NTNU

Norwegian University of
Science and Technology

Edge Diffraction Implementation by Semi-Transparent Surfaces in Geometrical Room Acoustics

Anders Kristoffer Isebakke

Master of Science in Electronics
Submission date: May 2011
Supervisor: Peter Svensson, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

In room acoustic softwares such as CATT-Acoustic and Odeon, geometrical room acoustics is the basis for prediction of the sound field calculations. However, a great disadvantage of this method is that geometrical room acoustics is not able to handle diffraction effects. Further, extended methods, still based on geometrical room acoustic assumptions, have been developed to successfully include diffraction. However, the subsequent calculations have been found hard to implement in existing room acoustic software. This thesis should study alternative ways of modelling diffraction, which somehow could be fairly easily-implemented in CATT-Acoustic.

Assignment given: February 2011
Supervisor: Peter Svensson, IET

Abstract

This report presents a potential method to efficiently implement edge diffraction from a noise barrier into geometrical room acoustic softwares. The modelling is based on semi-transparent surfaces, and the classic digital signal processing multipath transmission equation has been employed to describe in a mathematical term the presented method. The basic idea is to subdivide the noise barrier into a number of subareas, and then give each subarea an optimalized transmission coefficient for building the best possible output impulse response.

To evaluate the proposed semi-transparent modelling, a Matlab simulation model of an infintite noise barrier case has been developed, and the corresponding simulations have been compared with the ideally correct solution. Accordingly, it is stated that there seems to be a clear positive potential in the proposed modelling technique. However, the results also reveal a somewhat instability in the modelling, which is expected to appear mainly for rare critical source and receiver positions.

A main goal has been to develop a method that can easily be implemented in the existing calculation algorithms of today's commercial software developers. For verification, the proposed modelling has by discussion been associated with the often employed diffuse rain method. However, since no true implemenation in geometrical room acoustic software has been performed, further studies are required.

To maintain efficiency and reliability, another desired outcome of the presented modelling has been that is should function for a general one-to-all source-receiver condition. Surely, the modelling seems to give fairly good results for symmetric source and receiver positions, but as the receiver is moved away from these symmetric conditions some unwanted errors occur, especially at higher frequencies.

Main focus has been given to receiver positions located in the shadow zone, but some simulations and discussion has also been given to receiver positions located near the source-receiver sight line - at where direct sound energy contributions are also included and an interference pattern arises. To cope with this interference

pattern, a polarity shift is proposed, which gives a clear improvement at low frequencies.

One certainly interesting feature of the presented modelling technique is that it involves a broadband-based simulation method, which means that it gives the full frequency response by running only one simulation. Indeed, this is advantageous regarding calculation efficiency, but it does however also introduce some issues regarding a potential future software implementation - as the common case in geometrical room acoustics is to run individual octaveband-based simulations.

Contents

Chapter 1

Introduction

This chapter starts with a review on the background and motivation for the research presented in this report. Further, it gives a formal description of the current task, and argues for the main focus of the present research process. Finally, this chapter presents the outline for the rest of this report.

1.1 Background

Indeed, the noise barrier edge diffraction case this is a very classic acoustic situation. In room acoustics, such noise barriers are common features in e.g. office landscapes and classroom environments - as an attempt to devide a room into a number of partly separated zones. Our daily experience reveals that such noise barriers aren't necessarily as acoustically efficient as desired, as a large amount of the sound energy tends to leak over the barrier due to edge diffraction. Diffraction is one of the most well-known qualities of the wave nature, and as acoustic waves involves fairly large wavelengths compared with potentially obstructing objects (such as noise barriers), the edge diffraction topic is highly relevant when studying noise reduction cases.

From an acoustic consultant's point of view it would come in handy to own software tools that can deal with such enclosed room edge diffraction cases. *Finite-element method*-based softwares often provide very accurate results in any type of acoustic situation. However, when dealing with three-dimensional enclosed spaces these calulations get way too complex and time-consuming - even for today's powerful computers. Fortunately, *geometrical room acoustics*-based softwares is an alternative, much more efficient method for solving the wave equation in enclosed spaces. The disadvantage of geometrical room acoustics is that it has

some limitations - among them not being able to handle diffraction phenomena in a proper way. Consequently, geometrical room acoustic simulations will typically return a sound pressure amplitude equal to zero for receivers locateded in the shadow zone of the discussed noise barrier edge diffraction case.

Previous studies have been published regarding the issue on how edge diffraction can be implemented in geometrical room acoustics. In 1999, *Svensson, Fred & Vanderkooy* [1] presented a mathematical solution that calculates the impulse response for an perfectly rigid edge diffraction case based on geometrical room acoustic assumptions. Their concept is based on the employment of a secondary edge source, which agrees with Huygen's wave propagation principle. However, due to an introduced complexity in the consecutive calculation algorithm, the method has been found inconvenient to implement in existing commercial room acoustic softwares.

Further, in *Dammerud's* [2] PhD thesis a smart and easily implemented trick was introduced in order to mimic the diffraction phenomena of sound propagation through a symphony orchestra. A symphony orchestra includes a large number of sound sources and obstructing objects - hence a very complex acoustic case. Briefly, diffraction was here implemented by claiming that the sound waves propagates *through* the obstucting objects rather than around them. In principle, this means that an assumption was made that *edge diffraction is a quality of the barrier* rather than a quality of the wave nature. Indeed, this is easy to imagine for very small obstructing objects, as they will appear "invisible" for frequencies of somewhat much longer wavelengths. However, Dammerud's study indicated that the method also seemed to function for obstructing objects of a somewhat larger size. Even though such a mimetic edge diffraction implementation of course will introduce some errors compared to the physical nature of wave propagation, the method is still interesting and valuable, both because it is very efficient, and because too accurate results often is outside the scope of interest when dealing with room acoustics.

Moreover, a research by *Isebakke* [3] studied whether the semi-transparent diffraction modelling technique also could be employed to simulate the locally perceived acoustic conditions in a public hall audience seating area. Clearly, such a complex geometrical case will be affected by diffraction effects. First, the seat benches can be regarded as a line of obstructing objects. In addition, a characteristic *seat dip* [4, 5, 6] diffraction effect is often found. Isebakke's research also revealed somewhat convincing results, and consequently it was ensured a general potential in the semi-transparent diffraction modelling technique.

However, the drawback in both *Dammerud* [2] and *Isebakke* [3] is that their results were achieved in a somewhat tentatively proceeding form of research, where available parameters in the simulations were utilized in order to optimally

tune the model relative to a measurement session of the same acoustic case. Accordingly, it would at this point be interesting to obtain a more in-depth mathematical linkage between the physical edge diffraction behavior and the available semi-transparent simulation parameters. The trend in the previous studies is that the sound propagation is obstructed in a very complex way, which makes it difficult to derive simple relationships. Hence, it could be interesting to study a simpler diffraction case, such as the simple noise barrier, in search for a rational design procedure for the introduced semi-transparent surface property.

1.2 Current task description

The superior intention of the presented research is to investigate whether the simple noise barrier edge diffraction case can be implemented in geometrical room acoustic softwares by using semi-transparent surfaces. A somewhat theoretical approach to the developed modelling is desired. Further, a vital goal will be to develop a method that easily can fit into the already existing calculation algorithms of today's commercial geometrical room acoustic softwares. Accordingly, it will also be of interest to obtain a result that function for a general one-to-all source-receiver condition.

Main foucs will be given to receiver positions located in the shadow zone - i.e. positions where the direct sound contributions are obstructed by the noise barrier. In the shadow zone, it will be assumed that *only* edge diffraction energy contributions appear at the receiver. Still, some simulations and discussion will also be presented regarding receiver positions located around the limit of the source-receiver sight line.

The research process will consider the frequency range within the $63 - 8000Hz$ octavebands, as this is the frequency range in which edge diffraction leakage is most relevant. Of course, frequencies even further down in the frequency range will also be diffracted over the edge, but due to the general limitations in geometrical room acoustics, these lowest frequencies have been ignored.

The research will be carried out by developing a geometrical room acoustic Matlab simulation model of the noise barrier edge diffraction case. In order to evaluate the research outcome, the *Edge diffraction Matlab toolbox* [7] will be employed as the reference solution.

1.3 Outline

Chapter 2 gives a detailed review on the theory and methods that have been used in the research process. Hence, the chapter includes all the required acknowledgements in order to fully grasp the content of the research process and further discussion. The final proposed semi-transparent noise barrier modelling is described in detail. Chapter 3 gives the achieved results, presented as octaveband energy level errors relative to the Edge diffraction Matlab toolbox. Chapter 4 gives an extensive discussion on the results, with main focus on validation of the proposed modelling. Possible improvements and practical software implementation issues are also discussed. Finally, Chapter 5 contains the overall conclusion. In the Appendix all belonging CATT-Acoustic and Matlab files can be found.

Chapter 2

Methods

This chapter describes the methods and theory that has been used, and is essential reading in order to fully grasp the content of this report. A detailed review is given on the proposed semi-transparent noise barrier modelling, including the basic idea, mathematical terms and final design. Also, this chapter accounts and argues for all assumptions that have been made.

2.1 Brief review on geometrical room acoustics

Geometrical room acoustics is a software tool that has been developed in order to obtain efficient simulations in room acoustic cases. It has the advantages of being both fast and powerful, and is often capable to bring out somewhat satisfying results for complicated enclosed room shapes. In 1968, pioneer research was performed by *Krokstad, Strøm & Sørsdal* [8]. Today, *CATT-Acoustic*, *Odeon* and *Ease* are some of the most well-known commercial developers.

Briefly, geometrical room acoustics solves the wave equation based on the assumption that all room modes can be ignored - i.e. that all frequencies of current interest lie above the Schroeder frequency. This assumption can only be justified if the dimensions of the room are large compared with the wavelength of the sound. Accordingly, the assumption may cause a somewhat incorrect result at lower frequencies.

There are in principle two different calculation methods being used in geometrical room acoustic softwares: the *ray-tracing method* and the *image-source method*. The ray-tracing method will be the employed platform for the presented modelling in this report. In short, ray-tracing regards sound propagation as rays

travelling normal to the wave front. This is implemented by a defined point source that sends out a large number of rays distributed in all directions. Each ray then represents a certain angle of a spherical wave. Subsequently, all rays are traced throughout their travel in the room - until all energy has died out. For non-rigid surfaces a mixture of specular reflections (the angle of the reflected wave is found by Snell's law) and diffuse reflections (the angle of the reflected wave is random) can be found. This decision is implemented by a dedicated surface *scattering coefficient* set between 0% and 100% with a corresponding number of rays speculary and randomly reflected. Finally, a receiver of a given spherical expansion is employed to ensure that a certain stochastic number of rays actually do hit the receiver. Apparently, it is profitable to employ a fairly large number of rays to reduce stochastic variation.

A simplified way to describe how the ray-tracing method builds an impulse response can be given by the classic digital signal processing *multipath transmission* equation:

$$h(n) = \sum_{k=1}^{K} c_k \delta(n - n_k) \tag{2.1}$$

where K is the number of rays sent out by the receiver, c_k is the energy contribution of ray k as it strikes the receiver, and n_k is the sample number for when ray k strikes the receiver. For a source placed in an enclosed space the ray energy contribution c_k is mainly affected by *spherical propagation damping* and *surface absorption* due to surface hits.

Note that eq. (2.1) is not a totally correct mathematical representation of the classic ray-tracing technique, as it is not for granted that each ray will strike the receiver. However, an improved method called *diffuse rain* ray-tracing (by *Heinz* [9]) has been developed to ensure that each ray indeed will contribute in the consecutive impulse response. In short, diffuse rain replaces the standard method of checking each ray for hits with the receiver sphere during the ray's propagation. Instead, for each ray's wall hit, a separate wall-to-receiver ray is constructed which makes sure that every single wall hit will generate a subsequent hit at the receiver (as long as the wall hit point has an unobstructed path to the receiver point). This separate ray is valid for diffusely reflecting wall's and hence the name of the method.

As mentioned, geometrical room acoustics is not able to handle diffraction phenomena. When sound waves are regarded as rays the phase quality of the wave nature is ignored, which often results in low frequency errors as interference patterns often occur and affect this frequency region.

Dammerud [2], in his mimetic semi-transparent diffraction modelling, took advantage of an octaveband-dependent *semi-transparent* quality that geometrical room acoustic surfaces can be dedicated. According to *CATT-Acoustic user's manual* [10, 11], semi-transparent surfaces are implemented in the software by introducing a *transparency coefficient* τ, being a property of the surface, defined specificly for each octaveband. The range of τ goes from $\tau = 0$ (zero transparency, only reflected sound and absorption) to $\tau = 1$ (full transparency, no reflected sound). The direct sound goes deterministically through a maximum of one semi-transparent surface and is then attenuated by $(1 - \alpha)\tau$. Higher order transmission is random depending on the transmission coefficient.

For further insight on geometrical room acoustics: check out Chapter 3 and 4 in *Kuttruff: Room acoustics* [12] and Chapter 4 in *Vigran: Bygningsakustikk* [13].

2.2 Infinite noise barrier edge diffraction

In the presented study a *hard rigid infinite noise barrier on a totally absorptive floor and otherwise free-field conditions* has been studied. A simple geometrical cross-section model of the given case can be seen in Figure 2.1. It is highly recommended to give this model a close look, as the presented variables and their notations will be frequently referred to in this report.

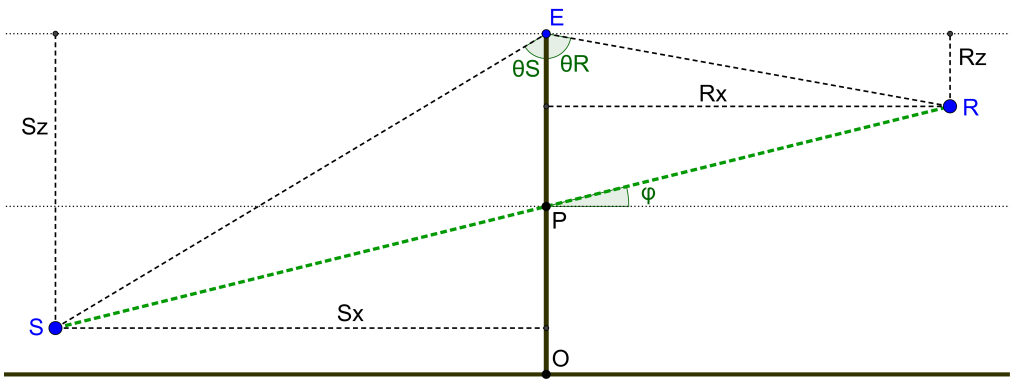

Figure 2.1: *The infinite noise barrier geometry*

A typical impulse response of the edge diffraction shadow zone case can be seen in Figure 2.2, and the consecutive frequency response can be seen in Figure 2.3. Note that the frequency response slope appears to decrease for an increased frequency. This means that the noise barrier behaves like a typical lowpass filter, which indeed agrees with our daily experience.

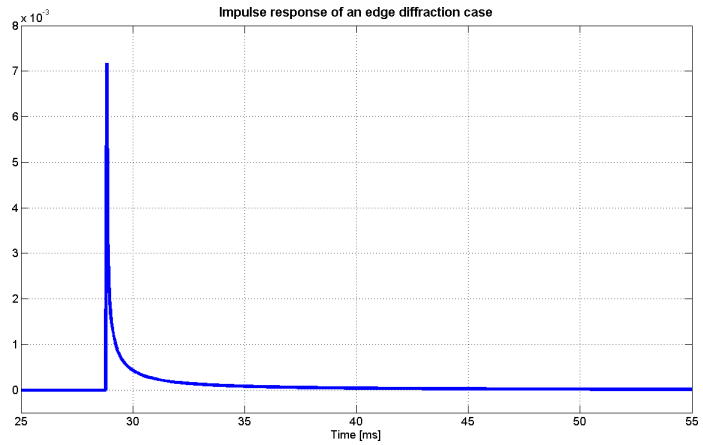

Figure 2.2: *Typical impulse response of the edge diffraction case*

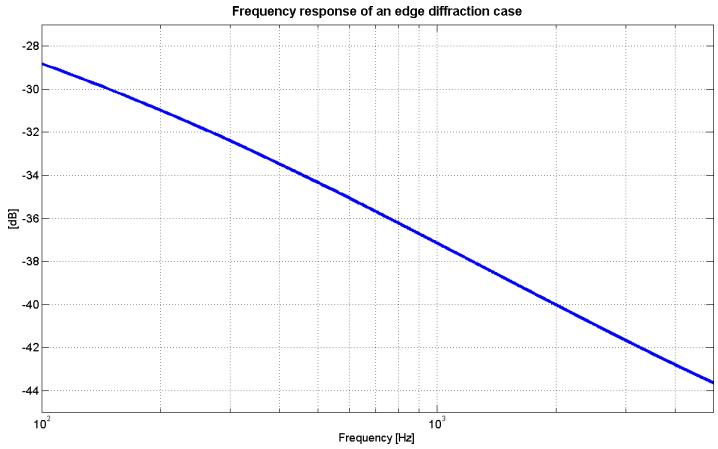

Figure 2.3: *Typical frequency response of the edge diffraction case*

2.3 Proposed semi-transparent modelling technique

In order to easily implement noise barrrier edge diffraction in geometrical room acoustic softwares, the most efficient solution would be to simply take advantage of those properties that already exist in the software. Accordingly, it would be highly profitable to develop an approach that fits right into eq. (2.1).

As described in Chapter 1.1, the assumption that edge diffraction is a quality of the barrier rather than a quality of the wave nature is now implemented. Accordingly, it is claimed that no sound energy is radiated from a secondary edge source, and instead that the barrier is semi-transparent - i.e. that some sound energy is radiated *through* the barrier. In principle, this means that the physical properties of the barrier is manipulated so that it behaves more in the sense of a digital signal processing filter than in the genuine sense of a hard rigid infinite barrier. Of course, this is not true by nature, but it could somehow be justified in the virtual dimensions of the simulation software - as long as it gives satisfying results.

2.3.1 Semi-transparent division

In order for the introduced approach to function, it is essential that the noise barrier is given a somewhat functional semi-transparent design. One possible action is then to divide the total area of the noise barrier surface S_{tot} into a number of subareas, and give each subarea a suitable transmission coefficient. For simplicity, it is assumed that each subarea dS_k will be hit by *one* ray k and contribute with *one* certain amount of energy c_k at the receiver position. Apparently, this assumption can be justified by the mentioned diffuse rain algorithm of geometrical room acoustic softwares. Strictly, this diffuse rain approvement is neither required, as the stochastic differences compared with classic ray-tracing will dimish as the number of rays is increased. The arrival time n_k of ray k will be given by the sum of the distance between the source S and the barrier hit P_k and the distance between the barrier hit to the receiver R. Consequently, the resulting impulse response will be given directly by eq. (2.1).

To grasp a simple understanding of the concept, Figure 2.4 gives a helpful illustration on this type of division. The barrier is here divided into a number of equally large rectangular subareas. Due to the simple geometry of this rectangular tessellation pattern, this will also be the employed type of division for the final noise barrier design presented in this report.

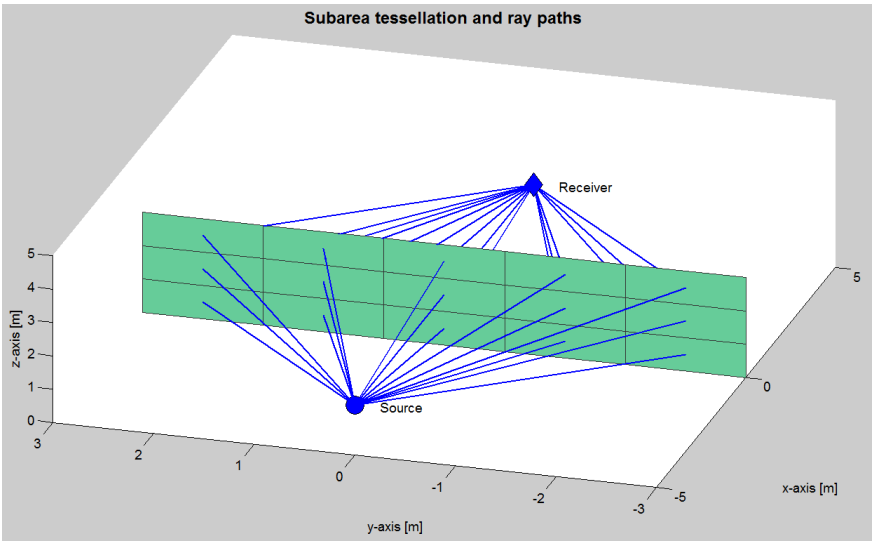

Figure 2.4: *Ray paths of the proposed model*

For each ray k, the value c_k in eq. (2.1) will be given by the following formula:

$$c_k = \frac{1}{SP_k \cdot RP_k} \cdot \tau_k \cdot s_k \cdot dS_k \tag{2.2}$$

where τ_k is the transmission coefficient of the subarea dS_k that ray k propagates through, and s_k is a scattering coefficient out from the noise barrier. These two parameters will be further described in Chapter 2.3.3 and 2.3.4. SP_k is the length of the propagation path between the source and the subarea, and RP_k is the length of the propagation path between the subarea and the receiver.

Note, that the application of eq. (2.2) somehow corresponds more in the sense of a Rayleigh integral than in the sense of classic ray-tracing, as the equation represents the *sound pressure* rather than the *squared sound pressure*. However, the classic *squared impulse response as primary quantity* will be further described in Chaper 2.5.

The value n_k in eq. (2.1) will be given by the following formula:

$$n_k = \frac{f_s}{c_{air}} \cdot (SP_k + RP_k) \tag{2.3}$$

where f_s is the sample frequency, and c_{air} is the speed of sound.

Now, in theory it should be possible to create any type of impulse response as

long as the number of subareas is large enough. As the noise barrier is devided into an increased number of subareas, the travel length difference between two succeeding incomming rays will naturally decrease. Therefore, as the number of subareas grows towards infinity, a number rays will even arrive within the same sample, which means that every sample n in the impulse response $h(n)$ will hold a certain number of ray energy contributions. Accordingly, the impulse response values can be optimally tuned as the subareas are dedicated optimal transmission coefficients τ_k.

2.3.2 *n*th sample barrier intersection area

To give the parameter τ_k optimalized values, there will be a vital task to map all rays/subareas that contribute within the same sample. These will be bounded by a certain barrier projection, centered by the barrier intersection of the direct sound propagation path SR. To grasp a deeper understandig, Figure 2.5 illustrates the nth sample's barrier relationship.

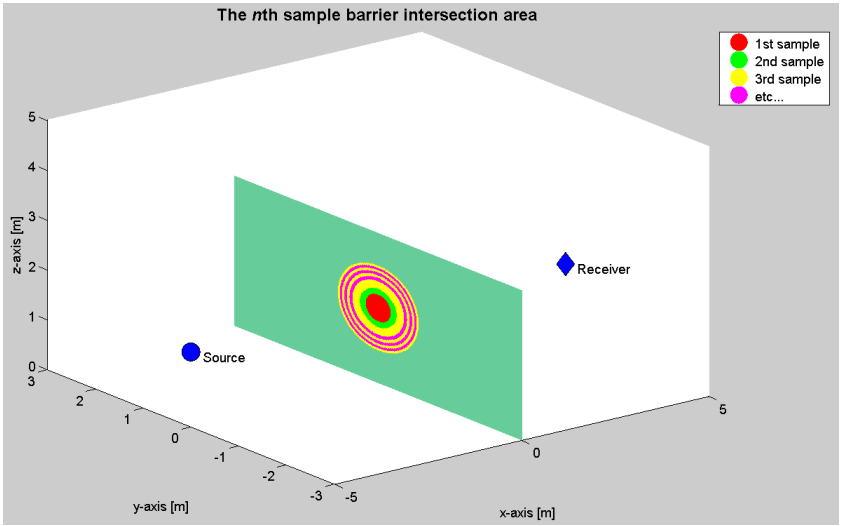

Figure 2.5: *The nth sample barrier intersection area*

In theory, the exact shape and size of each nth sample barrier intersection area will be bounded by an elliptic radiation pattern quite similar to a *Fresnel zone* (commonly referred to in optics and radio communications), which in this particular case will be given by the source position $\{S_x, S_y, S_z\}$ and the receiver position $\{R_x, R_y, R_z\}$, as well as the sample frequency f_s and the speed of sound c_{air}. A cross-section of this elliptic zone can be seen in Figure 2.6.

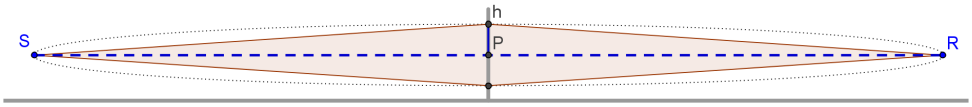

Figure 2.6: *Cross-section of nth sample barrier intersection area*

For simplicity, it is now assumed that the nth sample barrier intersection is a circular desk. In reality, this is only the true case for source and receiver positions that are symmetric in proportion to both the y-axis and the z-axis. Still, for relatively large $S_x + R_x$ it will be a fairly good assumption. Be aware, however, that it may introduce some errors if a small $S_x + R_x$ is combined with a comparatively large $S_y - R_y$ and/or $S_z - R_z$.

By employing the Pythagorean Theorem to the cross-section in Figure 2.6, the nth sample barrier intersection radius Ph_n can be expressed by the following relationship:

$$\sqrt{SP^2 + {Ph_n}^2} + \sqrt{RP^2 + {Ph_n}^2} - SR = \frac{c_{air}}{f_s} \cdot n \tag{2.4}$$

Since the set $X = \{SP, RP\} \gg Ph_n$, the following Taylor/Maclaurin series approximation [14] can be employed:

$$X\sqrt{1 + \left(\frac{Ph_n}{X}\right)^2} \approx X\left(1 + \frac{1}{2}\left(\frac{Ph_n}{X}\right)^2\right) \tag{2.5}$$

Accordingly, eq. (2.4) can be re-writttten into the following expression:

$$Ph_n \approx \sqrt{2 \cdot \frac{SP \cdot RP}{SR} \cdot \frac{c_{air}}{f_s} \cdot n} \tag{2.6}$$

The nth sample barrier intersection area will then be found by the following equation:

$$S_n = \left(\pi (Ph_n)^2 \cdot trunc_n\right) - \sum_{i=1}^{n-1} S_i \tag{2.7}$$

where $trunc_n \in [0, 1]$ is a *truncation factor* that appears because the radius Ph_n at some point may exceed the fixed z-axis size of the noise barrier. This will typically take place for somewhat distant source and/or receiver positions. As truncation finds place, the region S_n will appear as an area that consists of a circle sector S_{sec} and an isosceles triangle S_{tri}, as illustrated in Figure 2.7.

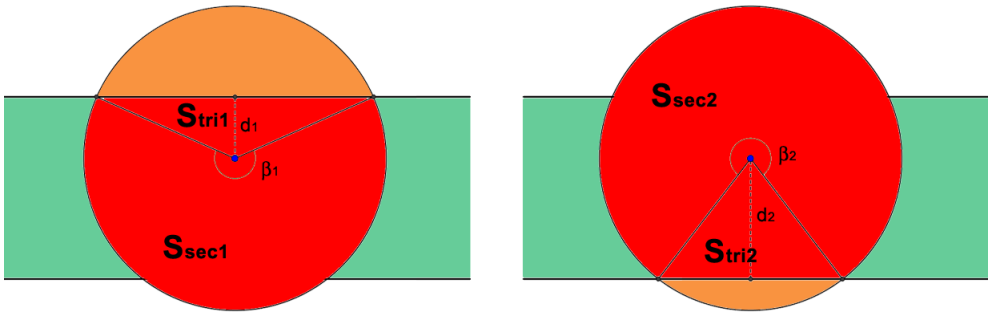

Figure 2.7: *The truncated nth sample intersection area*

Note that the n-notation in S_n temporary has been removed. This is to avoid confusion between *edge vs. floor truncation sectors in Figure 2.7* and *the sample parameter n*.

The angles β_1 and β_2 of the circle sector can easily be found by the following equations:

$$\beta_1 = 2\left(\pi - \cos^{-1}\left(\frac{d_1}{Ph}\right)\right) \quad , \quad d_1 = Ph - PE \tag{2.8}$$

$$\beta_2 = 2\left(\pi - \cos^{-1}\left(\frac{d_2}{Ph}\right)\right) \quad , \quad d_2 = Ph - PO \tag{2.9}$$

Hence, the area of the two truncated circle sectors in Figure 2.7 will be given by the following equations:

$$S_{sec1} + S_{tri1} = \beta_1 \frac{Ph^2}{2} + d_1\sqrt{Ph^2 - {d_1}^2} \tag{2.10}$$

$$S_{sec2} + S_{tri2} = \beta_2 \frac{Ph^2}{2} + d_2\sqrt{Ph^2 - {d_2}^2} \tag{2.11}$$

Accordingly, the truncation factor *trunc* will be given by the following equation:

$$trunc = \frac{S_{sec1} + S_{tri1} + S_{sec2} + S_{tri2} - \pi(Ph)^2}{\pi(Ph)^2} \tag{2.12}$$

2.3.3 Transmission coefficient

At this moment, the model will be considered by an *one-to-one* source-receiver condition. Later, the desired, more complex general *one-to-all* source-receiver condition will be considered. For now, also assume that the scattering coefficient of eq. (2.2) is neglected by a constant value $s_k = 1$.

As mentioned, a major challenge in the present reseach process will be to dedicate optimal transmission coefficients τ_k to each subarea dS_k. In order to build a correct impulse response, the idea is then that the sum of all rays contributions c_k within a sample n should result in a value that corresponds to the true impulse response value $h_{ref}(n)$. Accordingly, an equation that gives the transmission factor for the true impulse response $h_{ref}(n)$ being considered as a semi-transparent noise barrier is needed:

$$T_n = \frac{SP \cdot RP}{S_n} \cdot h_{ref}(n) \tag{2.13}$$

where S_n is the nth sample barrier intersection area given by eq. (2.7).

Further, for the simulated impulse response $h(n)$ it would now be appropriate to claim that all ray contributions that arrive within the same sample have an equal transmission coefficient τ_k. This means that all subareas that contribute within the nth sample can be dedicated the following transmission coefficient:

$$\tau_k = T_n \iff n_k = n \tag{2.14}$$

By introducing this relationship, there should in theory be possible to build a perfect impulse response $h(n) = h_{ref}(n)$ regarding a one-to-one source-receiver condition. However, things get more complicated as the desired multiple receivers condition is introduced in the next section.

2.3.4 Scattering coefficient

As mentioned, one of the main goals in this study is to maintain the efficient general one-to-all source-receiver quality of geometrical room acoustic softwares.

A first step towards this achievement would be to claim that *the ray energy is scattered out from the noise barrier* - i.e. that the energy out from each subarea will contribute at each potential receiver position. Apparently, this assumption can be justified by the mentioned diffuse rain method. Hence, the scattering coefficient s_k in eq. (2.2) is introduced.

For simplicity, it would be favourable to assume a simple cosinus-relatied scattering pattern:

$$s_k = \cos \phi_k \tag{2.15}$$

where ϕ_k is the ray k's angle out from the subarea dS_k.

The drawback of the introduced scattering is that further attention must be given to the semi-transparent design of the noise barrier. Indeed, it must be stated that it is a complex case to make a noise barrier modelling design that works fairly well at every receiver position. Still, a proposed modelling will be presented in the next section.

2.3.5 Final semi-transparent noise barrier design

To create a functional multiple receivers situation, the employed idea is to allow a small general error ahead of potentionally very large errors at certain distinct receiver positions. Accordingly, the following semi-transparent noise barrier design procedure is proposed:

1. A simple source and an array of receivers are employed to dedicate transmission coeffients. These source and receiver objects, S_G and R_G, will now be regarded as the noise barrier's "modelling generator" - almost like a building kit. Originally, the intention of the introduced building kit was that the barrier modelling should be adjusted only as a function of the source position S_G, and that the consecutive model should give acceptable results for all employed receiver positions. However, the receiver array R_G must also be placed somewhere along the x-axis, and accordingly the results will also vary to some extent as a function of R_G.

S_G is a source randomly placed within the intervals $S_x = [-30.0, -1.0]m$ and $S_z = [0.25, 2.75]m$. R_G is a receiver array of nine units along the z-axis, equally distributed within the interval $R_z = [0.5, 2.5]\,m$, placed at a random position within the interval $R_x = [1.0, 30.0]m$. S_G and R_G are placed symmetric regarding the y-axis.

2. The noise barrier is devided into two different zones, named the *mid/high peak zone* and the *bass tail zone*. As known from *quasi-anechoic recording techniques* [15, 16, 17], a Fourier transform of the initital samples of a somewhat time-smeared impulse response will only give a complete energy representation for relatively high frequencies. Evidently, this can be associated with the ray contributions that arrive earliest at the receiver. Consequently, it can also be claimed that all ray contributions that arrives somewhat further out in the impulse response will only perform the accomplishment of "boosting" the lower frequencies to a more and more correct representation. Based on this, the utilized idea is that the mid/high peak zone - represented by the ray contributions that arrive very early at the receiver, should give a somewhat flat frequency response level, while the bass tail zone - represented by the more delayed ray contributions, should give the frequency response the more typical shape of a lowpass filter. Hence the adopted names. To illustrate this proposed semi-transparent modelling concept, Figure 2.8 gives a sketch of the zone division.

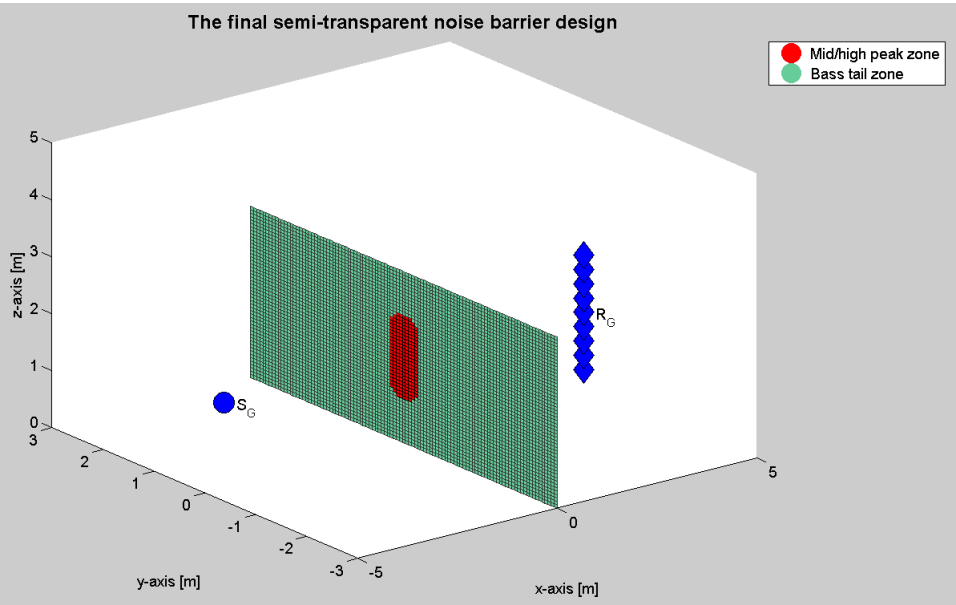

Figure 2.8: *The final semi-transparent noise barrier design*

Further, Figure 2.9 gives a rough sketch of the concept of an impulse response being shaped by the two transmission zones. Note however, that there in practice will be a much more blurred overlap in the transition between the two zone's impulse response contributions, and that the final curve shape will be expected somewhat less smooth.

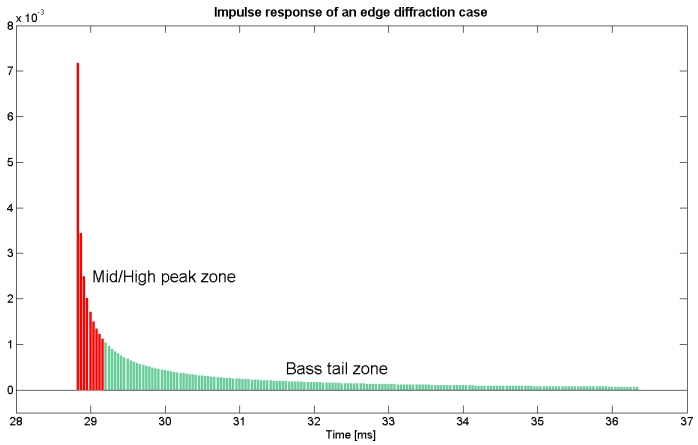

Figure 2.9: *Ideal impulse response build by two transmission zones*

3. The mid/high peak zone: is bounded by the union of the 1st sample barrier intersection areas of all the nine receivers in R_G. Further, the transmission factor of the impulse response peak value T_1 is found at each receiver. Then, a *1st order polynomial fit* is given to the resulting array of transmission factors T_1, using the Matlab function **polyfit()**. Subsequently, the subareas included in the mid/high peak zone are given transmission coefficients depending on their location along the z-axis. By such a transmission coefficient distribution, the property that receivers located closer to the limit of the shadow zone should hold a somewhat larger portion of mid/high frequency energy, was to some extent considered treated. Also, this difference will be expected to smear out for a more distant receiver location along the x-axis.

4. The bass tail zone: involves all subareas of the noise barrier that are not occupied by the mid/high peak zone. The bass tail zone is given a *constant transmission coefficient* which is found by the mean value of the array of nine transmission factors T_1 multiplied by a factor 0.1. The mean value is found using the Matlab function **mean()**. The intention is that the continous damped tail of the impulse response should appear due to the steadily increased spherical propagation damping of succeeding rays, in addition to an increased scattering influence.

2.4 Polarity error

Eventually, when studying a noise barrier and moving the receiver closer to the source-receiver sight line, direct sound contributions will appear in the frequency response. As described in Chapter 2 in *Kuttruff: Room acoustics* [12], an inter-

ference pattern will occur due to the opposite polarities of the direct sound and the diffracted sound. This intereference effect will typically be strongest around the source-receiver sight line, where both direct sound and diffracted sound contain much energy. For further insight, a study by *Lokki, Svensson & Savioja* [18] gives an example that illustrates this interference effect. Note, that this study at the same time proves that the edge diffraction solution by *Svensson, Fred & Vanderkooy* [1] is capable to bring out a fairly correct representation of the given case.

Unfortunately, when dealing with classic geometrical room acoustic softwares, such a polarity shift will not be displayed, since the rays don't consider the wave's phase property. As described in Chapter 2.1, this assumption is fair for high frequencies, but it also means that some low frequent errors may occur.

In order to check how crucial this polarity error is, a polarity-check was implemented in the Matlab code. The polarity-check simply checks whether direct sound contributions are included in the impulse response, and does in that case make sure that the diffracted sound contribution is multiplied by a factor -1. The succeeding results are presented in Chapter 3.2.

2.5 Squared impulse response as primary quantity

The fact that interference phenomena are neglected in geometrical room acoustics, means in practice that the squared impulse response could be considered as the primary quantity, instead of the impulse response. In fact, this also seems to be the common case in commercial geometrical room acoustic softwares. Most often, this squared impulse response quantity is referred to as the *echogram*.

As a consequence, in a potential implementation of the proposed semi-transparent noise barrier modelling, the T_n and τ_k values in eq. (2.13) and eq. (2.14) may have to be re-written into the following echogram form:

$$T'_n = \frac{SP^2 \cdot RP^2}{S_n} \cdot h_{ref}(n)^2 \tag{2.16}$$

$$\tau'_k = T'_n \iff n_k = n \tag{2.17}$$

which leads to the re-written c_k values:

$$c'_k = \frac{1}{SP^2 \cdot RP^2} \cdot \tau'_k \cdot s'_k \cdot dS_k \tag{2.18}$$

and finally the succeeding re-written impulse response:

$$h(n)' = \sqrt{\sum_{k=1}^{K} c'_k \delta(n - n_k)} \tag{2.19}$$

In order to review the scope of the introduced differences by the alternative echogram quantity, a plot of the given case is presented in Chapter 3.3.

2.6 Reference model: Edge diffraction Matlab toolbox

As a reference model the **EDBtoolbox** [7] (developed by the Acoustics Group, NTNU) was employed. This toolbox is a set of Matlab functions that gives the impulse response for a point source in an environment of rigid, plane surfaces. The calculations are based on the secondary source method by *Svensson, Fred & Vanderkooy* [1], and accordingly the toolbox is supposed to give a correct result regarding the rigid infinite noise barrier case. The EDBtoolbox performs the calculations based on a **.cad** geometry file (which easily can be created using CATT-Acoustic) and two vectors/matrices of employed sources and receivers. All adopted sources and receivers are omni-directive.

As it is impossible to create an infinte noise barrier in CATT-Acoustic, the employed reference model **.cad** geometry file contained a $120m$ wide noise barrier. However, the EDBtoolbox includes a smart feature that gives the opportunity to exclude edge diffraction contributions from all edges that are built by corners with corner numbers higher than a specified value. Therefore, the outcome will in fact give solutions for a true infinite noise barrier.

To work with a Matlab toolbox as a reference model, rather than e.g. an acoustic measurement session, was considered appropriate because of the fast procedure and flexible possibilities to quickly make adjustments.

2.7 Validation of the further presented results

A sample frequency $f_s = 24000Hz$ was used in all the Matlab calculations. According to Nyquist's theorem, this should correspond to a correct energy representation for all frequencies $f \leq 12000Hz$, which means that the desired $63 - 8000Hz$ octavebands are covered. The speed of sound was given the value $c_{air} = 343m/sec$.

Furthermore, a $3m$ high noise barrier was studied. Indeed, this is not the typical indoor office landscape noise barrier height. However, in order to capture a detailed review of the edge diffraction phenomena as function of the receiver height R_z, such a tall barrier was still employed.

As it is very unpractical and somehow pointless to run calculations on a truely infinte noise barrier, the barrier width of the developed semi-transparent noise barrier model was strongly truncated to a total width of $12m$. In other words, it was assumed that all subareas outside the limits $y \in [-6, 6]m$ could be given a transmission coefficient equal to zero. Hence, the total number of subareas could be set to a somewhat applicable value. By experiments, the noise barrier was finally devided into a total number of 20301 elements, 201 along the y-axis and 101 along the z-axis.

For the outcome of the simulations, it was considered appropriate to present octaveband energy level-based results. Consequently, the broadband impulse responses were octaveband-filtered by the function **oktavbandfilter.m** (developed by the Acoustics Group, NTNU). Note, that due to the increased bandwidth of an increased octaveband, the octaveband energy level will constantly increase with $3dB$ for each octaveband. However, when comparing results from two different calculation methods such knowledge is irrelevant. The *energy level* of each octaveband impulse response was found by the following equation:

$$E_{oct.band} = 10 \cdot \log \left(\sum_{n=1}^{N} |h_{oct.band}(n)|^2 \right) \tag{2.20}$$

where $h_{oct.band}(n)$ is the octaveband-filtered impulse response.

In order to present reliable results for general source and receiver positions, the presented octaveband error plots are all based on the averaged outcome of a total number of 100 simulation sessions. To obtain random source and receiver positions the Matlab function **rand()** was employed.

All relevant Matlab scripts can be found in the Appendix. Also, the applied CATT-Acoustic geometry file can be found here.

Chapter 3

Results

This chapter gives a presentation of the obtained results from the proposed semi-transparent noise barrier modelling presented in Chapter 2.3.5. As mentioned, the EDBtoolbox serves as the reference for all the presented results. Be aware that the distributions of S_G and R_G were defined in Chapter 2.3.5. Further, some supplementary comments are included in order to clarify possible vaguenesses related to the presented figures.

3.1 Errors in the shadow zone

As a first evaluation for the proposed method, a set of simulation sessions were performed for receivers located in the shadow zone, which means that no direct sound contributions are included. Furthermore, the proposed modelling was evaluated for three different scenarios of increased complexity.

First, Figure 3.1 gives a plot of the mean octaveband error for the receiver array R_G. In addition, two extended receivers at $R_z = 0.25m$ and $R_z = 2.75m$ are also included. S_G is the employed source. Accordingly, it can be stated that Figure 3.1 holds a multiple receiver quality regarding the z-axis.

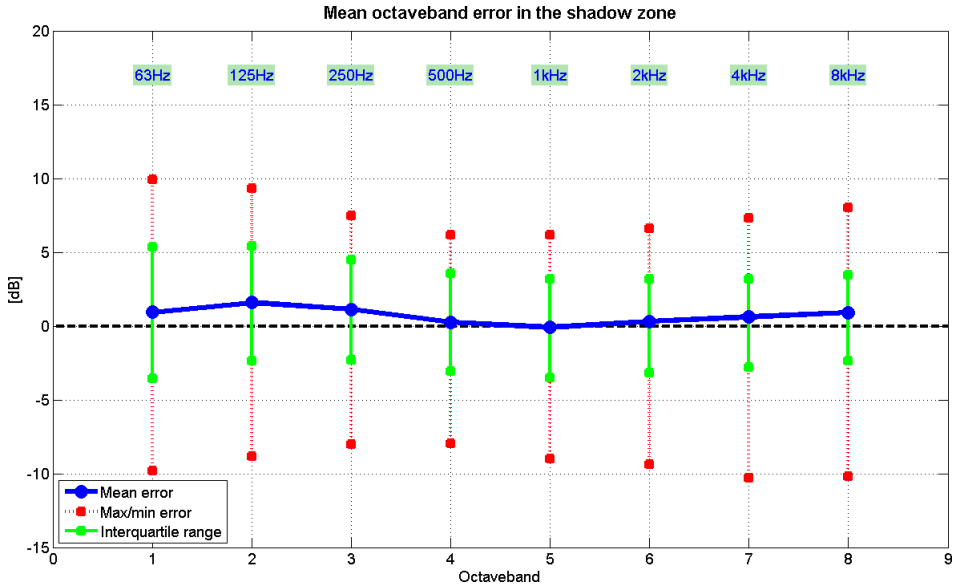

Figure 3.1: *Mean octaveband error*

Secondly, Figure 3.2 gives a plot of the mean octaveband error as the receiver is placed at different positions along the y-axis. Again, S_G is the employed source. However, all the presented receivers are now set up in addition to the receiver array R_G. The receivers are placed along an array $R_y = \{0, 2, \cdots, 8\}m$, at the same x-axis value as R_G, somewhere randomly within the interval $R_z = [0.25, 2.75]m$. Accordingly, it can be stated that Figure 3.2 holds a multiple receiver quality regarding the y-axis.

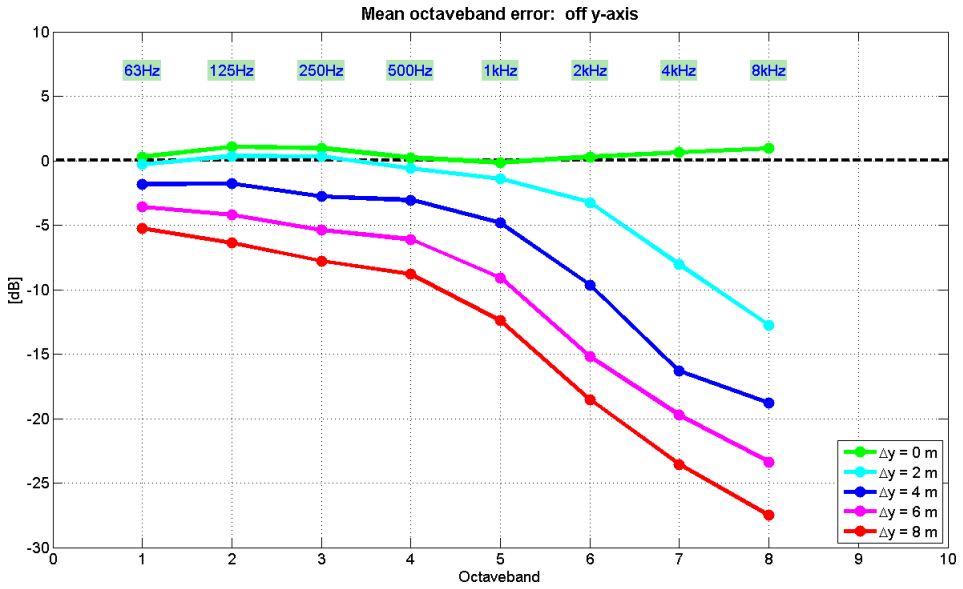

Figure 3.2: *Mean octaveband error - off y-axis*

Thirdly, Figure 3.3 gives a plot of the mean octaveband error as the receiver is placed at different positions along the x-axis. As in the non-symmetric y-axis case, all the presented receivers in Figure 3.3 are placed in addition to the receiver array R_G. The receivers are placed along an array which is displaced by a factor Δx relative to the x-axis value of R_G. Further, the receivers are placed at a random position within the interval $R_z = [0.25, 2.75]m$, at $R_y = 0m$. Accordingly, it can be stated that Figure 3.3 holds a multiple receiver quality regarding the x-axis. In order to enable the presented receiver position $\Delta x = -10m$, the range of the receiver array R_G was truncated to $R_x = [11.0, 30.0]m$.

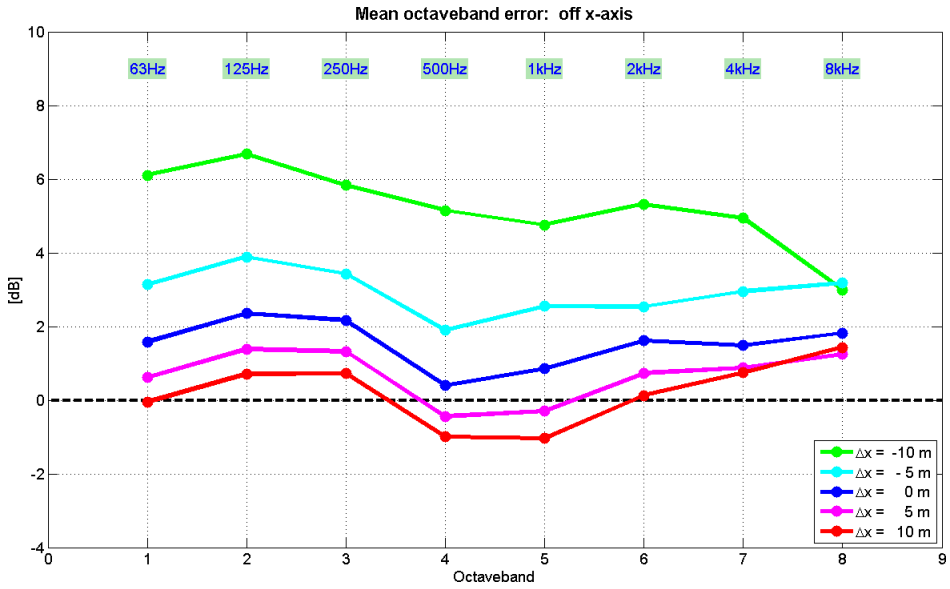

Figure 3.3: *Mean octaveband error - off x-axis*

3.2 Errors at source-receiver sight line

Figure 3.4 gives a plot of the errors that occur for receivers locateded close to the source-receiver sight line. The presented receivers are placed at the same position as the receiver array R_G regarding the x-axis and y-axis. In order to fix the presented receivers relative to the source-receiver sight line, a parameter z_{sz} is now introduced - given by the interesection between the line determined by the source S_G and the barrier edge E: $\overleftrightarrow{SE}$, and the floor perpendicular at R_G.

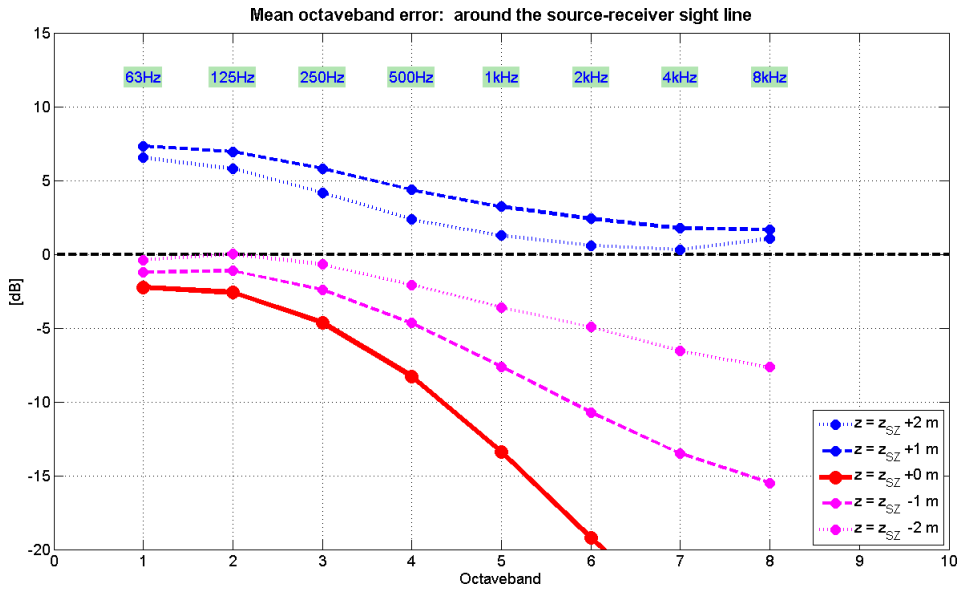

Figure 3.4: *Mean octaveband error - around the source-receiver sight line*

Figure 3.5 gives a plot of the receiver positions just above the source-receiver sight line presented *with* and *without* the polarity correction described in Chapter 2.4. The presented results are based on the same simulation series as in Figure 3.4.

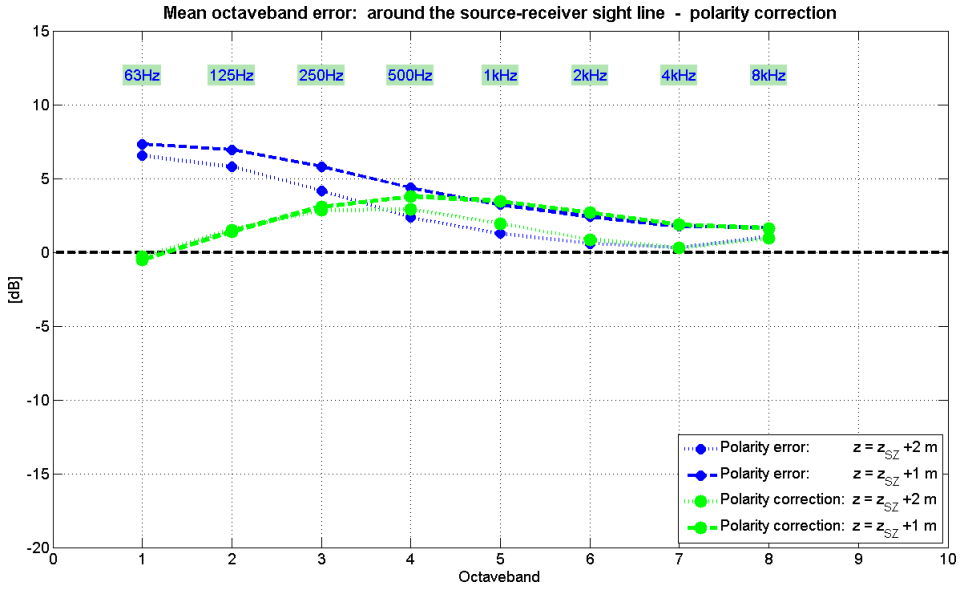

Figure 3.5: *Error plot for the polarity correction case*

3.3 Squared impulse response as primary quantity

Figure 3.6 gives a plot of the mean octaveband error by the squared impulse response/echogram as primary quantity. Hence, the calculations are based on the list of equations given in Chapter 2.5. Otherwise, the presented result in Figure 3.6 is achieved by the exact same simulation setup as in Figure 3.1.

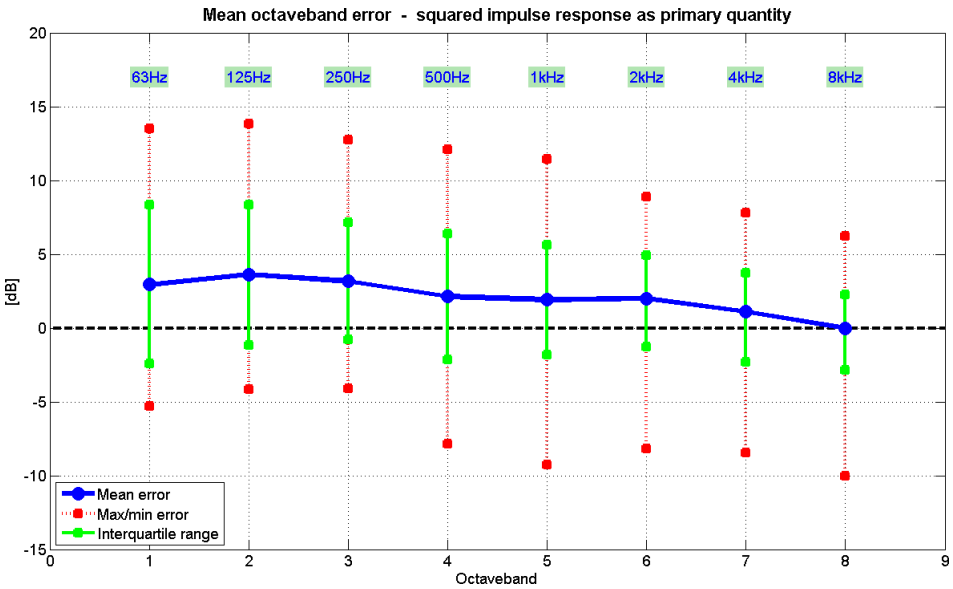

Figure 3.6: *Mean octaveband error - squared IR as primary quantity*

Chapter 4

Discussion

This chapter gives a discussion on the achieved results. The chapter focuses to a great extend on the advantages, limitations and possible side effects caused by the proposed semi-transparent noise barrier modelling. Furthermore, discussion is given to some practical issues regarding a potential future implementation of the semi-transparent edge diffraction modelling in commercial geometrical room acoustic softwares.

4.1 First impression

By first impression it must be argued that the errors in Figure 3.1 reveals a fairly hopeful result. The *mean*-value errors are very small and almost negleciable. Sadly, large errors are introduced for the max/min-values, as $10dB$-errors cannot be tolerated. In addition, the range between max and min is also large, which indicates a disturbing instability in the modelling. The main reason for these errors can probably be explained by an introduced uncertainty in the applied semi-transparent noise barrier modelling procedure of Chapter 2.3.5. On the other hand, the performed simulation sessions do include a large amount of critical rare positions, where there in any case is assumed that the modelling will function badly. Errors are expected for positions close to the barrier, as well as for positions close to the floor and barrier height. These expected errors are caused partly due to the assumptions introduced for the nth sample barrier intersection area S_n in eq. (2.7), and partly due to a potential unfavourable scattering behaviour. Especially, for a combination of an odd source position and an odd receiver position there must be expected a misleading impulse response shape. Fortunately, most of these odd source and receiver positions are indeed rare in any practically relevant source-receiver setup. Accordingly, it may be stated that the

largest errors will not appear very frequently, and that the large max/min errors probably will vanish by a slightly truncated range of source/receiver positions. Further, the interquartile range also indicates a somewhat improved impression of the overall stability of the modelling. A $5dB$-error is of course not a desired outcome, but it is still within a fairly reasonable range.

Nevertheless, it must also be restated that geometrical room acoustics in any case often introduces some errors in the lower octavebands, as room modes, comb filtering effects, ect. are neglected. It could therefore be questioned whether the diffraction errors of Figure 3.1 at all should be questioned ahead of these already existing uncertainties? Moreover, it will perhaps also be advantageous to implement a mimetic edge diffraction modelling that introduces some errors and uncertainties - versus a modelling that yields no edge diffraction energy contributions at all.

4.2 Multiple receivers case

By studying Figure 3.2, it must be stated that the presented semi-transparent modelling has some limitations regarding a non-symmetric y-axis case. Especially for the highest octavebands, the non-symmetric y-axis case reveals large errors. However, this is not a very surprising outcome, since the samples that arrives first at the receiver in the symmetric y-axis case will not arrive first in the non-symmetric y-axis case. Accordingly, the desired pulse shape in Figure 2.9 will not arise, as the mid/high peak zone will be more smeared out and arrive further out in the impulse response. In addition, the energy contributions c_k of the mid/high peak zone rays will be further affected by the scattering coefficient s_k, due to an increased angle out from the noise barrier. However, this scattering damping may actually not be such a bad approach to how the higher frequencies decrease along the non-symmetric line. Hence, the major concern should most likely be given to the mentioned delayed arrival time of the mid/high peak zone rays.

Further, the non-symmetric y-axis errors in the lower octavebands can easily be justified by emphasizing that the simulated infinite noise barrier is not infinite for real. As mentioned in Chapter 2.7, each subarea outside the barrier region $y \in [-6, 6]m$ is given a transmission coeffiecient τ_k equal to zero. Surely, for the non-symmetric y-axis case this implemented simplification will introduce side effects. Therefore, the lower octaveband errors could probably have been avoided by just adding the bass tail zone transmission coefficient to a larger region of the infinite noise barrier. Furthermore, it may perhaps be stated that the lower octavebands are modelled in a somewhat satisfying way by the applied constant transmission coefficient of the bass tail zone.

Figure 3.3 reveals that the proposed modelling seems a bit more stabile regarding movement along the x-axis. Still, some errors do appear - especially for receivers located closer to the noise barrier relative R_G. The presented $6dB$-errors at $\Delta x = -10m$ cannot be tolerated. Further, it is noticeable that the errors at $\Delta x = 5m$ and $\Delta x = 10m$ in fact are smaller than at $\Delta x = 0m$. This may indicate a somewhat non-optiamlized semi-transparent noise barrier modelling regarding the the x-axis. Another interesting observation is that the presented receiver $\Delta x = 0m$ yields a different result than the presented *mean* error in Figure 3.1. Most likely, this appears due to the truncated range of R_G along the x-axis.

4.3 Interference issues at source-receiver sight line

As expected, Figure 3.4 reveals that serious errors are introduced for receivers located close to the source-receiver sight line. First, it is evident that the modelling does not deliver enough energy for the receiver postions just below the source-receiver sight line - i.e. at positions where direct sound contributions still are obstructed and the diffracted sound contributions are appreciable. Accordingly, it can be concluded that the proposed modelling fails to deliver enough energy in this critical region of the shadow zone.

Secondly, it is obvious that the modelling gives a fatal mismatch along the source-receiver sight line z_{sz}, and especially for the higher octavebands. Most likely, this is caused by the total lack of direct sound contributions, which indeed not is the case for the true acoustic conditions. Maybe an additional scattering coefficient at the edge of the barrier could mend the faulty modelling? However, for the presented results in this report, this given error must again be justified by the somewhat rare receiver positions. In addition, this highly critical region concerns only a very limited receiver range.

Thirdly, Figure 3.4 also displays noticeable errors at the lowest octavebands for the receivers located just above the source-reiver sight line. These errors are most certainly caused by the polarity error described in Chapter 2.4. Clearly, the improved result of the polarity correction plot in Figure 3.5 argues in favour of this statement. As expected, the improved outcome of the polarity correction curve is best for the lower octavebands, while it tends to diminish at higher frequencies. The reason is that the higher octavebands will not be as strongly affected by the introduced interference effect. Moreover, it must be stated that an improvement of about $6dB$ in the two lowest octavebands is quite noteworthy. Accordingly, it could be further questioned whether it would be profitable to study a potential polarity correction implementation for the edge diffraction energy in geometrical room acoustic software. Indeed, this correction is easy to treat in the developed

Matlab script, but it may however disturb the simple calculation algorithms that makes the ray-tracing method so efficient.

4.4 Semi-transparent noise barrier design improvements

One action to potentially lower the presented errors could of course be to study whether there exists alternative noise barrier modelling methods that function better. As mentioned, the presented model is based on a two-zones transmission coefficient mapping, where the idea is that the first zone mainly builds the energy of the mid and high frequencies, whereas the second zone adds requisite energy to the lower frequencies. Even though this solution turns out to give an applicable result, there may still exist other noise barrier designs that imitates the true noise barrier behaviour even better. Perhaps a more complex design of even more zones could be profitable? Likewise, maybe entirely different, more clever modelling concepts could be developed?

By contrast, one obvious alternative modelling technique could be to develop a noise barrier design that involves an octaveband-based transmission coefficient. Such a solution corresponds to both the previous semi-transparent edge diffraction modelling studies by *Dammerud* [2] and *Isebakke* [3]. As mentioned, an octaveband-dependent semi-transparent feature already exists in CATT-Acoustic, which surely is beneficial. However, the disadvantage of such a modelling technique is that the succeeding pulse shape of the impulse response is less controlled. In addition, it must also be stated that a broadband simulation technique surely is advantageous in many ways.

It is evident that the presented modelling technique introduces errors at the highest receiver positions in the shadow zone, as too little energy is added to the early samples of the impulse response. One way to possibly improve these errors could be to study whether the high/mid frequency region could distribute the energy even more accurately. Maybe the mid/high peak zone should be more semi-transparent in the higher region, and less semi-transparent in the lower region? For the presented modelling, the transparency distribution is based on a 1st order polynomial fit to the transmission factors T_1 of the reciever array R_G. Perhaps another more progressive/complex curve fit could do a better job?

Moreover, to lower the non-symmetric errors in Figure 3.2, one possible improvement could be to devide the barrier into a number of additional mid/high peak zones - arranged in a pattern to optimally fit the hig/mid frequency expansion along the y-axis. In order to maintain the natural damping for an increased receiver position along the y-axis, the mid/high peak zones could have been dedicated a suitable damping factor as function of distance along the y-axis.

4.5 Echogram issues

By Figure 3.6 it must be stated that a semi-transparent noise barrier modelling given by the squared impulse response/echogram as primary quantity introduces some additional octaveband errors. An additional error of about $3dB$ can be found at the lower octavebands. As stochastic differences are expected to have a minor impact for the averaged outcome of a number of 100 simulations, these introduced differences are most probably caused by the difference in eq. (2.1) and eq. (2.19).

First, as the scattering coefficient will now have a $\sqrt{\cos\phi_k}$ influence (instead of a $\cos\phi_k$ influence) on the ray contributions of the impulse response $h(n)'$ there might be expected a somewhat different resulting impulse response: $h(n)' \neq h(n)$. Furthermore, as $\sqrt{\cos\phi_k} > \cos\phi_k$ for all $\phi_k \in \langle 0, \frac{\pi}{2} \rangle$, the re-written echogram quantity-based impulse response $h(n)'$ will probably include a somewhat larger energy amount: $h(n)' > h(n)$. Indeed, by comparing Figure 3.6 and Figure 3.1, this also seems to be the case. However, it might be a bit harsh to claim that this scattering coefficient inequality alone will constitute in a $3dB$ difference.

Yet another interesting difference that may appear by the echogram as primary quantity is a *coherent ray addition problem*. In ordinary ray-tracing, energy contributions can safely be added to the impulse response, since all the individual contributions typically correspond to very different propagation paths, and thereby very different phase shifts. Then, on average, energy addition will be correct. Here, on the other hand, all the contributions that arrive within the same sample will have exactly the same phase shift, which leads to a coherent ray addition. Fortunately, however, the calculation of the transmission coefficients is based on such a coherent addition for the receiver array R_G. Therefore, the coherent ray addition does give close to the expected results.

Accordingly, there should probably be stated that the main reason for the introduced additional errors are caused by a somewhat non-optimalized noise barrier modelling for the echogram as primary quantity. Note that the entire procedure of Chapter 2.3.5 in this case was based on the reference signal $h^2_{ref}(n)$, which means that τ'_k not necessarily equals to $\tau^2_k \cdot dS_k$. Clearly, such a conditional equality would conflict the entire comprehension of *primary quantity*. Further, the proposed semi-transparent noise barrier modelling in Chapter 2.3.5 is surely not developed by any exact mathematical background, and instead given by a basic two-zones division concept and a succeeding trial by error procedure. Especially, the introduced constant 0.1 in the bass tail zone transmission coefficient remains an unknown property. By lowering this value an improved result could be expected in Figure 3.6. In conclusion, it does not seem to be vitally impractical to implement the proposed modelling into the perferred echogram quantity.

4.6 Propagation phase error

As the first incomming ray will now arrive at a time that corresponds to the source-receiver propagation path SR, rather than a time that corresponds to the true source-edge-receiver propagation path $SE + RE$, a propagation phase error has been introduced, which is given by the following equation:

$$\Delta t_{error} = \frac{SE + RE - SR}{c_{air}} \tag{4.1}$$

The propagation phase error lies somewhere in the region $\Delta t_{error} \in \langle 0, \frac{2 \cdot OE}{c_{air}} \rangle$, which corresponds to some milliseconds. To be more specific, the studied case of a $3m$ high noise barrier will return a maximum propagation error of $17.5msec$. However, the error will most likely be further reduced in any practically relevant source-receiver setup.

Now, which effects will this propagation phase error cause? In an enclosed room case, as early reflections from walls and ceilings are included, the propagation phase error will cause a small time shift for the edge diffraction energy contribution relative to the rest of the room's impulse response. Therefore, if Δt_{error} is large, there may be expected some errors in the auralized sound quality of the simulation relative to the true acoustic conditions. As an indicator on the scope of this error, it may be appropriate to consider the well-established room acoustic parameters *Deutlichkeit* D_{50} and *Clarity* C_{80}. Accordingly, since these parameters operates with time intervals of $[0, 50]msec$ and $[0, 80]msec$, there may be reasonable to assume that the propagation phase error will be of minor importance.

Some interference pattern errors may also appear due to the propagation phase error. However, as interference patterns are generally neglected in geometrical room acoustics, these potential interference pattern errors will not be further discussed.

4.7 Finite noise barrier issues

When considering a *finite* noise barrier case some new troubles are introduced, and the presented modelling may not be able to comply. In the presented results, only the infinite noise barrier case has been simulated. Evidently, this is an ideal situation and a rare case in room acoustics. In the more applicable case of a finite noise barrier, edge diffraction contributions will also arise from the two sides of the barrier. By the presented noise barrier modelling technique, the

potential number of three edge diffraction contributions, all arriving at different propagation times, is clearly not supported.

Again, one possible solution could be to reconsider the semi-transparent noise barrier modelling design. Maybe it could be more profitable to apply a semi-transparent design that is most semi-transparent close to the edges? However, more attention would then be required regarding the overall ray arrival sample n_k - as it is highly desired to obtain the typical impulse response shape of the edge diffraction case.

Anyhow, for the presented semi-transparent noise barrier modelling it must be stated that only 1st order edge diffraction from a surface can be applied. Apparently, this again corresponds to a reliance on the room acoustic parameters Deutlichkeit D_{50} and Clarity C_{80}. Whether or not this is a good approach regarding auralization could perhaps be further discussed.

4.8 Further transmission coefficient studies

Interestingly, a consecutive assumption by the proposed semi-transparent modelling is that there seems to be a strong correlation between the peak value of the impulse response $h_{ref}(n_{peak})$ and the entire impulse response $h_{ref}(n)$. This assumption is given by the fact that the entire simulated impulse response $h(n)$ is constructed based on the receiver array R_G's impulse response peak values. Consequently, also the bass tail zone transmission coefficient is based on the outcome of these peak values. However, it is a well-known fact that the frequency response will vary to some extent as the receiver is moved closer to the barrier edge, since the efficient screening height is then lowered and more mid frequency energy is added. Therefore, it should be expected that the assumed peak value correlation will lead to a certain amount of errors. Fortunately, as the above discussion claims, the peak correlation assumption still seems to work out fairly well.

Now, as the true impulse response peak value T_1 in eq. (2.13) plays an important role in the proposed semi-transparent modelling, there should also be of vital interest to study whether there is possible to derive a simple relationship between T_1 and the simple geometry of the infinite noise barrier case. The preferred variables would then typically be the source position $\{S_x, S_z\}$ and the receiver position $\{R_x, Rz\}$. However, the total number of four input parameters will make it a bit complicated to derive a simple relationship. Therefore, a great advantage would be to assume a relationship that can be expressed by e.g. the product of two transmission coefficients:

$$T_1(S_x, S_z, R_x, R_z) = T_{S1}(S_x, S_z) \cdot T_{R1}(R_x, R_z) \tag{4.2}$$

At an early stage in this research process, some simulation sessions were carried out in order to seek such a relationship. Sadly, no usable material was found, and as this was not considered main focus in the research process it was eventually decided to instead take advantage of the results from the pre-processed reference Edge diffraction toolbox values. Such an alternative could be associated with an implemented table sheet in geometrical room acoustic softwares. However, for advantage, further studies should be performed in search for a possible $T_1(S_x, S_z, R_x, R_z)$ relationship. A reasonable proposal would perhaps be to work with spherical coordinates - centered at the noise barrier edge. The reason is that there may be assumed that the angles θ_S and θ_R will contain essential information.

4.9 Software implementation issues

As mentioned, a main goal of this study has been to find an edge diffraction modelling that easily can be implemented in existing commercial geometrical room acoustic softwares. Likewise, it has been stated that the proposed modelling concept seems to correspond with the often applied diffuse rain algorithm. However, the presented semi-transparent noise barrier modelling results have all been built by a Matlab script, and no serious attempt has so far been made on a true geometrical room acoustic software implementation. Therefore, this final discussion will be dedicated to possible issues regarding a potentional future semi-transparent edge diffraction implementation.

Interestingly, the presented modelling is based on a broadband simulation technique, which to some extent is advantageous, but which also will cause some issues regarding a potentional room acoustic software implementation. Normally, to imitate an octaveband-dependent absorption factor related to each surface, simulations are run separately for each octaveband. From there, the echogram values of each octaveband are post-processed to form an interpolated frequency response. Finally, this frequency response is combined with a digital signal processing minimum-phase filter to generate a continuous broadband impulse response curve. Accordingly, the question is then how a broadband simulation technique can be added to this applied octaveband simulation technique? One possible solution could perhaps be to run a separate "edge diffraction" simulation session in addition to all the octaveband simulations, and then somehow merge the edge diffraction impulse response to the final output impulse response by post-processing. Unfortunately, at this moment it remains unknown whether

such an operation successfully can be implemented, but hopefully some future studies will deal with this current problem.

For simplicity, rays are usually distributed with a constant mutual angle out from the source in geometrical room acoustic softwares. This means that each ray represents a certain constant angle of the sphereical wave. However, in the developed model - as each ray k is linked to a fixed size subarea dS_k of the noise barrier, the rays do not represent a uniformly distributed angle of a spherical wave source. However, this fact should not cause troubles in a potential future geometrical room acoustic software implementation, as the only adaptation would be to give dS_k a variable value.

Further studies should be performed regarding the reverbant consequencies of the presented edge diffraction modelling. In this study only early edge diffraction energy contributions are considered. How is the reverbant field affected by this modelling technique? Moreover, since this is a broadband simulation method, what will happen as these rays hit a frequency-dependent absorptive surface?

Chapter 5

Conclusion

To summarize, a general potential in the proposed semi-transparent edge diffraction modelling technique has been revealed. The presented results and discussion indicate a somewhat successful modelling of the infinite noise barrier edge diffraction case. However, some unwanted errors are introduced at rare source-receiver positions. Also, some errors are found by the adoption of a multiple receivers quality, especially for a non-symmetric y-axis. Thus, possible future studies should be performed in search for a somewhat improved modelling procedure.

As desired, there seems to be possible to implement the presented edge diffraction modelling in existing geometrical room acoustic softwares. Apparently, the modelling tends to agree with the often applied diffuse rain algorithm. However, no attempt has so far been made on a true geometrical room acoustic software implementation. Some possible complications are introduced as the presented modelling is a broadband ray-tracing simulation method, whereas the common procedure is to perform individual octaveband simulations. In conclusion, further research should study to which extent the proposed modelling technique actually can be implemented in commercial geometrical room acoustic softwares.

A main outcome of this report reveals that a fairly satisfying model of the noise barrier edge diffraction case can be devoloped based only the reference model's impulse response peak value. Accordingly, there should be of further interest to find a somewhat simple relationship between the impulse response peak value and the simple geometry of the given infinite noise barrier case.

Bibliography

[1] U. P. Svensson, R. I. Fred, and J. Vanderkooy, *An analytic secondary source model of edge diffraction impulse responses.* J. Acoust. Soc. Am. 106, 2331-2344, 1999.

[2] J. J. Dammerud, *Stage acoustics for symphony orchestras in concert halls.* Ph.D. thesis, University of Bath, England, 2009.

[3] A. Isebakke, *Modelling audience seating area in geometrical room acoustic software.* Faculty of information technology, mathematics and electrical engineering, Norwegian university of science and technology, Norway, 2010.

[4] T. J. Schultz and B. G. Watters, *Propagation of sound across audience seating.* J. Acoust. Soc. Am. 36, 885-896, 1964.

[5] G. M. Sessler and J. E. West, *Sound transmission over theatre seats.* J. Acoust. Soc. Am. 36, 1725-1732, 1964.

[6] J. S. Bradley, *Some further investigations of the seat dip effect.* J. Acoust. Soc. Am. 90, 324-333, 1991.

[7] U. P. Svensson, "http://www.iet.ntnu.no/∼svensson/software/index.html."

[8] A. Krokstad, S. Strøm, and S. Sørsdal, *Calculating the acoustical room response by the use of a ray tracing technique.* J. Sound and Vibration 8, 118-125, 1968.

[9] R. Heinz, *Binaural room simulation based on an image source model with addition of statistical methods to include the diffuse sound scattering of walls and to predict the reverberant tail.* J. Applied Acoustics 38, 2-4, 145-159, 1993.

[10] CATT-Acoustic, *CATT-Acoustic v8.0 user's manual*, 2002.

[11] CATT-Acoustic, *Addendum to CATT-Acoustic manual*, 2010.

[12] H. Kuttruff, *Room acoustics, 5th edition.* Spon Press, 2009.

[13] T. E. Vigran, *Bygningsakustikk, 1st edition.* Tapir Akademisk Forlag, 2002.

[14] K. Rottmann, *Matematisk formelsamling, page 115.* Spektrum Forlag, 2003.

[15] J. M. Berman and L. R. Fincham, *The application of digital techniques to the measurement of loudspeakers.* J. Audio Eng. Soc. 25, 370-384, 1977.

[16] L. R. Fincham, *Refinements in the impulse testing of loudspeakers.* J. Audio Eng. Soc. 33, 133-140, 1985.

[17] E. Benjamin, *Extending quasi-anechoic electroacoustic measurements to low frequencies.* AES Convention paper 6218, 2004.

[18] T. Lokki, U. P. Svensson, and L. Savioja, *An efficient auralization of edge diffraction*. Proceedings of AES 21st Conference on Arhitectural Acoustics and Sound Reinforcement, St. Petersburg, Russia, 2002.

Appendix A

CATT-Acoustic files

A.1 CATT-Acoustic geometry file

This is the CATT-Acoustic **.geo** file that was exported into a **.cad** file and used as the geometry file in the **EDBmainISESx** Matlab toolbox.

```
;MASTER.GEO
;PROJECT=NoiseBarrier

;INCLUDE

;OFFSETCO
;OFFSETPL

;MIRROR co_add pl_add

ABS wall <10 10 10 10 10 10> ;L <10 10 10 10 10 10>

CORNERS

;id x y z
1    0    -60   3
2    0    60    3
11   0    -60   0
12   0    60    0
101      0   0   0


PLANES

;[id name  /   / absname  ]
[1 sourceSide    / 1 2 12 101 11 / wall ]
[2 receiverSide  / 1 11 101 12 2 / wall ]
```

Appendix B

Matlab files

B.1 The EDBtoolbox setup file

A file that was run as the setup file by the **EDBmainISESx** function in the EDBtoolbox. The file includes the source S_G and receiver array R_G. For those presented figures with additional receiver positions, the additional receivers where simply added by an extended receiver matrix.

```
%% NOISE BARRIER - infinite in free field
% by Anders Isebakke

global FSAMP CAIR RHOAIR SHOWTEXT
FSAMP = 24000;  % Sample frequency
CAIR = 343;     % Speed of sound
RHOAIR = 1.21;  % Rho air
SHOWTEXT = 3;   % Determines how much text will be printed on the screen


%% Input/Output files

% INPUT: CAD geometry file
CADfile = 'C:\Anders\Skole\Master\Simulations\NoiseB\NoiseBarrierGEO.CAD';

open_or_closed_model = 'o';     % Specify if the model is open or closed
int_or_ext_model = 'e';         % Specify if you are interested in the
                                % interior or exterior of the model.

% OUTPUT filepath
Filepath = 'C:\Anders\Skole\Master\Simulations\NoiseB\MatlabOUT\';

% OUTPUT filestem (output files will start with this name)
Filestem = 'NoiseBarrier_infinite_freefield_OUT';


%% Source and receivers

% SOURCE: SG
sources = [-((30-1)*rand(1,1) + 1) 0 ((2.75-0.25)*rand(1,1)+0.25)];


% RECEIVERS: RG
xREC = ((30-1)*rand(1,1) + 1);
```

```
yREC = zeros(1,9);
zREC = linspace(0.5,2.5,9);

receivers = zeros(length(zREC),3);
receivers(:,1) = xREC;
receivers(:,2) = yREC;
receivers(:,3) = zREC;


%% Calculation parameters

EDcalcmethod = 'n';      % 'n' = the new method by Svensson et al.
directsound = 1;         % 1 to include the direct sound
                         % 0 to exclude the direct sound
specorder = 2;           % The highest number of specular reflections.
difforder = 1;           % The highest number of edge diffractions.
                         % Note that specorder must be >= difforder.
elemsize = 1;            % Accuracy parameter for each order of edge diffraction.
                         % The vector must start with 1. The value 0.5 decides how
                         % small edge elements will be used for second order
                         % diffraction. A higher number gives more accurate
                         % results but takes much longer time.
                         % For third-order: elemsize = [1 0.5 0.25] for
                         % instance.
nedgesubs = 2;           % This is a parameter that decides how many parts
                         % each edge is divided into for the
                         % part-visibility check of edges. A higher number
                         % gets more accurate but takes much longer time.
                         % Minimum = 2.
calcpaths = 1;           % If you want to run the first calculation step
                         % (find the paths), set the value 1.
                         % If you have run this part earlier and just want
                         % to change some setting for the second calculation
                         % step, then you can re-use the first step and set
                         % this value to 0.
calcirs = 1;             % If you want to run the second calculation step
                         % (construct IRs), set the value 1, otherwise 0.

firstskipcorner = 10;    % exclude all corners with corner numbers higher than this.

Rstart = 0;              % All impulse responses will have a lot of zeros
                         % at the start, if the distance from source
                         % to reciever is5 long, and the sampling frequency
                         % is high. By setting Rstart to some non-zero
                         % value, the impulse responses will all start at
                         % the time that corresponds to this distance in
                         % meters. It is important to set this longer than
                         % the minimum that can ever happen since there
                         % might be some cryptic error message otherwise.
```

B.2 Typical impulse and frequency response

A file that was used for to make the figures of the typical impulse response in Figure 2.2 and frequency response in Figure 2.3.

```
%% A typical example of noise barrier edge diffraction
% by Anders Isebakke


%% Initialize the IR matrix

longestIR = load(['MatlabOUT\NoiseBarrier_infinite_freefield_OUT_1_', ...
    num2str(length(receivers(:,1))),'_ir.mat']);

% Impulse response vector
ir = zeros(int64(1.5*length(longestIR.irtot)),length(receivers(:,1)));

% Time vector
tvec = 0:1/FSAMP:(length(ir(:,1))-1)/FSAMP; % Time vector


for ii = 1:length(receivers(:,1))
    temp = load(['MatlabOUT\NoiseBarrier_infinite_freefield_OUT_1_', ...
        num2str(ii),'_ir.mat']);
        ir(1:length(temp.irtot),ii) = temp.irtot;
end



%% Preform the Fast Fourier transform

nfft = FSAMP/4;      % Number of FFT elements
fvec = 1:4:FSAMP;    % Frequency vector


% The frequency response vector
FreqResp = zeros(nfft,length(ir(1,:)));
for ii = 1:length(FreqResp(1,:))
    FreqResp(:,ii) = fft(full(ir(:,ii)),nfft);
end



%% Make figures

figure(1);  % Impulse responses
plot(1000.*tvec,ir,'b','LineWidth',4);
set(gca,'FontSize',14);
xlabel('Time [ms]');
title('Impulse response of an edge diffraction case', ...
    'FontWeight','bold','Fontsize',16);
axis([25 55 -0.0005 0.008]);
grid on;

figure(2);  % Frequency responses
semilogx(fvec,20.*log10(abs(FreqResp)),'b','LineWidth',4);
axis([100 5000 -45 -27]);
set(gca,'FontSize',14);
xlabel('Frequency [Hz]');
ylabel('[dB]');
title('Frequency response of an edge diffraction case', ...
    'FontWeight','bold','Fontsize',16);
grid on;
```

B.3 Ray paths illustration

A file that was used to build the ray paths illustration in Figure 2.4.

```
%% A script that illustates the ray paths
% by Anders Isebakke


%% The geometry

FSAMP = 24000;  % Sample frequency
CAIR = 343;     % Speed of sound

src = [-5 0 1.5];       % Sources
rec = [5 0 1.5];        % Receiver

edge = 3;           % The barrier height


barrierH = 3;       % Barrier height
barrierW = 6;       % Barrier width

numbOfElemZ = 3;    % Number of elements along z-axis
numbOfElemY = 5;    % Number of elements along y-axis

elemArea = (barrierH/numbOfElemZ)*(barrierW/numbOfElemY);   % Element area



%% Ray hits at barrier - [x y z]

rayHits = zeros(numbOfElemY*numbOfElemZ,3);
for jj=0:numbOfElemZ-1
    for ii=0:numbOfElemY-1
    rayHits((ii+1)+(numbOfElemY*jj),2) = (0.5*barrierW/numbOfElemY) + ...
        (ii.*barrierW/numbOfElemY) - (barrierW/2);
    end
end
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        rayHits((1+(jj*(numbOfElemY))):(numbOfElemY+(jj*(numbOfElemY))),3) = ...
            - (0.5*barrierH/numbOfElemZ) + ((jj+1).*barrierH/numbOfElemZ);
    end
end



%% Noise Barrier Color Mapping

wallCorners = zeros(numbOfElemY+1,numbOfElemZ+1);
for ii=1:numbOfElemY+1
    wallCorners(ii,:) = linspace(0,barrierH,numbOfElemZ+1);
end


zVec = linspace(0,barrierH,numbOfElemZ+1);
xVec = zeros(1,4);
yVec = linspace(-barrierW/2,barrierW/2,length(wallCorners(:,1)));


patchesY = zeros(numbOfElemY*numbOfElemZ,4);
for jj = 0:numbOfElemZ-1
    for ii = 1:numbOfElemY
        patchesY(ii+(numbOfElemY*jj),:) = [yVec(ii) yVec(ii+1) yVec(ii+1) yVec(ii)];
    end
end


patchesZ = zeros(numbOfElemY*numbOfElemZ,4);
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        patchesZ(ii+(jj*(numbOfElemY)),:) = [zVec(jj+1) zVec(jj+1) zVec(jj+2) zVec(jj+2)];
    end
end
```

```
%% A plot of the ray paths

edgeColoor = [0.2 0.2 0.2];

figure(2);
plot3(src(1),src(2),src(3),'k o','MarkerSize',20,'MarkerFaceColor','b');
text(src(1),src(2)-0.25,src(3),'\color{black}Source','FontSize',14, ...
    'HorizontalAlignment','left');
for ii = 1:length(rec(:,1))
    hold on;
    plot3(rec(ii,1),rec(ii,2),rec(ii,3),'k d','MarkerSize',20, ...
        'MarkerFaceColor','b');
end
text(rec(1),rec(2)-0.25,rec(3),'\color{black}Receiver','FontSize',14, ...
    'HorizontalAlignment','left');
for ii = 1:numbOfElemY*numbOfElemZ
    hold on;
    patch(xVec,patchesY(ii,:),patchesZ(ii,:),[0.4 0.8 0.6], ...
        'EdgeColor',edgeColoor);
end
for ii = 1:length(rayHits(:,1))
    hold on;
    plot3([src(1) rayHits(ii,1) rec(1)],[src(2) rayHits(ii,2) rec(2)], ...
        [src(3) rayHits(ii,3) rec(3)],'b','LineWidth',2);
end
set(gca,'FontSize',14);
hold off;
axis([-5 5 -3 3 0 5]);
xlabel('x-axis [m]');
ylabel('y-axis [m]');
zlabel('z-axis [m]');
title('Subarea tessellation and ray paths','FontWeight','bold','Fontsize',18);
grid off;
```

B.4 *n*th sample barrier intersection area illustration

A file that was used to build the *n*th sample barrier intersection area illustration in Figure 2.5.

```
%% A script that illustates the different nth sample regions of the barrier
% created on Apr. 28, 2011 by Anders Isebakke


%% The geometry

FSAMP = 24000;  % Sample frequency
CAIR = 343;     % Speed of sound

src = [-5 0 1.5];       % Sources
rec = [5 0 1.5];        % Receivers

edge = 3;               % The barrier height


barrierHit = zeros(length(rec(:,1)),1);
for ii = 1:length(barrierHit)
    barrierHit(ii) = ...
        (rec(ii,3) - rec(ii,1)*((rec(ii,3)-src(3))/(rec(ii,1)-src(1))));
end

distSRCtoP = zeros(length(rec(:,1)),1);
for ii = 1:length(distSRCtoP)
    distSRCtoP(ii) = norm([0 0 barrierHit(ii)] - src);
end

distPtoREC = zeros(length(rec(:,1)),1);
for ii = 1:length(distPtoREC)
    distPtoREC(ii) = norm([0 0 barrierHit(ii)] - rec(ii,:));
end

sampelRadius = zeros(length(rec(:,1)),1);
for ii = 1:length(sampelRadius)
    sampelRadius(ii) = ...
        sqrt(2.*(CAIR/FSAMP).*((distPtoREC(ii)*distSRCtoP(ii))./ ...
        (distPtoREC(ii)+distSRCtoP(ii))));
end


barrierH = 3;       % Barrier height
barrierW = 6;       % Barrier width

numbOfElemZ = 201;   % Number of elements along z-axis
numbOfElemY = 401;   % Number of elements along y-axis

elemArea = (barrierH/numbOfElemZ)*(barrierW/numbOfElemY);   % Element area


%% Ray hits at barrier - [x y z]

rayHits = zeros(numbOfElemY*numbOfElemZ,3);
for jj=0:numbOfElemZ-1
    for ii=0:numbOfElemY-1
    rayHits((ii+1)+(numbOfElemY*jj),2) = ...
        (0.5*barrierW/numbOfElemY) + (ii.*barrierW/numbOfElemY) - (barrierW/2);
    end
end
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        rayHits((1+(jj*(numbOfElemY))):(numbOfElemY+(jj*(numbOfElemY))),3) = ...
            - (0.5*barrierH/numbOfElemZ) + ((jj+1).*barrierH/numbOfElemZ);
    end
end



%% Ray propagation Time and Sample

rayDelays = zeros(length(rayHits(:,1)),length(rec(:,1)));
```

```
raySampleFloat = zeros(length(rayDelays),length(rec(:,1)));
for jj = 1:length(rec(:,1))
    for ii = 1:length(rayHits(:,1))
        rayDelays(ii,jj) = ...
            (norm(rayHits(ii,:)-src) + norm(rec(jj,:)-rayHits(ii,:)))/CAIR;
        raySampleFloat(ii,jj) = FSAMP.*rayDelays(ii,jj);
    end
end


minRaySampleFloat = min(raySampleFloat)-floor(min(raySampleFloat));


% Ray sample number
raySample = zeros(length(rayHits(:,1)),length(rec(:,1)));
for ii = 1:length(rec(:,1))
    raySample(:,ii) = floor(raySampleFloat(:,ii)-minRaySampleFloat(ii));
end
minRaySample = min(raySample);



%% Noise Barrier Color Mapping

wallCorners = zeros(numbOfElemY+1,numbOfElemZ+1);
for ii=1:numbOfElemY+1
    wallCorners(ii,:) = linspace(0,barrierH,numbOfElemZ+1);
end


zVec = linspace(0,barrierH,numbOfElemZ+1);
xVec = zeros(1,4);
yVec = linspace(-barrierW/2,barrierW/2,length(wallCorners(:,1)));


patchesY = zeros(numbOfElemY*numbOfElemZ,4);
for jj = 0:numbOfElemZ-1
    for ii = 1:numbOfElemY
        patchesY(ii+(numbOfElemY*jj),:) = ...
            [yVec(ii) yVec(ii+1) yVec(ii+1) yVec(ii)];
    end
end


patchesZ = zeros(numbOfElemY*numbOfElemZ,4);
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        patchesZ(ii+(jj*(numbOfElemY)),:) = ...
            [zVec(jj+1) zVec(jj+1) zVec(jj+2) zVec(jj+2)];
    end
end



%% A plot of the nth sample noise barrier intersection

edgeColoor = 'none';

figure(2);
plot3(src(1),src(2),src(3),'k o','MarkerSize',20,'MarkerFaceColor','b');
text(src(1),src(2)-0.25,src(3),'\color{black}Source','FontSize',14, ...
    'HorizontalAlignment','left');
for ii = 1:length(rec(:,1))
    hold on;
    plot3(rec(ii,1),rec(ii,2),rec(ii,3),'k d','MarkerSize',20, ...
        'MarkerFaceColor','b');
end
text(rec(1),rec(2)-0.25,rec(3),'\color{black}Receiver','FontSize',14, ...
    'HorizontalAlignment','left');
for ii = 1:numbOfElemY*numbOfElemZ
    hold on;
    patch(xVec,patchesY(ii,:),patchesZ(ii,:),[0.4 0.8 0.6], ...
        'EdgeColor',edgeColoor);
    for jj = 1:length(minRaySample)
        if raySample(ii,jj)-minRaySample(jj) == 0
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'r', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 1
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'g', ...
                'EdgeColor',edgeColoor);
```

```
        end
        if raySample(ii,jj)-minRaySample(jj) == 2
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'y', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 3
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'m', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 3
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'y', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 4
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'m', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 5
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'y', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 6
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'m', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 7
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'y', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 8
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'m', ...
                'EdgeColor',edgeColoor);
        end
        if raySample(ii,jj)-minRaySample(jj) == 9
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'y', ...
                'EdgeColor',edgeColoor);
        end
    end
end
hold on;
sector1 = plot3(1000,1000,1000,'r o','MarkerSize',18,'MarkerFaceColor','r');
hold on;
sector2 = plot3(1000,1000,1000,'g o','MarkerSize',18,'MarkerFaceColor','g');
hold on;
sector3 = plot3(1000,1000,1000,'y o','MarkerSize',18,'MarkerFaceColor','y');
hold on;
sector4 = plot3(1000,1000,1000,'m o','MarkerSize',18,'MarkerFaceColor','m');
hold off;
set(gca,'FontSize',14);
axis([-5 5 -3 3 0 5]);
legend([sector1 sector2 sector3 sector4],'1st sample','2nd sample', ...
    '3rd sample','etc...');
xlabel('x-axis [m]');
ylabel('y-axis [m]');
zlabel('z-axis [m]');
title('The {\itn}th sample barrier intersection area', ...
    'FontWeight','bold','Fontsize',18);
grid off;
```

B.5 Noise barrier zone division sketch

A file that was used to build the noise barrier zone division sketch in Figure 2.8.

```
%% A script that builds a sketch of the noise barrier zone division
% by Anders Isebakke

FSAMP = 24000;
CAIR = 343;


%% The Geometry

src = [-4.5 0 1.5];       % Source position

xREC = 4.5;
yREC = 0;
zREC = linspace(0.5,2.5,9);

rec = zeros(length(zREC),3);     % Receiver position
rec(:,1) = xREC;
rec(:,2) = yREC;
rec(:,3) = zREC;

edge = 3;   % The barrier height


% Coordiates at where the direct sound rays hits the barrier
barrierHit = zeros(length(rec(:,1)),1);
for ii = 1:length(barrierHit)
    barrierHit(ii) = ...
        (rec(ii,3) - rec(ii,1)*((rec(ii,3)-src(3))/(rec(ii,1)-src(1))));
end

% Distance between SRC and barrier hit
distSRCtoP = zeros(length(rec(:,1)),1);
for ii = 1:length(distSRCtoP)
    distSRCtoP(ii) = norm([0 0 barrierHit(ii)] - src);
end

% Distance between REC and barrier hit
distPtoREC = zeros(length(rec(:,1)),1);
for ii = 1:length(distPtoREC)
    distPtoREC(ii) = norm([0 0 barrierHit(ii)] - rec(ii,:));
end

% The 1st sample barrier intersection radius
sampelRadius = zeros(length(rec(:,1)),1);
for ii = 1:length(sampelRadius)
    sampelRadius(ii) = ...
        sqrt(2.*(CAIR/FSAMP).*((distPtoREC(ii)*distSRCtoP(ii))./ ...
        (distPtoREC(ii)+distSRCtoP(ii))));
end



%% The simulated semi-transparent barrier

barrierH = 3;        % Barrier height
barrierW = 6;       % Barrier width

numbOfElemZ = 51;    % Number of elements along z-axis
numbOfElemY = 101;    % Number of elements along y-axis

elemArea = (barrierH/numbOfElemZ)*(barrierW/numbOfElemY);    % Element area



%% Ray hits at barrier - [x y z]

rayHits = zeros(numbOfElemY*numbOfElemZ,3);
for jj=0:numbOfElemZ-1
    for ii=0:numbOfElemY-1
    rayHits((ii+1)+(numbOfElemY*jj),2) = ...
```

```
        (0.5*barrierW/numbOfElemY) + (ii.*barrierW/numbOfElemY) - (barrierW/2);
    end
end
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        rayHits((1+(jj*(numbOfElemY))):(numbOfElemY+(jj*(numbOfElemY))),3) = ...
            - (0.5*barrierH/numbOfElemZ) + ((jj+1).*barrierH/numbOfElemZ);
    end
end



%% Time dealys and sample numbers of the rays

% The time delay of each ray
rayDelays = zeros(length(rayHits(:,1)),length(rec(:,1)));

% The incomming sampel of each ray - float number!
raySampleFloat = zeros(length(rayDelays),length(rec(:,1)));

for jj = 1:length(rec(:,1))
    for ii = 1:length(rayHits(:,1))
        rayDelays(ii,jj) = ...
            (norm(rayHits(ii,:)-src) + norm(rec(jj,:)-rayHits(ii,:)))/CAIR;
        raySampleFloat(ii,jj) = FSAMP.*rayDelays(ii,jj);
    end
end

% A parameter used to obtain the 1st sample regions (a small time-shift)
minRaySampleFloat = min(raySampleFloat)-floor(min(raySampleFloat));


% The incomming sampel of each ray - integer number!
raySample = zeros(length(rayHits(:,1)),length(rec(:,1)));
for ii = 1:length(rec(:,1))
    raySample(:,ii) = floor(raySampleFloat(:,ii)-minRaySampleFloat(ii));
end

% The direct sound rays' sample number
minRaySample = min(raySample);



%% Noise Barrier color mapping

wallCorners = zeros(numbOfElemY+1,numbOfElemZ+1);
for ii=1:numbOfElemY+1
    wallCorners(ii,:) = linspace(0,barrierH,numbOfElemZ+1);
end


zVec = linspace(0,barrierH,numbOfElemZ+1);
xVec = zeros(1,4);
yVec = linspace(-barrierW/2,barrierW/2,length(wallCorners(:,1)));


patchesY = zeros(numbOfElemY*numbOfElemZ,4);
for jj = 0:numbOfElemZ-1
    for ii = 1:numbOfElemY
        patchesY(ii+(numbOfElemY*jj),:) = ...
            [yVec(ii) yVec(ii+1) yVec(ii+1) yVec(ii)];
    end
end


patchesZ = zeros(numbOfElemY*numbOfElemZ,4);
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        patchesZ(ii+(jj*(numbOfElemY)),:) = ...
            [zVec(jj+1) zVec(jj+1) zVec(jj+2) zVec(jj+2)];
    end
end



%% A plot of the noise barrier zone division

edgeColoor = [0.2 0.2 0.2];

figure(1);
```

```
plot3(src(1),src(2),src(3),'k o','MarkerSize',20, ...
    'MarkerFaceColor','b');
text(src(1),src(2)-0.25,src(3),'\color{black}S_G', ...
    'FontSize',14,'HorizontalAlignment','left');
for ii = 1:length(rec(:,1))
    hold on;
    plot3(rec(ii,1),rec(ii,2),rec(ii,3),'k d', ...
        'MarkerSize',20,'MarkerFaceColor','b');
end
text(rec(6,1),rec(6,2)-0.25,rec(6,3),'\color{black}R_G', ...
    'FontSize',14,'HorizontalAlignment','left');
for ii = 1:numbOfElemY*numbOfElemZ
    hold on;
    patch(xVec,patchesY(ii,:),patchesZ(ii,:),[0.4 0.8 0.6], ...
        'EdgeColor',edgeColoor);
    for jj = 1:length(minRaySample)
        if raySample(ii,jj)-minRaySample(jj) == 0
            patch(xVec,patchesY(ii,:),patchesZ(ii,:),'r');
        end
    end
end
hold on;
zone1 = plot3(1000,1000,1000,'r o','MarkerSize',18,'MarkerFaceColor','r');
hold on;
zone2 = plot3(1000,1000,1000,'o','MarkerSize',18, ...
    'MarkerEdgeColor',[0.4 0.8 0.6],'MarkerFaceColor',[0.4 0.8 0.6]);
hold off;
set(gca,'FontSize',14);
axis([-5 5 -3 3 0 5]);
legend([zone1 zone2],'Mid/high peak zone','Bass tail zone');
xlabel('x-axis [m]');
ylabel('y-axis [m]');
zlabel('z-axis [m]');
title('The final semi-transparent noise barrier design', ...
    'FontWeight','bold','Fontsize',18);
grid off;
```

B.6 Impulse response shaped by two transmission zones

A file that was used to build the sketch of the concept of an impulse response being shaped by the two transmission zones in Figure 2.9.

```
%% A script on the concept of a two-zones shaped impulse response
% by Anders Isebakke


%% Initialize the IR matrix

longestIR = load(['MatlabOUT\NoiseBarrier_infinite_freefield_OUT_1_', ...
    num2str(length(receivers(:,1))),'_ir.mat']);
ir = zeros(int64(1.5*length(longestIR.irtot)),length(receivers(:,1)));

tvec = 0:1/FSAMP:(length(ir(:,1))-1)/FSAMP; % Time vector

for ii = 1:length(receivers(:,1))
    SRCtemp = load(['MatlabOUT\NoiseBarrier_infinite_freefield_OUT_1_', ...
        num2str(ii),'_ir.mat']);
        ir(1:length(SRCtemp.irtot),ii) = SRCtemp.irtot;
end

[maxIR,maxIRsamp] = max(ir);    % The value and sampel nr of the peak



%% Make a figure

figure(1);  % Impulse responses -two zones
plot1 = stem(1000.*tvec(maxIRsamp+9:maxIRsamp+180), ...
    ir(maxIRsamp+9:maxIRsamp+180),'Color',[0.4 0.8 0.6],'LineWidth',3, ...
    'MarkerEdgeColor','none');
text(1000.*tvec(maxIRsamp+67),2.5*ir(maxIRsamp+67), ...
    '\color{black}Bass tail zone','FontSize',20, ...
    'HorizontalAlignment','left');
hold on;
plot2 = stem(1000.*tvec(maxIRsamp:maxIRsamp+8), ...
    ir(maxIRsamp:maxIRsamp+8),'r','LineWidth',3,'MarkerEdgeColor','none');
text(1000.*tvec(maxIRsamp+2) + 0.1,ir(maxIRsamp+2), ...
    '\color{black}Mid/High peak zone','FontSize',20, ...
    'HorizontalAlignment','left');
hold off;
set(gca,'FontSize',14);
xlabel('Time [ms]');
title('Impulse response of an edge diffraction case', ...
    'FontWeight','bold','Fontsize',16);
axis([28 37 -0.0005 0.008]);
```

B.7 Final developed semi-transparent modelling

A file that includes the entire procedure/processing of the final developed semi-transparent noise barrier modelling.

```
%% The Semi-Transparent Noise Barrier Modelling
% by Anders Isebakke


%% Import the IR_ref files

longestIR = load(['MatlabOUT\NoiseBarrier_infinite_freefield_OUT_1_', ...
    num2str(length(receivers(:,1))),'_ir.mat']);

% The IR_ref matrix
ir = zeros(int64(1.5*length(longestIR.irtot)),length(receivers(:,1)));
for ii = 1:length(receivers(:,1))
    SRCtemp = load(['MatlabOUT\NoiseBarrier_infinite_freefield_OUT_1_', ...
        num2str(ii),'_ir.mat']);
    for jj = 1:length(SRCtemp.irtot)
        ir(jj,ii) = full(SRCtemp.irtot(jj));
    end
end


[maxIR,maxIRsamp] = max(ir);    % The value and sampel nr of the peak


irPeak = zeros(1,length(maxIR));
for ii = 1:length(irPeak)
    irPeak(ii) = ir(maxIRsamp(ii),ii) + ir(maxIRsamp(ii)-1,ii);
end



%% The Geometry

src = sources;      % Source position
rec = receivers;    % Receiver positions (an array along z-axis)

edge = 3;           % The barrier height


% Coordiates at where the direct sound rays hits the barrier
barrierHit = zeros(length(rec(:,1)),1);
for ii = 1:length(barrierHit)
    barrierHit(ii) = (rec(ii,3) - rec(ii,1)* ...
        ((rec(ii,3)-src(3))/(rec(ii,1)-src(1))));
end

% Distance between SRC and barrier hit
distSRCtoP = zeros(length(rec(:,1)),1);
for ii = 1:length(distSRCtoP)
    distSRCtoP(ii) = norm([0 0 barrierHit(ii)] - src);
end

% Distance between REC and barrier hit
distPtoREC = zeros(length(rec(:,1)),1);
for ii = 1:length(distPtoREC)
    distPtoREC(ii) = norm([0 0 barrierHit(ii)] - rec(ii,:));
end

% The 1st sample barrier intersection radius
sampelRadius = zeros(length(rec(:,1)),1);
for ii = 1:length(sampelRadius)
    sampelRadius(ii) = sqrt(2.*(CAIR/FSAMP).* ...
        ((distPtoREC(ii)*distSRCtoP(ii))./(distPtoREC(ii)+distSRCtoP(ii))));
end



%% The truncated dS
```

```
% dS - untruncated
dStemp = pi.*(sampelRadius.^2);

d1 = barrierHit;        % distance from wall intersection to floor
d2 = edge-barrierHit;  % distance from wall intersection to wall edge

% angle of the included circle sector - given by floor limitation
beta1 = 2.*(pi-real(acos(d1./sampelRadius)));

% angle of the included circle sector - given by wall height limitation
beta2 = 2.*(pi-real(acos(d2./sampelRadius)));


% The sector S1
Ssec1 = dStemp.*(beta1/(2*pi));
Stri1 = real(d1.*sqrt((sampelRadius.^2)-(d1.^2)));

% The sector S2
Ssec2 = dStemp.*(beta2/(2*pi));
Stri2 = real(d2.*sqrt((sampelRadius.^2)-(d2.^2)));


% The trunctation factor: (S1+S2-S)/S
trunc = ((Ssec1+Stri1)+(Ssec2+Stri2)-(dStemp))./dStemp;


% dS - truncated
dS = dStemp.*trunc;



%% The tremble/mid region transmission factors T1

% The transmission factors found by EDBmainISESx
tau1 = irPeak(1:9)'.*distSRCtoP(1:9).*distPtoREC(1:9)./dS(1:9);

% The polynomial fitted curve being used to build the tremble/mid region
tau1PolyCoeff = polyfit(edge-rec(1:9,3),tau1,1);
tau1Poly = tau1PolyCoeff(1).*(edge-rec(1:9,3)) + tau1PolyCoeff(2);


% A plot of the polynomial fitted curve
figure(1);
plot(edge-rec(1:9,3),tau1);
hold on;
plot(edge-rec(1:9,3),tau1Poly,'r');
hold off;



%% The simulated semi-transparent barrier

barrierH = 3;       % Barrier height
barrierW = 6;       % Barrier width

numbOfElemZ = 101;   % Number of elements along z-axis
numbOfElemY = 201;   % Number of elements along y-axis

elemArea = (barrierH/numbOfElemZ)*(barrierW/numbOfElemY);   % Element area



%% Ray hits at simulated barrier - [x y z]

rayHits = zeros(numbOfElemY*numbOfElemZ,3);
for jj=0:numbOfElemZ-1
    for ii=0:numbOfElemY-1
    rayHits((ii+1)+(numbOfElemY*jj),2) = (0.5*barrierW/numbOfElemY) + ...
        (ii.*barrierW/numbOfElemY) - (barrierW/2);
    end
end
for jj=0:numbOfElemZ-1
    for ii=1:numbOfElemY
        rayHits((1+(jj*(numbOfElemY))):(numbOfElemY+(jj*(numbOfElemY))),3) = ...
            - (0.5*barrierH/numbOfElemZ) + ((jj+1).*barrierH/numbOfElemZ);
    end
end
```

```
%% Time delays and sample numbers of the rays

% The time delay of each ray
rayDelays = zeros(length(rayHits(:,1)),length(rec(:,1)));

% The incomming sampel of each ray - NB: float number!
raySampleFloat = zeros(length(rayDelays),length(rec(:,1)));

for jj = 1:length(rec(:,1))
    for ii = 1:length(rayHits(:,1))
        rayDelays(ii,jj) = (norm(rayHits(ii,:)-src) + ...
            norm(rec(jj,:)-rayHits(ii,:)))/CAIR;
        raySampleFloat(ii,jj) = FSAMP.*rayDelays(ii,jj);
    end
end


% A parameter used to obtain the 1st sample regions (a small time-shift)
minRaySampleFloat = min(raySampleFloat)-floor(min(raySampleFloat));


% The incomming sampel of each ray - integer number!
raySample = zeros(length(rayHits(:,1)),length(rec(:,1)));
for ii = 1:length(rec(:,1))
    raySample(:,ii) = floor(raySampleFloat(:,ii)-minRaySampleFloat(ii));
end


% Distance and Samples between SRC and REC
distSRCtoREC = zeros(length(rec(:,1)),1);
sampSRCtoREC = zeros(length(rec(:,1)),1);
for ii = 1:length(distSRCtoREC)
    distSRCtoREC(ii) = norm(rec(ii,:) - src);
    sampSRCtoREC(ii) = ...
        floor(FSAMP*distSRCtoREC(ii)/CAIR - minRaySampleFloat(ii));
end


% The direct sound rays' sample number
minRaySample = min(raySample);



%% The Transmission Coefficients

% An initial constant "bass tail zone" value given to all subareas
tauSubArea = (1/10).*mean(tau1).*ones(1,length(rayHits(:,1)));


% The variable "mid/high peak zone" transmission coeff is added
for ii = 1:numbOfElemY*numbOfElemZ
    for jj = 1:length(rec(1:9,3))
        if raySample(ii,jj)-minRaySample(jj) == 0
            tauSubArea(ii) = ...
                tau1PolyCoeff(1).*(edge-rayHits(ii,3)) + tau1PolyCoeff(2);
        end
    end
end

tauSubAreaMatrix = zeros(length(rec(:,1)),length(tauSubArea));

for ii=1:length(rec(:,1))
    tauSubAreaMatrix(ii,:) = tauSubArea;
end



%% The Scattering Coefficients

% Values based on cosinus scattering
rayScatt = zeros(length(rec(:,1)),length(rayHits(:,1)));
for jj=1:length(rec(:,1))
    for ii=1:length(rayHits(:,1))
        rayScatt(jj,ii) = (rec(jj,1)./norm(rayHits(ii,:)-rec(jj,:)));
    end
end
```

```
%% The noise barrier Filtering Process

% Initialize the rays by Dirac pulses (input)
rayDiracs = ones(length(rec(:,1)),length(rayHits(:,1)));


% Find travel lengths for SRCtoHIT and HITtoREC
RraySRCtoHIT = zeros(length(rec(:,1)),length(rayHits(:,1)));
RrayHITtoREC = zeros(length(rec(:,1)),length(rayHits(:,1)));
for jj = 1:length(rec(:,1))
    for ii = 1:length(rayHits(:,1))
        RraySRCtoHIT(jj,ii) = norm(rayHits(ii,:)-src);
        RrayHITtoREC(jj,ii) = norm(rayHits(ii,:)-rec(jj,:));
    end
end



% The filtering of each ray
rayTemp = ...
    (rayDiracs./(RraySRCtoHIT.*RrayHITtoREC)).*tauSubAreaMatrix.*rayScatt*elemArea;



% The simulated IR - being built on basis of all the rays
simuIR = zeros(2*max(raySample(end,:)),length(rec(:,1)));
for jj=1:length(rec(:,1))
    for ii=1:length(rayHits(:,1))
        simuIR(raySample(ii,jj),jj) = ...
            simuIR(raySample(ii,jj),jj)+rayTemp(jj,ii);
    end
end



% A plot of the IRs
count = 0;
figure(2);
for ii=10:2:length(rec(:,1))
    count = count+1;
    subplot(2,3,count);
    plot(ir(:,ii),'r','LineWidth',5);
    hold on;
    plot(simuIR(:,ii),'g','LineWidth',3);
    hold off;
end
clear count;



%% Perform Fourier Transform

nfft = FSAMP/4;      % Number of FFT elements
fvec = 1:4:FSAMP;    % Frequency vector


% Frequency Responses (by FFT)
irFreq = fft(full(ir),nfft);
simuIRFreq = fft(simuIR,nfft);


% A plot of the FRs
count = 0;
figure(4);
for ii=10:2:length(rec(:,1))
    count = count+1;
    subplot(2,3,count);
    semilogx(fvec,20.*log10(abs(irFreq(:,ii))),'r','LineWidth',3);
    hold on;
    semilogx(fvec,20.*log10(abs(simuIRFreq(:,ii))),'g','LineWidth',3);
    hold on;
    axis([45 10000 -60 -20]);
    hold off;
    grid on;
end
clear count;



%% Octave band filtering
```

```
% Octave band filtered IRs - EDBmainISESx
trueIR_63 = zeros(size(ir));
trueIR_125 = zeros(size(ir));
trueIR_250 = zeros(size(ir));
trueIR_500 = zeros(size(ir));
trueIR_1k = zeros(size(ir));
trueIR_2k = zeros(size(ir));
trueIR_4k = zeros(size(ir));
trueIR_8k = zeros(size(ir));

% Octave band filtered IRs - Semi-Transparent Method
simuIR_63 = zeros(size(simuIR));
simuIR_125 = zeros(size(simuIR));
simuIR_250 = zeros(size(simuIR));
simuIR_500 = zeros(size(simuIR));
simuIR_1k = zeros(size(simuIR));
simuIR_2k = zeros(size(simuIR));
simuIR_4k = zeros(size(simuIR));
simuIR_8k = zeros(size(simuIR));


for ii=1:length(rec(:,1))
    trueIR_63(:,ii) = oktavbandfilter(full(ir(:,ii)),1);
    trueIR_125(:,ii) = oktavbandfilter(full(ir(:,ii)),2);
    trueIR_250(:,ii) = oktavbandfilter(full(ir(:,ii)),3);
    trueIR_500(:,ii) = oktavbandfilter(full(ir(:,ii)),4);
    trueIR_1k(:,ii) = oktavbandfilter(full(ir(:,ii)),5);
    trueIR_2k(:,ii) = oktavbandfilter(full(ir(:,ii)),6);
    trueIR_4k(:,ii) = oktavbandfilter(full(ir(:,ii)),7);
    trueIR_8k(:,ii) = oktavbandfilter(full(ir(:,ii)),8);

    simuIR_63(:,ii) = oktavbandfilter(simuIR(:,ii),1);
    simuIR_125(:,ii) = oktavbandfilter(simuIR(:,ii),2);
    simuIR_250(:,ii) = oktavbandfilter(simuIR(:,ii),3);
    simuIR_500(:,ii) = oktavbandfilter(simuIR(:,ii),4);
    simuIR_1k(:,ii) = oktavbandfilter(simuIR(:,ii),5);
    simuIR_2k(:,ii) = oktavbandfilter(simuIR(:,ii),6);
    simuIR_4k(:,ii) = oktavbandfilter(simuIR(:,ii),7);
    simuIR_8k(:,ii) = oktavbandfilter(simuIR(:,ii),8);
end



% Octaveband energy level - EDBmainISESx
energyTrueIR_63 = 10.*log10(sum(abs(trueIR_63).^2));
energyTrueIR_125 = 10.*log10(sum(abs(trueIR_125).^2));
energyTrueIR_250 = 10.*log10(sum(abs(trueIR_250).^2));
energyTrueIR_500 = 10.*log10(sum(abs(trueIR_500).^2));
energyTrueIR_1k = 10.*log10(sum(abs(trueIR_1k).^2));
energyTrueIR_2k = 10.*log10(sum(abs(trueIR_2k).^2));
energyTrueIR_4k = 10.*log10(sum(abs(trueIR_4k).^2));
energyTrueIR_8k = 10.*log10(sum(abs(trueIR_8k).^2));

% Octaveband energy level - Semi-transparent method
energySimuIR_63 = 10.*log10(sum(abs(simuIR_63).^2));
energySimuIR_125 = 10.*log10(sum(abs(simuIR_125).^2));
energySimuIR_250 = 10.*log10(sum(abs(simuIR_250).^2));
energySimuIR_500 = 10.*log10(sum(abs(simuIR_500).^2));
energySimuIR_1k = 10.*log10(sum(abs(simuIR_1k).^2));
energySimuIR_2k = 10.*log10(sum(abs(simuIR_2k).^2));
energySimuIR_4k = 10.*log10(sum(abs(simuIR_4k).^2));
energySimuIR_8k = 10.*log10(sum(abs(simuIR_8k).^2));


% Octaveband energy level ERROR
oBandErr = [(energySimuIR_63 - energyTrueIR_63) ; ...
    (energySimuIR_125 - energyTrueIR_125) ; ...
    (energySimuIR_250 - energyTrueIR_250) ; ...
    (energySimuIR_500 - energyTrueIR_500) ; ...
    (energySimuIR_1k - energyTrueIR_1k) ; ...
    (energySimuIR_2k - energyTrueIR_2k) ; ...
    (energySimuIR_4k - energyTrueIR_4k) ; ...
    (energySimuIR_8k - energyTrueIR_8k)];


% A plot of the octaveband energy level error
figure(5);
plot([0 10],[0 0],'k --','LineWidth',3);
hold on;
plotH1 = plot(oBandErr(:,end),': o b','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','b');
hold on;
```

```
plotH2 = plot(oBandErr(:,end-2),': o g','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','g');
hold on;
plotH3 = plot(oBandErr(:,end-4),': o r','LineWidth',5,'MarkerSize',8, ...
    'MarkerFaceColor','r');
hold on;
plotH4 = plot(oBandErr(:,end-6),': o m','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','m');
hold on;
plotH5 = plot(oBandErr(:,end-8),': o c','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','c');
hold off;
set(gca,'FontSize',14);
legend([plotH1 plotH2 plotH3 plotH4 plotH5],'rec nr1','rec nr2', ...
    'rec nr3','rec nr4','rec nr5','Location','SouthEast');
axis([0 10 -20 20]);
text(1,8,'\color{blue}63Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(2,8,'\color{blue}125Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(3,8,'\color{blue}250Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(4,8,'\color{blue}500Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(5,8,'\color{blue}1kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(6,8,'\color{blue}2kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(7,8,'\color{blue}4kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(8,8,'\color{blue}8kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
xlabel('Octave band');
ylabel('[dB]');
title('Mean octaveband error','FontWeight','bold','Fontsize',16);
grid on;
```

B.8 Mean octaveband error calculation scripts

A file that was used to create the mean octaveband error plots in Figure 3.1 and Figure 3.6.

```
%% A script for Mean Octaveband Error Analyze
% by Anders Isebakke



%% Initialize the error matrix

numbOfFiles = 100;
errorMatrix = zeros(8,11*numbOfFiles);

for ii = 1:numbOfFiles
    errTemp = load(['oBandErr',num2str(ii),'.mat']);
    errorMatrix(:,1+(11*(ii-1)):11+(11*(ii-1))) = errTemp.oBandErr;
end



%% Mean, Min and Max

meanError = mean(errorMatrix,2);
maxError = max(errorMatrix,[],2);
minError = min(errorMatrix,[],2);



%% Interquartile range

interquartileLower = zeros(8,1);
interquartileUpper = zeros(8,1);
for ii=1:length(errorMatrix(:,1))

    [y,x] = hist(errorMatrix(ii,:),50);
    y = y/sum(y);

    cy = cumsum(y);

    lo = find(cy>0.125);
    lo2 = lo(1);
    lo1 = lo2-1;
    polyFitCoeffLo = polyfit([x(lo1) x(lo2)],[cy(lo1) cy(lo2)],1);
    interquartileLower(ii) = (0.125 - polyFitCoeffLo(2))/polyFitCoeffLo(1);

    hi = find(cy>0.875);
    hi2 = hi(1);
    hi1 = hi2-1;
    polyFitCoeffHi = polyfit([x(hi1) x(hi2)],[cy(hi1) cy(hi2)],1);
    interquartileUpper(ii) = (0.875 - polyFitCoeffHi(2))/polyFitCoeffHi(1);

end



%% A plot of the mean octaveband error

figure(1);
plot([0 10],[0 0],'k --','LineWidth',3);
hold on;
for ii=1:8
    plotH1 = plot([ii ii],[minError(ii) maxError(ii)],': s r', ...
        'LineWidth',3,'MarkerFaceColor','r');
    hold on;
end
for ii=1:8
    plotH2 = plot([ii ii],[interquartileLower(ii) interquartileUpper(ii)], ...
        '- s g','LineWidth',3,'MarkerFaceColor','g');
    hold on;
end
```

```
plotH3 = plot(meanError,'- o','LineWidth',5,'MarkerSize',10, ...
    'MarkerFaceColor','b');
hold off;
set(gca,'FontSize',14);
axis([0 9 -15 20]);
text(1,17,'\color{blue}63Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(2,17,'\color{blue}125Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(3,17,'\color{blue}250Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(4,17,'\color{blue}500Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(5,17,'\color{blue}1kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(6,17,'\color{blue}2kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(7,17,'\color{blue}4kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(8,17,'\color{blue}8kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
legend([plotH3 plotH1 plotH2],'Mean error','Max/min error', ...
    'Interquartile range','Location','SouthWest');
xlabel('Octaveband');
ylabel('[dB]');
title('Mean octaveband error in the shadow zone', ...
    'FontWeight','bold','Fontsize',16);
grid on;
```

A file that was used to create the mean octaveband error plots in Figure 3.2 and Figure 3.3.

```
%% A script for Mean Octaveband Error Analyze
% by Anders Isebakke



%% Initialize the error matrix

numbOfFiles = 100;
errorMatrix = zeros(8,14*numbOfFiles);
errorCorrPMatrix = zeros(8,14*numbOfFiles);



for ii = 1:numbOfFiles
    errTemp = load(['oBandErr',num2str(ii),'.mat']);
    errorMatrix(:,1+(14*(ii-1)):14+(14*(ii-1))) = errTemp.oBandErr;
end



%% Pick out the wanted off-axis receiver errors

yadd0 = zeros(8,numbOfFiles);
yadd2 = zeros(8,numbOfFiles);
yadd4 = zeros(8,numbOfFiles);
yadd6 = zeros(8,numbOfFiles);
yadd8 = zeros(8,numbOfFiles);

for ii = 1:numbOfFiles
    yadd0(:,ii) = errorMatrix(:,10+14*(ii-1));
    yadd2(:,ii) = errorMatrix(:,11+14*(ii-1));
    yadd4(:,ii) = errorMatrix(:,12+14*(ii-1));
    yadd6(:,ii) = errorMatrix(:,13+14*(ii-1));
    yadd8(:,ii) = errorMatrix(:,14+14*(ii-1));
end

meanYadd0 = mean(yadd0,2);
meanYadd2 = mean(yadd2,2);
meanYadd4 = mean(yadd4,2);
meanYadd6 = mean(yadd6,2);
meanYadd8 = mean(yadd8,2);



%% Make a plot of the mean octaveband error - off-axis

figure(1);
plot([0 10],[0 0],'k --','LineWidth',3);
hold on;
plotH1 = plot(meanYadd0,'- o g','LineWidth',4,'MarkerSize',8, ...
    'MarkerFaceColor','g');
hold on;
plotH2 = plot(meanYadd2,'- o c','LineWidth',4,'MarkerSize',8, ...
    'MarkerFaceColor','c');
hold on;
plotH3 = plot(meanYadd4,'- o b','LineWidth',4,'MarkerSize',8, ...
    'MarkerFaceColor','b');
hold on;
plotH4 = plot(meanYadd6,'- o m','LineWidth',4,'MarkerSize',8, ...
    'MarkerFaceColor','m');
hold on;
plotH5 = plot(meanYadd8,'- o r','LineWidth',4,'MarkerSize',8, ...
    'MarkerFaceColor','r');
hold off;
set(gca,'FontSize',14);
legend([plotH1 plotH2 plotH3 plotH4 plotH5], ...
    '\Deltay = 0 m','\Deltay = 2 m','\Deltay = 4 m','\Deltay = 6 m', ...
    '\Deltay = 8 m','Location','SouthEast');
axis([0 10 -30 10]);
text(1,7,'\color{blue}63Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(2,7,'\color{blue}125Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
```

```
text(3,7,'\color{blue}250Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(4,7,'\color{blue}500Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(5,7,'\color{blue}1kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(6,7,'\color{blue}2kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(7,7,'\color{blue}4kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(8,7,'\color{blue}8kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
xlabel('Octaveband');
ylabel('[dB]');
title('Mean octaveband error:  off y-axis', ...
    'FontWeight','bold','Fontsize',16);
grid on;
```

A file that was used to create the mean octaveband error plots in Figure 3.4 and Figure 3.5.

```
%% A script for Mean Octaveband Error Analyze
% by Anders Isebakke



%% Initialize the error matrix

numbOfFiles = 100;
errorMatrix = zeros(8,18*numbOfFiles);
errorCorrPMatrix = zeros(8,18*numbOfFiles);

for ii = 1:numbOfFiles
    errTemp = load(['oBandErr',num2str(ii),'.mat']);
    errCorrPTemp = load(['oBandErrCorrP',num2str(ii),'.mat']);

    errorMatrix(:,1+(18*(ii-1)):18+(18*(ii-1))) = errTemp.oBandErr;
    errorCorrPMatrix(:,1+(18*(ii-1)):18+(18*(ii-1))) = ...
        errCorrPTemp.oBandErrCorrP;
end



%% Pick out the wanted near SZ receiver errors

SZmin2 = zeros(8,numbOfFiles);
SZmin1 = zeros(8,numbOfFiles);
SZadd0 = zeros(8,numbOfFiles);
SZadd1 = zeros(8,numbOfFiles);
SZadd2 = zeros(8,numbOfFiles);

SZadd1corrP = zeros(8,numbOfFiles);
SZadd2corrP = zeros(8,numbOfFiles);

SZadd2(:,1) = errorMatrix(:,18);

for ii = 1:numbOfFiles
    SZadd2(:,ii) = errorMatrix(:,18+18*(ii-1));
    SZadd1(:,ii) = errorMatrix(:,16+18*(ii-1));
    SZadd0(:,ii) = errorMatrix(:,14+18*(ii-1));
    SZmin1(:,ii) = errorMatrix(:,12+18*(ii-1));
    SZmin2(:,ii) = errorMatrix(:,10+18*(ii-1));

    SZadd2corrP(:,ii) = errorCorrPMatrix(:,18+18*(ii-1));
    SZadd1corrP(:,ii) = errorCorrPMatrix(:,16+18*(ii-1));
end

meanSZadd2 = mean(SZadd2,2);
meanSZadd1 = mean(SZadd1,2);
meanSZadd0 = mean(SZadd0,2);
meanSZmin1 = mean(SZmin1,2);
meanSZmin2 = mean(SZmin2,2);

meanSZadd2corrP = mean(SZadd2corrP,2);
meanSZadd1corrP = mean(SZadd1corrP,2);



%% Make a plot of the mean octaveband error - near SZ

figure(1);
plot([0 10],[0 0],'k --','LineWidth',3);
hold on;
plotH1 = plot(meanSZadd2,': o b','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','b');
hold on;
plotH2 = plot(meanSZadd1,'-- o b','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','b');
hold on;
plotH3 = plot(meanSZadd0,'- o r','LineWidth',5,'MarkerSize',10, ...
    'MarkerFaceColor','r');
hold on;
plotH4 = plot(meanSZmin1,'-- o m','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','m');
```

```
hold on;
plotH5 = plot(meanSZmin2,': o m','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','m');
hold off;
set(gca,'FontSize',14);
legend([plotH1 plotH2 plotH3 plotH4 plotH5], ...
    'z = z_{SZ} +2 m','z = z_{SZ} +1 m','z = z_{SZ} +0 m', ...
    'z = z_{SZ} -1 m','z = z_{SZ} -2 m','Location','SouthEast');
axis([0 10 -20 15]);
text(1,12,'\color{blue}63Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(2,12,'\color{blue}125Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(3,12,'\color{blue}250Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(4,12,'\color{blue}500Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(5,12,'\color{blue}1kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(6,12,'\color{blue}2kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(7,12,'\color{blue}4kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(8,12,'\color{blue}8kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
xlabel('Octaveband');
ylabel('[dB]');
title('Mean octaveband error:  around the source-receiver sight line', ...
    'FontWeight','bold','Fontsize',16);
grid on;



%% Make a plot of the mean octaveband error  -  near SZ

figure(2);
plot([0 10],[0 0],'k --','LineWidth',3);
hold on;
plotH6 = plot(meanSZadd2,': o b','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','b');
hold on;
plotH7 = plot(meanSZadd1,'-- o b','LineWidth',3,'MarkerSize',8, ...
    'MarkerFaceColor','b');
hold on;
plotH8 = plot(meanSZadd2corrP,': o g','LineWidth',4,'MarkerSize',10, ...
    'MarkerFaceColor','g');
hold on;
plotH9 = plot(meanSZadd1corrP,'-- o g','LineWidth',4,'MarkerSize',10, ...
    'MarkerFaceColor','g');
hold off;
set(gca,'FontSize',14);
legend([plotH6 plotH7 plotH8 plotH9], ...
    'Polarity error:           z = z_{SZ} +2 m', ...
    'Polarity error:           z = z_{SZ} +1 m', ...
    'Polarity correction:  z = z_{SZ} +2 m', ...
    'Polarity correction:  z = z_{SZ} +1 m','Location','SouthEast');
axis([0 10 -20 15]);
text(1,12,'\color{blue}63Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(2,12,'\color{blue}125Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(3,12,'\color{blue}250Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(4,12,'\color{blue}500Hz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(5,12,'\color{blue}1kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(6,12,'\color{blue}2kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(7,12,'\color{blue}4kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
text(8,12,'\color{blue}8kHz','FontSize',14, ...
    'HorizontalAlignment','center','BackgroundColor',[.7 .9 .7]);
xlabel('Octaveband');
ylabel('[dB]');
title('Mean octaveband error:  ...  polarity correction', ...
    'FontWeight','bold','Fontsize',16);
grid on;
```