells form a feedback system that most likely allows for amplifications of certain sounds.

2.1.3. The Frequency Map

Another important characteristic of the cochlea as it relates to the current study is the frequency map of the cochlea. The frequency range of human hearing is from 20 Hz and an upper limit of 22 kHz. The area in the cochlea close to the base is attributed to higher frequencies, and the region of the cochlea close to the apex to low frequencies. It is important to keep this in mind. The typical frequency content plot in audio processing places the low frequencies on the left and the high frequencies on the right. The models used in this study are typically viewed with the base depicted on the left and the apex/helicotrema on the right. Note that in this orientation of viewing a 1-D cochlea model the region responsible for the high frequencies would be viewed on the left and the low frequencies on the right, and that this is mirrored in an audio frequency content plot.

Fortunately the place frequency map of the cochlea follows a logarithmic scale. This is nice as this is also the scale that is frequently used in audio signal processing. Greenwood[4] was one of the first to publish the

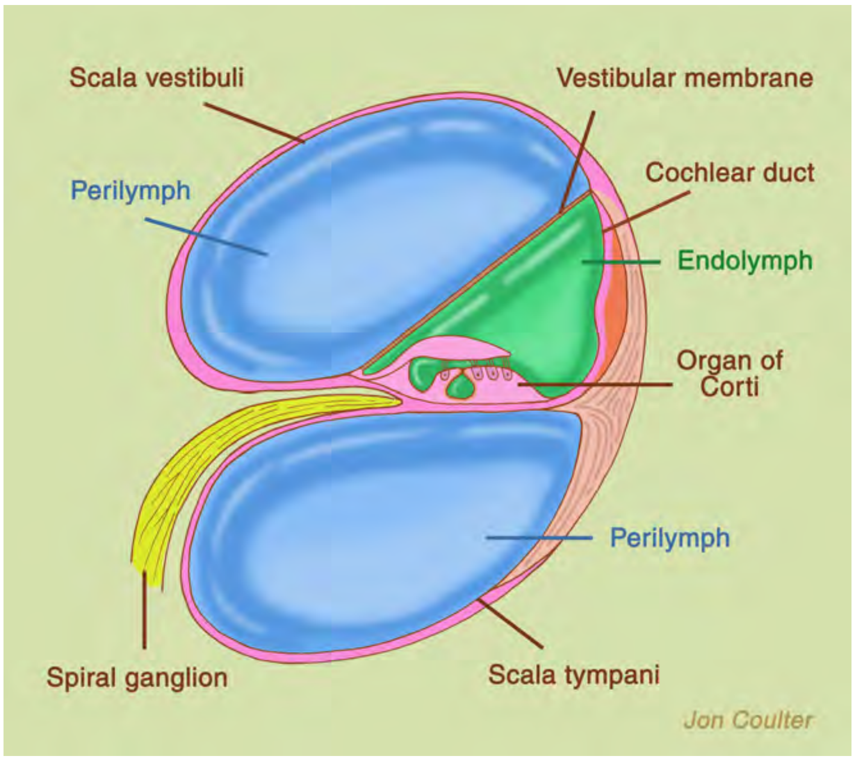

Figure 2.2: A cross section of the cochlea, Image source: [1]

place frequency map of the cochlea. A function for the frequency map is as follows:

$$f_c(x) = 165.4(10^{0.06(35-x)} - k) \tag{2.1}$$

Where $k = 0.85$ is the apical limit parameter. For a visual reference see Figure 2.3. In this picture the highest frequency is taken as 18kHz instead of the 21kHz simulated in the models used.

2.1.4. The Cochlear Nerve

There are just a few things that need to be mentioned related to the cochlear nerve. Obviously the sound signal is transferred from the cochlea to the brain via this nerve. Lopez and Barrios[7] showed that deafferentation of the auditory nerve can also be a source of hearing loss. An accurate hearing loss simulator would therefore need to take this into account. In this project this aspect of hearing loss is not included. Future work in this area should aim to simulate this mode of hearing loss as well.

2.2. Cochlear Emissions

From what is described until this point one may be tempted to believe that the cochlea is a relatively straight forward sound encoding organ. In truth however it is an exceedingly intricate and very finely tuned encoding organ capable of amplifying some sounds and damping others. Indicators of this are various otoaucustic emissions that were first experimentally verified by Kemp in the late 70s[5]. In essence, an otoaucustic emission is a sound that is generated from within the ear. There are two basic types of emissions the first being evoked emissions and the second being spontaneous emissions. Interestingly when the cochlea is damaged these emissions tend to decrease or entirely disappear. This is why measuring cochlear emissions has become a technique for measuring the health of the inner ear. And this is also why cochlear emissions are important in relation to a hearing loss simulator as one would want to use the information from an emissions test to set the parameters of the hearing loss simulator. Listed below are a few emissions that an accurate hearing loss simulator should take into account. This is, however, not an exhaustive list. For further information I recommend Chapter 4 of Cochlear Mechanics by Duifhuis[2].

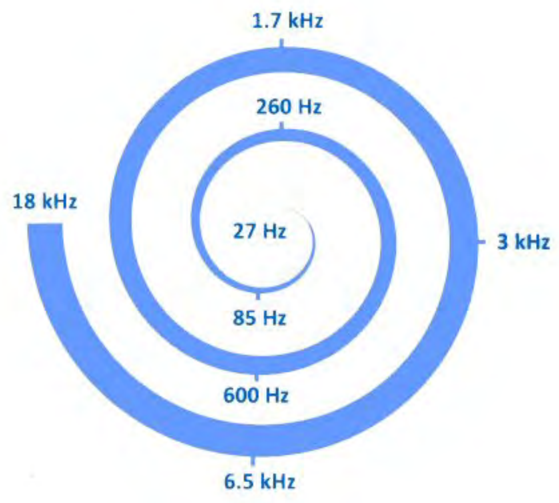

Figure 2.3: A visual place frequency map

2.2.1. Distortion Product Otoacoustic Emissions (DPOAEs)

One of the more important of the various emissions is the distortion product otoaucustic emission (DPOAE). This emission is evoked using two tones at frequencies f_1 and f_2 with $f_1 < f_2$. The result is that there are a number of emissions of the nonlinear mechanics of the organ of Corti at different frequencies, also referred to as combination tones. These combination tones are at frequencies $mf_1 - nf_2$ where m and n are integers and $mf_1 > nf_2$. The combination tone is referred to as even or odd if $m+n$ is even or odd respectively [2, p. 68]. Although there are a number of combination tones the ones that are understood the best are $2f_1 - f_2$ called the “cubic” distortion tone, and $f_2 - f_1$ which is referred to as the quadratic combination tone. These are likely to play a key role in parameter adjusting in the future.

2.2.2. Spontanious Otoacoustic Emissions (SOAEs)

Unlike the case of DPOAEs some cochlear emissions occur without being evoked, hence the term Spontaneous Otoacoustic Emission (SOAE). This implies the generation of sound in the cochlea without there being an input sound. This in turn is strong evidence for a cochlear amplifier.

2.2.3. Cochlear Amplifier

Although the best method to model a cochlear amplifier is not yet well defined, the existence of the physical amplifier and its workings are widely accepted although definitive experimental proof is still lacking. On the organ of Corti the outer hair cells form a positive electromechanical feedback mechanism that allows it to increase movement of the basilar membrane and thereby amplify the sound.

2.3. Hearing Loss

Although there are some cases of hearing loss that have relatively simple solutions like an obstructed ear canal, others are much more serious and do not have an easy fix. The severity of hearing loss is classified as mild, moderate, severe, profound, and totally deaf. Causes of hearing loss include aging, noise, chemicals, medications, illnesses, genetics, and trauma. Treatment depends, of course, on the condition. Some conditions benefit from surgery, for example, to remove scar tissue on the eardrum. Many conditions, however, cannot be directly treated and are alternately relieved by the use of hearing aids or even a cochlear implant. Although they may relieve the severity of the disability hearing aids and cochlear implants hardly restore the full function of the auditory system.

2.3.1. Age

Age is perhaps the most common source of hearing loss. This is most likely due to damage or wear on the delicate structures within the cochlea such as the inner and outer hair cells that do not regenerate when damaged. This type of hearing loss may start in the late twenties to early thirties and is typically progressive. Hearing loss due to age tends to affect the sensitivity to the higher frequencies primarily.

2.3.2. Noise Exposure

Hearing loss due to noise exposure is dependent on the dB level of the noise and the length of time that an individual is exposed to the noise. A 3 dB increase in level constitutes a doubling of the intensity of the sound and thus will do as much damage in half the exposure time. According to the Exposure Action Value, an eight hour exposure of noise at the 85 dB level is considered safe. However a conservative estimate for a safe exposure time at a 91 dB noise level is only two hours.

Oishi and Schacht[10] place noise related hearing loss at 5% of the global population. Unlike hearing loss due to age, loss due to noise exposure affects mainly the frequencies in the 3,000 - 6,000 Hz range. Also unlike loss due to age, loss due to noise exposure is something over which an individual has much control. Proper choices related to using ear protection or limiting exposure to loud music would significantly reduce later hearing loss.

Today there is an increased awareness of this type of hearing loss as well as better education related to it. One potential use for a high quality hearing loss simulator would be in educating young people of the damages of noise exposure. Physicians, parents, or even teachers could use such a tool to give young people a good impression of what it is like to live with hearing loss, and thereby encourage them to prevent such loss in the first place.

2.3.3. Inner Hair Loss

Related to the above causes of hearing loss, what really happens within the ear is that there is damage to the hair cell structures. Depending on where the damage is, there is a differing loss of function. Should there be damage to the inner hair cells then there is a direct loss to the electrical encoding mechanism. Although the BM may vibrate at the location of the given hair cell, it is unable to prompt a resulting electrical stimulus.

2.3.4. Outer Hair Loss

However in cases where the damage is related to the outer hair cells then there is a loss of functionality of the cochlear amplifier. The positive feedback brought about by the outer hair cells is reduced and this in turn decreases the sensitivity of the ear.

2.3.5. Auditory Deafferentation

Auditory deafferentation or the loss of nerve fibers is another area of hearing loss that is being studied. One very troublesome affect of hearing loss is decreased speech perception in noisy environments. A possible cause for this loss of perception is due to auditory deafferentation [8] [6] [7]. Furthermore, Lopez-Poveda and Barrios showed that this can be modeled by stochastically undersampling the sound waveform. As mentioned before, any advanced future hearing loss simulator must take this phenomenon into account.

2.3.6. Other Hearing Problems

Although simulating hearing loss is the main focus of this report, hearing problems are not limited solely to hearing loss. Should a hearing loss simulator be developed that took a nonlinear model as a base then the simulation of Tinnitus and Hyperacusis might also be a realizable objective.

Tinnitus

Tinnitus, or ringing of the ear, is one hearing problem not related to hearing loss. In essence a person perceives a sound despite the sound not really being there. Although it has many potential causes a common cause is noise induced hearing loss. There are two classifications of tinnitus one being subjective and the other being objective. Subjective tinnitus might be quite hard to accurately simulate as this is purely subjective, and therefore there is no way to measure or test for it. Although objective tinnitus can be tested for similarly to otoaucustic emissions the frequency and level cannot be measured directly. So any simulations would be qualitative.

Hyperacusis

Hyperacusis is an over sensitivity to sound. It is possible that for an individual it will only affect certain frequencies or volumes of sounds. Some theories place the fault of hyperacusis in the brain's perception of sound. Others link it to a fault in the tensor muscle in the middle ear. Because both are not directly related to a model of the cochlea this would require additional consideration and will therefore not be given precedence in this study.

3
Modeling the Cochlea

A wide range of cochlea models have been developed over the years. For a concise presentation of the history of these models Chapters 3 and 5 of *Cochlear Mechanics*[2] is a good reference. These models range from simple 1-D filter bank models to 3-D nonlinear models. Some model the cochlear cross-section and include micro mechanics. Most 3-D models simplify the scala media and the two membranes into one cochlear partition and do not account for the micro mechanics of the cochlea. For the hearing loss simulations in this report linear and nonlinear 1-D transmission line cochlea models are considered. The hearing loss simulations performed were only using the linear model. However, future hearing loss simulators should also include back-transformations from nonlinear models. For this reason both linear and nonlinear models are presented here.

The main equation of the transmission line cochlear model is:

$$\frac{\partial^2 p}{\partial x^2}(x,t) - \frac{2\rho\partial^2 y}{h\partial t^2}(x,t) = 0, \quad 0 \le x \le L, \quad t \ge 0 \tag{3.1}$$

In this equation $y(x,t)$ is the excitation of the oscillator (to be defined shortly, see Section 3.1), ρ represents the density of the cochlear fluid, and h is the height of the scala. The length of the cochlea is denoted by L and the position in the cochlea by x. The pressure on the cochlear partition $p(x,t)$ can be written as:

$$p(x,t) = m\ddot{y}(x,t) + d(x)\dot{y}(x,t) + s(x)y(x,t) \tag{3.2}$$

Here m is the mass per unit area and s and d are stiffness and damping terms respectively. It should be noted that both the stiffness and damping terms depend on the position x. Nonlinearities in the model are introduced via the parameters s and d and will be presented shortly.

To solve the equations we introduce a new term as the sum of the stiffness and damping terms as follows:

$$g(x,t) = d(x)\dot{y}(x,t) + s(x)y(x,t) \tag{3.3}$$

With that, $m\ddot{y}(x,t)$ can be rewritten as:

$$m\ddot{y}(x,t) = p(x,t) - g(x,t) \tag{3.4}$$

This allows us to then rewrite Equation 3.1 as in Equation 3.5 where $\kappa = \frac{2\rho}{hm}$.

$$\frac{\partial^2 p}{\partial x^2}(x,t) - \kappa p(x,t) = -\kappa g(x,t) \tag{3.5}$$

The Equation 3.5 at a single time point is an ODE (ordinary differential equation). The approach to solve the model of the cochlea is to discretize this ODE in space and solve the system in each time step. Furthermore, the Equation 3.2 can be transformed into two first order equations in time. Defining v as the partial derivative of $y(x,t)$ with respect to time gives the following two equations:

$$v(x,t) = \frac{\partial y(x,t)}{\partial t} \tag{3.6}$$

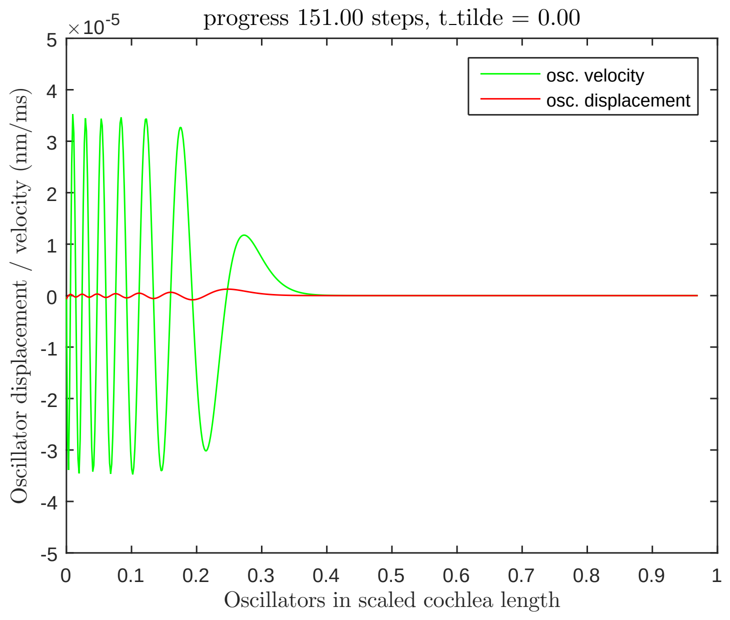

Figure 3.1: The movement of the cochlear partition in response to an impulse

$$\frac{\partial v(x,t)}{\partial t} = \frac{p(x,t) - g(x,t)}{m} \tag{3.7}$$

Equations (3.6) and (3.7) can be solved with RK4 (Classical 4th order Runge Kutta). Of course the actual solving of this system would require the inclusion of boundary conditions and some finer details of discretization.

3.1. Coupled Oscillators as the Spatial Discretization

The equation of the cochlea model, Equation 3.1, is numerically solved in discrete time steps as mentioned in the previous section. It is furthermore discretized in space into a set of N oscillators. Theoretically speaking what is referred to as an oscillator is just any point on the cochlear partition. In the discrete model, however, each discrete point on the partition can be viewed as an oscillator. For the rest of the report frequent use is made of this term “oscillator“.

Each oscillator in the model has a specific damping and stiffness governing its behavior as described by Equations 3.9 and 3.8. In addition each oscillator is coupled to the oscillators around it via the simulation of the cochlear fluid. This means that the simulated cochlear partition is not simply a set of independent oscillators that can move independently of each other solely according to their characteristic frequency determined by their respective stiffness. Rather, the movement of the oscillators in relation to each other looks similar to a traveling wave on a string. Figures 3.1 - 3.4 show the response of the model to an impulse in successive time steps and give an impression of the movement of the oscillators in relation to each other.

Almost any number of discrete points N can be used. Typically the model becomes more accurate with an increase in the number of oscillators used. For a linear model, as few as $N = 300$ oscillators gives good results. For a nonlinear model more oscillators are required. Using $N = 600$ is quite standard. For the simulation of some Otoacoustic Emissions even more are used, up to $N = 1200$. As long as the results are accurate enough it is beneficial to use fewer oscillators to minimize computation time and also to decrease memory demand.

For the rest of this report a model with $N = 600$ oscillators is used. This choice was made in an effort to keep parameters simpler with the thought that nonlinear simulations would also be conducted.

3.2. A Linear Model

In this research two cochlear models will be used. The first being a linear and the second being a nonlinear model. The main equation to be solved still remains Equation (3.1). However, the specific terms and arguments change.

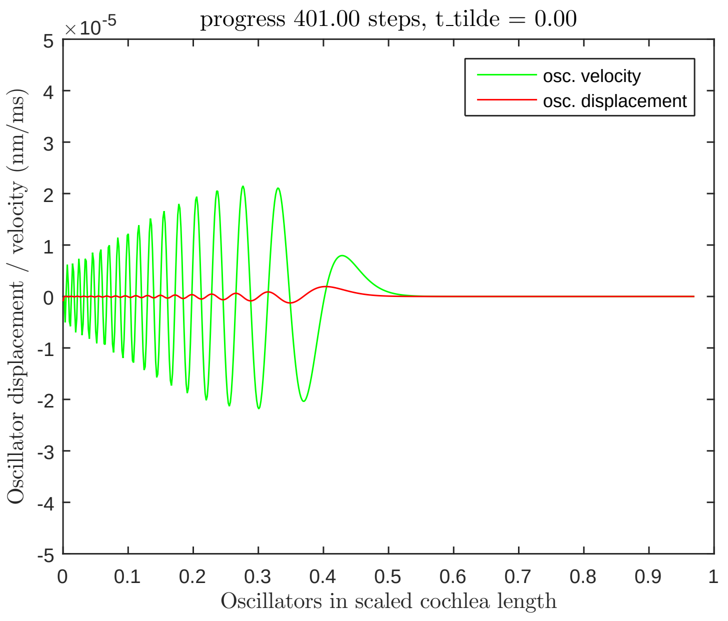

Figure 3.2: The movement of the cochlear partition in response to an impulse

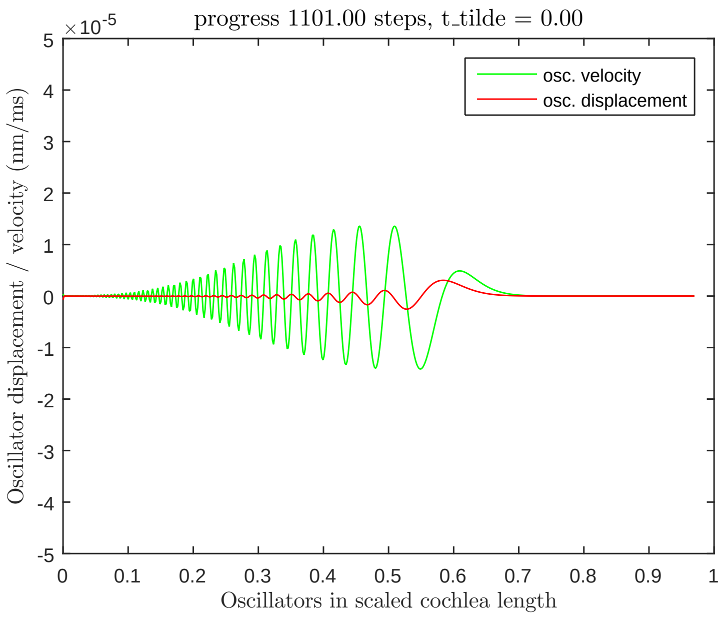

Figure 3.3: The movement of the cochlear partition in response to an impulse

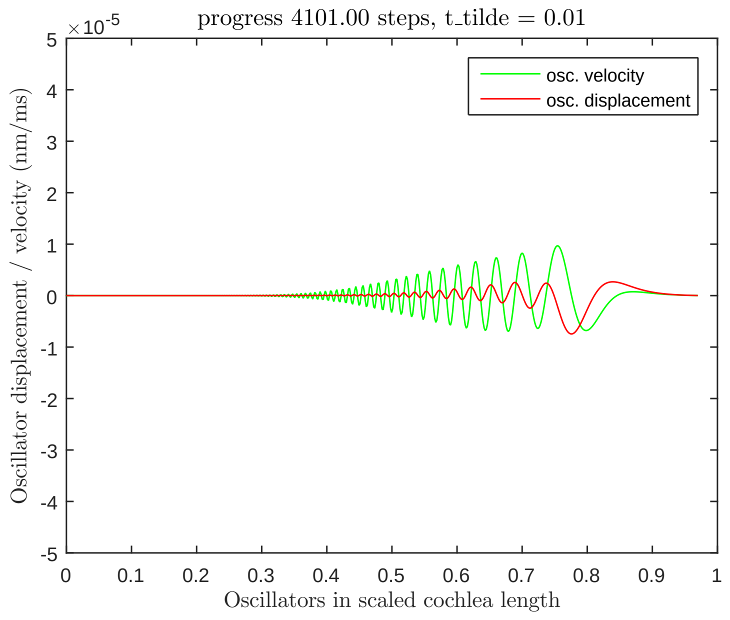

Figure 3.4: The movement of the cochlear partition in response to an impulse

Parameter	Value	Description
	Middle Ear:	
A_{st}	$3\times10^{-6}\mathrm{m}^2$	Stapes area
A_{tm}	$60\times10^{-6}\mathrm{m}^2$	Tympanic membrane area
ME_Q	0.4	Middle ear quality factor
ME_N	30	Middle ear transformer
	Cochlea:	
x	$0\le x\le 35\times10^{-3}\mathrm{m}$	Longitudinal coordinate
$\tilde{x}$	$0\le\tilde{x}\le 1$	Normalized length
m	$0.5\ kg\ \mathrm{m}^{-2}$	Membrane mass per unit area
ϵ	5×10^{-2}	Modulation factor for the impulse resonator
s_0	$10^{10}\frac{Pa}{\mathrm{m}}$	First stiffness term
λ	$300\mathrm{m}^{-1}$	Relationship between frequency and location x

Table 3.1: Parameter set of the linear cochlea model

3.2.1. A Linear Parameter Set

For the linear model the following are the parameters to be used. The mass m is as found in Table 3.1. The stiffness term is defined as follows:

$$s(x) = s_0 e^{-\lambda x} \tag{3.8}$$

And the damping term to be used is:

$$d(x) = \epsilon\sqrt{m\ s(x)} \tag{3.9}$$

3.2.2. Shortcomings of a Linear Model

From a modeling perspective it would be nice if a linear model could capture all the necessary processes of the cochlea. If this were the case then many classical techniques of digital signal processing would be sufficient to simulate hearing loss accurately. However, there are a number of phenomenon that simply cannot be captured with a linear model.

Among the things that a linear model cannot account for are [2, p. 60-61]:

Parameter	Value	Description
	Middle Ear:	
A_{st}	$3 \times 10^{-6}\text{m}^2$	Stapes area
A_{tm}	$60 \times 10^{-6}\text{m}^2$	Tympanic membrane area
ME_Q	0.4	Middle ear quality factor
ME_N	30	Middle ear transformer
	Cochlea:	
x	$0 \leq x \leq 35 \times 10^{-3}\text{m}$	Longitudinal coordinate
$\tilde{x}$	$0 \leq \tilde{x} \leq 1$	Normalized length
m	$0.375\ kg\ \text{m}^{-2}$	Membrane mass per unit area
$\tau(x)$	$1.742/f_R(x)$	Delay of feedback stiffness
γ	0.12	Amplitude plateau
δ_{sat}	$0.2 \times 10^{0.52\tilde{x}}$	Damping saturation
δ_{neg}	$-0.12 \times 10^{-0.17\tilde{x}}$	Negative damping
α	40	Middle ear turning point velocity
μ_α	6	Slope at lower turning point
β	-10	Lower turning point velocity
μ_β	5	Slope at lower turning point
A	$20\,832\ s^{-1}$	x-f map coefficient
λ	$60\ m^{-1}$	x-f map length constant
κ	$145.5\ s^{-1}$	x-f map correction
σ_{zweig}	$0.1416 \times 10^{-0.17\tilde{x}}$	Feedback stiffness amplitude
ϵ	0.01	Roughness scaling coefficient

Table 3.2: Parameter set of the nonlinear cochlea model

- Dynamic Range, that is, the compression of sound in the 100 - 120 dB level to and output of 30 - 50 dB in the auditory nerve.

- Combination tones

- Cochlear Emissions

Interestingly, with damage to the cochlea the above nonlinear processes tend to linearize.

Because of the shortcomings of the linear model, a nonlinear model should eventually also be considered. This research, however, will focus mainly on providing a solution to the linear case which should serve as a stepping stone to finding a back transformation to a nonlinear cochlea model.

3.3. A Nonlinear Model

A nonlinear model tries to account for the phenomenon evident in the cochlea that a linear model cannot capture. Over the years a number of models have been proposed. A particular nonlinear model that is regarded as the gold standard of transmission line models is set forth here.

3.3.1. The Parameter Set used by Epp et al.

The specific nonlinear model that we will consider is the one used by Bastian Epp and Jesko L. Verhey[3], and based on work by van den Raadt and Duifhuis (1990); van Hengel et al. (1996)[15]; Mauermann et al. (1999)[9] and Talmadge et al. (1998)[14].

Unlike the linear model, the damping term is not as straight forward for this model. The reason for this is that negative damping is used at low cochlear partition velocity. This is what allows the modeling of the cochlear amplifier as well as spontaneous otoaucustic emissions. The nonlinear damping term looks like this:

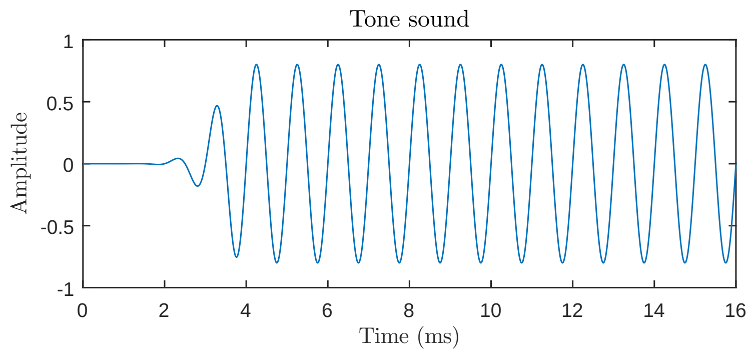

Figure 3.5: A simple tone sound wave with a gradual onset

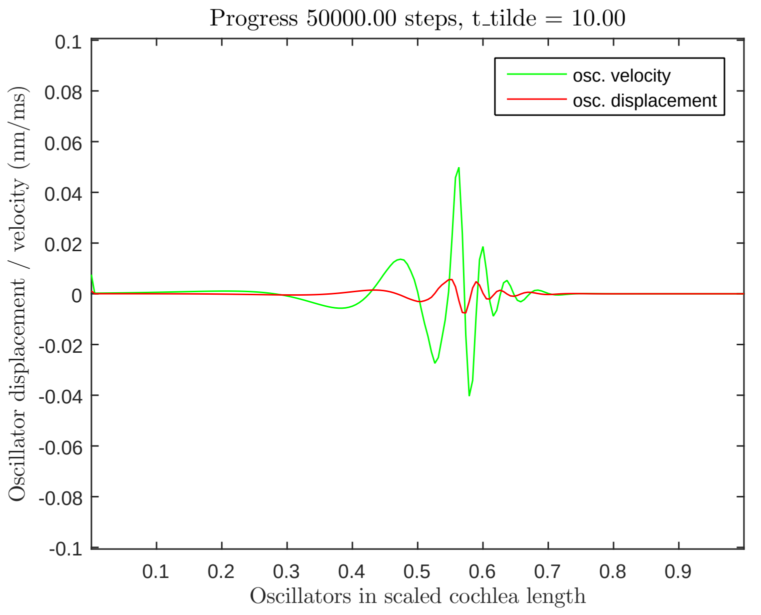

Figure 3.6: Modeling the response of the basilar membrane to the tone in Figure 3.5

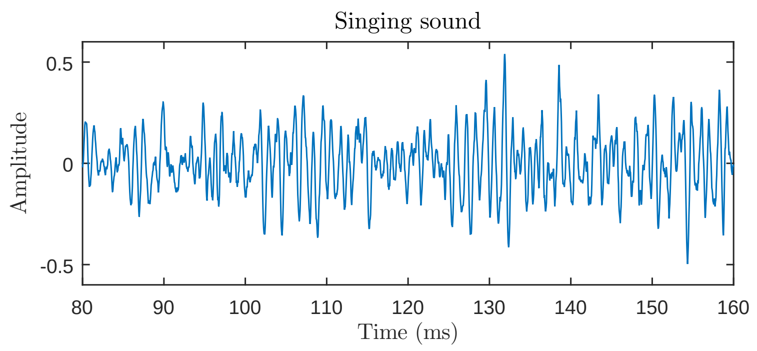

Figure 3.7: A short snippet of singing sound

$$d(x,\dot{y}) = \Big\{(1-\gamma)(\delta_{sat}-\delta_{neg})\left[1-\frac{1}{1+e^{(\Lambda-\alpha)/\mu_\alpha}}\right] + \gamma(\delta_{sat}-\delta_{neg})\left[1-\frac{1}{1+e^{\Lambda-\beta/\mu_\beta}}\right]+\delta_{neg}\Big\}\sqrt{m\ s(x)} \tag{3.10}$$

Where Λ is the velocity level in dB and defined by:

$$\Lambda = 20\log_{10}\left(\frac{|\dot{y}|}{y_0}\right), \quad y_0 = 10^{-6}\frac{m}{s} \tag{3.11}$$

Considering that there is negative damping below a certain threshold should immediately cause one to question stability. Indeed with just the introduction of negative damping there would be a stability problem. Therefore the following delayed feedback stiffness term c is needed, which vanishes at higher levels:

$$c(x,\dot{y}) = \Big\{(1-\gamma)(\sigma_{zweig})\left[\frac{1}{1+e^{(\Lambda-\alpha)/\mu_\alpha}}\right] + \gamma(\sigma_{zweig})\left[\frac{1}{1+e^{\Lambda-\beta/\mu_\beta}}\right]\Big\} \tag{3.12}$$

With the inclusion of this c term, then, the pressure term (see: Equation 3.2) for a specific time point changes to:

$$p(x) = m\ddot{y}(x) + d(x,\dot{y})\dot{y}(x) + s(x)[y(x) + c(\dot{y})y(t)|_{t-\tau}] \tag{3.13}$$

The delay in stiffness in the c term is due to the time shift or a delay ramp introduced by the use of the constant τ.

3.4. Model Inputs and Outputs

For the introduction to the model output let us first consider two sound inputs. The first of these sound inputs is a very simple tone sound with a gradual onset as seen in Figure 3.5. The second sound that will be considered is a more complex sound snippet from a song. This sound is seen in Figure 3.7.

The time representation of these two sounds, as seen in Figures 3.5 and 3.7, is very common in sound engineering. On the x-axis is time and on the y-axis is amplitude. Before digital sound representation, the amplitude axis was a plot of the volts content of a signal as received from a microphone. In digital sound representation this is simply a one dimensional array of samples in the range $[-1,1]$ in time. Note that this representation is not in dB scale.

Assume the input sound to the cochlear model is a 100 millisecond sound a. Assume further that the cochlear model runs at a sample rate of $50kHz$. If the sound a does not have a sample rate of 50 kHz then it would first need to be resampled to this sample rate. Strictly speaking, because the method used to solve the equations of the model is RK4, which uses two sample points per integration time step, the sound a needs to be resampled to a sample rate twice that of the model. That is, a would need to be resampled to $100kHz$. The

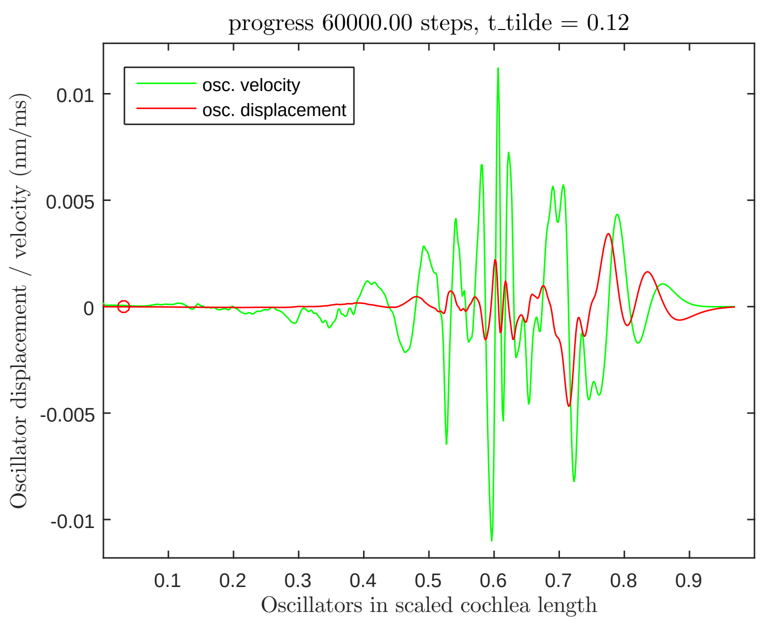

Figure 3.8: Modeling the response of the basilar membrane to a sound clip from a song. The red circle indicates the oscillator for which the trace is shown in Figure 3.9

input to the model would therefore be a numerical array $-1 \leq a(t) \leq 1$ of length $2l = (2)(100/1000)$ (50000) = 10000.

Because the first sound is a tone with a constant frequency the response within the cochlea model is seen mostly in a particular section of the cochlear partition. This is evident in Figure 3.6 by the fact that activity on the cochlear partition is restricted to a small area approximately in the middle of the cochlea. It should also be noted that the response here is shown only for a particular timestep and that at each timestep the response changes. The plot shows both the excitation of the cochlear partition as well as its velocity. The y-axis represents the excitation in nanometers and the velocity in nanometers per millisecond. The x-axis is the position in the cochlea and bears a little more explanation.

This report will make use of a few different scales of the position in the cochlea. Physically the cochlea is approximately 35mm in length. For the most part, plots in the report will not feature this length of 35mm. Rather, two other lengths will be used. The first, also the one seen in Figure 3.6, will be referred to as the "scaled length" of the cochlea. In this format, whatever the length of the cochlea is, it is scaled to fall in the range of $[0,1]$. The other format that will be used is according to the number of oscillators in the model. For example $[0,N]$ which will be $[0,600]$ in the simulations that follow.

A more intricate sound, like one of a group of people singing, would of course have a more interesting cochlear response than the one seen in Figure 3.6 as seen in Figure 3.8.

Given the input a mentioned earlier the output from the model would be an array $C(t)$ of traces from the $N = 600$ oscillators of size $(N \times l) = (600 \times 5000)$. As can be imagined the output from the cochlea model can quickly become a very large array, should the input sound be a multisecond sound. If not all the oscillator traces are needed for a computation it is advantageous to not record all N oscillator traces.

An oscillator trace that is a response to the sound in Figure 3.7 can be seen in Figure 3.9. Although this oscillator trace is a part of the model output, it is not the whole output. The entire output consists of N such traces. A very good way to view all of the data from the output of the model in a single picture is by a cochleogram.

A cochleogram is a comprehensive representation of the output of the cochlea model. On the x-axis is time, typically represented in milliseconds. On the y-axis is each oscillator. Then on the z-axis is the position of the given oscillator at a given point in time. In Figure 3.11 the z-axis is visible only as a color scale.

The process of computing a cochleogram is outlined in Figure 3.10. Basically, it is a positive envelope of an oscillator trace calculated from the analytic signal. The analytic signal $u_a(t)$ of a source signal $u(t)$ is a complex signal used commonly in signal processing and consists of $u(t)$ for the real part and the Hilbert transform of $u(t)$ for the imaginary part ($u_a(t) = u(t) + i \cdot H(u(t))$) where H denotes the Hilbert Transform. As a result, what is represented in the cochleogram is not the oscillations of the oscillators themselves but rather,

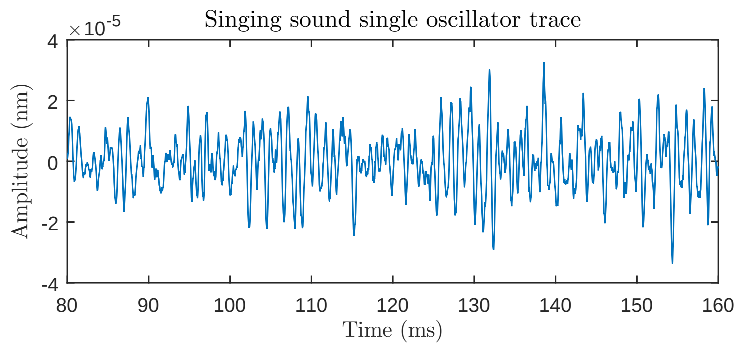

Figure 3.9: A trace from a single oscillator, the one marked by the red circle in Figure 3.8, corresponding to the sound in Figure 3.7

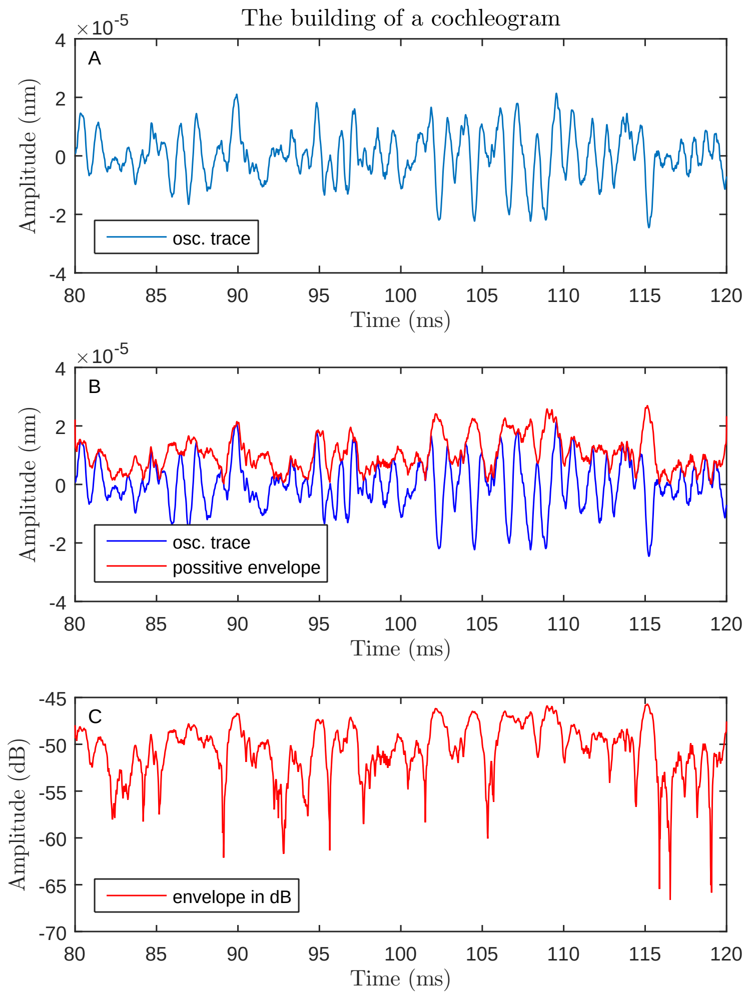

Figure 3.10: The method of building a cochleogram. (A) shows a slightly zoomed in version of Figure 3.9. (B) shows the same oscillator trace as seen in (A) with an included envelope. (C) shows the envelope in (B) but in dB scale. All 600 envelopes of the oscillator traces are combined to form the cochleogram seen in Figure 3.11

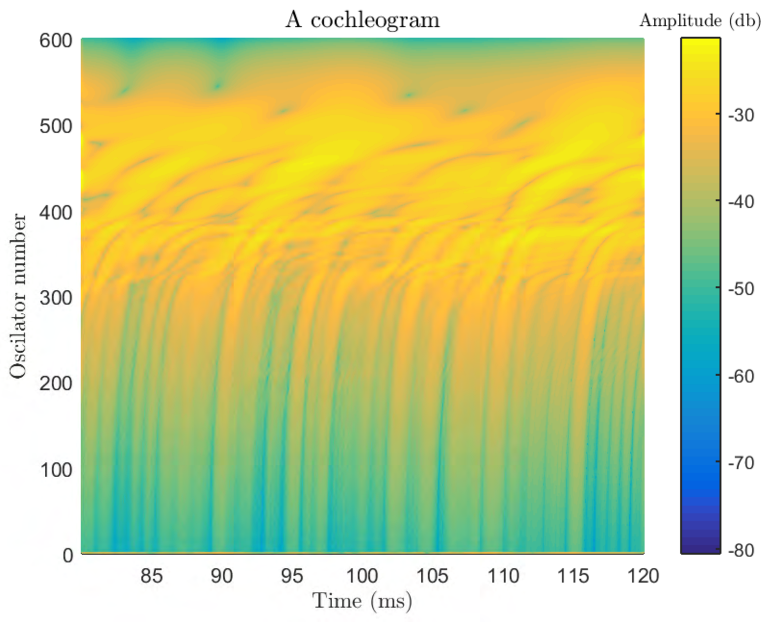

Figure 3.11: A full cochleogram

the envelope that indicates stimuli that affected a given oscillator. The envelope is then scaled to dB representation. The envelopes of all the oscillators are then plotted side by side and this gives a comprehensive picture of the stimuli and responses of the entire CM. An example of a cochleogram is seen in Figure 3.11.

4 Solution Method

This chapter will be a quick introduction and explanation of some digital signal processing (DCP) techniques in acoustics. The reader who is well versed on the topic could consider skipping to the next chapter.

For a lot of DCP work that relates to signals in time there are actually two different domains in which to work with such signals. The first domain is the time domain and the second is the frequency domain. Sound signals can be represented in either domain. A signal in the time domain can be transferred to the frequency domain via a Fourier Transform (FT) and a signal in the frequency domain can be transferred to the time domain via an inverse Fourier Transform (IFT). Because these transformations are very fast either domain can be used for signal processing.

An operation of two signals resulting in a third signal is referred to as a convolution when done in the time domain. The result of the cochlea model on the input signal is a convolution. Different convolution operations have a varying degree of complexity. From the description of the cochlea model in Chapter 3 one can imagine that the convolution operation of the cochlea model is a very complex one. When a convolution operation has taken place, the operation of negating the convolution is called a deconvolution.

4.1. Convolution and Deconvolution vs. Multiplication and Division

In many ways it is easier to edit sound in the frequency domain. Take for example a high frequency filter. If implemented in the time domain a moving average filter might be used. Such a filter iterates over all the sample points and averages each sample point in comparison to the neighboring sample points. As a result the high frequency sound waves would be smoothed or filtered out.

An alternate way to implement the same filter would be to transfer the sound signal to the frequency domain and then point-wise multiply the signal with an appropriate boolean vector such that the desired high frequency range was set to zero. The result of the multiplication can then be transferred back to the time domain where the targeted frequency will no longer be in the signal.

Note that of the two operations that were described above, the operation in the time domain falls into the category of a convolution whereas the operation in the frequency domain is a multiplication. Performing an inverse operation of the filter would be extremely difficult in the time domain as it is not so easy to find an inverse of a convolution operation.

However, the inverse of a multiplication operation is simply division, something that is easy to find, granted of course that one does not run into a divide by zero problem. This makes the frequency domain the domain of choice when an inverse operation is needed. In this project an inverse cochlea model is what is needed and therefore the domain of choice for much of the signal processing will be the frequency domain

4.2. The Cochlea Model as a Convolution

As mentioned earlier the result of the cochlea model can be viewed as a convolution operation. Let an input signal to the cochlea model be $a(t)$ where t denotes time and let the operation of the cochlea model be denoted by $b(t)$. For the rest of this report the symbol $*$ is used to denote a convolution operation. Furthermore, let the result of $a(t) * b(t)$ be $c(t)$. Then the operation of the cochlea model upon the input can be denoted by Equation 4.1.

$$c(t) = a(t) * b(t) \tag{4.1}$$

In short what is needed to simulate hearing loss is a method to deconvolve the signal $c(t)$, as in Equation 4.2.

$$a(t) = c(t) * b^{-1}(t) \tag{4.2}$$

The first step to simulating hearing loss, then, is to find a way to determine $b^{-1}(t)$. Although finding a deconvolution filter and doing a back transformation is a common DSP problem, one that involves a cochlea model is not without its complications (see Chapter 5).

For the linear case the operations described in Equation 4.1 can also be performed in the frequency domain by transferring $c(t)$, $a(t)$, and $b(t)$ to the frequency domain. Equation 4.1 then becomes Equation 4.3 where f denotes a particular frequency.

$$C(f) = A(f) \cdot B(f) \tag{4.3}$$

Here $B(f)$ can be referred to as the frequency response of the cochlea model. In essence $b(t)$ is the time domain response to an impulse. This is also called an impulse response. We use the frequency response of the cochlea model, which is just the impulse response $b(t)$ in the frequency domain, to an impulse to find $B^{-1}(f)$ as described in Equation 4.4, and consequently $b^{-1}(t)$ by IFT as described in Equation 4.5. Once the inverse filter has been built, then $a(t)$ can be found via deconvolution, Equation 4.6, given the output of the cochlea model c.

$$B^{-1}(f) = \frac{1}{B(f)} \tag{4.4}$$

$$b^{-1}(t) = IFT(B^{-1}(f)) \tag{4.5}$$

$$a(t) = c(t) * b^{-1}(t) \tag{4.6}$$

An impulse in the continuous time domain is also referred to as a Dirac delta function. This function is one with an infinitely narrow and tall spike at the origin and zero everywhere else and has an integral of 1 over the entire real line. The discrete delta function used here, common in DSP, is like a sound starting with value 1 at $t = 0$ and zero everywhere else.

4.3. Previous Work by incas^3

Some time ago incas^3 developed methods to find which parts of the cochleogram were most likely dominated by the sound of a single (target) source, and which were not. They created masks based on that information to be used in a speech recognition system. Because they wanted to have a way of telling how good these masks were, they tried to reconstruct an audio signal from a cochleogram multiplied by the mask. There was a problem in that they calculated the cochleogram and masks at a lower sampling frequency than the cochlea model was using.

Their work attempted to resolve this problem from two slightly different angles. In one approach movement of the basilar membrane was used to calculate an impulse response and from that an inverse filter. The other approach multiplied an envelope of the cochleogram with sin functions having a frequency that matched the center frequency of the cochleogram segments. The inverse filter method in this case has a longer delay and start-up time than does the cochleogram version.

They got to a certain point with the methods but eventually dropped the work in part because the resynthesized sound had click artifacts that made it unusable (see an example in Figure 6.6). This and other problems are solved in the following chapters.

4.4. Sound Resynthesis in the Frequency Domain

In the linear case the algorithm for sound reconstruction follows the pseudo-code given in Algorithm 1. The parameters that are required for this version of the algorithm are POS, $samples$, and $signalFile$. POS here is an array of the oscillators to be used for the reconstruction. If this is a single number then only the information from a single oscillator will be used for the reconstruction. $Samples$ indicates the length of the impulse response to be used for the reconstruction, and $impulse$ and $signal$ are the impulse and signal sound files respectively. The first thing that is done is to get the response of the model to the impulse Line 5. The next thing that is done is to also get the model output in response to the $signal$ sound file Line 6. In Lines 8 - 15 is the construction of the inverse filter or $H(f)$ for the frequency domain version. Firstly, as much of the impulse response as is needed is selected as indicated by $samples$. A Fourier transform transfers the response to the frequency domain. Then there is the need to bound the response in the frequency domain to diminish noise that would be introduced by numerical error in the part of the response that is almost zero. This is done in the if - $else$ statement on Lines 10 - 14. In the frequency domain, the rest of the algorithm is quite simple. The response of the cochlea model to the signal is transferred to the frequency domain in Line 16 and the result is made as a simple element-wise multiplication by the inverse filter in Line 17. This result is transformed back into sound by inverse Fourier transform in Line 18. Given that the response of the cochlea model is proper and a good choice for $samples$ and $bound$ was used then this is as simple as it should be in the Linear case where the computation is done in the frequency domain and the resynthesis is to be done with only one oscillator trace. To simulate hearing loss accurately more than one oscillator must contribute to the resynthesis, this is more complicated and is examined in Chapter 6.

Algorithm 1 Resynthesis in Frequency Domain

```
Require: POS, samples, impulse, signal
 1: POS                                    ▷ int or array of ints, the oscillator(s) to use
 2: samples                                ▷ the length of impulse response to use
 3: signalFile                             ▷ the signal sound file
 4:
 5: $impulse \leftarrow$ CM(impulseFile, POS)
 6: $amplitude \leftarrow$ CM(signalFile, POS)
 7:
 8: $h \leftarrow$ impulse(:,1:samples)                           ▷ MatLab notation
 9: $f \leftarrow$ fft(h, length($amplitude$))                     ▷ zero padded fft
10: if $f_i \geq bound$ then
11:     $f_i \leftarrow f_i$
12: else
13:     $f_i \leftarrow bound$                       ▷ Cut off $f$ at certain dB below peak
14: end if
15: $fi \leftarrow \frac{1}{f}$
16: $Amplitude \leftarrow$ fft($amplitude$)
17: $Res \leftarrow Amplitude\ .\cdot\ fi$                         ▷ MatLab notation
18: $res \leftarrow$ real( ifft($Res$) )
19: return res
```

4.5. Sound Resynthesis in the Time Domain

The algorithm where sound reconstruction is done in the time domain starts very similarly to the previous algorithm. In fact it is exactly the same until Line (9). At this point instead of transferring the cochlea model response to the sound "$amplitude$" to the frequency domain for sound reconstruction the inverted f, fi, is transferred back to the time domain to become the inverse filter called hi in the algorithm (See Algorithm 2). This, in mathematical speak, is simply $b^{-1}(t)$ as used in Section 4.2. This inverse filter must then be smoothed by a window, for example a Gaussian or Hanning window, in Line (18). The result is then calculated in a standard finite impulse response (FIR) deconvolution filter operation in Line (19). The final step is to normalize the amplitude of the reconstructed sound to have a peak of ± 1, which is done in Line (20).

Both of the algorithms presented here provide the overarching concept or approach to sound reconstruction. Both, however, are inadequate for actual hearing loss simulation because they do not provide a way to do sound reconstruction from multiple oscillators. In the following chapters sound reconstruction using multiple oscillators will be considered further.

Algorithm 2 Resynthesis by Deconvolution

```
Require: POS, samples, impulse, signal
 1: POS                                   ▷ int or array of ints, the oscillator(s) to use
 2: samples                               ▷ the length of impulse response to use
 3: impulseFile                           ▷ the impulse sound file
 4: signalFile                            ▷ the signal sound file
 5:
 6: $impulse \leftarrow$ CM(impulseFile, POS)
 7: $amplitude \leftarrow$ CM(signalFile, POS)
 8:
 9: $h \leftarrow$ impulse(:,1:samples)   ▷ MatLab notation
10: $f \leftarrow$ fft(h)
11: if $f_i \geq bound$ then
12:     $f_i \leftarrow f_i$
13: else
14:     $f_i \leftarrow bound$            ▷ Cut off $f$ at certain dB below peak
15: end if
16: $fi \leftarrow \frac{1}{f}$
17: $hi \leftarrow$ real( ifft($fi$) )
18: $hiw \leftarrow hi$ .· $window$       ▷ MatLab notation, window can be Gauss, Hann, etc
19: $res \leftarrow amplitude * hiw$
20: $res \leftarrow res$ ./ max($|res|$)  ▷ just normalizing the result
21: return res
```

5

Requirements for Simulating Hearing Loss

In order for the back transformation of the model output into sound to work, the right parameters for building the inverse filters must be chosen. As is often also the case in research, this project faced its fair share of obstacles which needed to be overcome. Some of these turned out to not be of much consequence. Others if left unsolved would make simulating hearing loss impossible.

5.1. The Choice of Window and Length of Impulse Response

As mentioned in Section 4.3, incas[3] has attempted sound resynthesis before. Their code provided the starting point for this project. The code followed Algorithm 2 from Section 4.5. One of the first problems that became evident when using this code was that the quality of sound reconstruction was very sensitive to the type of window (see an example of a Gaussian window in Figure 5.4) used to smooth the inverse filter. The operation in question is in Line 18 of Algorithm 2. The sound quality was even more sensitive to the window breadth. The breadth of the window also corresponds to the length of the impulse response.

In Figure 5.1 is a very accurate sound reconstruction using a window breadth of 1,776 samples. The same reconstruction is done again and shown in Figure 5.2 using only 10 samples less. As is evident, the sound is very different. This shows how sensitive the original code was to the length of the impulse response, or the breadth of the window.

Much better results were acquired when using a Gaussian window than when using a Hanning window. However, the type of window being used should have almost no affect on the quality of the sound reconstruction. Fortunately, this problem, as well as the problem of sensitivity to the breadth of the window, was easily solved, and was related to the default ordering of MatLab's inverse Fourier transform function.

In the inverse Fourier transform (see: Line 17 of Algorithm 2) the default ordering of MatLab's function puts the important data of the inverse filter at the endpoints of the array as seen in Figure 5.3. The very next operation in Algorithm 2 smooths the endpoints of the inverse filter. This is done with a point wise multiplication by a window, for example a Gaussian window as seen in Figure 5.4. Because MatLab put the

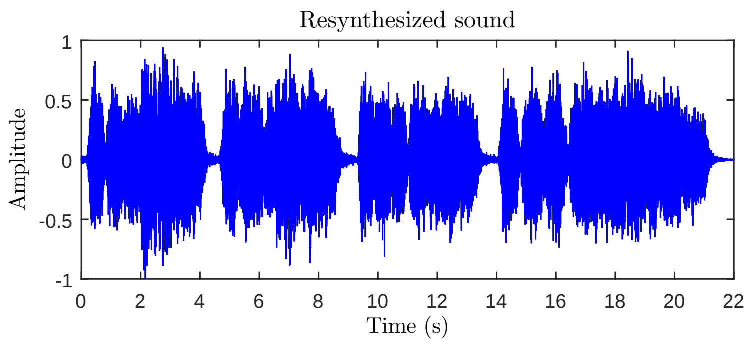

Figure 5.1: Sound resynthesis using the 3rd oscillator and a window breadth of 1776 samples

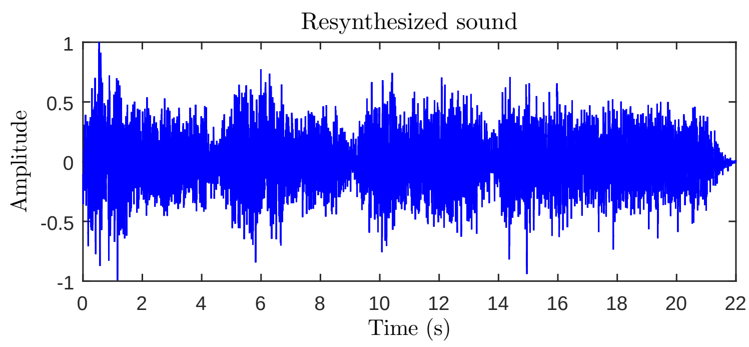

Figure 5.2: Sound resynthesis using the 3rd oscillator and a window breadth of 1766 samples

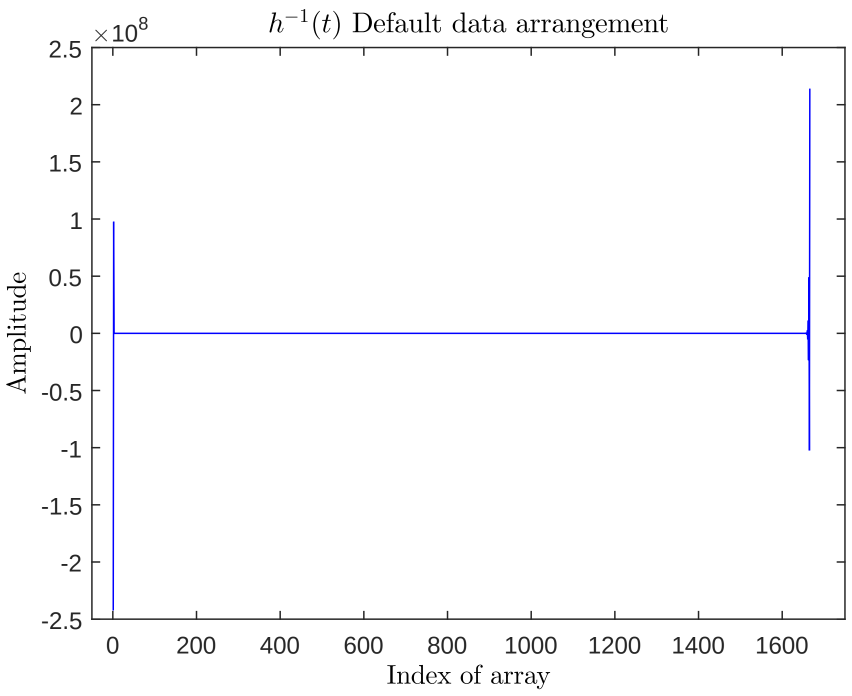

Figure 5.3: MatLab's ifft default data arrangement

important data at the endpoints of the array, the very part that the window smooths, the inverse filter function was greatly damaged. By shifting the arrangement of the inverse filter to the order seen in Figure 5.5, where the endpoints are shifted to the middle, this problem is fixed. This reordering needs to be carried out anyway, and it is crucially important to do the reordering prior to the application of the smoothing window.

With this alteration, the type of smoothing window no longer has a significant affect on the quality of the results. The length of the oscillator trace is also not nearly as sensitive. Mainly, as long as the oscillator comes to rest then it will suffice.

A part of the research question is to determine the affects that the choice of smoothing window and the breadth of the window has on the quality of sound reconstruction. The answer to this part of the research question is that when the window is properly applied, the choice of the type of window does not affect the results significantly. With that, the breadth of the window is not significant as long as it is as wide as the inverse filter it is supposed to smooth.

5.1.1. Optimal Length of Impulse Response

Another of the research questions is, what is a good length of impulse response. Because the damping term in the cochlea model decreases at an exponential rate, it is expected that the needed length of the impulse

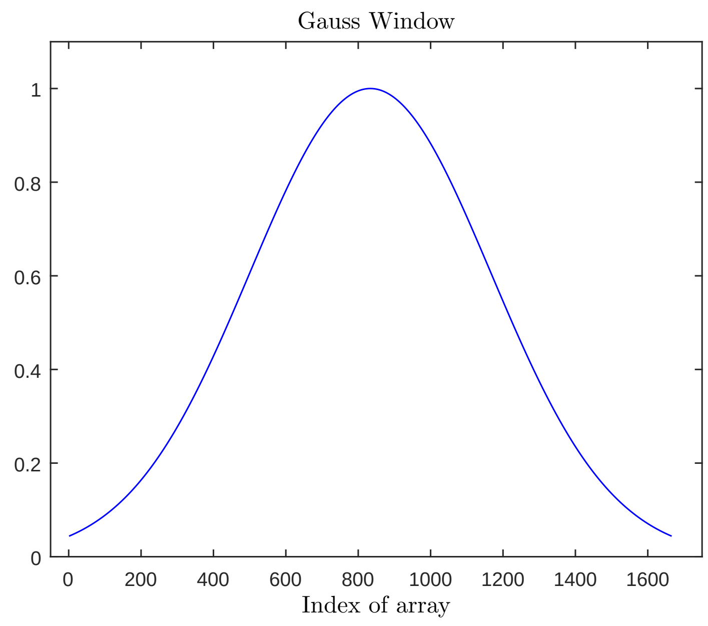

Figure 5.4: An example of a smoothing window as used in Algorithm 2

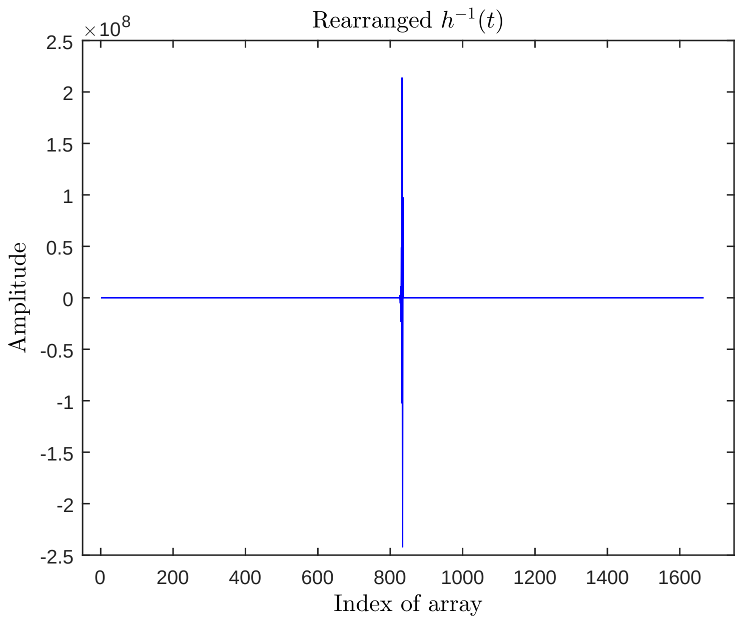

Figure 5.5: A properly rearranged inverse transfer function

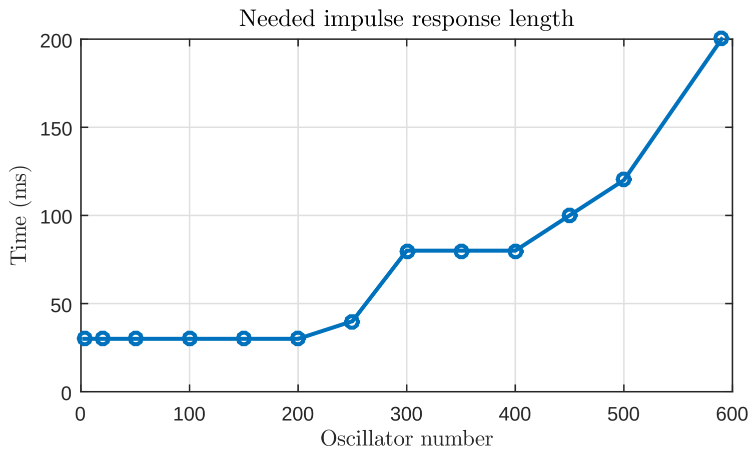

Figure 5.6: Good lengths of impulse responses in relation to oscillator number

response would increase inversely proportionally to the damping term for the length of the cochlea. For a finite impulse response filter to work properly, a clean impulse response is needed, one where the oscillations completely die down. An example of a clean impulse response is seen in Figure 5.10.

Due to energy reflection at the helicotrema, a topic that will be given more attention in Section 5.4, there is some difficulty in getting clean impulse responses. In the sound reconstruction this is evident in some low frequency error. An example of a moderate amount of this error is seen in Figure 6.23 at approximately 0.022 kHz (= 22 Hz). It appears that for certain ranges of lengths of impulse responses this low frequency error is disproportionately high.

To get an approximation of good impulse response lengths relative to an oscillator, tests were conducted with a cochlea model running at 500 kHz. The number of samples or the length of the impulse response was then increased and the quality of sound reconstruction observed. From the perspective of the frequency content of the sound, there should be little error in the frequencies above 100 Hz. And with respect to the time representation of the sound, the profile of the sound snippet should be similar to that in the original sound. The profile of the sound was visually compared to the profile of the original, with both plotted side by side. When these requirements were met, then the number of samples was increased until the low frequency error was minimized. The results are summarized in the Figure 5.6.

Low frequency error is well minimized with impulse responses shorter than 4 milliseconds. But low frequencies are not well represented when using such a short impulse response. That disqualifies such a short impulse response. Especially if the sound being reconstructed has noticeable content in this frequency range. For the range of impulse response lengths between 4 milliseconds and 30 milliseconds low frequency error is very high. Because of this, oscillators 2 to 200 need an impulse response of 30 milliseconds to keep this error lower.

For oscillators beyond 200 the curve represented by good choices of sample numbers appears exponential, as is the expectation. Oscillators 300 and 350 need a slightly longer impulse response than the curve suggests. This is again due to there being a range between 50 milliseconds and 80 milliseconds where low frequency error is too dominant.

The length of impulse response that is needed is important if this technique of hearing loss simulation should ever be used to do real time simulations. Real time simulations would be more feasible using the approach of Algorithm 2. For this method the inverse filtering method needs a start-up time for the inverse filter to become accurate, at least as long as the impulse response is. Hence, a minimum amount of delay in simulation is the length of the impulse response.

5.2. Scaling of the Model and N

The model of the cochlea was found to be unstable for $N = 600$ oscillators. The threshold where the model became unstable was $N \approx 428$. The first oscillator in the model represents the middle ear, or the stapes. The stapes size which is supposed to be $\frac{1}{N}$ was hard coded to be $\frac{1}{400}$. As long as $N < 428$ was used, the model was stable, albeit inaccurate for values of N not equal to 400. In future work this detail should be checked in the

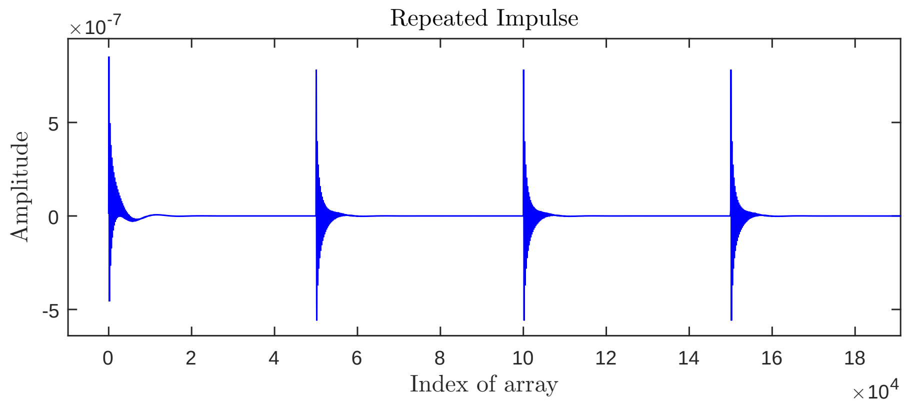

Figure 5.7: A faulty repeated impulse response

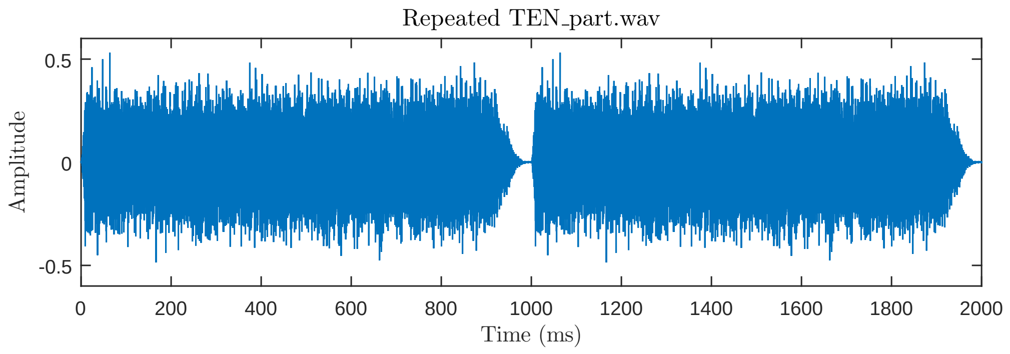

Figure 5.8: A repeated sound signal

model.

5.3. Using a Repeated Signal

Some problems in this project did not have an easy solution. These include some errors in the original adaptation of the cochlea model to accept general input, as well as a problem with energy reflection, discussed in Section 5.4. The cochlea model that was supplied for this project was not able to accept arbitrary sound files as input. Because during the initial update of the model, to accept general input, some errors were made, it seemed like a repeated input signal affected the model output. This was not the case, but it raised the question if there were other effects of using a repeated signal.

5.3.1. Using a Repeated Signal

In an effort to simulate hearing loss, every day sounds should be valid as input to the model. However, the initial model supplied for this project operated on a pure tone signal that was generated repeatedly, twice at every time step of the simulation, once for the the full step of the four step Runge-Kutta (RK4) method and then again with a half step time shift, as required for the RK4 method used by the model. Changes were made to the model itself to accept an arbitrary sound input, with the requirement that the input has a sample rate twice that of the model. In this way, the additional sample points are used as the half time step values for the RK4 method.

Due to errors in updating the model initial outputs contained inaccurate waves. However, when input to the model was repeated these effects tended to be reduced as is evident in Figure 5.7. After correcting the flaws in the model updates and benchmarking the model, these effects were greatly reduced. However, there remains a difference when using a repeated signal versus using a single signal.

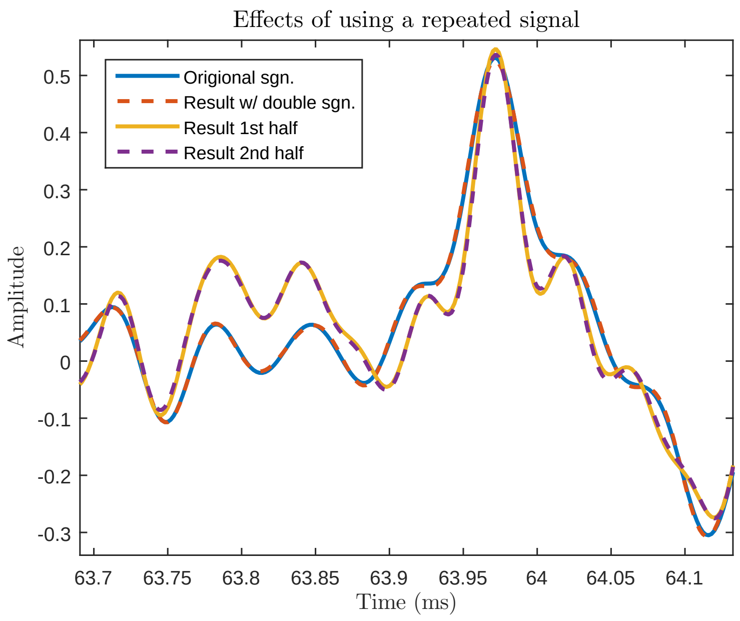

Figure 5.9: A comparison between four different sounds. The original sound is plotted in blue. Reconstructed sounds using the first half and last half of a repeated signal are plotted in yellow and purple respectively. Sound reconstructed using a repeated cochlea output is shown in red

5.3.2. Actual Effects of a Repeated Signal

Using a double signal, like the one seen in Figure 5.8, to generate the model output and then using this to reconstruct sound, shows that there is a difference in the reconstructed sound and that the accuracy is improved when using a repeated signal.

When comparing the repeated parts of the model output, almost no difference is found between the two. However, when using a repeated output to do sound reconstruction versus using only a single output shows that there are noticeable differences. Let a be the original signal from which the cochlea output is generated. In Figure 5.9 a is plotted in blue. Now let r_1 be the reconstructed sound using the first half of the cochlea output. It is plotted in yellow in Figure 5.9. Let r_2 be the sound reconstructed using the second half of the cochlea output, plotted in a purple dashed line. Lastly, let r_D be the sound that is reconstructed using the repeated cochlea model output, plotted in a red dashed line. There is not a noticeable difference between r_1 and r_2. However, there is a very noticeable difference between these two and r_D. The question then is, which sound is more accurate? To test this, the original sound a is superimposed onto the other results. Zooming in on the three results shows that r_D is strikingly close to a as is evident in Figure 5.9. Both r_1 and r_2 are very similar to each other, but very different from a.

Therefore, as far as the cochlea model output is concerned there is not a significant difference when using a repeated signal. But in sound reconstruction there is a difference when a repeated output is used. Furthermore a repeated output increases the accuracy of the sound resynthesis.

5.4. Energy Reflection

One of the biggest problems encountered in the project was a problem of energy being reflected at the helicotrema (the end of the cochlea opposite of the base or opposite from where the sound energy enters via the oval window). This had a few consequences, some of which were not immediately apparent, and will be discussed here.

5.4.1. The Problem

The first, and rather obvious problem of energy being reflected at the helicotrema, is related to the generation of a clean impulse response. For the creation of an inverse filter a finite impulse response is needed. Furthermore, because contributions from all oscillators is necessary to simulate hearing loss, creating inverse filters from all the oscillators must be possible.

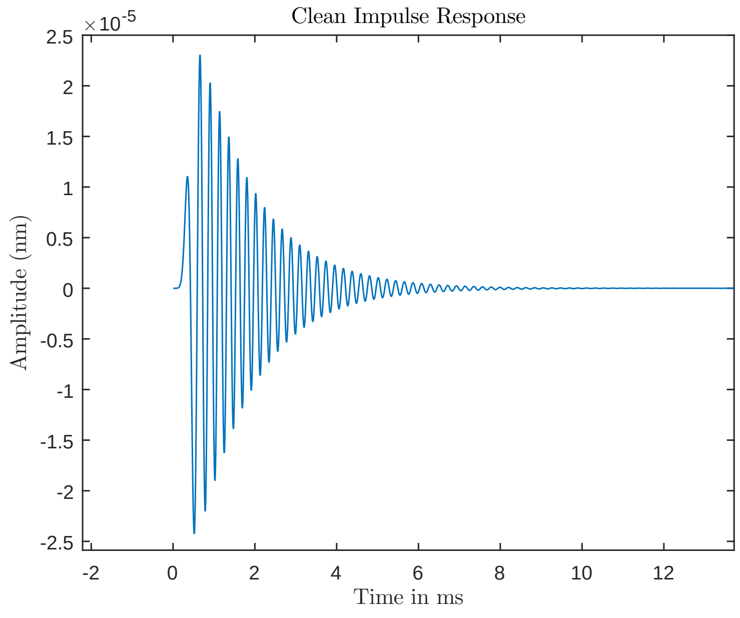

Figure 5.10: A clean finite impulse response

For the later oscillators, impulse responses look like the one seen in Figure 5.20 due to energy being reflected. What is seen in this figure is that the oscillator movement dies down well, but at a certain point large low frequency oscillations take over. The original oscillations do eventually die out, but the low frequency oscillations become increasingly distinct. For the earlier oscillators in the model a clean response is possible, at least until the energy bouncing back from the helicotrema arrived at the given oscillator. If an oscillator comes to rest before the reflected energy begins to affect it, then making a good inverse filter is possible by limiting the length of the impulse response. The length of the impulse response must be long enough to allow the oscillator to come to rest but not so long as to include reflected energy.

Although using the right length of impulse response worked for the high frequency oscillators it does not work for the low frequency oscillators. The reason for this is that these oscillators do not come to rest before being affected by the reflected energy.

In addition to the problem of not being able to create good inverse filters, the energy reflected at the helicotrema also affects the model response to the input sound. Because not all the energy in the model is damped out, a continued input of energy from a longer input source eventually causes the output from the model to be blown out of proportion. Such output closely resembles output from an unstable model.

This unstable looking behavior is seen, for example, in Figure 5.11. Additionally, when the integration time step is reduced then it seems as though the output becomes more stable, as seen in Figure 5.12. Using a timestep that is too big, is a frequent problem in numerical simulations that causes a model to become unstable. Repeatedly decreasing the time step in the cochlear model, eventually rendered output that looked fine. However, the results were still not useable because energy was still reflected. The reason that the output looked good is that a smaller time step forced the input to be shorter because of the increased demand on computer memory. This shorter time horizon minimized the effect of the build-up of residual energy in the model.

What is now abundantly clear is that there was a problem with the cochlea model. Without a fix for energy reflection the hope of using the cochlea output to simulate hearing loss would not be realized.

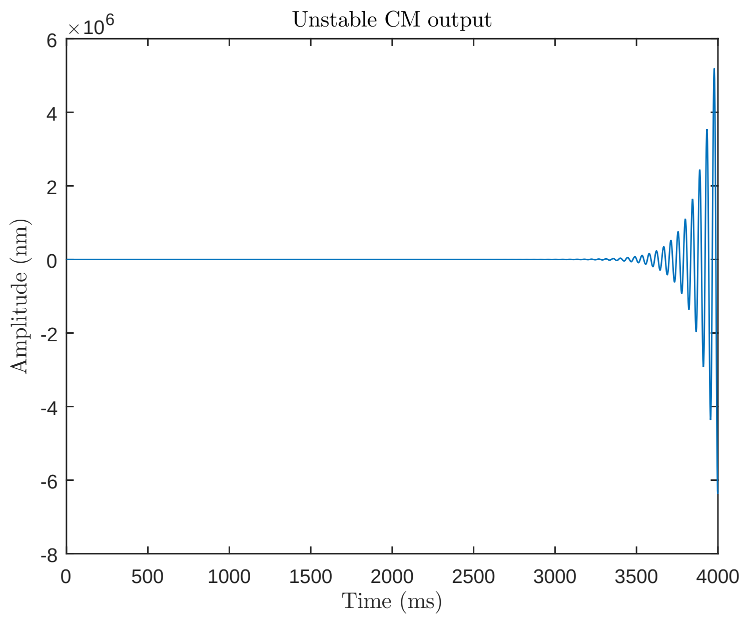

Figure 5.11: Cochlea model output that is not stable due to energy build-up in the model

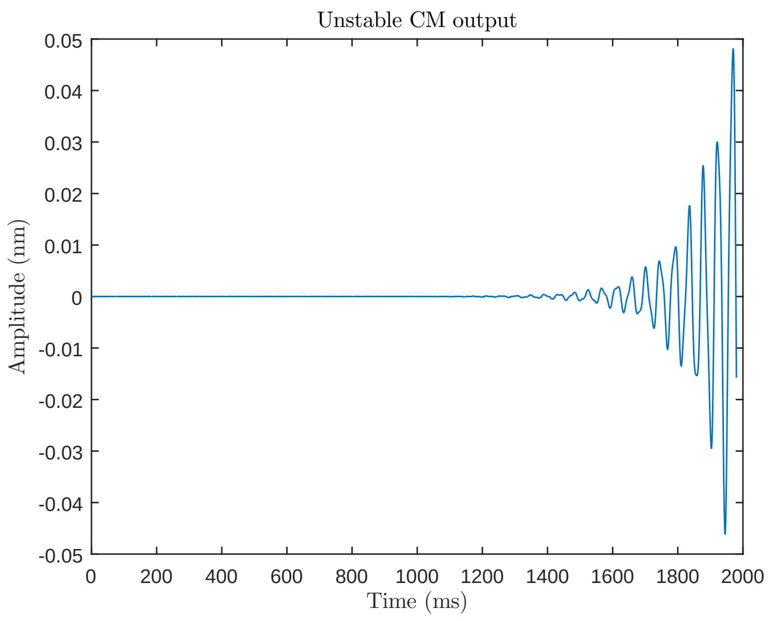

Figure 5.12: CM output that is not stable but with smaller error than in Figure 5.11 due to either smaller time step or shorter time horizon

5.4.2. The Solution

Before moving on to the solution to the energy reflection problem here are a few key terms that need to be introduced. In the linear model there is a constant mass term m and position dependent stiffness and damping terms $s(x)$ (see: Equation 3.8) and $d(x)$ (see: Equation 3.9) respectively. The profiles of these terms over the length of the cochlea are as seen in Figures 5.13 and 5.14. Here $0 \leq x \leq 35$ is the position on the cochlear partition. Because in the model the length of the cochlear partition is scaled to $0 \leq \tilde{x} \leq 1$, $\tilde{x}$ is used to denote this scaled position.

A term used to describe the relation between these two terms, d and s, is the Quality Factor $Q(\tilde{x})$ or just Q defined as follows:

$$Q(\tilde{x}) = \frac{\sqrt{m_s s(\tilde{x})}}{d(\tilde{x})}, \quad 0 \leq \tilde{x} \leq 1 \tag{5.1}$$

Using the definition for the quality factor Q it is easy to see that Q will be constant at $\frac{1}{\varepsilon}$.

Energy reflection has been a problem before in the use of the transmission line cochlea model. To counter this problem Hendrikus Duifhuis introduced a modification to the damping term [16]. The modification that he made was, that a factor was introduced to the d term such that the quality factor instead of being constant went from 20 to 0.5. Lets call this factor f and it is defined as seen in Equation (5.2).

$$f(\tilde{x}) = 1 + 39e^{-\alpha(1-\tilde{x})} \tag{5.2}$$

The parameter α in f determines over how much of the damping term the update will be noticeable. Setting $\alpha = 25$ makes the change such that it significantly affects approximately the last $1/4$ of the oscillators.

The above definitions for the damping and the stiffness are not exactly as those in the cochlea model. Rather, the stiffness and damping as defined in the code is as follows.

$$d(\tilde{x}) = \varepsilon \hat{t}_0 \sqrt{\frac{s_0}{m_s}} e^{-0.5\tilde{\lambda}\tilde{x}} \quad 0 \leq \tilde{x} \leq 1 \tag{5.3}$$

With the variables defined as follows:

$$\varepsilon = 0.05 \tag{5.4}$$
$$\hat{t}_0 = \frac{1}{1000} \tag{5.5}$$
$$s_0 = 1 \times 10^{10} \tag{5.6}$$
$$m_s = 0.5 \tag{5.7}$$
$$\lambda = 10.5 \tag{5.8}$$

With this line of code:

```
d = epsilon*t_0_hat*sqrt(s_0/m_s)*exp(-0.5*lambda_tilde.*x_tilde(2:end));
```

And the stiffness is defined in the code as follows:

$$s(\tilde{x}) = \frac{\hat{t}_0^2}{m_s} s_0 e^{-\tilde{\lambda}\tilde{x}} \quad 0 \leq \tilde{x} \leq 1 \tag{5.9}$$

With this line of code:

```
s = t_0_hat^2 / m_s * s_0 .*exp( - lambda_tilde.*x_tilde(2:end) );
```

With the added parameters of m_s and $\hat{t}_0$ the “unchanged” quality factor in the model is constant at 14.1421. And the stiffness and damping profiles can be seen in Figures (5.15) and (5.16). The updated damping profile according to the recommendation by Duifhuis is as seen in Figure (5.18).

The updated Q when applying the same factor (Equation 5.2) goes from 14.1421 to 0.3536 as seen in Figure (5.17)

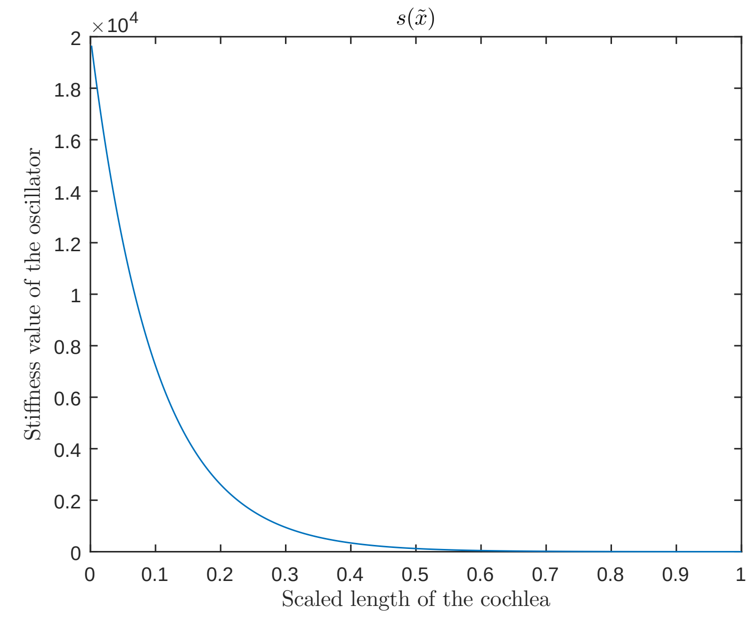

Figure 5.13: The profile of the stiffness term $s(\tilde{x})$

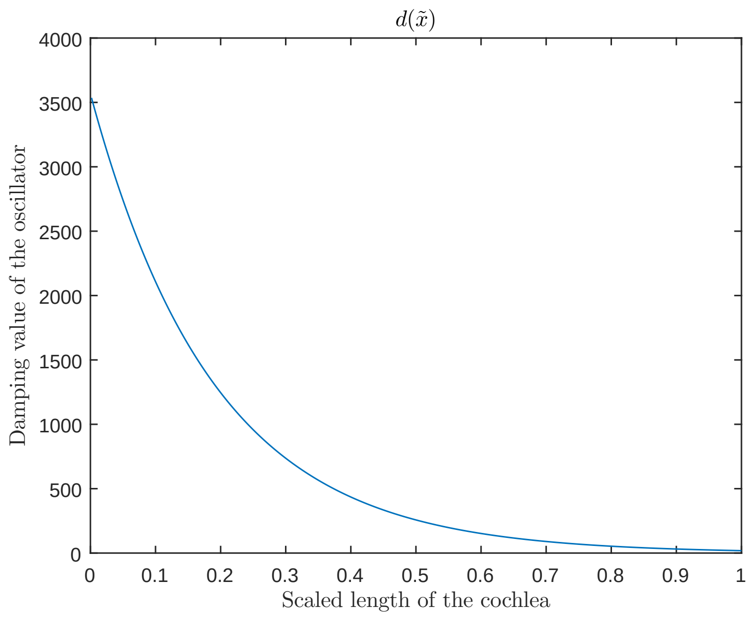

Figure 5.14: The profile of the damping term $d(\tilde{x})$

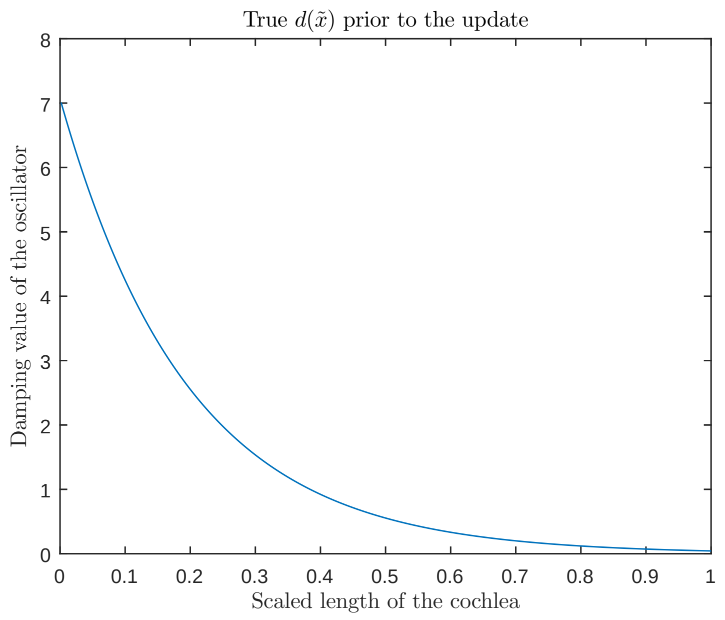

Figure 5.15: The damping profile prior to the update, according to the actual code

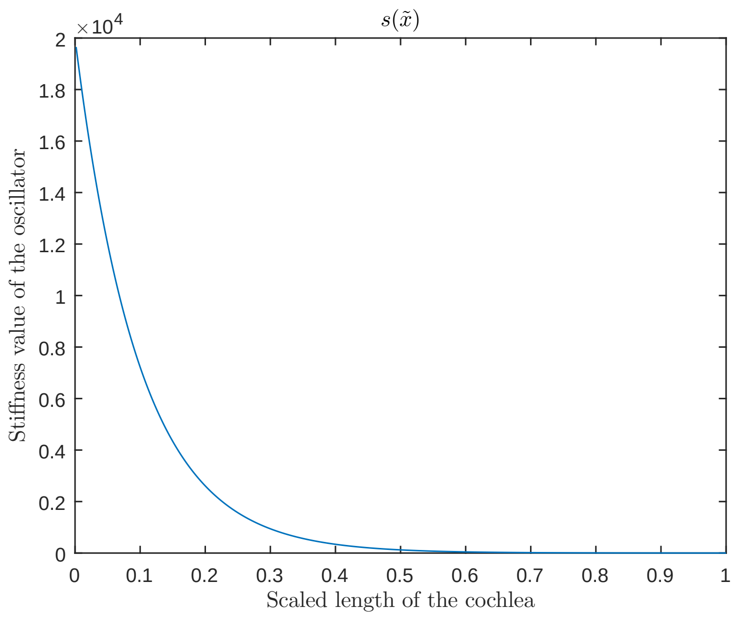

Figure 5.16: The stiffness profile according to the actual code

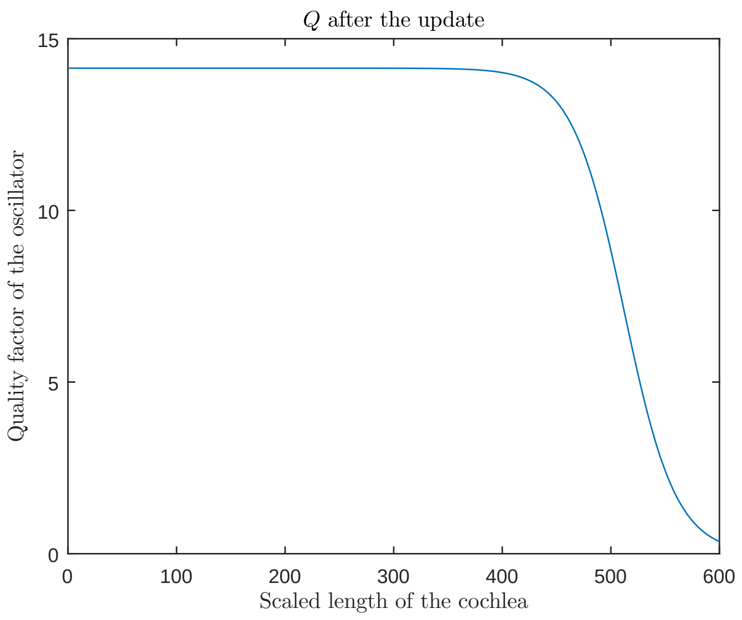

Figure 5.17: The quality factor after the update according to the actual code

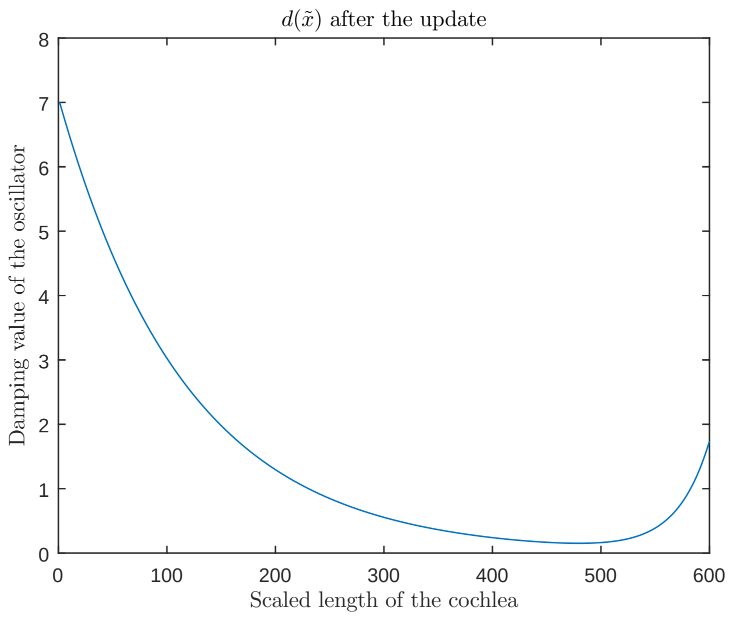

Figure 5.18: The damping profile according to the recommended modification

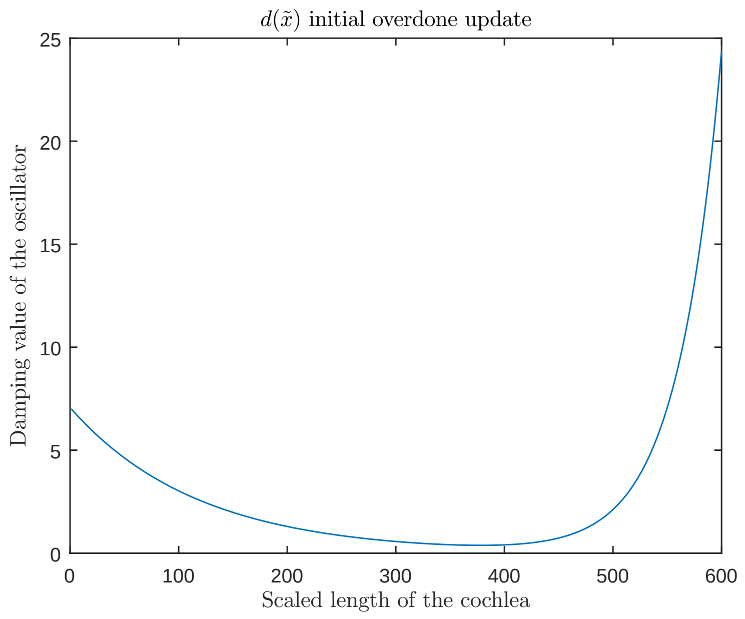

Figure 5.19: The damping profile with the initial exaggerated update

However, this update was not sufficient to produce stable results. In Figure 5.20 there is an oscillator trace of the model without any modification. As is clear in this figure the impulse response is not finite. With the update that was used by Duifhuis, the impulse response becomes better but is still not finite. The result from this modification can be seen in Figure 5.21. This implies that there needs to be significantly more damping to really “kill” enough energy before it reaches the helicotrema. An even greater modification was made to the damping term, that is, the factor f was made to be as defined in Equation 5.10. Such an exaggerated modification allowed to have good results that are very promising for sound reconstruction. The damping profile using this f is as seen in Figure 5.19. And the results on an impulse response is as seen in Figure 5.22. There still remain small oscillations on the impulse response, however, sound reconstruction is possible with an acceptable level of error.

$$f(\tilde{x}) = 1 + 560e^{-\alpha(1-\tilde{x})} \tag{5.10}$$

5.4.3. The Best Solution

The best solution to the problem of energy being reflected at the helicotrema would be to extend the model to include ±100 extra oscillators on which the damping would be modified to reduce energy reflection. Doing it in this way would keep the simulation of the cochlea relatively unaffected by the damping modification.

Adding an artificial damping layer is an approach also used in other applications, like seismic survey simulations. Such a damping layer is often referred to as a sponge layer, or sometimes, as a sponge boundary condition. Optimal sponge layers depend on the application, and remains a topic of research.

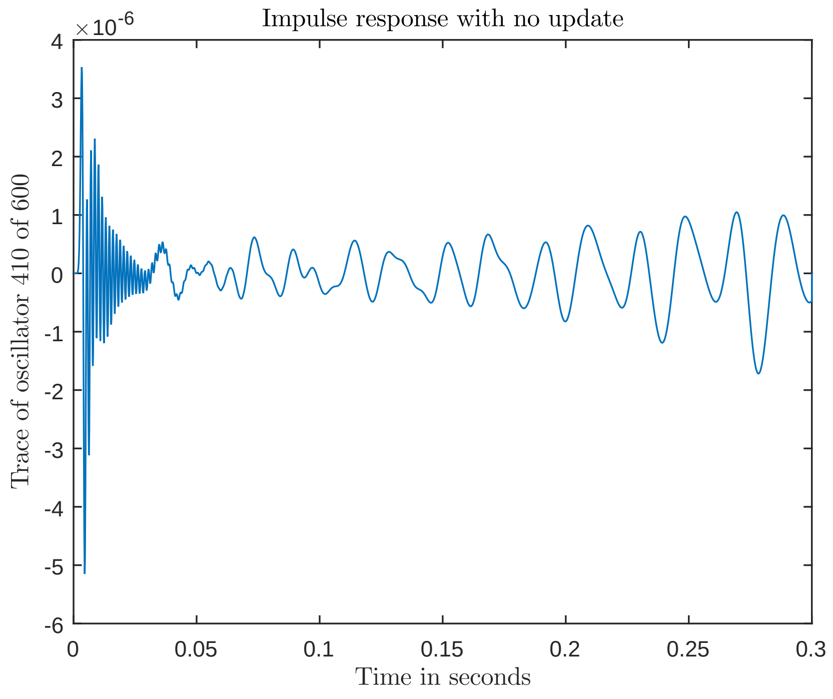

Figure 5.20: An impulse response without an update, oscillator 410 of 600

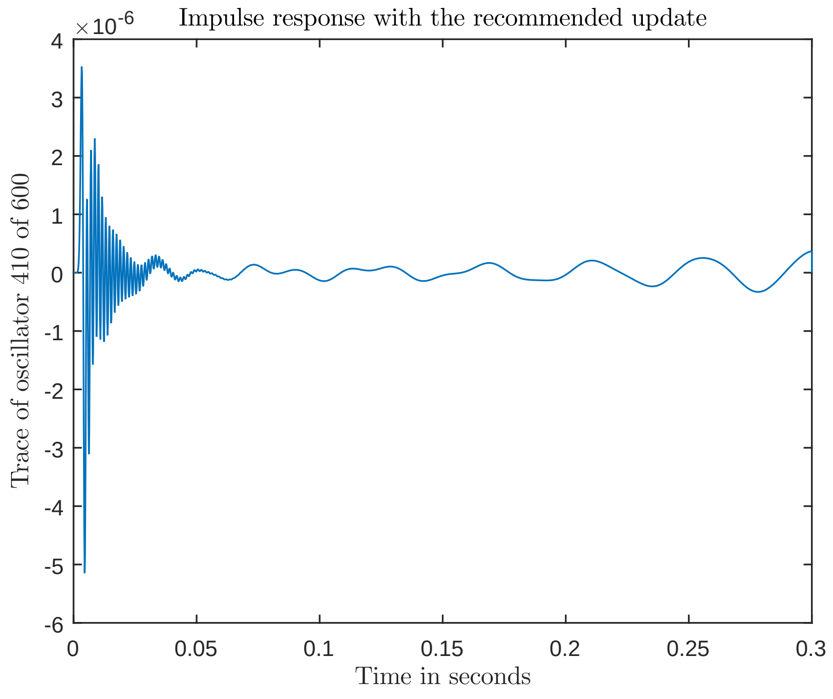

Figure 5.21: An impulse response with the recommended update, oscillator 410 of 600

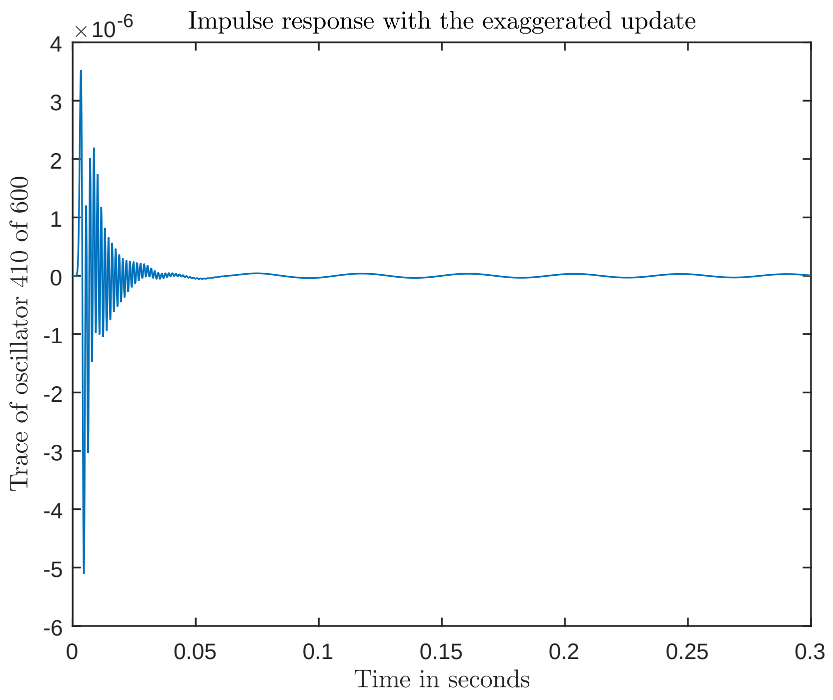

Figure 5.22: An impulse response with the exaggerated update, oscillator 410 of 600

6 Results and Simulations

With the corrections and modifications to the cochlear model outlined in Section 5.4 the focus shifts to the problem of reconstructing the input signal a from a set of oscillator traces C. For this we want to construct a viable inverse model that utilizes a broad range of oscillators for the reconstruction. Eventually the developed method will be used to simulate hearing loss.

In Section 3.1 the output of the cochlea model is described. Given an input sound a consisting of l sample points, then the output from the cochlea model is a set of N oscillator traces C of length l. An inverse model, then, is one that can generate the sound a given the set of oscillator traces C. In practice it is very difficult to resynthesize the input sound a exactly. Therefore $\tilde{a}$ shall be used to refer to the resynthesized version of a.

When using the solution method described in Section 4 the sound $\tilde{a}$ can be generated using a single trace c where c is a single oscillator trace from the set of traces C. But as described later, this will not suffice to simulate hearing loss. Thus, there is the need for $\tilde{a}$ to be reconstructed from a broad range of oscillators.

6.1. Effects of Choice of Oscillator

One of the main sound files that will be used is a 1 second snippet from the wave file "TEN_Part", or just "TEN", a sound file with broad spectrum noise. This sound is sometimes referred to as "threshold equalizing noise". It was supplied by incas[3], originally from Moore et al [12]. The time representation of this sound file can be seen in in Figure 6.1, and the frequency domain representation can be seen in Figure 6.2.

The frequency content plots of sound that follow are such that the the relevant frequencies are plotted on the x-axis in logarithmic scale. This is common practice in acoustical signal processing. Frequency content plots are often also plotted in dB scale. For sound content plots the dB scale is not used in this report. However, tuning curves and inverse filter functions are represented in dB scale. Some of the figures show the frequency content of sound with an inverse filter super-imposed in the plot. In such a case the sound content is not in dB scale, but the inverse filter is.

As explained in Section 4 the way that the inverse finite impulse response (inverse FIR) filters are found

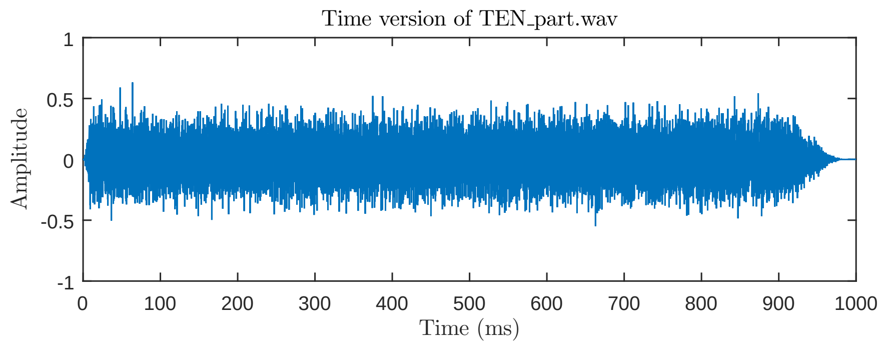

Figure 6.1: Time representation of the sound file "TEN_Part.wav"

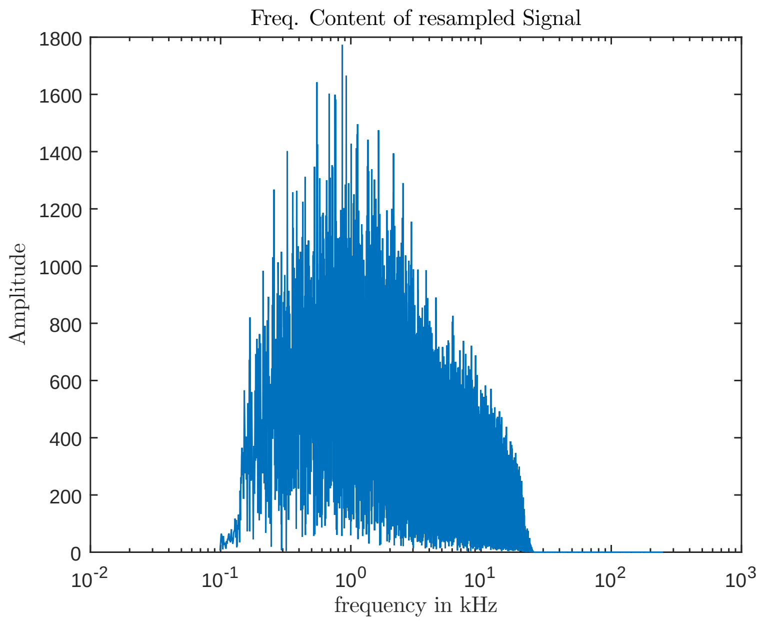

Figure 6.2: Frequency domain representation of the sound file "TEN_Part.wav"

is by the response of an oscillator from the model to an impulse sound. This is called an "impulse response" and is like the ones shown in Section 5.4.2, of which the impulse response in Figure 5.10 is a good example. Frequency domain representations of the impulse responses are as seen in Figure 6.3. These curves are called the tuning curves of a given oscillator, because they will have their maximum frequency content close to the oscillator's characteristic frequency. Notice that after this maximum point the tuning curve drops off, indicating that there is very little response to frequencies higher than the characteristic frequency. These tuning curves refer to the model as $B(f)$ in Section 4. Inverting the tuning curve gives the inverse filter $B^{-1}(f)$.

A given inverse filter can be used to resynthesize the sound a when multiplied with the corresponding oscillator trace c. Using a high frequency oscillator gives a resulting sound with frequency content profile almost identical to the original seen in Figure 6.2. However, when a lower frequency oscillator is used then a few things should be noticeable.

In Figure 6.5 is the frequency content of the reconstructed sound $\tilde{a}$ using a mid-frequency oscillator. The first change to notice is that the high frequency content of the original sound is not represented. This is evident by the flat line starting at 7 kHz. The reason that $\tilde{a}$ contains so little content beyond 7 kHz is due to the high frequency energy being dissipated on higher frequency oscillators earlier in the cochlea.

The second noticeable thing, is that there is a large amount of error introduced where the tuning curve drops off, at approximately 5 kHz. In the inverse filter that drop off point corresponds to a large amplification factor which is the peak seen on the inverse filter in Figure 6.5. There is also a corresponding peak in the frequency spectrum of $\tilde{a}$ that is so great that it shoots off the plot. This inaccuracy peak is brought in by the amplification of numerical error by the inverse filter. The peak of the error here is not seen in the graph, but goes to a value of $+19,000$.

The amplification of numerical error is what gives rise to clicks in the resynthesized sound. The time domain click of the error seen in Figure 6.5 is as seen in Figure 6.6 at $t \approx 9800$. Because the human hear is very sensitive to clicks such as these, this is a problem that must be solved for a hearing loss simulator.

A solution to the problem of error amplification is to choose a maximum amplification value and all frequencies where the inverse filter is greater than this maximum value is set to it. Two inverse filters are in Figure 5.11, where one is shown without a bound and the other is bounded at 50 dB above minimum amplification value. This method certainly works very well when using a single oscillator. Experiments using this method were performed but results are not presented in this report.

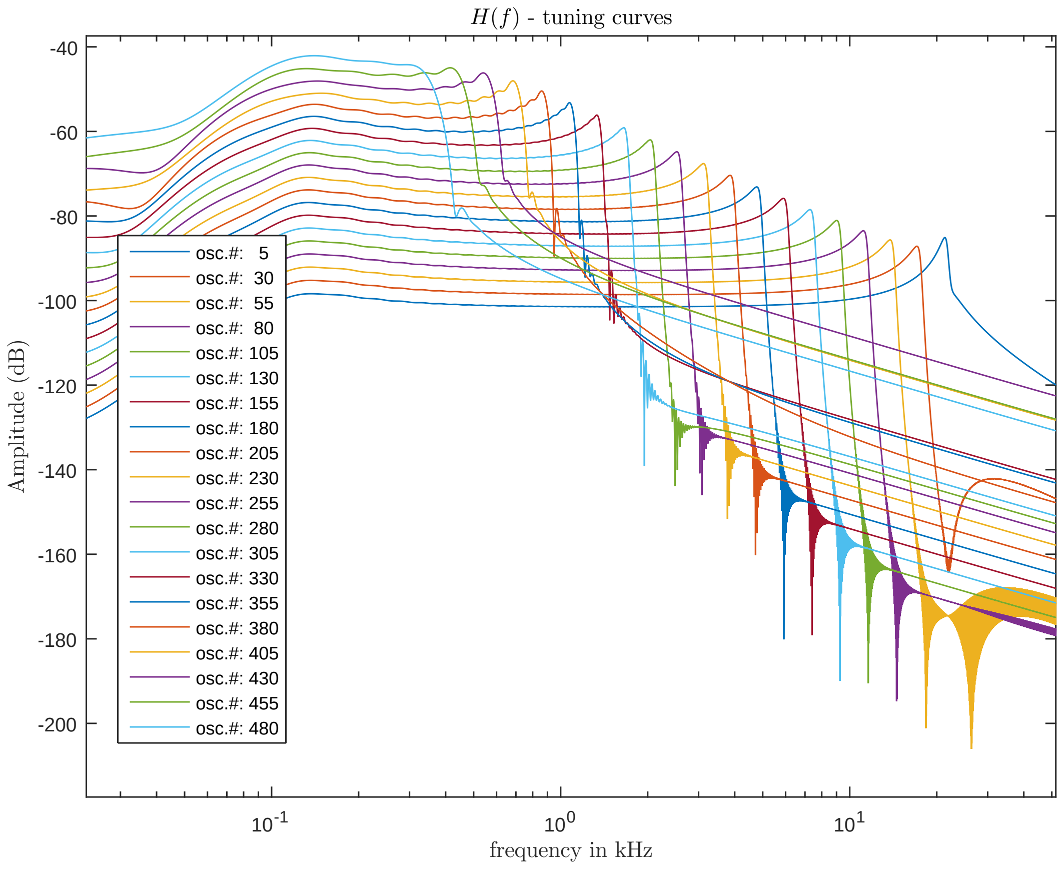

Figure 6.3: These are the tuning curves of the different oscillators, the equivalent of $H(f)$

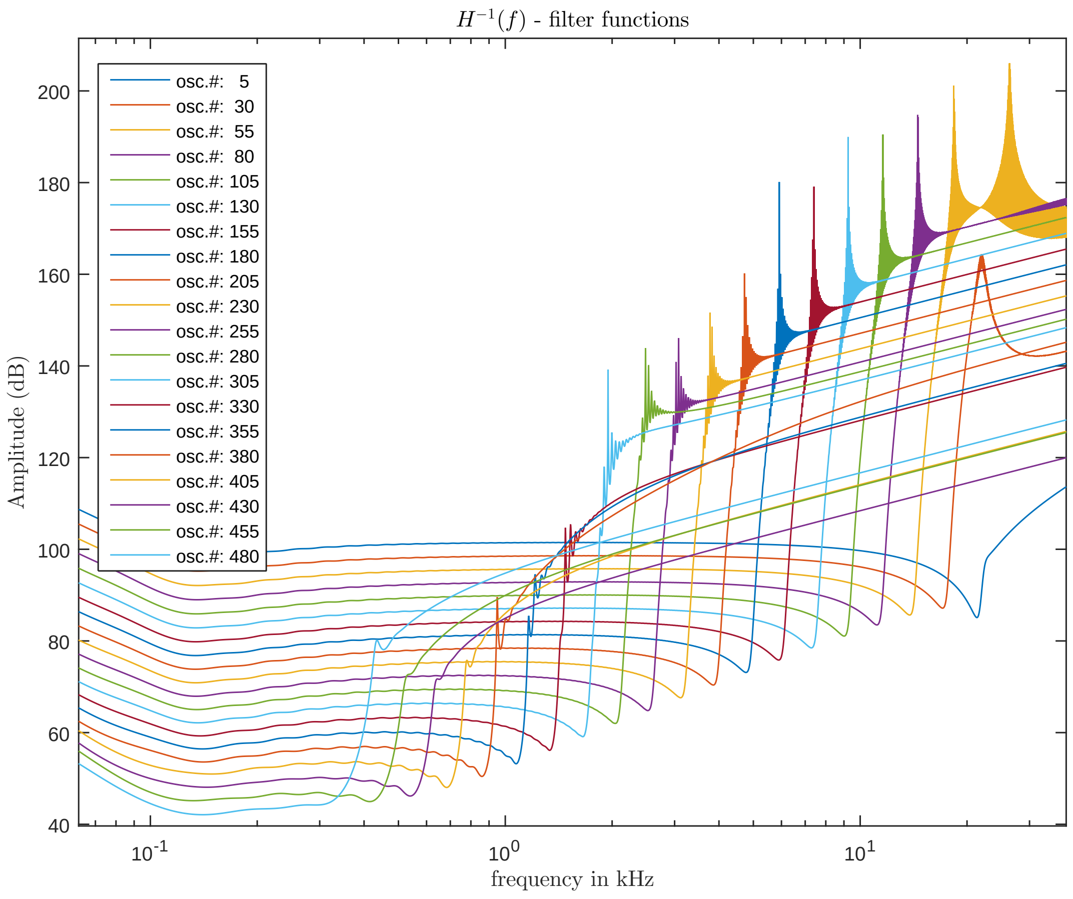

Figure 6.4: Inverting the tuning curves makes them the inverse filters $H^{-1}(f)$

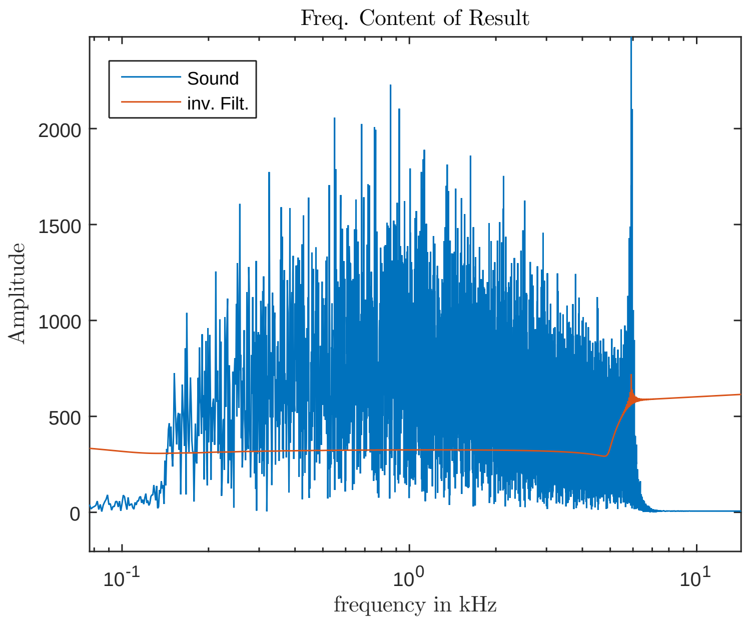

Figure 6.5: Frequency content of the result, using oscillator 180. Filter represented in (dB), not according to scale. Y-axis of sound content is not in (dB) scale

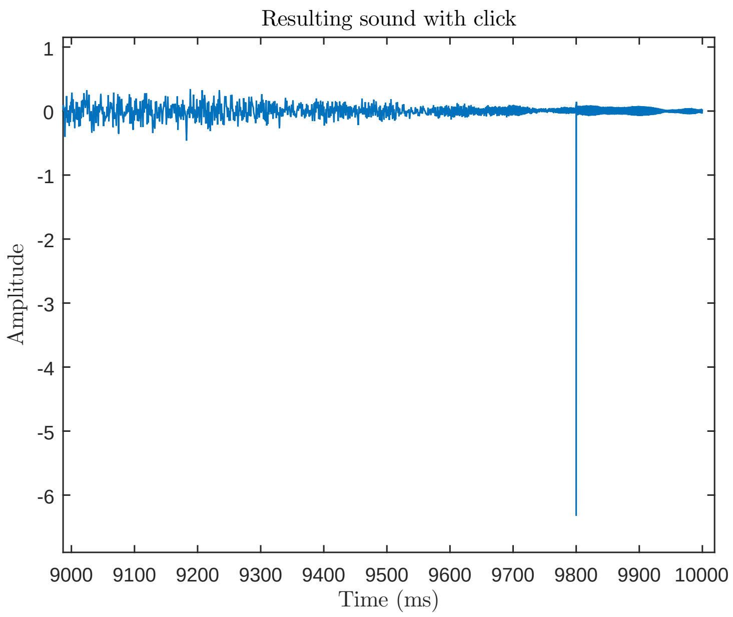

Figure 6.6: Sound with an error click seen in the time domain

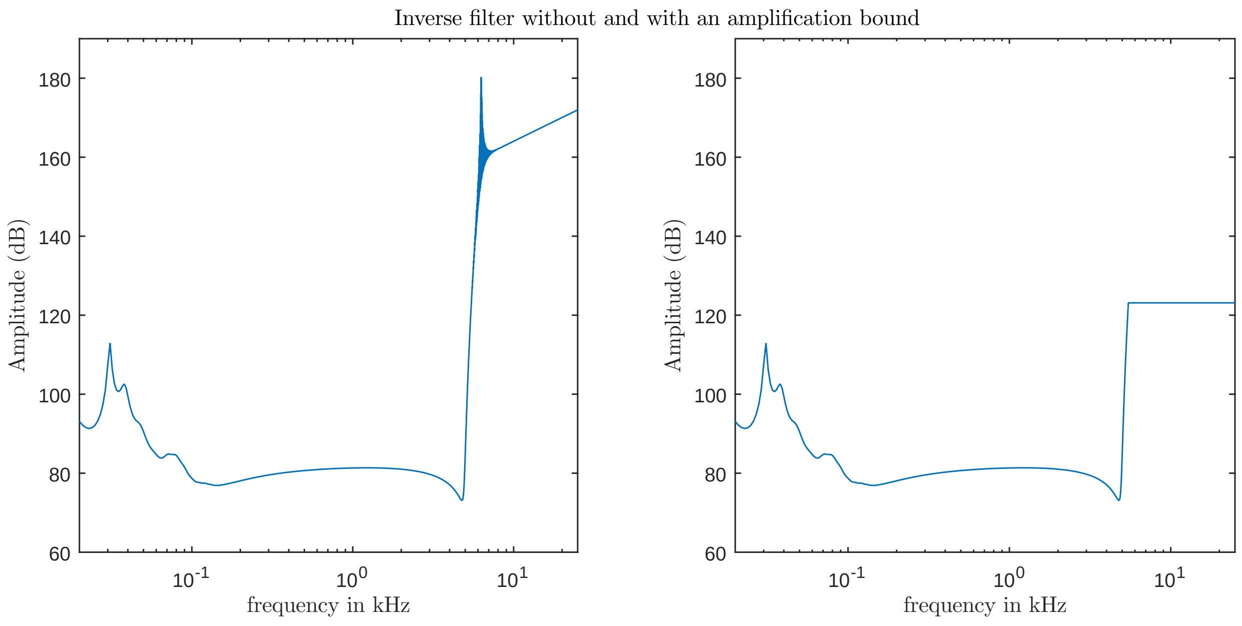

Figure 6.7: An inverse filter with no amplification bound (L), and an inverse filter with an amplification bound of 50 dB above the minimum point (R)

6.2. Combining the Resynthesized Sound

In the previous section it is indicated what are some of the effects of the choice of oscillator in sound resynthesis. It is pointed out that some oscillators will not represent high frequency sound content, and that there is a danger for error amplification. There was also a solution to lower the error when using a single oscillator.

However, in order to simulate hearing loss there is a need to combine the results from multiple oscillators. This is necessary because hearing loss often affects certain areas of the cochlea more than others. Assume that the middle of the cochlea is affected. Then, if a single or even multiple oscillators are chosen not from this area then the hearing damage will not be simulated. Because of this, it is vital that a way is found to combine contributions from multiple oscillators for sound reconstruction. Furthermore, it is logical that contributions from a particular oscillator would represent frequencies close to its characteristic frequency. Out of necessity, it should not contribute to the frequency range where it has a high error amplification value.

Given a set of N oscillators that are to contribute to the sound reconstructions, an algorithm that works well is as follows. First, determine the frequency f at which the tuning curve of the N^{th} oscillator drops off. Because the N^{th} oscillator corresponds to the lower frequencies it should be used for all sound with frequencies lower than f. Then, declare a $\tilde{f} = f$. Next, reset f to the frequency where the tuning curve for oscillator $N-1$ drops off. Oscillator $N-1$ is then used for all frequencies in the range of $(\tilde{f}, f]$. This process continues, except for oscillator 2 (given that oscillator 2 is in the set of oscillators that contributes to sound reconstruction) which is used for all the frequencies higher than $\tilde{f}$. The reason that oscillator 2 is used as the last oscillator instead of oscillator 1, is because the first oscillator in the model corresponds to the middle ear, not to the cochlea.

A good way to determine the contribution boundaries f and $\tilde{f}$ is to use the minimum value of the inverse filter as the boundary (see: an example in Figure 6.8). In this way, each oscillator contributes not exactly to the frequencies around its characteristic frequency, rather it contributes to frequencies lower than the characteristic frequency. This, however, should not be a big problem as long as the set of contributing oscillators does not have large gaps. Furthermore, as long as the goal is to just resynthesize the original sound, large gaps should not pose a problem. But when a hearing loss simulation is attempted then the regions where there is hearing damage should have a sufficient number of representing oscillators. To save computing time and to minimize memory usage it it beneficial to use less oscillators, but care must be taken to have sufficient representation. Using every third oscillator gave good results for the simulations that follow.

It is possible to also use a region around the minimum of the inverse filter as the contributing region for an oscillator. But using this kind of region becomes tricky because the drop-off slope is much steeper for low

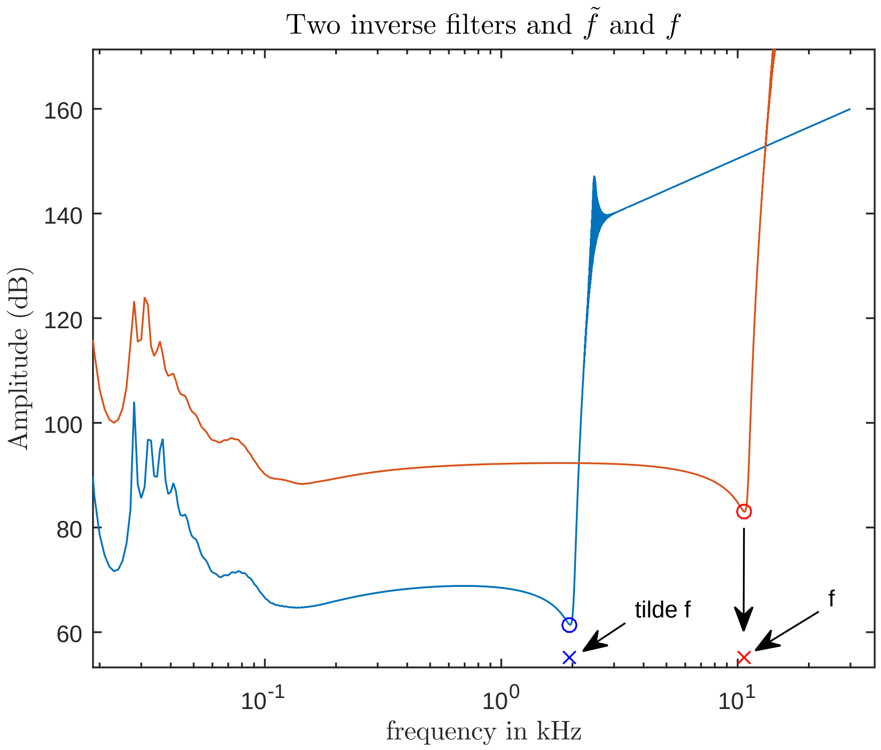

Figure 6.8: Two example inverse filter functions with the value for $\tilde{f}$ as the index point denoted by the blue x and f as the index point denoted by the red x

frequency oscillators than it is for high frequency oscillators. It quickly becomes complicated to compute such a region and ensure that for every oscillator the error amplification area is avoided. Good results were obtained by first finding the frequency of the minimum value of the inverse filter. And then finding the frequency where the slope is steepest beyond the minimum point and then using a weighted average between these two frequencies as the boundary f. Using the average between these frequencies works. Skewing it to be closer to the minimum of inverse filter seemed more accurate. But all of these trials did not work as well as just using the minimum value of the inverse filter for f as described in the previous paragraph.

An alternate way to find the boundaries f and $\tilde{f}$ would be to use the parts of a filter where it has the smallest value of all the filters. This method of selecting the boundaries would be very similar to using the minimum point, but would not skew contributions to the lower frequencies as much.

Due to the way the highest frequencies are reconstructed it is paramount that one of the first oscillators is included in the list of contributing oscillators. The recommendation is to always include the second oscillator. If none of the first oscillators is included then any high frequency content of the input sound a that is higher than the drop off value of the first used oscillator cannot be represented in $\tilde{a}$.

By using the boundaries f and $\tilde{f}$ in the way described above the need for a maximum amplification value is no longer relevant. This is because a given oscillator simply does not contribute to the frequency range where it amplifies error. In Figure 6.9 is the frequency content of a resynthesized version of “TEN_Part.wav” along with the filter functions of the oscillators that contributed. In this figure the frequency content of $\tilde{a}$ is shown not in dB. The inverse filter functions are shown in dB, but not according to scale.

Close inspection of Figure 6.9 indicates that the inverse filter functions are shown in groups of four. In this particular simulation the contributions from the inverse filters within each group were averaged with the hope that it would make for increased accuracy. There was, however, not a noticeable accuracy gain which is why averaging results was not continued.

With these added updates to the reconstruction method, a possible algorithm is as seen in Algorithm 3. This algorithm starts very similarly to Algorithms 1 and 2 from Chapter 4. In Line 12 this algorithm starts to change. This line is simply the definition of the length of the input signal. Line 13 is needed for the scaling of the inverse filters in Line 21. In Line 19 the inverse filters are set to zero, except in the frequency ranges where

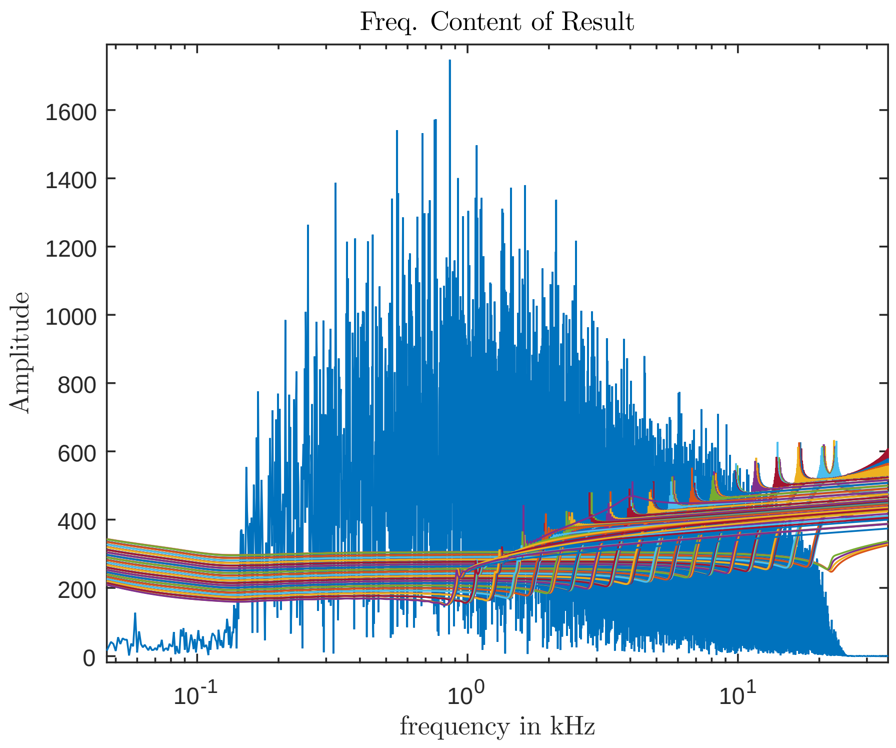

Figure 6.9: Sound reconstructed with contributions from a set of oscillators

they contribute to sound reconstruction. The other thing that changes, is the finding of the frequency ranges that each filter does contribute to. For this $\tilde{f}$ and f are used. They are updated in Lines 16, 18, and 24.

6.3. Filtering for Combining the Result

In Section 6.2 it was mentioned that a specific oscillator was to contribute to the reconstructed sound $\tilde{a}$ for frequencies in a range $(\tilde{f}, f]$. This means that the frequencies outside of this range must be filtered out by some means. The simplest filter to implement in MatLab is to just send the time signal through a standard MatLab band pass or band reject filter function with the needed parameters of what frequencies to filter.

Although easy to implement, there are a few reasons why this is not a good idea. First, the simplest of FIR filters will introduce a time delay into the signal. Especially when multiple segments are combined, the summing of multiple delayed signals renders the result unusable.

As an example, the sound $\tilde{a}$ represented in Figure 6.10 has a frequency spectrum that very closely matches that of the original signal a. But in the time representation it is evidently not similar to the original, as seen when comparing $\tilde{a}$ in Figure 6.10 with a in Figure 6.1. In Figure 6.10 at the start of the sound clip there are 10 milliseconds (until $t = 10$) of complete quiet. In contrast a has a sharp start of sound such that at $t = 10$ it is at full volume. After the 10 milliseconds of quiet in $\tilde{a}$ (Figure 6.10) there is a gradual increase in sound volume that is visible as a taper until about $t = 250$. Also, where a has a clear decrease in sound content starting at $t \approx 875$, there is only a slight and cluttered taper in $\tilde{a}$. By this, it is clear that not only should the frequency content of $\tilde{a}$ be similar to that of a, but the time domain sound must also be similar.

A second reason why it can be complicated to filter out the unwanted frequencies in the time domain is that a very high order filter is needed to minimize either gaps or double representation in the frequencies close to the boundaries f and $\tilde{f}$, where content is taken from different oscillators. An example of this can be seen in Figure 6.11. In this figure, the blue in the background is the frequency content of the original signal and the orange is the frequency content of the reconstructed signal. In the forefront are the filter functions that contributed to the the resynthesis. The frequency content gaps, seen here at frequencies 3.25, 3.8, 4.6, 5.4, 6.4, 7.6 and 9 kHz, correspond to the boundaries where on one side of the gap the reconstruction was from one oscillator and on the other side of the gap from another oscillator.

Finally, the typical approach to solving the delay problem is to do a backward-forward filtering to offset

Algorithm 3 Resynthesis in Frequency Domain using Multiple Oscillators

```
Require: POS, samples, impulse, signal
 1: POS                                        ▷ int or array of ints, the oscillator(s) to use
 2: samples                                    ▷ the length of impulse response to use
 3: signalFile                                 ▷ the signal sound file
 4:
 5: $impulse \leftarrow$ CM(impulseFile, POS)
 6: $amplitude \leftarrow$ CM(signalFile, POS)
 7:
 8: $h \leftarrow$ impulse(:,1:samples)                        ▷ MatLab notation
 9: $filter \leftarrow$ fft(h, length($amplitude$))           ▷ zero padded fft
10: $invFilter \leftarrow \frac{1}{filter}$
11: $Amplitude \leftarrow$ fft($amplitude$)
12: $length \leftarrow lengthOf(\ signal\ )$
13: $signalInFreqDomain \leftarrow fft(signalFile)$
14: $res \leftarrow zeros$
15:
16: $\tilde{f} \leftarrow \min(\ invFilter_{nSeg,j}\ )\quad 1 \le j \le length/2$
17: for $ind = nSeg - 1 : 1$ do
18:     $f \leftarrow \min(\ invFilter_{ind,j}\ )\quad 1 \le j \le length/2$
19:     $invFilter_{ind,j} \leftarrow 0\quad j < \tilde{f}\ or\, j \ge f$     ▷ set inverse filter to 0 except for where it contributes
20:     $Res \leftarrow Amplitude\ .\cdot\ invFilter$
21:     $invFilter \leftarrow invFilter \cdot \frac{signalInFreqDomain_{(\tilde{f}+f)/2}}{Res_{(\tilde{f}+f)/2}}$     ▷ scaling the filter
22:     $Res \leftarrow Amplitude\ .\cdot\ invFilter$
23:     $res \leftarrow res + ifft(Res)$
24:     $\tilde{f} \leftarrow f$
25: end for
26: return res
```

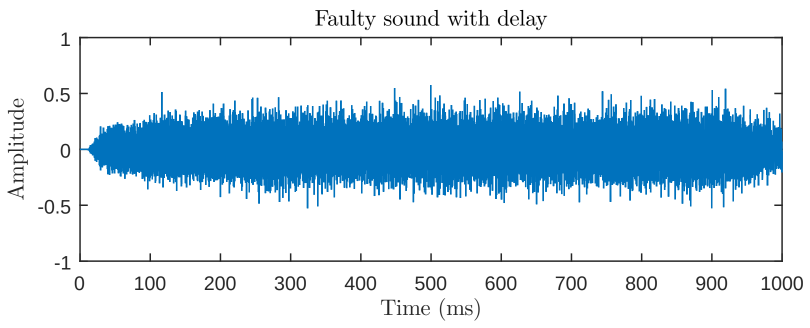

Figure 6.10: Reconstructed sound with a FIR filter induced delay

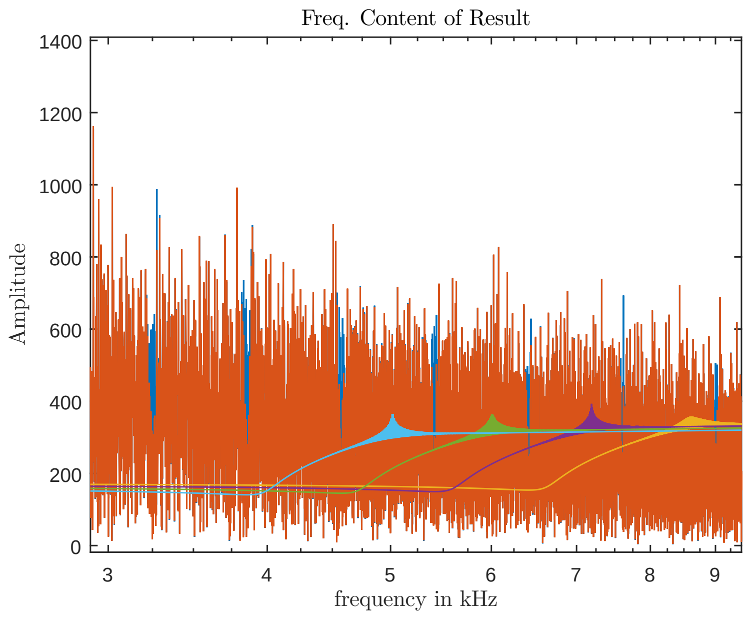

Figure 6.11: Sound with some frequency gaps

the delay. That is, the sound is filtered once backward. This will introduce a negative delay. The sound is then filtered forward and the forward delay will cancel the negative delay from the backward filtering. Because backward-forward filters are computationally expensive this is not an optimal solution.

For these reasons it is not practical to filter the resynthesized sound repeatedly in order to combine contributions from specific oscillators. A different approach must be used.

6.4. Filtering in the Frequency Domain

One way to circumvent the above filtering complications is to do all the computations strictly in the frequency domain. This is perfectly viable as long as the entire signal is available at the time of computation.

The filtering that was attempted in the previous section in the time domain is easy in the frequency domain. That is, the index from the minimum point of the previous contributing inverse filter $\tilde{f}$ up until the index of the minimum point of the current inverse filter f are the frequencies that the respective inverse filter contributes to.

Perhaps the most intuitive way to combine the results is to set the unwanted frequencies of the result to zero in the frequency domain prior to the transfer to the time domain. The results from all the oscillators can then be summed in the time domain to form the final result $\tilde{a}$.

The smarter approach, however, is to set the values of the inverse filter to zero for the unwanted frequencies outside of the range $(\tilde{f}, f]$. This leads directly to a minimization of needed multiplication operations and assignment operations.

6.5. Scaling the Result

Another important consideration is the scaling of the resulting sound. In a typical time domain wave-form the amplitude should fall in the range $-1 \leq a(t) \leq 1$. The scaling of the result must be taken into consideration, because when the result does not fall into the stated range then some audio converters will clip audio. That is, values $\tilde{a}(t) < -1$ are set to -1 and values $1 < \tilde{a}(t)$ are set to 1. This is how distortion, also called “clipping”, is introduced.

At the very least, the result must be scaled such that it falls into the $-1 \leq \tilde{a}(t) \leq 1$ range. This can be done using the operation of the following equation:

$$\tilde{a}(t) = \frac{\tilde{a}(t)}{\max\limits_{0 \leq t_i \leq t_f} (|\tilde{a}(t_i)|)} \tag{6.1}$$

Where $\tilde{a}$ is the reconstructed sound and t_f is the final time point in $\tilde{a}$

This is also how incas[3] scaled their result (see: Section 4.3). The reason that scaling in this fashion is inadequate is that to determine the accuracy of $\tilde{a}$, comparisons of the frequency spectrum of the signal a and the frequency spectrum of the result $\tilde{a}$ is very helpful. With a poorly scaled $\tilde{a}$ this comparison is challenging because the frequency spectrums, given that $\tilde{a}$ is accurate, will be scalar multiples of each other.

A better way to do the scaling is to scale the value $\max(|\tilde{a}|)$ to $\max(|a|)$, as in Equation 6.2. As long as the time domain result is very close to that of the signal this method works well.

$$\tilde{a}(t) = \frac{\tilde{a}(t)}{\max\limits_{0\le t_i\le t_f}(|\tilde{a}(t_i)|)} \max_{0\le t_i\le t_f}(|a(t_i)|) \tag{6.2}$$

However, the time representation of the result $\tilde{a}$ is very susceptible to small changes that might not be audible to the best of listeners. As soon as there is a small error in the resynthesis, scaling according to Equation 6.2 will result in the frequency spectrum of the result $\tilde{a}$ being a scalar multiple of the spectrum of the signal a.

The best way to scale the result $\tilde{a}$ is to use the values of the frequency spectrum. For example, to scale the contribution of a given oscillator, the maximum value of the frequency spectrum, in the part of the spectrum to which the oscillator contributes, is picked. Then, using the ratio of this value from the spectrum of a with the corresponding value from the spectrum of $\tilde{a}$ to scale $\tilde{a}$ gives the ratio by which $\tilde{a}$ should be amplified. To improve robustness, an average of a few sample points from the frequency spectrums of a and $\tilde{a}$ can be used to derive the amplification ratio.

Because the scaling is linear between the time and frequency domain representations of the same sound, and because the basic operation of the inverse filter when working in the frequency domain is multiplication, the scaling can be applied directly to the inverse filter function. This has a desired effect in two ways. First it reduces the computation of resynthesis after the filter functions have been built. And secondly, it generalizes the inverse filter function, eliminating the need to compare the amplitude of $\tilde{a}$ with that of a.

It is important to scale the filter function instead of the result. The reason is that the goal is to not just be able to invert the operation of the cochlea model on a sound input. If this were the goal, then there is no problem with scaling the resulting sound in comparison with the original sound. But, because the goal is to simulate damage, the resulting sound is intended to be different from the original sound. Thus, by definition it is a bad idea to scale the result $\tilde{a}$ to the original signal a.

6.6. Simulating Hearing Loss

A simple way to simulate hearing loss in the transmission line cochlea model is to update the damping term to have increased damping in the “damage” region. The damping term of the model can be seen in Figure 5.14. Of course, the actual damping profile used is as seen in Figure 5.19.

For the damage simulations in this report, the damping term is multiplied with a Gaussian hump (see: Figure 6.12) centered in the cochlea length that affects approximately 10% of the cochlea length. This creates the theoretical damping profile as seen in Figure 6.13. Of course, the damping profile, with the sponge layer, in actual use is that seen in Figure 6.14.

In Figure 6.15 is a cochleogram of the sound seen in 6.1. In Figure 6.16 is a cochleogram of the same sound snippet as in Figure 6.15 except here hearing loss is being simulated. There are four lines super-imposed on this Figure. These four lines correspond to oscillators 250 and 350, plotted in red, and oscillators 282 and 319, plotted in green. These lines also correspond with the red and green x’s that are seen in Figure 6.12. That is, the data in the cochleogram that falls between the red lines is in the region affected by the update to the damping term. The data that falls between the green lines is in the region where the damping term was modified to a factor greater than 2.

When comparing Figure 6.15 with Figure 6.16 it can be seen that the wave patterns follow a more vertical (parallel to the Y-axis) orientation in the damage region. This is because sound at the frequency corresponding to the damage region will still excite the oscillators, but the oscillations die out much more rapidly due to the increased damping. Waves on the cochlear partition that start at a higher frequency (lower oscillator number) and travel as a wave, along the partition, tend to not have as much of a representation in the damage region. This is why there is less of the horizontal “drifting” of the wave patterns in the damage region. Energy from oscillators prior to the damage region does traverse the region, and will continue to the oscillators beyond the region, however, with less energy than if there was not the damage region.

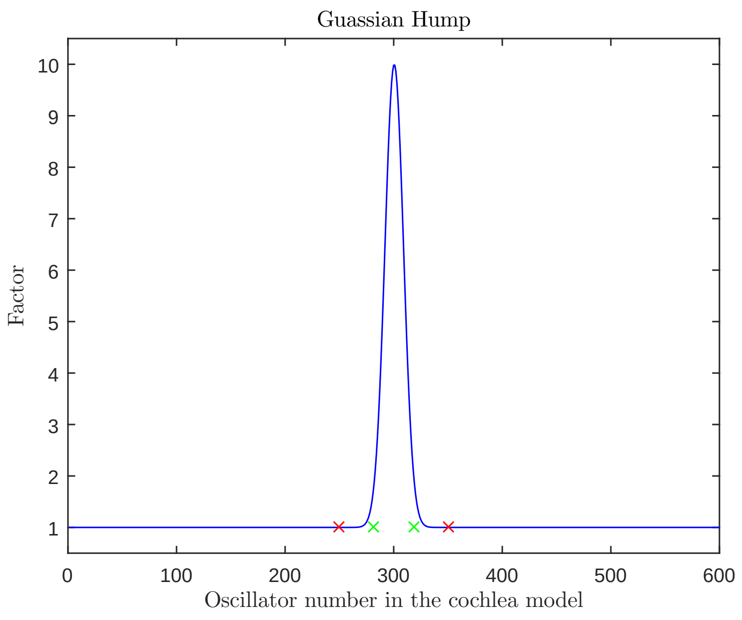

Figure 6.12: Gaussian hump factor used to modify the damping profile of the model, the two red x's here indicate points where the hump starts and ends. The green x's indicate the points between which the value will be more than doubled from the normal. On the cochleograms that follow, these values are indicated by similarly colored lines

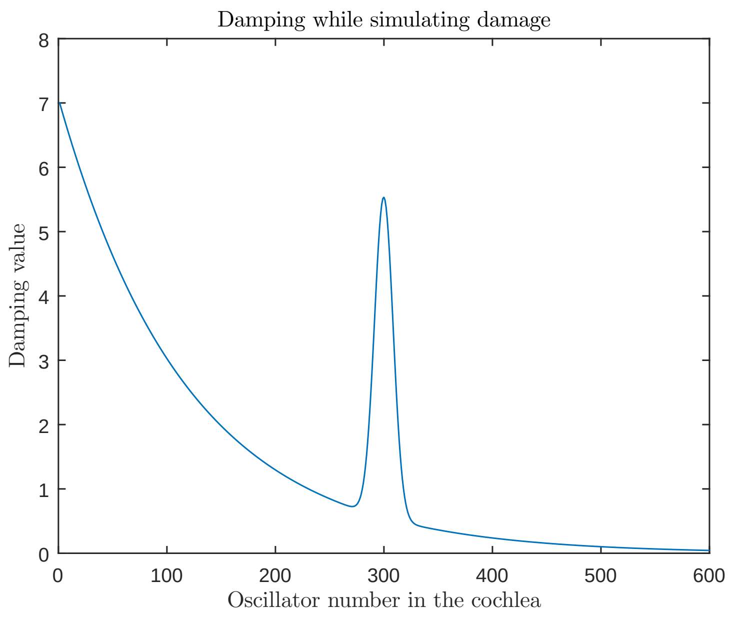

Figure 6.13: The theoretical damping profile of a model that simulates hearing loss

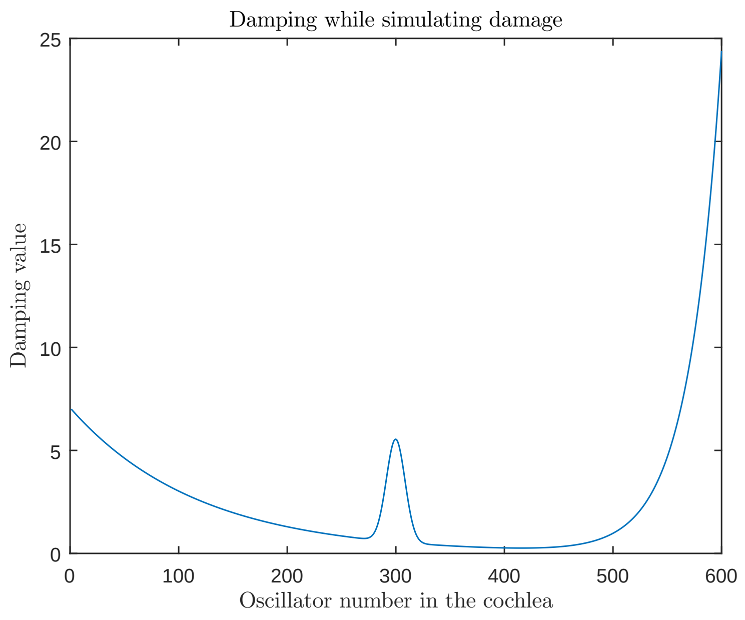

Figure 6.14: The actual damping profile used to simulate hearing loss

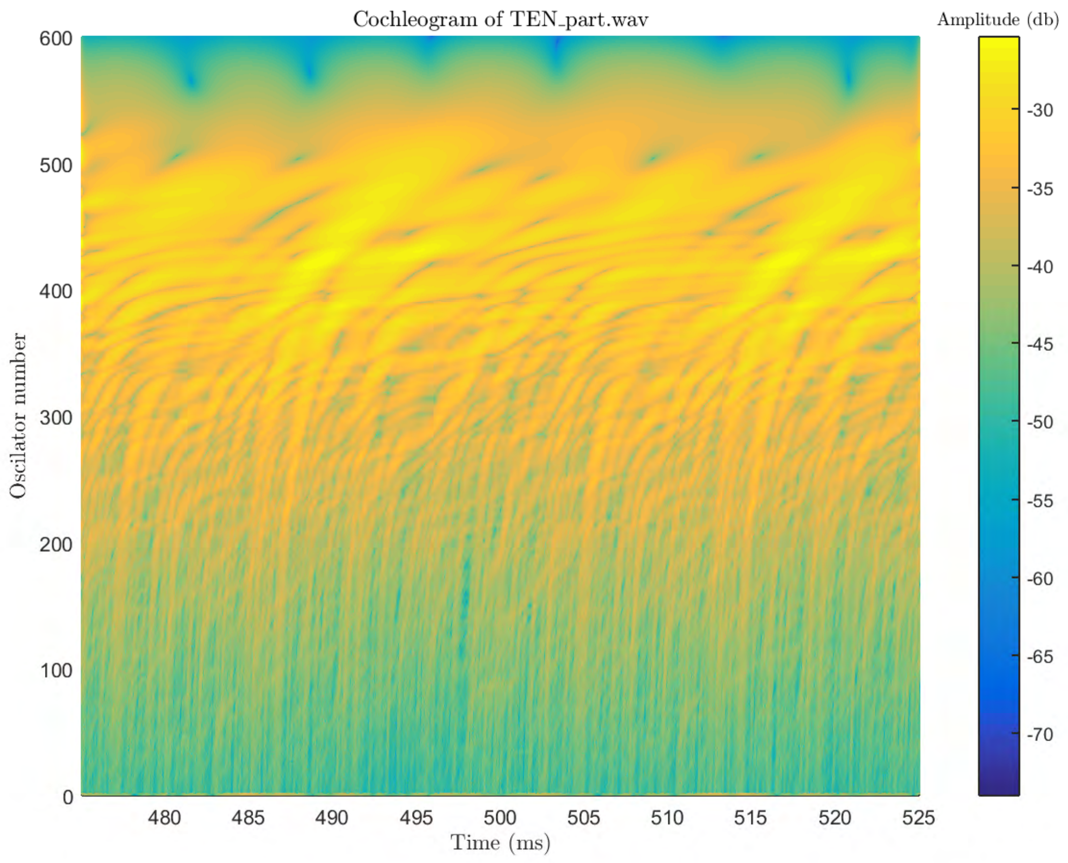

Figure 6.15: The cochleogram of "Ten_part.wav". A cochleogram is a comprehensive visualization of the output from a transmission line cochlea model. On the x-axis is time, on the y-axis the oscillators of the model. On the z-axis is the position of the given oscillator in time.

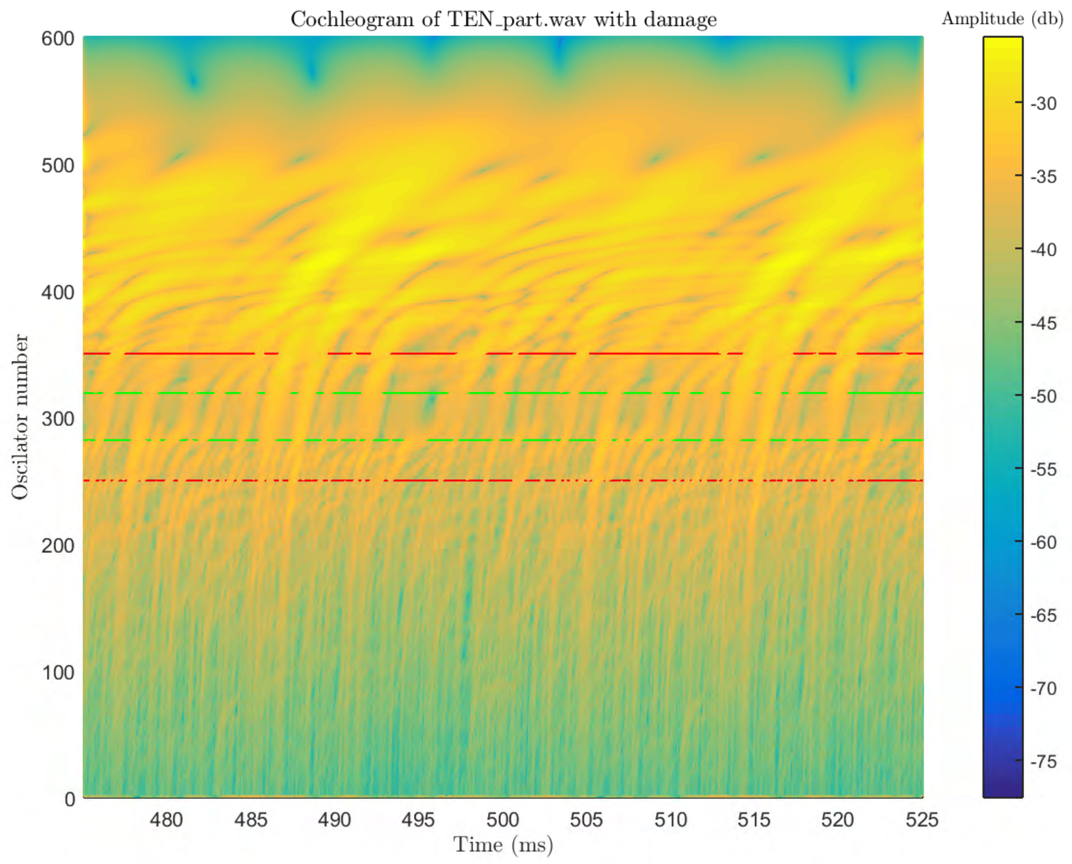

Figure 6.16: Cochleogram of "Ten_part.wav" while simulating hearing loss. The data that falls between the red lines is in the region where the damping term was affected. The data that is between the green lines is the area where the damping was increased beyond a factor of 2. The lines corresponding to oscillators 250, 282, 319, and 350 are plotted at a depth of approximately -36.4, -34:9, -35.0, and -32.6 dB

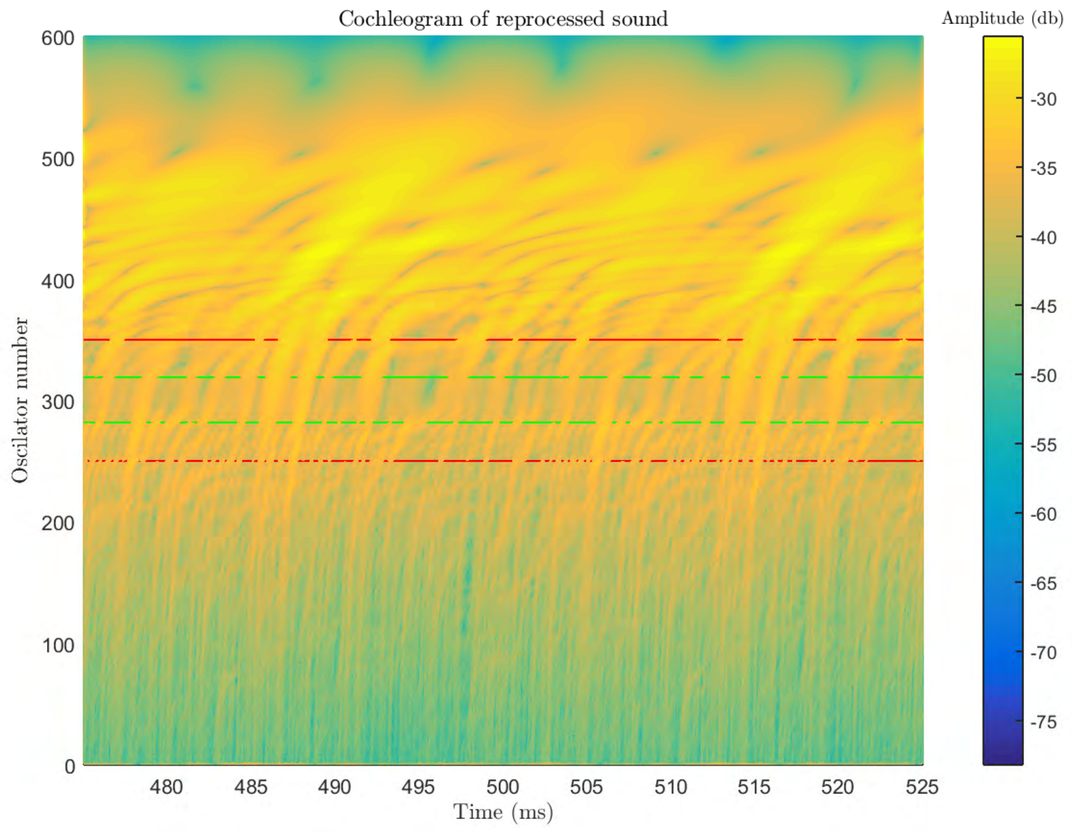

Figure 6.17: The reconstructed sound sent through a normal hearing model.

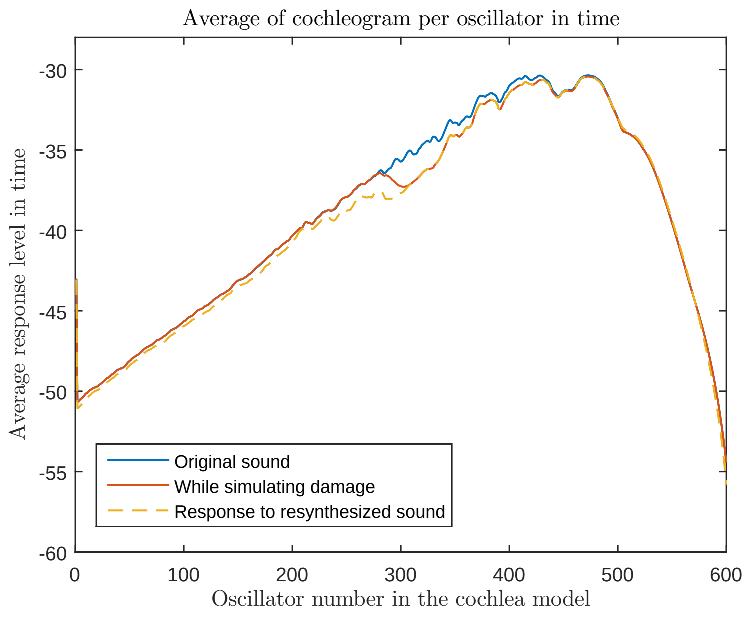

Figure 6.18: Here the time dimension of the cochleograms seen in Figures 6.17, 6.16, and 6.17 is averaged giving a 2-D representation of the cochleograms

Using the inverse filters built on the original sound, based on the cochleogram of normal sound (see: Figure 6.15) to resynthesize sound based on the cochleogram that simulates hearing loss (see: Figure 6.16) the resulting sound indeed has much less content in the frequency range corresponding to the damage region as seen in Figure 6.19. In this figure a deep notch in the frequency content is visible, centered at approximately 1.06 kHz.

The resynthesized sound $\tilde{a}$ (not shown) is almost identical to the original a as in Figure 6.1. When the sound $\tilde{a}$ is run through the cochlea model again with the settings on the model set to simulate a healthy ear then a cochleogram almost identical to the one in Figure 6.16 is produced. This cochleogram can be seen in Figure 6.17. These two different cochleograms match very closely.

Just by looking at the Figures 6.16 and 6.17 it is hard to tell how similar, or different, the two cochleograms are. Therefore, a way to make the comparison easier, is to average the cochleogram in time. In this way the 3-D cochleogram is visualized in 2-D, where the response in time is averaged. In Figure 6.18 three cochleograms are compared. The cochleogram in Figure 6.15 is a response to the sound a, and is represented by the blue line. Let this model output be denoted by MO_1. The cochleogram in Figure 6.16, which is the model output, lets call it MO_2, from which the damage sound $\tilde{a}$ is resynthesized from, is represented by the red line. Finally, the cochleogram from Figure 6.17 is represented by the dashed yellow line. This is the cochleogram in response to the sound $\tilde{a}$. Let it be denoted by MO_3. MO_1 and MO_2 are very similar up until the damage region, at approximately oscillator 275. Beyond the damage region, the average of MO_2 is lower than that of MO_1 until about oscillator 500. At this point the two are very similar. Interestingly, the average of MO_3 is lower up until the middle of the damage area, at oscillator 300. Until oscillator 200 the average is just slightly below that of both MO_1 and MO_2. But it seems that the damage region in $\tilde{a}$ starts a little early with a gap between MO_2 and MO_3 between oscillators 250 and 300. This corresponds to the frequencies higher than the simulated damage area, for which, the frequency spectrum plot of the sound $\tilde{a}$ (see: Figure 6.20) showed it was almost identical to the original sound a. Because of that, this gap between MO_2 and MO_3 seems unexpected. Beyond oscillator 300 the average of MO_2 and MO_3 are almost identical. These results would suggest that the resynthesized sound has a slight exaggeration in simulated damage in frequencies just above the damage area.

The fact that the cochleogram in Figure 6.16, as the result from the model that simulates a damaged ear and that the cochleogram seen in Figure 6.17 as the result of the “damaged sound” $\tilde{a}$ on a model of a normal cochlea, are so similar, is strong proof that the sound $\tilde{a}$ is an accurate representation of the damage being simulated.

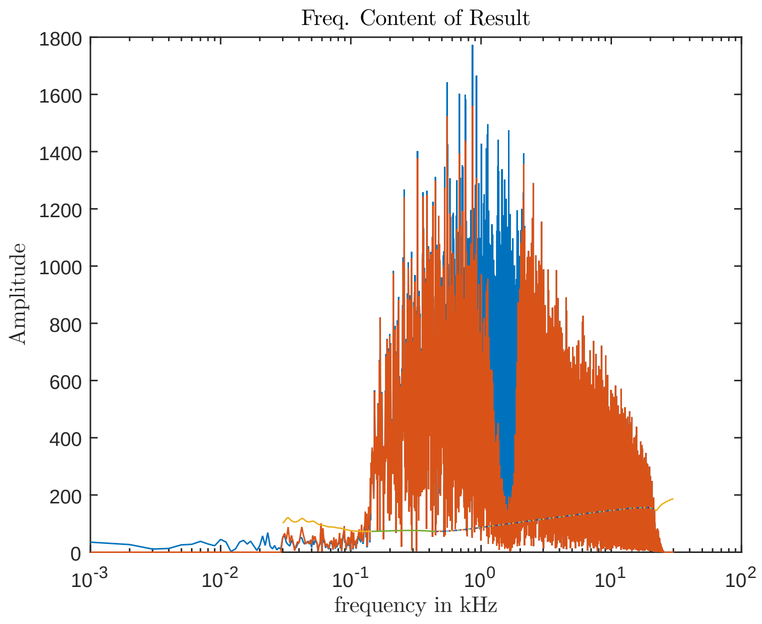

Figure 6.19: The frequency content of the sound with simulated hearing loss. The data plotted in blue is the frequency content of the original sound, and the data in orange is the frequency content of the sound with damage being simulated

The main objective of this research project was to be able to produce the sound as a person with hearing loss would hear normal sound. This sound is referred to as $\tilde{a}$. This objective is met for linear transmission line cochlea models as evidenced in the previous paragraph. An accurate tool to turn damage that is being simulated into sound is hereby provided. Future research that investigates the changes needed in the model to simulate various kinds of hearing loss could therefore use the tool created here to listen to the effects of the damage.

6.7. The Damaged Sound

In Figure 6.19 the frequency content of both the original signal as well as the frequency content of the damaged sound is seen. “Damaged sound” refers to sound that is the result of simulating hearing loss, or hearing damage. Let the original sound again be referred to as a, and let $\hat{a}$ be the sound where hearing loss is simulated. The sound $\hat{a}$ indeed has less content in the damage region. Careful inspection of Figure 6.19 shows additional features that are of interest. Zooming in on this figure to the higher frequencies (see Figure 6.20) shows that $\hat{a}$ is almost identical in the higher frequencies. These frequencies correspond to the oscillators that come before the damage region in the cochlea model. However, inspecting the low frequencies shows that there is a range of frequencies lower than the damage region in which there is less content than is in the origional signal. This can be seen in Figure 6.21 in the frequencies in the range of 0.5 – 0.8 kHz (500 – 800 Hz).

A plot of the ratio between the frequency content of the sound $\hat{a}$ to a is shown in Figure 6.22. In this figure, it is clearly evident that the ratio between the two is close to 1 for the frequencies higher than the damage region. At the lowest point the ratio is almost 0.1, which is close to the inverse of the multiple, by which the damping term in the model was modified at a factor of 10. In the frequencies below the damage region, which correspond to the oscillators after the damage region in the model, the ratio tends to be below 1, even in the region outside of where damage was simulated. This indicates that by using a transmission line model, effects from the damage region propagate to areas not in the damage region. Such propagation of effects is not possible when simulating hearing loss through simpler filter bank signal processing.

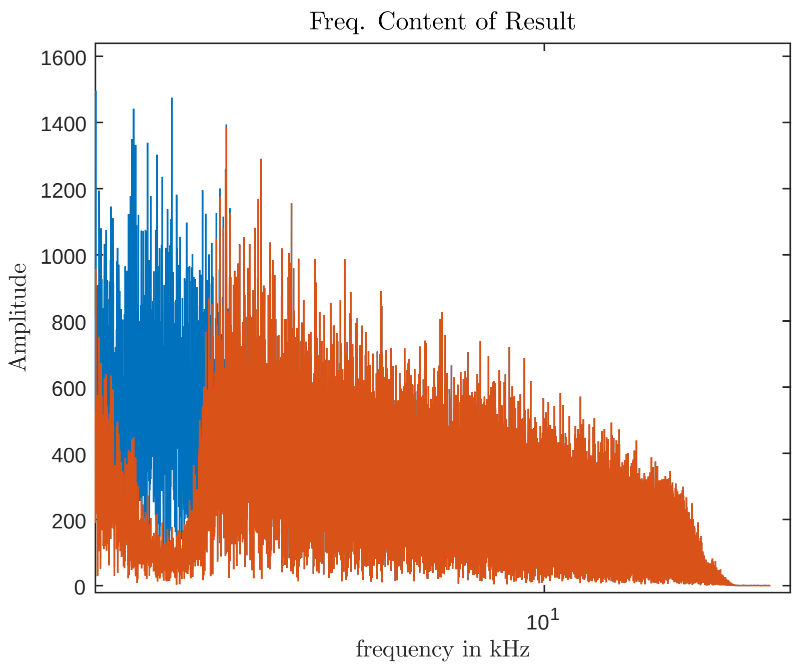

Figure 6.20: High frequency content comparison between a and $\hat{a}$

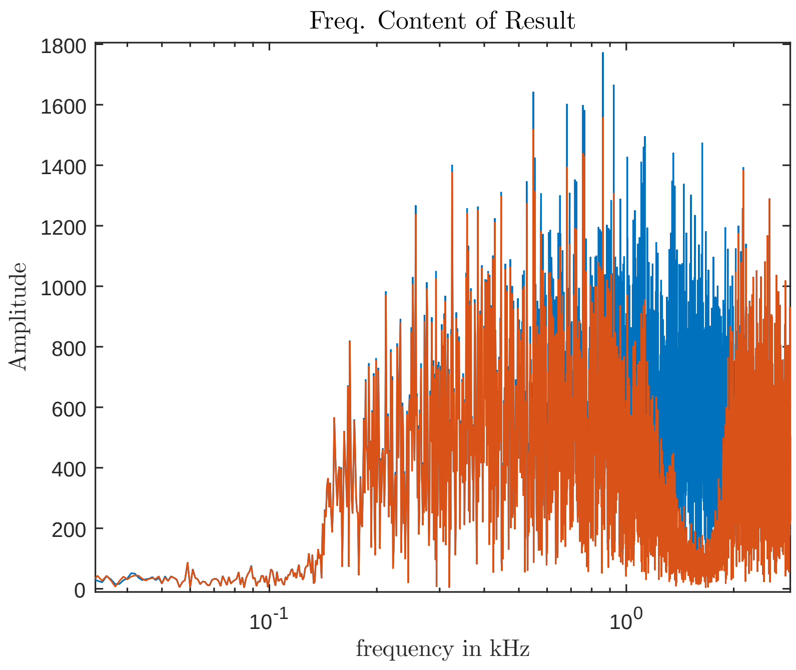

Figure 6.21: Low frequency content comparison between a and $\hat{a}$

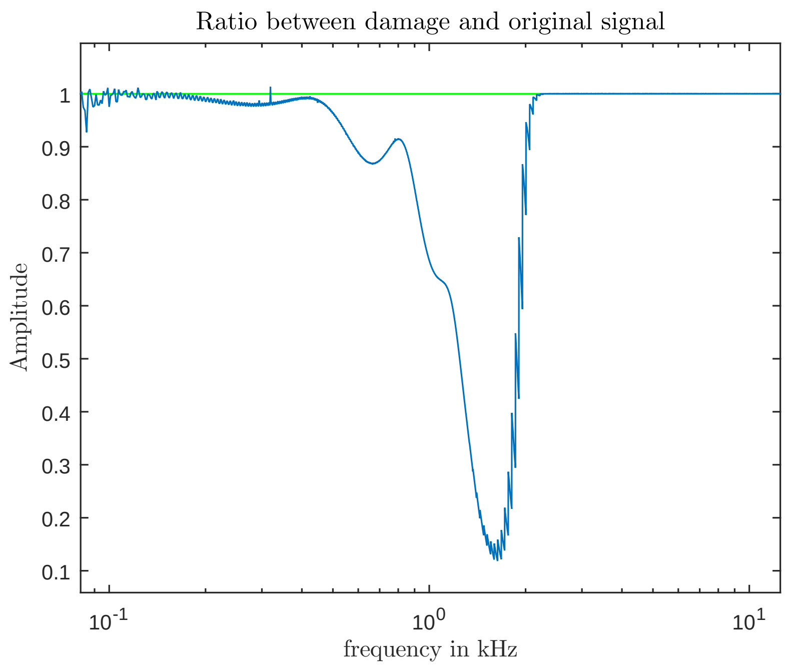

Figure 6.22: Ratio of the frequency content of the sound $\hat{a}$ to a. A green line is plotted at $Amplitude = 1$ for reference. (Every third oscillator is used, in the damage region, for sound reconstruction)

6.8. The Low Frequencies

In the low frequency range, that is, frequencies below approximately 100Hz, it is surprisingly difficult to get an accurate reconstruction of sound. This is evidenced in Figure 6.23. The result is very accurate for frequencies as low as 53Hz (0.053 kHz) but for frequencies lower than this, the result is not very accurate. Increasing the length of the impulse response was tried, and did not improve the result. The length of impulse response for the result seen in Figure 6.23 was $100,000$ samples which showed a much more accurate result than when an impulse response of length $140,000$ was used (not pictured in this report).

It might be tempting to claim that hearing loss does not usually affect the region of the cochlea responsible for this frequency range, so these frequencies can simply be ignored in the reconstruction. It is, furthermore, even possible that the average listener would not be able to tell if this frequency range was missing from a complex sound. However, when comparing cochleograms of the reconstructed signal a very significant difference is seen. For example, in Figure 6.24 when looking at the oscillator range of $550 - 600$ there is not much activity. This cochleogram is an example of reconstructed sound where all content below 30Hz was ignored. In contrast, in the original sound there was excitation of those oscillators (see: Figure 6.16). By including the frequencies as low as 21Hz a much better result was obtained, as seen in Figure 6.17. For this reason, it is important to not ignore low frequency content even if it contains some error.

6.9. Number of Representing Oscillators

A claim was made that using every third oscillator in the sound reconstruction preserved a sufficient level of accuracy. Keep in mind that a given oscillator contributes to frequencies strictly lower than its characteristic frequency in the resynthesis. Therefore, if not every oscillator is used, then the effect of the simulated damage region will be skewed to the lower frequencies.

Consider the case where every tenth oscillator is used in the sound reconstruction. Assume further that a damage region is being simulated such that oscillator x is still just inside the healthy zone. Let the next oscillator be $x+1$ and let it be well in the damage zone. Let the characteristic frequencies of the above oscillators be f and $f+1$ respectively. The sound reconstruction from oscillator x will contribute to the sound in the frequencies from $f+1$ to f. The frequencies close to $f+1$ are in the damage zone, but because they are reconstructed from an oscillator that is not in the damage zone it will not reflect damage. In the same way if x were to be in the damage zone and $x+1$ was already in the zone not affected by damage then the damage

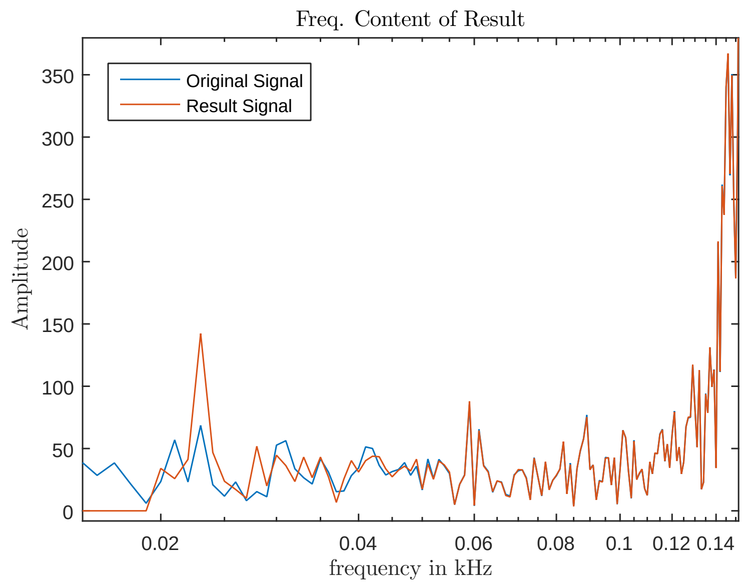

Figure 6.23: Error of the reconstructed (result) sound in the low frequencies

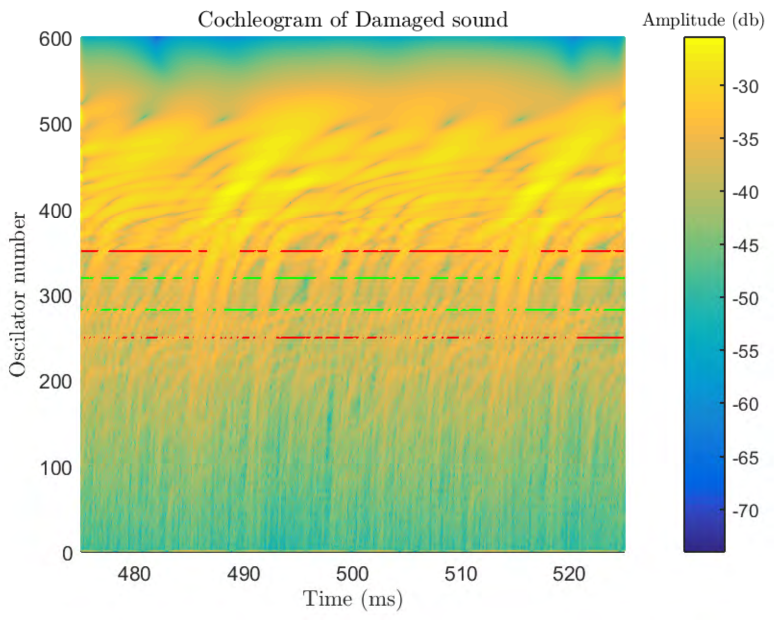

Figure 6.24: Cochleogram of sound reconstructed where sound content bellow 30 Hz was ignored. This can be seen in the oscillator range above 550

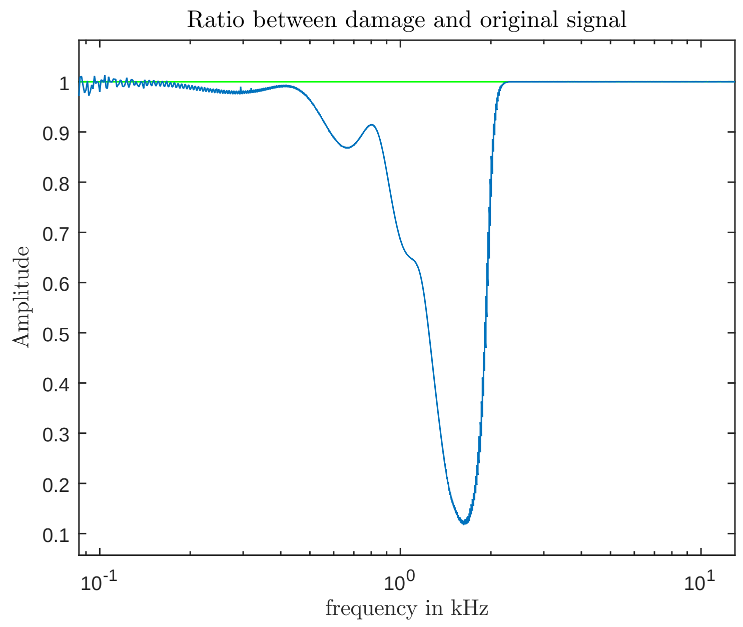

Figure 6.25: The ratio between the frequency spectrum content of the original sound a and the damage sound $\hat{a}$ where every oscillator is used in the damage region

would be projected into a frequency range where there is not to be damage.

This effect is present, and it is visible when carefully observing the frequency spectrums of the resultant damage sound. To see the improvement of using every oscillator versus using every third oscillator compare Figure 6.22, where every third oscillator is used, and Figure 6.25, where every oscillator is used. On the high frequency side of the damage notch, the ratio is clearly smoother with a denser oscillator concentration. Strangely, when comparing the cochleograms there was not a clear improvement by using more oscillators versus using fewer.

7 Conclusion

7.1. Discussion

Hearing loss is a significant global health problem of increasing prevalence. For those that suffer from it, it greatly decreases the quality of life. As such, research in this area will become more important in the coming years. A facet of this research will be the development of high quality hearing loss simulators. These are needed, both for educational purposes, and also to aid in the faster and more cost effective development of treatment for hearing loss. Because the human auditory system is amazingly complex, hearing loss simulators need to take as many physical phenomena into account as possible.

A practitioner in the field of hearing simulation must have a working knowledge of the human auditory system. In this report the needed groundwork, as it relates to hearing loss, is clearly and concisely presented. The transmission line cochlea model, also, is elaborated on and explained. Digital signal processing, an indispensable ingredient of simulating hearing loss, is also introduced.

This research project contributes to the goal of simulating hearing loss in a few ways. Firstly, the methods to simulate hearing loss are based on a mathematical model of the ear. This allows for more physical phenomena of hearing loss to be accounted for than do filter bank methods. To make the simulation of hearing loss, by use of the transmission line cochlea model, possible for the first time, a variety of problems are successfully dealt with. For the simulations to work, residual energy in the model needs to be dealt with properly. As long as this is not done, sound resynthesis could only be done for exceedingly short sound inputs, and only using a small part of the oscillators representing the cochlea. Hearing loss simulations would fail to be realized because a greater part of the cochlea must be represented. Furthermore, a technique to combine the contributions from multiple oscillators is developed, such that accurate sound can be reproduced from the model output. The resulting sound is accurate, both in time and frequency domain representations.

Using the developed methods for sound resynthesis to simulate hearing loss shows promise because unlike for filter bank methods, the effects of damage in one area of the cochlea propagates to other areas of the cochlea. Furthermore, the cochleogram from the model that simulated damage looks very similar to the cochleogram of a normal cochlea in response to the resynthesized damage sound. This demonstrates that the sound that is resynthesized accurately represents the damage that the model simulates.

7.2. Recommendations

This project lays the groundwork for a few areas of further research. Many of the hearing phenomena in the human auditory system are nonlinear. There are also nonlinear transmission line cochlea models. Hence, an important topic for further research would be to extend the back transformation methods developed in this project for linear models to work for nonlinear models. A good starting point would be to use linear inverse filters on nonlinear output. The time segments should be kept as short as possible. Each inverse filter would need to be individually scaled for each time segment of sound by comparing the result of normally resynthesized sound with the original input sound. To simulate damage, an inversely scaled filter should be used.

To simulate damage, the damping term of the cochlea is modified. Another needed research area, is to investigate the optimal levels and methods of modifying the damping term to simulate various known types of hearing loss. There is an abundance of literature on psychoacoustics (psychoacoustics is the study of sound perception) that could be used to calibrate the modifications of the damping term. This type of research is required to make hearing loss simulations, based on the transmission line cochlea model, a useful tool to the hearing industry.

A third area for further research relates to the method of negating the build-up of residual energy in the cochlea model. In the current project, a sponge layer of increased damping is used. Further research should answer if this is indeed the best way to handle residual energy. If so, what is the optimal application of this technique to match the conditions in the human cochlea. Further research should also investigate if energy reflection and the buildup of residual energy is also a problem in nonlinear transmission line cochlea models.

7.3. The End

For the first time sound has been successfully and accurately resynthesized from a linear transmission line cochlea model output. Such output has been analyzed by the use of cochleograms in the past. Resynthesized sound introduces a new way of exploring the output of such models. But, more importantly, a new method to simulate hearing loss hereby came into existence.

A

MatLab Code for Sound Resynthesis

MatLab code for the resynthesis algorithm. This code follows Algorithm 3 from Section 6.2.

```
% The code in this file computes inverse filters to do sound
   resynthisis
% from the output of a transmission line cochlea model (CM). To get the
    CM
% output it will either need to be used in conjuntion with the function
% cochleogram_dimensionless_RK4_func.m or the CM output will need to be
% uploaded into memory such that the MatLab workspace can access the
   needed
% variables.
% This file is based on functions provided by INCAS3

% Let us begin:
% Clear MatLab command screen and displaying progress message
clc;
disp('Working on it');
disp('-------------------------------------------')

%% Runtime Parameters:
% The current code assumes the CM uses 600 Osillators
POS = 1:600; % the oscillators to get traces for
pos = 1:60;  % the (subset of) oscillators to use for sound
   reconstruction

SIM_LOSS        = true;         % Set to true to use the contributions
   from multiple oscillators
samples         = 100000;       % Length of impulse response to use for
    reconstruction
GRAPH           = true;     % Plot the original sound and the
   reconstruction?
SOUNDORIG       = false;    % Playback the original sound before
   playing back the reconstruction
SOUND           = false;    % Playback the reconstructed sound
getFilter       = true;     % Set to true if the filter function are to
    be computed
getTrace        = true;         % Set to true if traces need to be made
    with the cochlea model
CUTOFFdB        = 140;      % The dB bound for the inverse filter
   functions
```

```
% Directry where sound files to be processed are
WorkingDir      = 'C:\Users\Leo\Dropbox\MastersProgram\Thesis\
   SimpleSoundFiles\';
% name of the signal to process
fileNameSignal  = 'DamageSimSparseOsc.wav';
% name of sound file if the sound is to be saved, leave blank if file
   should not be saved
filenameRes     = [];
% If loading amplitude from a .mat file
ampMatFile      = 'NormHearLin10Part1Sec';


%% Getting the impulse response and the oscillator trace from sound
   input
if getTrace
    disp('Getting oscillator trace of impulse');
    disp('--------------------------------------------')

    % Building the impulse "Sound signal"
    t = zeros(1,samples);
    t(1) = 1;

    % Getting the impulse response of the system
    [impulse, cm_sRate] = cochleogram_dimensionless_RK4_func(t,POS);

    % resampling the sound signal to the sample rate of the CM
    disp('Getting oscillator trace of sound')
    disp('--------------------------------------------')
    [signal,signal_sf] = audioread([WorkingDir fileNameSignal]);

    % Resampling is done in two steps, the ratio can be too large for a
        single step
    % Doing it in two steps is also more memory efficient
    if cm_sRate ~= signal_sf
        signal_cm_sRate = resample(signal, cm_sRate ,signal_sf);
        signal_2cm_sRate = resample(signal_cm_sRate, 2 ,1);
    end

    fprintf('Signal resampled from %i sRate to %i 2*cm_sRate\n',
       signal_sf, 2*cm_sRate);

    % Getting the oscillator traces for the singal sound file
    [amplitude, cm_sRate] = cochleogram_dimensionless_RK4_func(
       signal_2cm_sRate,POS);
end

% Uncomment this line to stop the program here and allow to save the
   variables
% generated up until this point for future use

%return;


if ~exist('amplitude')
    disp('Load a valid oscillator trace or set getTrace = true')
```

```
        return;
end

% Setting signal length and number oscillator traces to compute for
nSeg = length(pos);
ampLength = length(amplitude);


%% Computing the inverse filter; this was originally a separate file
   called get filter
if getFilter
    disp('Calculating tuning curves')
    disp('-------------------------------------------')

    if ~exist('impulse', 'var')
        disp('Load a valid impulse file');
                return;
    end

    % Getting the length of impulse response to be used for inverse
      filter
    h = impulse(pos,1:samples);

        % Making the tuning curves
    f = fft(h, ampLength, 2);

    % Bounding the inverse filters to decrease error amplification
    m = max(20 * log10(abs(f)),[],2) * ones(1,ampLength);

    % set f to 1 where it is zero to avoid divide by zero error
    f = f + (f==0);

    % computing the bound to eliminate error peaks
    b = (20 * log10(abs(f)) > (m - CUTOFFdB));

    % All values of the tuning curve below CUTTOFFdB less than the peak
       are
    % set to the value of bound
    bound = (10 .^ ((m - CUTOFFdB)/20));

    % the indices of f were b == 1 are set to bound
    f = (b .* f) + (~b .* bound);

    disp('Calculating inverse filter functions')
    disp('-------------------------------------------')

        % Inverting the tuning curve to create the invers filter
    inv_f = 1./f;
    fAmpl = fft(amplitude,[],2);
end

fax_Hz = (0:ampLength-1)*cm_sRate /(ampLength);
fax_kHz = fax_Hz/1000;

res = zeros(1,ampLength);
```

```
% This is the old way of combining the contributions used by INCAS3
if ~SIM_LOSS
    disp('Calculating results')
    disp('-------------------------------------------')
    Res = zeros(1,ampLength);

    % combining the result of the different segments in the frequency
       domain
    for seg = 1:nSeg
        Res = Res + fAmpl(seg,:) .* inv_f(seg,:);%(2:end);
    end

    res = real(ifft(Res)); % transferring the result in the frequency
       domain to the time domain
    % scaling the result in the time domain
    res = res  / ( 1/max(abs(signal)) * max(abs(res(10000:end-10000)))
       );

else % The modified method of combining the results

    disp('Calculating results')
    disp('-------------------------------------------')
    % Finding the charachteristic freq. of the segment
    fax_Hz = (0:ampLength-1)*cm_sRate / ampLength;
    fax_kHz = fax_Hz/1000;

    % boolean to control the low frequency reconstruction
    firstFilter = true;

    % \tilde f in the report, the lower bound that the specific filter
       is
    % to contribute to
    [M,oldI] = min(abs(inv_f(nSeg,1:floor(end/2))));

    % The first filter is not actually used
    inv_f(nSeg,:) = zeros(size(inv_f(nSeg,:)));

    % The number of iterations of scaling the inverse filter to use
    scaleIter = 5;

    % Combining the results
    for ind = nSeg-1:-1:1

        % Print the progress every 10 segments
        if rem( ind, 10 ) == 0;
            fprintf('Remaining steps: %d \n', ind );
        end

        % unless the filter has already been built and should no longer
            be touched
        if getFilter

            [M,I] =  min(abs(inv_f(ind,1:floor(end/2))));

            if (oldI + 2) >= I % A filter needs to contribute to at
               least two frequency points
            inv_f(ind,:) = zeros(size(inv_f(ind,:)));
            continue;
        end


        if firstFilter % if first filter then set everything below
           21 Hz to zero
            inv_f(ind,1:20) = zeros(1,20); % Getting rid of low
               freq. error
            inv_f(ind, I+1:end) = zeros(1,ampLength-I);
            firstFilter = false; % so there is only one first
               filter
        elseif ind == 1 % if this is the last filter then set
           frequencies above 30 kHz to zero
            inv_f(ind, 1:oldI) = zeros(1,oldI);
            inv_f(ind, 30000:end) = zeros(1,ampLength-30000+1);
        % otherwise set the filter to zero except in the frequency
           ranges that it contributes to sound reconstruction
        else
            inv_f(ind, 1:oldI) = zeros(1,oldI);
            inv_f(ind, I+1:end) = zeros(1,ampLength-I);
        end

        if I - oldI < 22
            pfas = floor((I - oldI)/2)-1; % think sample points to
               use in the average
        else
            pfas = 10; % think sample points to use in the average
        end

        % Used to get the ratio by which to scale the inverse
           filters
        freqContSignal = abs(fft(signal_cm_sRate'));
        avgStartInd = floor((oldI+I)/2-pfas);
        if avgStartInd < 21
            avgStartInd = 21;
        end
        avgEndInd = floor((oldI+I)/2+pfas);
        avgSign = mean( freqContSignal(avgStartInd:avgEndInd));

        % A few iterations of scaling the inverse filter
        for scalling = 1:scaleIter
            Res = fAmpl(ind,:) .* inv_f(ind,:);
            tempRes = real(ifft(Res));
            freqContRes = abs(fft(tempRes));
            avgRes = mean( freqContRes( avgStartInd:avgEndInd) );
            if scalling ~= scaleIter
                inv_f(ind,:) = inv_f(ind,:)*(avgSign/avgRes);
            end
        end

        % Constructing the time domain signal
        res = res + tempRes;

    else % The filters have already been built just construct the
       result
```

```
            if max(abs(inv_f(ind,1:floor(end/2)))) > 1
                Res = fAmpl(ind,:) .* inv_f(ind,:);
                tempRes = real(ifft(Res));
                res = res + tempRes;
            end
        end

        oldI = I;

        if GRAPH
            figure(3)
            plot(res);
            title('3', 'Interpreter','latex');
            drawnow;
        end
    end
end

disp('Calculations Done')
disp('--------------------------------------------')

foi = 30000;

%% Now the work is done the rest is just outputs and cleanup

% Plot graphs for report or the origional sound file with the
   reconstructed
% sound file
if GRAPH
    % plotting original sound and reconstructed sound
    figure(10)
    subplot(2,1,1)
    plot([0:ampLength-1]/(cm_sRate/1000), signal_cm_sRate)
    subplot(2,1,2)
    plot(res)
end

res_sf = ((length(res))/ampLength) * signal_sf;

% Playing the original sound
if SOUNDORIG
   sound(signal, res_sf);
   pause(18);
end

% Playing the reconstructed sound
if SOUND
        soundToPlay = resample(res,signal_sf,cm_sRate);
   sound(soundToPlay, signal_sf);
end

% write the reconstruction to a wav file
if ~isempty(filenameRes)
    soundToPlay = resample(res,signal_sf,cm_sRate);
    audiowrite([filenameRes '.wav'], soundToPlay, signal_sf);
end
```

B MatLab Code for the Updated Cochlea Model

MatLab code for the updated transmission line cochlea model. This code is set as a function with return variables. It also includes settings to allow for the sponge layer to be turned on and off. The sponge layer is as described in Section 5.4.2. It also allows for hearing loss to be simulated by updating the damping term.

```
% This implementation of the dimensionless Cochlea model followed the
% 16/1/1991 document by Marc van den Raadt as closely as possible.
%
% 19/3/2013: Added Zweig impedance infrastructure.
% This follows the FORTRAN 77 code (from INCAS3) as closely as possible
   .
%
% 10/20/2015: Added Sponge layer at the helicotrema and allows for the
% simulation of hearing loss by updating the damping term. It also is
% set as a function rather than a script. Furthermore, it is modified
   to
% accept an arbitrary sound file as either a path to a file, or as a
% numeric array
% - Leo Koop, Simulating Hearing Loss, October 20, 2015
%
%
% Author:        Jan Stegenga
% Version:       0
% Date:          1/2/2013
% Copyright:     INCAS3, The Netherlands
%
% The changes to linear version validated on 06/09/2015 (Leo)
%

function [ pos_trace, cm_sRate ] = cochleogram_dimensionless_RK4_func(
   filename, POS )
% INPUT:    filename -- is the sound file to pass into the cochlea
%           POS -- is the oscillator to get the trace of

    place_map       = 'fortran';
    LINEAR          = 1;
    ZWEIG           = 0;
    TIMESTEP        = 200;
    PLOT            = 0;
```

```
KILL_ENERGY      = 1;         % Set to '1' or true to add the sponge
   layer
SIMULATE_DAMAGE = 0;          % Set to '1' or true to modify the
   damping term to simulate damage

VSOM_CORR = 1;
% parameter list found on page A4 (van den Raadt [3])
% some parameters are explained on pages A1 to A3

% number of segments (oscillators) (excluding the midear and
   helicotrema)
N    = 600;
% [s]         simulation time step
dt   = 2.0e-6;
% [s]         time reference, used to normalize the time
t_0_hat       = 1e-3;
% [m]         length reference (x), used to normalize BM length
x_0_hat       = 35e-3;
% [m]         amplitude reference,
% used to normalize dynamic values of amplitude and amplitude
   change
y_0_hat       = 1e-9;
% [m]         width scalae (page 19)
b    = 1e-3;
%   BM width as function of segment (index) = scalae width
b_BM          =  b*ones(N,1);
% [m]         height of scalae (page 3)
h    = 1e-3;
% [m^2]       cross sectional area of the cochlea = b*h
s_sc          = 1e-6;
% [kg/m^3]  density of cochlear fluid
rho  = 1e3;
% [kg/m^3]  specifiek acoustic mass of BM
m_s  = 0.5;
% [m^2]       area of a segment (stape)
s_st          = 3e-6;
% [m^2]       area of eardrum
s_t  = 60e-6;
% [rad/s]    resonance frequency of midear
omega_rm      = 2*pi*2000;
%   Reciprocal of the Q factor of the midear
delta         = 2.5;
%   Transformationfactor of the midear
n_t  = 30;
% [Ns/m^3]  Specific acoustic impedance of air
Z_s  = 415;
%   Parameter created by normalization on x
alpha_squared        = 4900;
%   Parameter created by normalization on x
alpha_x_squared= alpha_squared;
% [Pa]        reference pressure, equals 0 dB SPL
p_ref         = 2e-5;
% [Pa/m]      constant for specific acoustic stiffness
s_0  = 1e10;
% [/m]        determines relationship between frequency and location
   (x)
```

```
lambda        = 300;
%   modulation factor for the impulse response of a resonator
epsilon       = 5e-2;
%   normalized midear segment (stape) length
dx_01_tilde = 1/N;
% demping at the helicotrema, according to the fortran code:
dhel = 2.2e7;


if place_map == 'fortran'
    % normalized stape length, following the f77 version
    dx_tilde          = (1-dx_01_tilde-(h/x_0_hat))/N;
    % definition x_tilde
    x_tilde_v         = [ 0, dx_01_tilde : dx_tilde : 1 - h/x_0_hat -
        dx_tilde ];
    % x_tilde has to be a column vector
    x_tilde       = x_tilde_v(:);
else
    % normalized stape length, following Van den Raadt
    dx_tilde          = 1/N;
    x_tilde_v         = 0:dx_tilde:1;
    % x_tilde has to be a column vector
    x_tilde       = x_tilde_v(:);
end

% parameters for the nonlinear part of the damping, page 34
mu_p1_tilde = 1;     % == chi_tilde
mu_p2_tilde = 1;     % == alhpa_tilde
mu_n1_tilde = 2;     % == 2*gamma_tilde
mu_n2_tilde = 1;     % == beta_tilde (or B_tilde)


lambda_tilde          = lambda*x_0_hat;
           % page 33
gamma         = m_s * s_t * omega_rm * delta / ( b * x_0_hat *
   dx_tilde * n_t^2 * Z_s );
% page 27, b_BM replaced by b
alpha_0       = gamma .* alpha_squared * dx_tilde;
           % page 27
alpha_x_squared= 2*rho*b_BM(1:end) * x_0_hat^2 / ( m_s * s_sc );
   % page 27

y_tilde       = zeros( N+1, 1 );       % page A3, volume displacement
   of BM
u_tilde       = zeros( N+1, 1 );       % page A3, volume velocity of
   BM
phi_tilde     = zeros( N+1, 1 );       % page 16, dimensionless phi,
% phi = pressure/specific mass, phi replaces accelaration in p = m*
   a

% page 21, specific acoustic mass
m_sm          = ( n_t^2 * Z_s * s_st ) / ( s_t * omega_rm * delta );
% page 6 and 7, g = damping * u + stiffness * y ->
% du/dt = 1/m*( p-g ), with g is dimensionless G
g   = zeros( N+1, 1 );
% page 25, A*phi_tilde = r; where r is represents the next state of
% uncoupled oscillators and A is the relation between neighbouring
    oscillators
r   = zeros( N+1, 1 );

% stimulus  parameters
% page 22, pure tone input: p_e = p_AM1*omega_1*cos( 2*pi*f_1*t )
        )
p_AM1        = p_ref*10^(60/20);             % [Pa]   stimulus: 60
    dB
omega_1      = 1;              % parameter to smooth-start the signal
f_1 = 1000;          % [Hz]

% dimensionless A construction               % page 26
% N+1 diagonal elements
A = diag( [          - ( 1 + alpha_0 * dx_01_tilde ); ...
- (2+alpha_x_squared(1:N-1) * dx_tilde^2 ); ...
- (1+alpha_x_squared(N)*dx_tilde^2+(4*x_0_hat*dx_tilde)/(pi*h+4*
    x_0_hat*dx_tilde)) ], 0 );

% Impedance matching at the helicotrema, according to the fortran
    code
A = diag( [          - ( 1 + alpha_0 * dx_01_tilde ); ...
- (2+alpha_x_squared(1:N-1) * dx_tilde^2 ); ...
- (2+alpha_x_squared(1) * dx_tilde^2 -1. + (1.69e5/ (1.69e5+8.26161
    e5))) ],0); ...
%- (1+alpha_x_squared(N)*dx_tilde^2+(4*x_0_hat*dx_tilde)/(pi*h+4*
    x_0_hat*dx_tilde)) ], 0 );

% add N above-diagonal elements
A = A + diag( ones(N, 1), 1 );
% add N below-diagonal elements
A = A + diag( ones(N, 1), -1 );
% invA is easiest way to solve the equation on page 26
invA = inv(A);

%% Pure Tone input that came with the file
%{
% input construction
t   = 0:dt:0.1;                              % time
t_tilde     = t/t_0_hat;                                % normalized
    time
dt_tilde    = dt/t_0_hat;                               % normalized
    timestep
p_e = p_AM1*omega_1*sin( 2*pi*f_1*t_0_hat*t_tilde );        %0.1*
    p_AM1*randn(size(t)) +
% is in fact in the unscaled time domain, since t = t_0_hat *
    t_tilde
taper       = exp(-(4-t_tilde).^2);          % taper, as used in the
     FORTRAN code.
% It prevents startup spurious events
taper(t_tilde>4) = 1;                                   % taper
P_e = ( t_0_hat^2/y_0_hat )*( n_t * p_e / m_s ).* taper;
% is scaled in time and amplitude
%}
% END of Pure Tone input that came with the file
%% From file or array input construction
```

```
    dt_tilde    = dt/t_0_hat;                           % normalized
       timestep

        % Modifications by Leo to allow for the input of arbitrary
           sound files or
        % sound represented as a numerical array
    if(isa(filename, 'char'))
        [P_e, sRate] = audioread(filename);
    elseif (isa(filename, 'double'))
        P_e = filename;
        sRate = 2*ceil(1/dt);
    end

        time_in_seconds = length(P_e)/sRate;

        % A few modifications to save some memory since t_tilde was
           used only for
        % the printing of the title of the plot at the end of the
           function
%        t       = 0:dt:time_in_seconds;                        % time
%        t_tilde         = t/t_0_hat;                           %
   normalized time
        %using the following instead of the above to try and save
           memory
        t_tilde = 0:dt:time_in_seconds;                         % time
        dt_tilde        = dt/t_0_hat;                           %
           normalized time


    % Resampling the sound file
    cm_sRate = ceil(1/dt);% I am pretty sure this should be just 1/dt
       instead of cm_sRate = ceil(1/(t_0_hat*dt));
    if 2*cm_sRate ~= sRate
        P_e = resample(P_e, 2*cm_sRate ,sRate);
    end
    %END From file or array input construction

    % Zweig impedance initialization, uses precomputed d and s
    d1  = - 0.12;       % = aplha (in FORTRAN 77 code)
    d2  =   0.01;       % = beta
    d3  =   0.062;      % = gamma
    d4  =   0.1416;     % = delta

    if ZWEIG
        d1 = d1+(epsilon-d1).*(exp(-20.*x_tilde(2:end))+exp(-20.*(1-
           x_tilde(2:end))));
        % modification to make the model more stable, FORTRAN
        d4 = d4.*(1-exp(-20.*x_tilde(2:end))-exp(-20.*(1-x_tilde(2:end)
           )));
        % modification to make the model more stable, FORTRAN
    end

    if LINEAR
        d_prof = 1; %d1;
        c_prof = 1; %d4;
```

```
end

% LEO EDIT: This is the damping profile of the code that I received
% Van den Raadt
d = epsilon*t_0_hat*sqrt(s_0/m_s)*exp(-0.5*lambda_tilde.*x_tilde(2:
    end));
s   = t_0_hat^2 / m_s * s_0 .*exp( - lambda_tilde.*x_tilde(2:end) )
    ;

% LEO EDIT: This is what I changed
if KILL_ENERGY && LINEAR                    % So I can turn off my
    changes

     % currently the sponge layer is only tested to work for the
         linear version of the code

     x_temp = linspace(0,1,600);

     % This is the change to d(x)
     d = d.*(1+560*exp(-25*(1-x_temp)))';
end

     % If damage is to be simulated the damping term is updated in
         the following lines
if SIMULATE_DAMAGE

               damage = gausswin(600,35); % Centered Guassian hump
                   affecting approx 10% of width
     damage = 1 + 9*damage;

%     figure(2)
%     plot(1:600, damage, 'b')
%     title('Guassian Hump','Interpreter','latex')
%     xlabel('Oscillator number in the cochlea model', 'Interpreter
    ','latex')
%     ylabel('Factor', 'Interpreter','latex')
%     axis([0 600 0.5 10.5])
%     hold on
%     plot(250, 1, 'rx', 350, 1, 'rx', 282, 1, 'gx', 319, 1, 'gx')

%     return;
%     d = d.*damage; % updating the damping to reflect the damage

%     figure(3)
%     plot(1:600, d)
%     title('Damping while simulating damage','Interpreter','latex')
%     xlabel('Oscillator number in the cochlea', 'Interpreter','
    latex')
%     ylabel('Damping value', 'Interpreter','latex')

end

if ZWEIG
     % implementation optimized for octave:
     d        = d./epsilon;
     % resonance frequency of elements
    freq    = sqrt( s./t_0_hat^2 )/2/pi;
    % formula in Zweig's article;  gives the delays required in
       Zweig's feedback
    per     = 1.75./( freq*dt );
    % roundoff; gives the length of shift register per element
    n_per   = floor( per + 1 );
    % per gives the percentage needed to approximate contribution
       if
    % delay(i) is not integer number of sample times
    per     = n_per - per;
    per     = per(:);
    n_per   = n_per(:);
    % gives much used indexes into ybuf array (a.o. edges of shift
       registers)
    cumsum_n_per    = cumsum( n_per );
    % all N shift registers are in a single column array
    ybuf    = zeros( sum(n_per), 1, 'double' );              % In
       matlab this needs to be changed 'double'
end

% yter is used for the Zweig-related update and is not in the van
   den Raadt document
yter = zeros( N, 1);

u_tilde_log = zeros(N+1, 1);

pos_trace = zeros(length(POS), ceil(length(P_e)/2));
% loop through the simulation
if PLOT
    figure(1);
end
timestep_last = length(P_e)-2;

% LEO EDIT:
% Modifications were made to allow for arbitrary sound input
% one change is to iterate over the sound file every two sample
   points
% make sure that the sound file has a sample rate 2 times that of
   the
% sample rate of the cochlea model
% Every 2nd sample point is used as the half step for the RK4
   method
% The updates to accept arbitrary input was benchmarked for the
   linear version
% but not specifically for the non-linear version of the model
for timestep = 1:2:timestep_last

    % LEO EDIT
    % Print a progress report
    if rem( timestep, 10000 ) <=1;
        fprintf('This is timestep: %d of %d \n', timestep,
           timestep_last);
    end

    % compute Zweig update and shift the corresponding registers;
    % implementation for octave optimality; see FORTRAN code
```

```
if ZWEIG
    % weighted update of yter; the feedback term
    yter          = per.*ybuf( -1 + cumsum_n_per ) + (1 - per).*
        ybuf( cumsum_n_per );
    % shift register:  ybuf(i) = ybuf(i-1), shift(..) is very
        slow
    ybuf          = [0; ybuf(1:end-1)];   % WAITING FOR
        CONFIRMATION ON THIS
    % insert new values at 'edges' of shift registers
    ybuf( cumsum_n_per - n_per + 1 ) =  y_tilde( 2:end );
end

%compute the nonlinearity contributions
if ~LINEAR
    % nonlinearities as in FORTRAN 77
    d_prof        = dnl( u_tilde(2:end)/100, d1, d2, d3 );
    c_prof        = cnl( u_tilde(2:end)/100, d2, d4 );
end

% NOTE: For now if using NL then do not use ZWEIG
if ZWEIG
    d_prof = d1;
    c_prof = d4;
end

% compute g
% g_0 on page 24; 0  u_tilde(1) = u_0_tilde, y_tilde(1) =
    y_0_tilde
% g_i  on page 36, part with yter is only in FORTRAN 77 code;
    yter will
% remain zero
g = [ (n_t^2*Z_s/m_s)*(s_st/s_t)*(t_0_hat*u_tilde(1)+omega_rm/
    delta*t_0_hat^2*y_tilde(1)); ...
       d_prof.*d.*u_tilde(2:end)+s.*y_tilde(2:end)+c_prof.*s.*
           yter ];

% compute r
% page 26;      r_0 = r(1) ; g(1) = g_0
r        = [ - alpha_0*dx_01_tilde*(P_e(timestep) + g(1)); ...
           - alpha_x_squared.*dx_tilde^2.*g(2:end) ];
% Some modification in the FORTRAN code
r(2)     = r(2)*dx_01_tilde/dx_tilde;
if VSOM_CORR
    Vsom = sum(u_tilde(2:end));
    Vsom = Vsom * dx_tilde * x_0_hat * b;
    r(end) = r(end) + (t_0_hat*dhel/m_s)*(-1.*s_st*u_tilde(1)-
        Vsom);
end

% solve A * phi_tilde = r      % page 25
phi_tilde = invA * r;

% Fourth-order Runge-Kutta method
% (Diependaal, Duifhuis, Hoogstraten, Viergever [2]).
% In this code: u(t) = y(t), v(t) = u(t) and omega[t, y(t), u(t
    )] =
        % [ ( m_s / m_sm )*( phi_tilde(1) - g(1) - P_e(timestep) );
        % phi_tilde(2:end) - g(2:end) ].
        functie1 = [ ( m_s / m_sm )*( phi_tilde(1) - g(1) - P_e(
            timestep) );
             phi_tilde(2:end) - g(2:end) ];
        y_tilde1 = y_tilde + 0.5* dt_tilde *u_tilde;
        u_tilde1 = u_tilde + 0.5*dt_tilde*( functie1 );

        % Determine g (again)

        g = [ (n_t^2*Z_s/m_s)*(s_st/s_t)*(t_0_hat*u_tilde1(1)+omega_rm/
            delta*t_0_hat^2*y_tilde1(1)); ...
                      d_prof.*d.* u_tilde1(2:end)+s.*y_tilde1(2:end)+
                          c_prof.*s.*yter ];
        % Determine p_e (t + 0.5*dt)
%%%         p_e   = p_AM1*omega_1*sin(2*pi*f_1*t_0_hat*(t_tilde+0.5*
    dt_tilde)); %0.1*p_AM1*randn(size(t))+
%%%         P_e   = ( t_0_hat^2/y_0_hat )*( n_t * p_e / m_s ).* taper;
        % Determine r (again)
        r = [  - alpha_0*dx_01_tilde*( P_e(timestep + 1) + g(1) )   ;
            ... % EDIT By LEO
                 - alpha_x_squared.*dx_tilde^2.*g(2:end)  ];
        r(2) = r(2)*dx_01_tilde/dx_tilde;

        if VSOM_CORR
             Vsom = sum(u_tilde1(2:end));
             Vsom = Vsom * dx_tilde * x_0_hat * b;
             r(end) = r(end) + (t_0_hat*dhel/m_s)*(-1.*s_st*u_tilde1(1)-
                 Vsom);
        end
        % Calculate phi_tilde
        phi_tilde = invA * r;

        functie2 = [ ( m_s / m_sm )*( phi_tilde(1) - g(1) - P_e(
            timestep + 1) );
             phi_tilde(2:end) - g(2:end) ];
        y_tilde2 = y_tilde + 0.5*dt_tilde*u_tilde1;
        u_tilde2 = u_tilde + 0.5*dt_tilde*( functie2 );

        % Repeat this for g, r and phi_tilde
        g = [ (n_t^2*Z_s/m_s)*(s_st/s_t)*(t_0_hat*u_tilde2(1)+omega_rm/
            delta*t_0_hat^2*y_tilde2(1)); ...
                 d_prof.*d.*u_tilde2(2:end)+s.*y_tilde2(2:end)+c_prof.*s
                     .*yter ];
        r = [ - alpha_0*dx_01_tilde*( P_e(timestep + 1) + g(1) )   ;
            ... % EDIT By LEO
                 - alpha_x_squared.*dx_tilde^2.*g(2:end)    ];
        r(2) = r(2)*dx_01_tilde/dx_tilde;

        if VSOM_CORR
             Vsom = sum(u_tilde2(2:end));
             Vsom = Vsom * dx_tilde * x_0_hat * b;
             r(end) = r(end) + (t_0_hat*dhel/m_s)*(-1.*s_st*u_tilde2(1)-
                 Vsom);
        end
        phi_tilde = invA * r;
        functie3 = [ ( m_s / m_sm )*( phi_tilde(1) - g(1) - P_e(
            timestep + 1) ); % EDIT By LEO
             phi_tilde(2:end) - g(2:end) ];
        y_tilde3 = y_tilde + dt_tilde*u_tilde2;
        u_tilde3 = u_tilde + dt_tilde*( functie3 );

        % Determine p_e again (t + dt)
%%%         p_e  = p_AM1*omega_1*sin(2*pi*f_1*t_0_hat*(t_tilde+dt_tilde)
    ); %0.1*p_AM1*randn(size(t))+
%%%         P_e  = ( t_0_hat^2/y_0_hat )*( n_t * p_e / m_s ).* taper;

        g = [ (n_t^2*Z_s/m_s)*(s_st/s_t)*(t_0_hat*u_tilde3(1)+omega_rm/
            delta*t_0_hat^2*y_tilde3(1)); ...
                d_prof.*d.*u_tilde3(2:end)+s.*y_tilde3(2:end)+c_prof.*s
                    .*yter ];
        r = [  - alpha_0*dx_01_tilde*( P_e(timestep + 2) + g(1) )  ;
            ... % EDIT By LEO
                - alpha_x_squared.*dx_tilde^2.*g(2:end)   ];
        r(2) = r(2)*dx_01_tilde/dx_tilde;
        if VSOM_CORR
            Vsom = sum(u_tilde3(2:end));
            Vsom = Vsom * dx_tilde * x_0_hat * b;
            r(end) = r(end) + (t_0_hat*dhel/m_s)*(-1.*s_st*u_tilde3(1)-
                Vsom);
        end
        phi_tilde = invA * r;
        functie4 = [ ( m_s / m_sm )*( phi_tilde(1) - g(1) - P_e(
            timestep + 2) ); % EDIT By LEO
             phi_tilde(2:end) - g(2:end) ];

        % Update y and u
        y_tilde = y_tilde + (dt_tilde/6)*(u_tilde+2*u_tilde1+2*u_tilde2
            +u_tilde3);
        u_tilde = u_tilde + (dt_tilde/6)*(functie1+2*functie2+2*
            functie3+functie4);

            if (timestep/2 * dt) > 0.095
                u_tilde_log = max(u_tilde_log, abs(u_tilde)); % Mimics
                    ichout = 3 in the fortran code
            end
        % plot every TIMESTEP timesteps

        if (rem( timestep,TIMESTEP )<= 1 && PLOT)
%             plot(x_tilde, u_tilde, 'g', x_tilde, y_tilde, 'r', x_tilde
    (2:end), yter, 'k', x_tilde(2:end), zeros(1,600), 'y');
            plot(x_tilde, u_tilde, 'g', x_tilde, y_tilde, 'r');
            title(sprintf('progress %.2f steps, t_tilde = %.2f ',...
                ceil(timestep/2), t_tilde(ceil(timestep/2))));
            ylabel('Oscillator displacement / velocity (nm/ms)', '
                Interpreter', 'Latex')
            xlabel('Oscillators in scaled cochlea length', 'Interpreter
                ', 'Latex')
            legend('osc. velocity', 'osc. displacement');
        %     axis([0 1 -1 1]);
            %axis([0 0.2 -Inf Inf ]);
            %axis([0 1 0.02*min(P_e) 0.02*max(P_e) ]);
      %       axis([0 1 -0.5 0.5 ]);
            axis([0 1 -0.00005 0.00005 ]);
            drawnow;
        end
        % The oscillator trace that this function currently returns is
           the position
        % of the basilar membrane not the velocity. This seemed to give
            better results
        % for simulating hearing loss than did velocity
        % Typically standard cochlea model output is the velocity of
           the cochlear partition
        % not the position
        pos_trace(:, floor(timestep/2)+1) = y_tilde(POS);
    end %for

    disp( 'end' );
        % the following code is used to benchmark the model
%    plot(u_tilde_log)
%    save('u_tilde_log_updated_code_clean_start_up', 'u_tilde_log');
end
```

Bibliography

[1] Jon P. Coulter. Sudden sensorineural hearing loss.

[2] Hendrikus Duifhuis. *Cochlear Mechanics, Introduction to a TIme Domain Analysis of the Nonlinear Cochlea*. Springer.

[3] Bastian Epp, Jesko L. Verhey, and Manfred Mauermann. Modeling cochlear dynamics: Interrelation between cochlea mechanics and psychoacousticsa). *The Journal of the Acoustical Society of America*, 128(4), 2010.

[4] Donald D. Greenwood. A cochlear frequency-position function for several species29 years later. *The Journal of the Acoustical Society of America*, 87(6), 1990.

[5] D. T. Kemp. Stimulated acoustic emissions from within the human auditory system. *The Journal of the Acoustical Society of America*, 64(5), 1978.

[6] Sharon G Kujawa and M Charles Liberman. Adding insult to injury: Cochlear nerve degeneration after "temporary" noise-induced hearing loss. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 29(45):14077–14085, November 2009.

[7] Enrique A Lopez-Poveda and Pablo Barrios. Perception of stochastically undersampled sound waveforms: A model of auditory deafferentation. *Frontiers in Neuroscience*, 7(124), 2013. ISSN 1662-453X. doi: 10.3389/fnins.2013.00124. URL `http://www.frontiersin.org/auditory_cognitive_neuroscience/10.3389/fnins.2013.00124/abstract`.

[8] Chadi A Makary, Jennifer Shin, Sharon G Kujawa, M Charles Liberman, and Saumil N Merchant. Age-related primary cochlear neuronal degeneration in human temporal bones. *Journal of the Association for Research in Otolaryngology*, 12(6):711–717, July 2011.

[9] Manfred Mauermann, Stefan Uppenkamp, Peter W. J. van Hengel, and Birger Kollmeier. Evidence for the distortion product frequency place as a source of distortion product otoacoustic emission (dpoae) fine structure in humans. i. fine structure and higher-order dpoae as a function of the frequency ratio f2/f1. *The Journal of the Acoustical Society of America*, 106(6), 1999.

[10] Naoki Oishi and Jochen Schacht. Emerging treatments for noise-induced hearing loss. *Expert Opinion on Emerging Drugs*, 16(2):235–245, June 2011.

[11] World Health Organization. 1.1 billion people at risk of hearing loss.

[12] B.C.J. Moore Professor, M. Huss, D. A. Vickers, B. R. Glasberg, and J.I. Alcántara. A test for the diagnosis of dead regions in the cochlea. *British Journal of Audiology*, 34(4):205–224, 2000. doi: 10.3109/03005364000000131. URL `http://www.tandfonline.com/doi/abs/10.3109/03005364000000131`. PMID: 10997450.

[13] Graham Road Surgery. Ear syringing.

[14] Carrick L. Talmadge, Arnold Tubis, Glenis R. Long, and Pawel Piskorski. Modeling otoacoustic emission and hearing threshold fine structures. *The Journal of the Acoustical Society of America*, 104(3), 1998.

[15] P. W. J. van Hengel, H. Duifhuis, and M. P. M. G. van den Raadt. Spatial periodicity in the cochlea: The result of interaction of spontaneous emissions? *The Journal of the Acoustical Society of America*, 99(6), 1996.

[16] Peter van Hengel. Private communication.