UNIVERSITY OF MUSIC AND PERFORMING ARTS GRAZ
INSTITUTE OF ELECTRONIC MUSIC AND ACOUSTICS

Simulating the sound radiating from an actuator-driven mobile device display

Master's thesis

Author: Patrick Heidegger, BSc. Mt.: 01531610

Supervisor: Ass.Prof. DI Dr.rer.nat. Franz Zotter

Assessor: O.Univ.Prof. Mag.art. DI Dr.techn. Robert Höldrich

Date: Graz, June 17, 2022

Acknowledgments

First and foremost, I want to express my deepest gratitude to my supervisor, Dr. Franz Zotter, for supporting me incredibly well throughout all aspects of this thesis, providing an extraordinary amount of motivation, creativity, and expertise. I am especially thankful for all the extensive and intriguing phone calls, the detailed feedback that substantially improved my writing, and the flexibly scheduled dedication to our discussions.

Further, a very special thanks to Sound Solutions Austria for giving me the great opportunity to work on this exciting topic. Thank you also for the financial support and for providing me with pleasant working conditions. Particularly, I wish to thank DI. Andreas Hintennach, my supervisor at Sound Solutions Austria, for his friendly and welcoming attitude and for flexibly supporting me with his expertise and practical knowledge throughout the practical part of this thesis. Thanks also to my other colleagues at Sound Solutions Austria for the interesting talks and the joint recreational activities.

I also like to thank my fellow students for having such a great time wrapping our heads around things we did not understand and for the evolved friendships that hopefully will remain in the future.

Thanks to my partner, Alexandra, who enriches every facet of my life and whose love and understanding supported me emotionally throughout this work.

Finally, my true thankfulness goes to my Parents, Raimund and Christine, who raised me with love, believed in me, and stood by my side all my life.

Thank you!

Abstract

The acoustic design of hands-free loudspeakers integrated into mobile devices such as smartphones and tablets is particularly challenging because of the trade-off between compactness and low-frequency output. In contrast to the commonly employed miniature voice-coil loudspeakers, an alternative approach considers vibrating actuators attached to the display to form a vibrating-display loudspeaker. The radiated sound field of a vibrating-display loudspeaker can be determined using acoustic measurements on a fine grid in the near field and far field. Further, the individual performance depends on the material properties of the display panel, various impedances defining the coupling to other surfaces of the device, and the particular actuator placement, requiring repeated assessment of the acoustic quality in the cyclic development process. To circumvent the need for extensive measurements that can only be done in acoustically treated rooms and exploiting specific mountings, one can simulate the sound radiation using laser-interferometry-based measurements of the device's surface vibration. This thesis studies five boundary-value-based simulation techniques: the Rayleigh I integral and the Equivalent Source Method (ESM) for half-space radiation in an infinite baffle, and taking into account the full-space setting: ESM, the Boundary Elements Method (BEM), and a combined-Rayleigh-integral formulation. The models are discussed and compared to sound-field measurements and finally employed to study the overall sound radiation of an actuator-driven smartphone dummy. The aim is to find a well-suited approach that allows for flexible application in the development process of vibrating-display loudspeakers and to gain a better understanding of their radiated sound field.

Kurzfassung

Die Entwicklung von Einbaulautsprechern für den Einsatz in mobilen Endgeräten wie Smartphones und Tablets beherbergt mehrere Problemstellungen. Ein Beispiel dafür ist der Kompromiss zwischen Lautsprechergröße und tieffrequenter Schallabstrahlung. Eine Alternative zu den konventionell eingesetzten miniatur-Tauchspulenwandlern besteht im Anbringen von Vibrationselementen (Aktuatoren) an der Rückseite des Bildschirms. Der angeregte Bildschirm fungiert dann als schallabstrahlender Plattenschwinger. Zur Bestimmung des abgestrahlten Schallfelds eines solchen Lautsprechers bedarf es akustischer Nah- und Fernfeldmessungen in einem fein abgetasteten räumlichen Raster. Die Schallabstrahlung hängt von den Materialeigenschaften, aber auch von verschiedenen Kopplungsimpedanzen zu den anderen Außenflächen, sowie der Platzierung der Aktuatoren ab. Der zyklische Entwicklungsprozess verlangt daher nach wiederkehrenden Messungen zur Bewertung der Qualität des abgestrahlten Schallfeldes. Solche Messungen sind oft langwierig und erfordern Räumlichkeiten mit entsprechender akustischer Ausstattung, sowie die Verwendung spezieller Befestigungen für das Gerät. Allerdings kann die akustische Messung mittels Laser-Interferometer-Messung der Oberflächenvibration und anschließender Simulation der Schallabstrahlung umgangen werden. Diese Arbeit untersucht insgesamt fünf auf Randwerte basierende Simulationstechniken: das Rayleigh I-Integral und die Äquivalentschallquellenmethode für ein Halbraumszenario, sowie Äquivalentschallquellenmethode, Randelementemethode und eine kombinierte Formulierung der beiden Rayleigh Integrale für ein Vollraumszenario. Die Modelle werden mit akustischen Referenzmessungen verglichen und schlussendlich angewandt, um das Schallfeld einer Aktuator-betriebenen Smartphone-Attrappe zu untersuchen. Ziel ist es, eine passende und flexibel einsetzbare Simulationsmethode für den Entwicklungsprozess der Lautsprecher zu finden und ein besseres Verständnis für deren abgestrahltes Schallfeld zu entwickeln.

Statuatory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

date	Patrick Heidegger

Contents

1 Introduction

This thesis was carried out in cooperation with Sound Solutions Austria[1]. Sound Solutions is a leading company in acoustics technology with focus on small-scale electroacoustic transducers. The company provided all tested devices, adapters, and measuring infrastructure, and it contributed the research question of this thesis.

The cyclic development process of loudspeakers requires recurring measurements to assess the acoustic quality of subsequent loudspeaker builds. This assessment is also necessary for loudspeakers used in hands-free devices, such as smartphones or tablets. While the traditional voice-coil-driven miniature loudspeakers are still the most common, a developing trend is to use vibrating-display loudspeakers instead. A vibrating-display loudspeaker constitutes a panel[2] loudspeaker that is formed by one or more vibrating actuators attached to the display of a hands-free device. The actuators introduce bending-wave vibrancy into the display and, via volumetric and structural coupling, into the device's back cover. Fig. 1.1 illustrates a vibrating-display loudspeaker, the internal coupling, and its sound radiation.

Acoustic measurements of vibrating-display loudspeakers can be more extensive than those of voice-coil-driven ones since here, the casing acts as an essential part of the loudspeaker. Hence, measurements cannot be conducted for the isolated loudspeaker surface but must include a whole casing that vibrates. Conventional small-scale loudspeakers are traditionally assessed by observing the acoustic response on a single concentric position in a short distance, which is possible due to their short measures and concentric directivity pattern. By contrast, due to the comparatively large measures of the casing, a vibrating-display loudspeaker requires acoustic measurements at larger distances, raising the demands placed on the measurement chamber. Further, the sound radiation of panel loudspeakers is more complex [1] and differs significantly from that of voice-coil loudspeakers. While panel loudspeakers ideally exhibit diffuse-like directivity patterns [2], [3], [4], they can also produce strong lobes at frequencies where few modes are excited in the material [5]. Hence, it is often not representative to assess the acoustic quality of vibrating-display loudspeakers only for a single observation point. Fig. 1.2 illustrates the different directivity patterns with balloon plots of a miniature voice-coil loudspeaker and an actuator-driven display.

Simulating the sound field from the measured surface velocities can be advantageous. Modern Laser-Doppler vibrometers produce low measurement noise, and no additional requirement for room size, nearby reflecting surfaces, or acoustic ground noise needs to be fulfilled. This is because the acoustic environment is of negligible influence on structural vibration. Further, a simulation can serve for more flexible analyses, and it is desirable to keep its cost low for, e.g., utilization in optimization. Evaluating the sound field on multiple receiver positions without remeasuring and the ability to link the radiated sound to surface parts allows the development of optimizing analysis tools, which are considered to be more practical when not having to perform repeated acoustic radiation measurements. Simulation moreover facilitates evaluating the sound radiation at arbitrarily large distances, without the need for far-field measurements in large anechoic chambers. Three approaches appear suitable for simulating the sound radiating from the device's vibrating surface: The Equivalent Source Method (ESM), Rayleigh integrals, and the Boundary Elements Method (BEM).

[1] Website of Sound Solutions Austria: www.sound-solutions-austria.com. Visited on: 10.05.2022

[2] Various names exist for this type of loudspeaker, e.g., flat-panel loudspeaker, distributed-mode loudspeaker, vibrating-panel loudspeaker, or, when multiple actuators are used, multi-actuator panel.

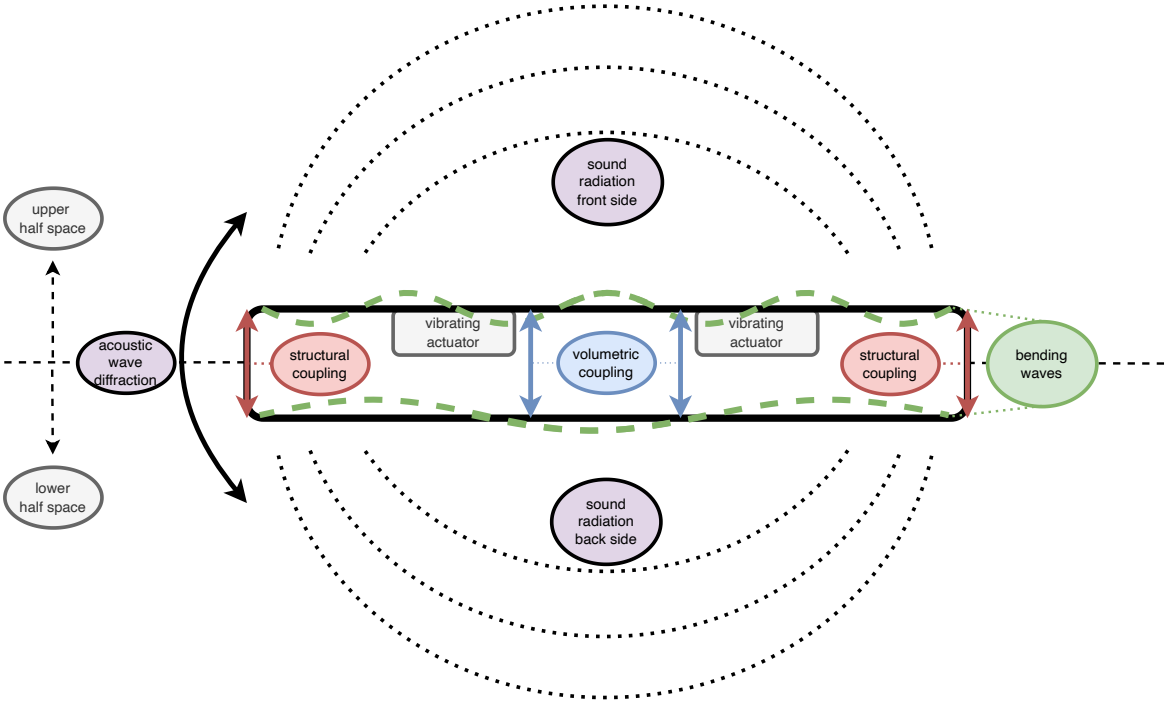
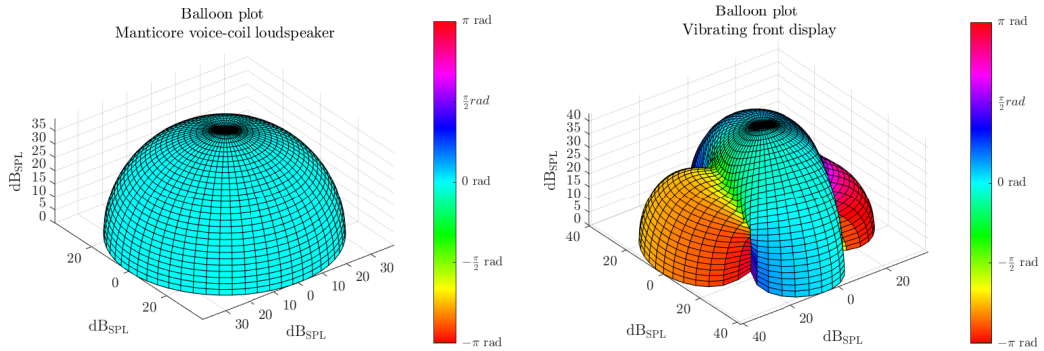


Figure 1.1 *Illustration of a vibrating-display loudspeaker, showing the cross section of a hand-held device. The vibrating actuators introduce bending waves into the front surface (i.e., the display). The bending waves transmit to the back cover via the internal volume and the frame. Both surfaces emanate acoustic waves into their respective half space. The full-space sound field can be seen as a superposition of the two half spaces and wave-diffraction effects. A possible sound radiation of the frame is neglected in the drawing.*



(a) (b)

Figure 1.2 *Hemispherical balloon plots, computed using the Rayleigh I integral. The shape of the balloon plot represents the radiated sound pressure in* $\mathrm{dB_{SPL}}$. *The coloration shows the relative phase to the front-centered observer position. The plots are evaluated at a hemispherical receiver arrangement with radius* $r = 0.91$ m, *at a frequency* $f = 3$ kHz. *(a) miniature voice-coil loudspeaker. (b) actuator-driven display.*

The Equivalent Source Method (ESM) is a simulation technique that mimics a vibrating surface by placing compact sources in the device's interior. A review of several frequency-domain and time-domain ESM approaches was done by Lee [6]. In [7, Sec. O.4], the Source Simulation Technique, a single-source ESM, is revisited in detail and developed into the ESM. The book includes sections about numerical errors, source positioning, and variants that use spherical wave functions for higher-order source types. The paper by Ochmann and Piscoya [8] introduces the complex sources, i.e., sources with complex-valued coordinates, to the ESM. The application of complex sources is further elaborated in [9, Chap. 9]. Numerical errors in the ESM are explained in [10], [11], and [12]. The latter suggests a treatment by a specific source constellation using a single-layer and a multi-layer potential. Zhang et al. [13] describe a method to determine the appropriate position and number of monopole sources, whereas Gounot and Musafir [14] introduce a global search tool for an optimal positioning using a genetic algorithm. Another paper by Gounot and Musafir [15] aims to find an optimal retreat distance for various planar source constellations in near-field acoustic holography. Optimal retraction distances and source positions for curved surfaces in near-field acoustic holography are investigated in [16]. No studies concerning the ESM for thin cuboid-like bodies could be found, as is the case for a vibrating-display loudspeaker.

The two Rayleigh integrals are simplifications of the Kirchhoff-Helmholtz Integral, assuming an infinitely extended planar body and applying either a Neumann or a Dirichlet boundary. Rayleigh I relates to the Neumann boundary and simulates the sound radiation from an infinite planar velocity boundary condition into a half space. Rayleigh II corresponds to prescribing an infinite sound pressure boundary condition, i.e., a Dirichlet boundary. There are existing open-source codes in Fortran [17], developed by Kirkup et al [18]. In an ongoing project on the platform Researchgate, Kirkup et al. is developing an open-source implementation in Matlab [19]. The repository provides links to literature and Matlab tutorials. Many studies are using Rayleigh integrals concerning the sound radiation of vibrating plates. Often, the Rayleigh integral serves as a reference for analyzing other simulation methods [5], [20], [21]. In [22], the implementation of the Rayleigh Integral Method is described, which includes a discretization procedure for the surface, allowing for arbitrarily shaped edges of the planar radiators. Discretization can either be done assuming constant vibration of surface elements, i.e., the midpoint method, or some weighted-residual approach as it is used in the Boundary Elements Method. Some studies aim to simulate the sound radiation from non-planar structures using Rayleigh integrals. In [23], a *"visible element Rayleigh integral"* is proposed, aiming to simulate the high-frequency sound radiation from three-dimensional bodies. The formulation exploits the projection of the body's outwards-normal vectors onto the observer coordinate to determine which surface part is visible to the observer. In [24], a formulation is proposed that allows for simulating arbitrarily curved surfaces using Rayleigh integrals, using wave-field extrapolation and matrix inversion. Rayleigh integrals seem promising for simulating a vibrating-display loudspeaker since they are easy to implement and require a low computational cost. However, their boundary conditions only allow for a half-space simulation and imply an infinitely-extended boundary. Including the contribution of the upper and lower surface appears reasonable and can be done by combining both Rayleigh integrals and exploiting the thin structure and parallel surfaces of the device.

The Boundary Elements Method (BEM) is the most sophisticated of the three methods. For acoustic problems, the BEM numerically evaluates the Kirchhoff-Helmholtz Integral. The BEM requires a three-dimensional surface mesh of the device. The meshing influences the quality of the simulation, as discussed in [25]. The BEM is accompanied by the non-uniqueness problem, which is a numerical issue occurring due to internal resonances and must be treated in most applications [26]. A common remedy for non-uniqueness is the 'combined Helmholtz Integral equation formulation' (CHIEF) [27], [28], which includes interior reference points, so-called CHIEF points, to suppress internal modes. The theory of BEM in acoustics is extensively discussed in [29], [30]. Due to its complexity, including

mesh creation, dealing with singular integrals, and inverting large matrices, implementing the BEM is more elaborate than the two previously described methods. However, it is also the physically most accurate when it is well tuned. Open-source implementations of the BEM exist, e.g., in Fortran [17] and Matlab [31].

This thesis revisits the three simulation approaches and observes their application to vibrating-display loudspeakers with measured surface vibration. The aim is to find a lightweight solution that is easy to integrate and can be employed flexibly in the cyclic development of vibrating-display loudspeakers. For this purpose, all simulations are implemented using the scientific programming language Matlab. The open-source toolbox OpenBem [31] is used for the core implementation of the BEM and the open-source finite-element platform Salome [32] is used for creating the surface mesh. The ESM and the Rayleigh are directly implemented into Matlab without the need for external software. For the ESM, a tailored source constellation is developed that leads to robust results taking into account the thin body. For the Rayleigh integrals, a corresponding combined formulation is found that allows for a full-space simulation and which is new to the author's knowledge. All models are validated by comparison to acoustic measurements in various constellations. Finally, the vibrating-display loudspeaker is further investigated using custom analysis tools, such as balloon plots, field plots, and phase analysis.

The thesis is structured as follows. Chapter 2 starts by describing the mathematical fundamentals of acoustic fields (Section 2.1), revisiting the Helmholtz equation and its fundamental solution, the free-space Green's function. After discussing the spatial derivatives of the Green's function and recalling the derivation of the Kirchhoff-Helmholtz Integral, the basic theory of the above-described simulation approaches is discussed: The ESM in Section 2.2, the conventional Rayleigh integrals in Section 2.3, the combined-Rayleigh formulation in Section 2.4, and finally, the BEM in Section 2.5. Chapter 3 first describes the devices under test and then discusses the measurement of surface velocities (Section 3.2) and acoustic responses (Section 3.3). An insight into how the simulations are implemented and their application is given in Chapter 4. Chapter 5 assesses and discusses all relevant simulation and measurement results. Finally, the conclusion and an outlook can be found in Chapter 6.

2 Theoretical Background

This chapter discusses the underlying theory necessary for simulating the sound-field radiation of a vibrating body. At first, the general theory about the description of acoustic fields is provided, followed by an explanation of the simulating techniques applied in this thesis: the Equivalent Source Method (ESM), the Rayleigh integrals, and the most sophisticated technique: the Boundary Elements Method (BEM).

2.1 Acoustic Fields

This section will derive the basic equations needed for the simulating approaches used in this thesis. First, the homogeneous and inhomogeneous Helmholtz equations are described, including the characteristic equations. We discuss the free-space Green's functions for acoustic monopoles and dipoles. In addition to their sound pressure, we derive the gradient as what would be received by dipole-directional observers of velocity sensors. Finally, we recall the Kirchhoff-Helmholtz integral, which allows for solving the Helmholtz equation by relating the boundary values of a surface to the acoustic field.

2.1.1 Euler's equation of motion

The Euler equation of motion is derived through the law of momentum conservation [33, Chap. 1,Sec. 3.1]. The equation states that the gradient of the sound pressure $\boldsymbol{\nabla} p$ must equal the negative particle acceleration $-\frac{\partial \boldsymbol{v}(\boldsymbol{x},t)}{\partial t}$, times the homogeneous fluid density ρ. In the frequency domain, the time-derivative simplifies to multiplication by $(j\omega)$. The Euler equation for the three-dimensional and one-dimensional case is, respectively [34, Part 3]

$$\boldsymbol{\nabla} p = -j\omega\rho\boldsymbol{v}, \tag{2.1}$$

$$\frac{\partial p}{\partial n} = -j\omega\rho v_n. \tag{2.2}$$

The $\boldsymbol{\nabla}(\cdot)$ operator in Eq. (2.1) represents the gradient of a function as vector operation,

$$\boldsymbol{\nabla}(\cdot) = \left[\frac{\partial(\cdot)}{\partial x}, \frac{\partial(\cdot)}{\partial y}, \frac{\partial(\cdot)}{\partial z}\right]^\top.$$

In Eq. (2.2), we symbolize the directional derivative into a single direction $\boldsymbol{n}^\top \boldsymbol{\nabla}(\cdot) = \frac{\partial(\cdot)}{\partial n}$. Throughout this thesis, we will use the equation to convert the normally-vibrating velocities on a radiator's surface into sound-pressure gradients and vice versa.

2.1.2 Plane waves: elementary solutions to the homogeneous Helmholtz equation

The Helmholtz equation consists of Eq. (2.1) and the compression equation $j\omega p = -\rho c^2 \boldsymbol{\nabla}^\top \boldsymbol{v}$ and is the space-frequency domain representation of a homogeneous sound field. With the definition of the

acoustic wave number $k^2 = \frac{\omega^2}{c^2}$, it states that for the sound pressure p in an obstacle- and source-free space,

$$\left(\Delta + k^2\right) p = 0. \tag{2.3}$$

It employs the symbol for the Laplacian that specifies the divergence of the gradient of a function,

$$\Delta(\cdot) = \boldsymbol{\nabla}^\top \boldsymbol{\nabla}(\cdot) = \frac{\partial^2(\cdot)}{\partial x^2} + \frac{\partial^2(\cdot)}{\partial y^2} + \frac{\partial^2(\cdot)}{\partial z^2}.$$

The characteristic equation for the Helmholtz equation, which any solution must fulfill if the field is homogeneous/source-free, can be found by guessing a product of exponentials as a solution:

$$p = \Psi \mathrm{e}^{jk_x x + jk_y y + jk_z z} = \Psi \mathrm{e}^{j\boldsymbol{k}^\top \boldsymbol{x}}. \tag{2.4}$$

Setting Eq. (2.4) into Eq. (2.3) yields the characteristic equation

$$k^2 = \left(k_x^2 + k_y^2 + k_z^2\right) = \boldsymbol{k}^\top \boldsymbol{k}. \tag{2.5}$$

The wave vector $\boldsymbol{k} = [k_x, k_y, k_z]^\top = k\boldsymbol{n}$ specifies the length and direction of a wave in three-dimensional space, where $\boldsymbol{n}$ is a unit vector that points to the direction from which the wave arrives and is always perpendicular to the plane wave fronts. $\boldsymbol{x} = [x, y, z]^\top$ is the three-dimensional position vector.

2.1.3 Green's function: an elementary solution to the inhomogeneous Helmholtz equation

The Helmholtz Equation Eq. (2.3) can be re-formulated to describe a homogeneous field that radiates from a point-like sound source. Only at this source point does the equation become inhomogeneous. We add an arbitrary source distribution $\delta(\boldsymbol{x})$ to the right-hand side of Equation 2.3 for a point source at the origin:

$$\left(\Delta + k^2\right) G(||\boldsymbol{x}||) = -\delta(\boldsymbol{x}). \tag{2.6}$$

The symbol $\delta(\boldsymbol{x})$ denotes the Dirac delta, a function/distribution that is non-zero only at an infinitely narrow volume around the coordinate $\boldsymbol{x}$,

$$\delta(\boldsymbol{x}) = \lim_{\epsilon \to 0} \begin{cases} a, & \text{for } ||\boldsymbol{x}|| \leq \epsilon, \\ 0, & \text{elsewhere,} \end{cases} \tag{2.7}$$

and whose infinite, non-zero value yields a normalized volume integral $\int \delta(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = 1$, i.e., $a = \lim_{\epsilon \to 0} \left[\frac{3}{4\pi\epsilon^3}\right]$. The elementary solution to the equation is denoted as $G(||\boldsymbol{x}||)$, and it is referred to as harmonic free-space Green's function. It constitutes *"...the expression for the pressure per unit source strength of a harmonic point monopole..."* [35, P. 105]. Throughout this thesis, we will frequently rely on this equation or its spatial derivatives. As of now, we will refer to it as Green's function. The space-frequency representation of the Green's function can be found as [35, Sec. 6.4.2]

$$G(r) = \frac{\mathrm{e}^{-jkr}}{4\pi r}. \tag{2.8}$$

r signifies the distance from the origin, or, when we shift the Green's function to a position $\boldsymbol{x}_0$, r signifies the distance between the shifted position and the point of observation $\boldsymbol{x}$. It is then

$$r = ||\boldsymbol{x} - \boldsymbol{x}_0|| = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}. \tag{2.9}$$

The Green's function has several properties. First, it fulfills the principle of acoustic reciprocity, *"...because the pressure at a receiver point is unchanged by interchange of source and receiver point locations."* [35, P. 105], i.e., it fulfills

$$G(||\boldsymbol{x} - \boldsymbol{x}_0||) = G(||\boldsymbol{x}_0 - \boldsymbol{x}||). \tag{2.10}$$

Secondly, we should note that the Green's function contains a singularity when evaluated at the source coordinate. Third, the Green's function fulfills Sommerfeld's radiation condition, which *"...broadly states that the wave fronts (surfaces of uniform phase) generated by any finite source region will become spherical at infinite distance and that no waves can approach the source region from infinity"*[35, P. 121]. Mathematically, this is described as [36, Sec. 4.5.4]

$$\lim_{r\to\infty}\left[r\left(\frac{\partial G(r)}{\partial r} + jkG(r)\right)\right] = 0, \tag{2.11}$$

and implies in three-dimensional space that the far-field amplitude must decay with $\frac{1}{r}$ and the phase must propagate with e^{-jkr} when using the signal-processing temporal Fourier transform conventions.

A balloon plot of the Green's function, visualizing the acoustic monopole in magnitude and phase, can be observed in Fig. 2.2 (a).

2.1.4 Gradient and directional derivative of the Green's function

The spatial rate of change of the sound pressure is proportionally related to the particle velocity, c.f. Eq. (2.1). Therefore, we can describe the particle velocity that is introduced by an acoustic monopole via the gradient of the Green's function,

$$\boldsymbol{\nabla} G(r) = \left[\frac{\partial G(r)}{\partial x_i}\right]. \tag{2.12}$$

We can calculate the gradient of the Green's function by the chain rule

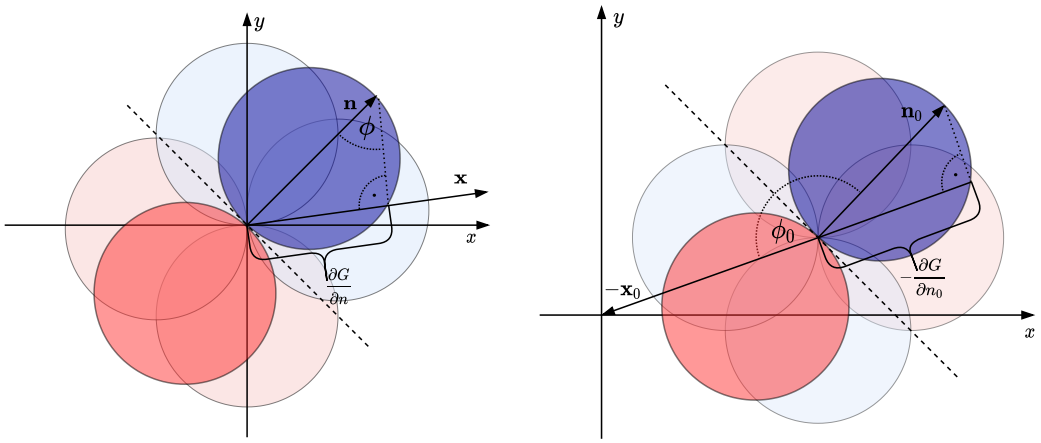
$$\boldsymbol{\nabla} G(r) = \boldsymbol{\nabla} r \frac{\partial G(r)}{\partial r}. \tag{2.13}$$

Assuming the source coordinate at the origin, $\boldsymbol{x}_0 = 0$, we find that the gradient of the radial distance to the observer is

$$\boldsymbol{\nabla} r = \left[\frac{\partial\sqrt{\sum_{i=1}^{3} x_i^2}}{\partial x_i}\right] = \frac{\boldsymbol{x}}{r}. \tag{2.14}$$

To derive the $\frac{1}{r}$ and e^{-jkr} components of the Green's function, we make use of the product rule

$$\frac{\partial G}{\partial r} = \frac{\partial \mathrm{e}^{-jkr}}{\partial r}\frac{1}{4\pi r} + \frac{\mathrm{e}^{-jkr}}{4\pi}\frac{\partial \frac{1}{r}}{\partial r} = -\left(jk + \frac{1}{r}\right)G(r). \tag{2.15}$$



(a) (b)

Figure 2.1 *Illustration of the normal derivative of the Green's function; for simplicity, the z-axis is omitted. Blue colors indicate in-phase monopoles; red colors indicate contra-phase monopoles. Brighter circles in the background show the gradient components, and darker circles show the resulting dipole characteristics of the normal derivative. Dashed lines represent the notional separating plane, pointed lines the angles and projection distances. (a): Derivative of the observing coordinate* $\frac{\partial G(r)}{\partial n}$*. Source term located in the origin* $\boldsymbol{x}_0 = \mathbf{0}$*. (b): Derivative of the shifted source coordinate* $\frac{\partial G(r)}{\partial n_0}$*. Observation from the origin* $\boldsymbol{x} = \mathbf{0}$*.*

The radial derivative of the Green's function fulfills the Sommerfeld condition, which can easily be verified anticipating the second radial derivative from Eq. (2.25),

$$\lim_{r\to\infty}\left[r\left(\frac{\partial^2 G(r)}{\partial r^2} + jk\frac{\partial G(r)}{\partial r}\right)\right] = 0. \tag{2.16}$$

The gradient function in Eq. (2.13) yields a $[3 \times 1]$ vector, consisting of three orthogonal dipoles. In many cases, as we will encounter in the following chapters, it is necessary to describe the spatial rate of change of $G(r)$ along a single direction. As in Eq. (2.2), we can find the directional derivative by projecting a unit-length direction vector onto $\boldsymbol{x}$. The result yields the spatial description of an acoustic dipole, which can be considered as the superposition of two odd-symmetrically radiating monopoles [36, Sec. 4.4]. We can consider the direction vector as the normal vector of a notional plane that tangentially separates the two monopoles. Hence, as of now, we will refer to the directional derivative as normal derivative $\frac{\partial}{\partial n}$ and the directivity vector as a normal vector $\boldsymbol{n}$. An illustration of the projection and the resulting dipole can be observed in Fig. 2.1 (a). A balloon plot that visualizes the magnitude and phase of the dipole can be observed in Fig. 2.2 (b). Mathematically, the projection is computed via the normalized scalar product of the two vectors, i.e.,

$$\begin{aligned}\frac{\partial G(r)}{\partial n} &= \frac{\boldsymbol{n}^\top \boldsymbol{x}}{r}\frac{\partial G}{\partial r}, \\ &= -\cos(\phi)\left(jk + \frac{1}{r}\right)G(r).\end{aligned} \tag{2.17}$$

2.1.5 Directional derivatives of Green's function with regard to both source and receiver locations

It is now easily possible to shift the source coordinate to any position outside the origin via substituting $\boldsymbol{x}$ to $\boldsymbol{x} - \boldsymbol{x}_0$. The gradient with respect to the source coordinate is then just the negative gradient

$$\begin{aligned}\boldsymbol{\nabla}_0 G(r) &= \left[\frac{\partial G(r)}{\partial x_{0,i}}\right] = \left[\frac{\partial (x - x_0)}{\partial x_{0,i}} \frac{\partial G(r)}{\partial x_i}\right] \\ &= \left[-\frac{\partial G(r)}{\partial x_i}\right] = -\boldsymbol{\nabla} G(r)\end{aligned} \tag{2.18}$$

Further, we can compute the normal derivative with respect to the shifted source coordinate $-\boldsymbol{x}_0$, which will provide the Green's function for a radiating dipole. We denote the normal vector that specifies the orientation of the dipole as $\boldsymbol{n}_0$. Via Eq. (2.17), we obtain

$$\begin{aligned}G_{dip}(r) = \frac{\partial G(r)}{\partial n_0} &= -\frac{\boldsymbol{n}_0^\top (\boldsymbol{x} - \boldsymbol{x}_0)}{r} \frac{\partial G}{\partial r}, \\ &= \cos(\phi_0) \left(jk + \frac{1}{r}\right) G(r).\end{aligned} \tag{2.19}$$

Fig. 2.1(b) shows an illustration of the normal derivative with respect to the source coordinate.

The radiated velocity of an acoustic dipole Eq. (2.19) can be described via its Hessian matrix, i.e.,

$$\boldsymbol{\nabla}\boldsymbol{\nabla}_0^\top G(r) = -\boldsymbol{\nabla}\boldsymbol{\nabla}^\top G(r) = \left[\frac{\partial^2 G(r)}{\partial x_i \partial x_{0,j}}\right]. \tag{2.20}$$

The directional derivative of the dipole can again be obtained via projection, now onto both $\boldsymbol{n}_0$ and $\boldsymbol{n}$,

$$\frac{\partial^2 G(r)}{\partial n \partial n_0} = -\boldsymbol{n}^\top \left[\boldsymbol{\nabla}\boldsymbol{\nabla}^\top G(r)\right] \boldsymbol{n_0}. \tag{2.21}$$

For the explicit computation of the directional derivative, it is helpful to expand the vector projection as a sum and calculate the derivative for a single entry:

$$\begin{aligned}\frac{\partial^2 G(r)}{\partial n \partial n_0} &= \sum_{i=1}^{3} n_i \frac{\partial}{\partial x_i} \left(\sum_{j=1}^{3} n_{0,j} \frac{\partial G(r)}{\partial x_{0,j}}\right) \\ &= \sum_{i=1}^{3} n_i \frac{\partial}{\partial x_i} \left(\sum_{j=1}^{3} n_{0,j} \frac{-(x_j - x_{0,j})}{r} \frac{\partial G(r)}{\partial r}\right) \\ &= \sum_{i=1}^{3}\sum_{j=1}^{3} n_i n_{0,j} \left(-\frac{1}{r}\frac{\partial G(r)}{\partial r}\frac{\partial (x_j - x_{0,j})}{\partial x_i} - (x_j - x_{0,j}) \frac{\partial \left(\frac{1}{r}\frac{\partial G(r)}{\partial r}\right)}{\partial x_i}\right) \\ &= \sum_{i=1}^{3}\sum_{j=1}^{3} n_i n_{0,j} \Bigg(\frac{-\delta_{i,j}}{r}\frac{\partial G(r)}{\partial r} + \frac{(x_j - x_{0,j})(x_i - x_{0,i})}{r^3}\frac{\partial G(r)}{\partial r} - \\ &\qquad \frac{(x_j - x_{0,j})(x_i - x_{0,i})}{r^2}\frac{\partial^2 G(r)}{\partial r^2}\Bigg).\end{aligned} \tag{2.22}$$

We then insert the solution back into vector-matrix notation, where the Kronecker delta $\delta_{i,j}$ becomes a (3×3) identity matrix $\boldsymbol{I}$. We define the source-to-receiver vector $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}_0$ and find

$$\frac{\partial^2 G(r)}{\partial n \partial n_0} = -\frac{\boldsymbol{n}^\top \boldsymbol{I} \boldsymbol{n}_0}{r} \frac{\partial G(r)}{\partial r} + \frac{\boldsymbol{n}_0^\top \boldsymbol{r}}{r} \frac{\boldsymbol{n}^\top \boldsymbol{r}}{r} \frac{1}{r} \frac{\partial G(r)}{\partial r} - \frac{\boldsymbol{n}_0^\top \boldsymbol{r}}{r} \frac{\boldsymbol{n}^\top \boldsymbol{r}}{r} \frac{\partial^2 G(r)}{\partial r^2}, \tag{2.23}$$

$$= -\cos(\eta) \frac{1}{r} \frac{\partial G(r)}{\partial r} + \cos(\phi_0)\cos(\phi) \left(\frac{1}{r} \frac{\partial G(r)}{\partial r} - \frac{\partial^2 G(r)}{\partial r^2} \right). \tag{2.24}$$

Where the vector projections result in direction cosines of the enclosed angles, as above. The second radial derive of the Green's function is found via application of the product rule, resulting in

$$\frac{\partial^2 G(r)}{\partial r^2} = \frac{\partial \left(-\frac{1}{r}\right)}{\partial r} G(r) - \left(jk + \frac{1}{r} \right) \frac{\partial G(r)}{\partial r},$$

$$= \left(-k^2 + \frac{2jk}{r} + \frac{2}{r^2} \right) G(r). \tag{2.25}$$

Finally, the second directional derivative consists of a linear combination of the entries in the Hessian, i.e., six functions corresponding to the second-order partial derivatives $\frac{\partial^2}{\partial x^2}$, $\frac{\partial^2}{\partial y^2}$, $\frac{\partial^2}{\partial z^2}$, $\frac{\partial^2}{\partial x \partial y}$, $\frac{\partial^2}{\partial x \partial z}$, and $\frac{\partial^2}{\partial y \partial z}$ [36, Sec. 4.4.3]. The result shows a quadrupole behavior whose shape can be irregular and is determined by the directions of derivatives. Representative balloon plots of quadrupoles can be observed in Fig. 2.2 (c) and (d).

2.1.6 Kirchhoff-Helmholtz integral

The Kirchhoff-Helmholtz integral *"...establishes a relationship between the pressure $p(P)$ outside a vibrating body, and the pressure $p(Q)$ and the normal velocity $v(Q)$ on the body"*[30, P. 22]. It can be seen as a *"...corollary of the governing partial-differential equation and of the Sommerfeld radiation condition"*[36, P. 209-210], rather than an acoustic boundary value problem, since the surface pressure will be determined by the velocity distribution. Further, the pressure derivatives are related to the velocity via the Euler equation Eq. (2.2). To emphasize this dependence, the derivation in this section will rely on using pressure derivatives instead of normal velocities.

To derive the Kirchhoff-Helmholtz integral, we combine Eq. (2.3) and Eq. (2.6). Thereby, the former equation is multiplied by $G(r)$ and the latter by $p(\boldsymbol{x})$. The terms containing the wave number cancel out, and we find

$$p(\boldsymbol{x})\delta(\boldsymbol{x} - \boldsymbol{x}_0) = G(r)\Delta p(\boldsymbol{x}) - p(\boldsymbol{x})\Delta G(r). \tag{2.26}$$

Exploiting the reciprocity of the Green's function, Eq. (2.26) can be transformed to an integral equation over a source volume, whereas the left-hand term results in the acoustic pressure at the point of observation, outside the source volume [34, Sec. 13.3], [37, Chap. 6]. We obtain

$$\int_{V_0} p(\boldsymbol{x}_0)\delta(\boldsymbol{x} - \boldsymbol{x}_0) \mathrm{d}V_0 = \int_{V_0} \Big(G(r)\Delta_0 p(\boldsymbol{x}_0) - p(\boldsymbol{x}_0)\Delta_0 G(r) \Big) \mathrm{d}V_0 = p(\boldsymbol{x}). \tag{2.27}$$

The Kirchhoff-Helmholtz integral can then be found using Gauss' Theorem [38, Sec. 3.8], which transforms the volume integral to a closed surface integral. The integral then contains the field values radiated by the (notional) sources in the source region, evaluated on the surface. The equation states

$$c(\boldsymbol{x})p(\boldsymbol{x}) = \oint_{\Gamma_0 \in \partial V_0} \left(G(r) \frac{\partial p(\boldsymbol{x}_0)}{\partial n_0} - p(\boldsymbol{x}_0) \frac{\partial G(r)}{\partial n_0} \right) \mathrm{d}\Gamma_0. \tag{2.28}$$

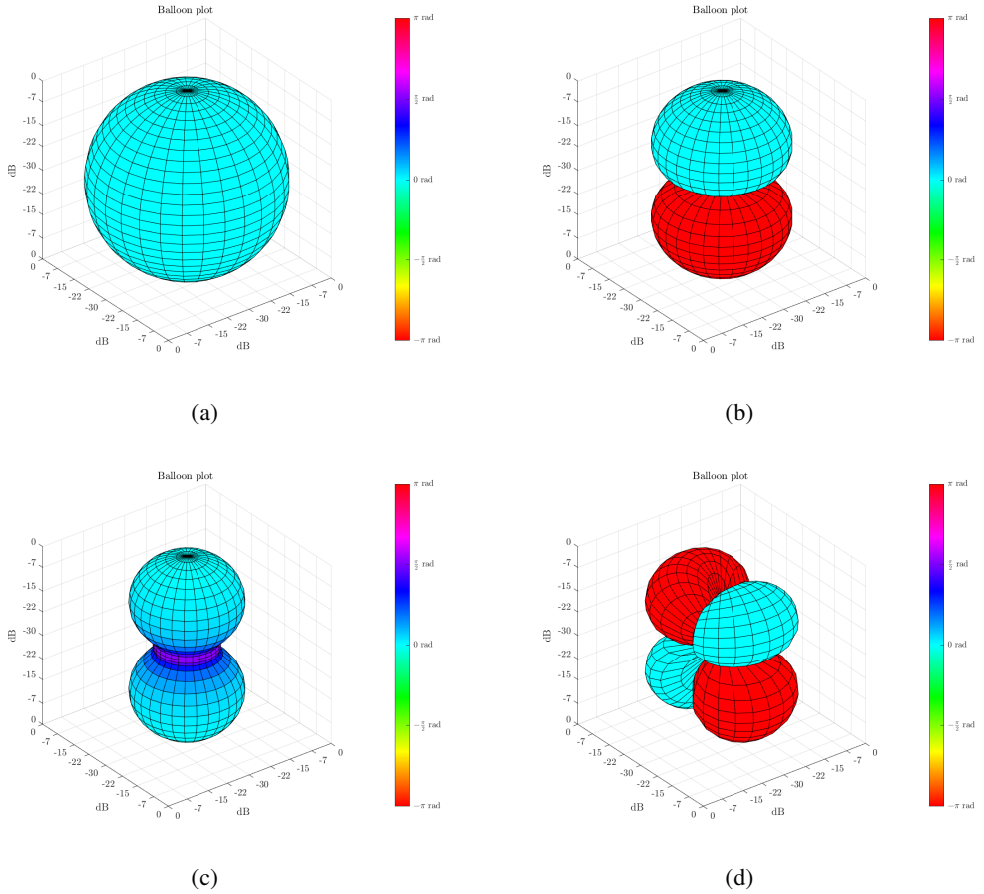


Figure 2.2 *Balloon plots of Green's function and derivatives. Colors represent the relative phase to the receiving position at $\boldsymbol{x}_0 = [0, 0, 1]^\top$. Radii of the balloons show the normalized levels of received sound pressures, with a zero offset of 30 dB. The Helmholtz number is $kr = 10$ for all subplots. (a): Green's function, representing an acoustic monopole. (b): normal derivative in direction $n = [0, 0, 1]^\top$, showing a dipole. (c): second normal derivative in same direction $n = n_0 = [0, 0, 1]^\top$, showing a longitudinal quadrupole. (d): second normal derivative in orthogonal direction $n_0 = [0, 0, 1]^\top$, $n = [0, 1, 0]^\top$, showing a lateral quadrupole.*

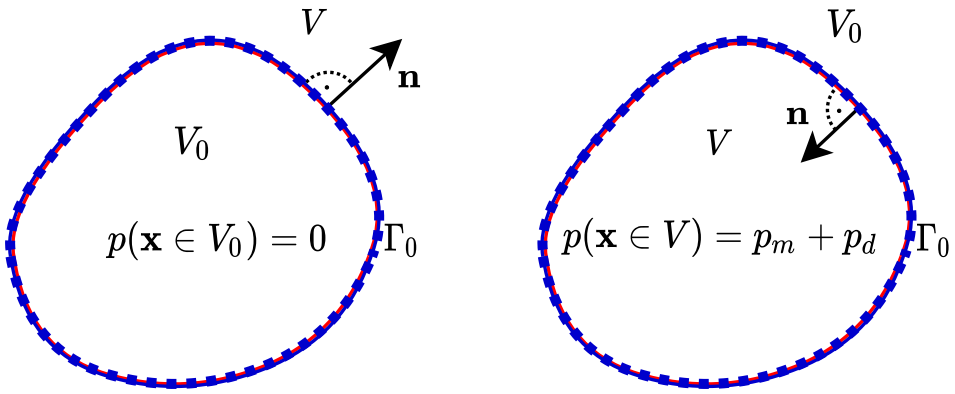


Figure 2.3 *Illustration of the Kirchhoff-Helmholtz integral, as two-dimensional projection; blue and red lines illustrate dipoles sitting on the boundary, dashed lines illustrate monopoles. Within V_0, monopole and dipole layer cancel out. In V, they sum up. left: formulation for an exterior radiation problem; right: interior radiation problem.*

The right-hand side represents the superposition of a single-layer and a double-layer potential, i.e., surface distributions of weighted monopoles and dipoles [29, Sec. 2.1]. Dipoles are weighted with the sound pressures on the surface, whereas monopoles are weighted with their normal derivatives. The boundary Γ_0 separates a source-free zone V and the source region V_0, enclosing all source positions $\boldsymbol{x_0}$, as in Eq. (2.27). V can either be the interior, finite volume, or the exterior, infinite volume, relating to formulations of an interior or exterior radiation problem. Thereby, the normal vectors point to V. $c(\boldsymbol{x})$ takes account for a discontinuity in the equation. Inside V_0, the monopole and dipole terms cancel out, whereas, in V, the terms add up to synthesize a homogeneous sound field. An illustration of the formulation for interior and exterior radiation problems can be found in Fig. 2.3.

On the infinitely thin surface, the resulting sound pressure diminishes because only a part of the receiver lies in V [29, Sec. 2.1]:

$$c(\boldsymbol{x}) = \begin{cases} 1 & \boldsymbol{x} \in V \\ 0 < c(\boldsymbol{x}) < 1 & \boldsymbol{x} \in \Gamma_0 \\ 0 & \boldsymbol{x} \in V_0 \end{cases} \tag{2.29}$$

If the receiver sits on an ideally smooth surface, $c(\boldsymbol{x}) = \frac{1}{2}$ [26]. Along a right-angled edge, it will be $\frac{3}{4}$ or $\frac{1}{4}$ and in a right-angled corner, $\frac{7}{8}$ or $\frac{1}{8}$, for an exterior and interior problem, respectively. An illustration for the definition of $c(\boldsymbol{x})$ can be found in Fig. 2.4. Note that in some literature [30, Sec. 2.3], [39, Sec. 2], $c(\boldsymbol{x})$ is formulated as the solid angle, which is due to a different normalization of the Dirac delta used there, to $\int \delta(\boldsymbol{x}) d\boldsymbol{x} = 4\pi$, instead of $\int \delta(\boldsymbol{x}) d\boldsymbol{x} = 1$ as it is used here.

The Kirchhoff-Helmholtz integral constitutes a closed-surface formulation for Huygen's principle [30, Sec. 2.4], which *"...states that each point on the wave front of a propagating wave can be replaced with a point source,...thus creating an array of wavelets whereby each wavelet is unaffected by the presence of all the other wavelets."* [34, P. 605]. It is the basic equation for the BEM, which will be covered in Section 2.5. The applications can be manifold, including sound radiation problems, scattering and edge diffraction analysis, and modal analysis of vibrating bodies.

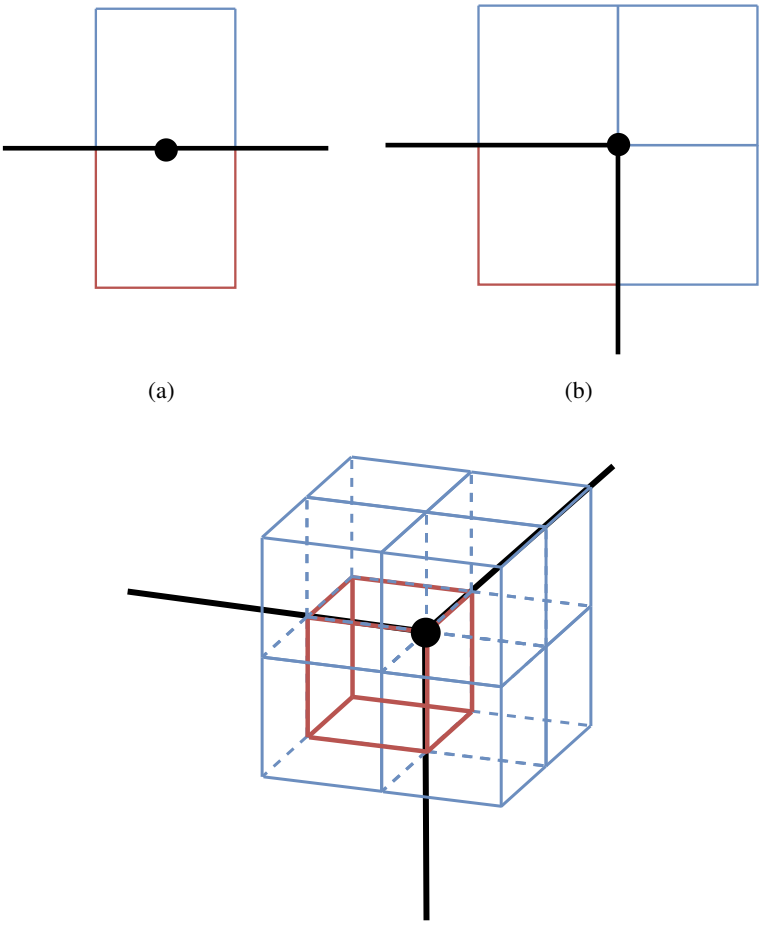


Figure 2.4 *Illustration for definition of $c(\boldsymbol{x})$, c.f. Eq. (2.29); black lines represent the surface, and black dots the position where the Kirchhoff-Helmholtz integral is evaluated; (a): smooth surface, half of the receiver lies in each subspace. (b): evaluation at an edge, one quarter of the receiver lies in the inner subspace and the rest in the outer subspace. (c): evaluation on a corner, one eighth of the receiver lies in the inner subspace and the rest in the outer.*

2.2 Equivalent Source Method (ESM)

The ESM is a simple-to-implement simulation technique for acoustic problems, i.a. sound-radiation problems. The basic idea is to approximate the sound radiation of a vibrating object by emulating its surface-normal velocities via so-called equivalent sources located in the interior. If the equivalent sources mimic the surface velocities well, their superimposed sound field aligns with the sound field radiated by the vibrating body.

Several techniques relate to the ESM but use different name conventions. While the terms Source Simulation Technique, Multipole Method, or Spherical Wave Synthesis relate to a single equivalent source lying at the origin [7, Sec. O.4], the terms Equivalent Source Method (ESM), Wave Superposition Method, and Method of Fundamental Solutions are used when systems of sources are employed [6].

A difficulty in using the ESM lies in determining an appropriate number, kind, and positioning of equivalent sources for the underlying problem [10], [11], [12], [13], [16]. After these preconditions are defined, the simulation carries out two steps: calibrating the source strengths to fulfill the boundary conditions and synthesizing the sound field by superimposing the radiated sound fields of each source.

2.2.1 Synthesizing the sound field

To begin with, let us resume the inhomogeneous Helmholtz equation for a single source, Eq. (2.6). Due to the linearity of the Helmholtz equation, it is possible to synthesize sound fields by superimposing weighted elementary solutions. Using the Green's function Eq. (2.8), we can thus distribute acoustic monopoles in space and calculate their total radiated sound pressure to a single location simply via vector multiplication:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} G(r_i) q_i = \boldsymbol{g}^\top \boldsymbol{q}. \tag{2.30}$$

The vector $\boldsymbol{g} = [G(r_1), \ldots, G(r_K)]^\top$ holds Green's functions while $\boldsymbol{q}$ contains the complex-valued source strengths.

2.2.2 Calibrating the source strengths

Let us assume a convex-shaped vibrating body whose normal surface velocities are known on L discrete surface nodes, i.e.,

$$v_n(\boldsymbol{x}) = \textstyle\sum_{l=1}^{L} v_{n,l}\delta(\boldsymbol{x}_l), \quad \boldsymbol{v}_n = [v_{n,1}, \ldots, v_{n,L}]^\top. \tag{2.31}$$

For a single position, the velocity radiated by the equivalent sources can be obtained via the projection of the pressure gradient, c.f. Section 2.1.5, and exploiting the Euler equation Eq. (2.2). This yields

$$v_n = \frac{1}{-j\omega\rho} \boldsymbol{h}^\top \boldsymbol{q}, \tag{2.32}$$

with $\boldsymbol{h} = \left[\frac{\partial G(r_1,\ldots,r_K)}{\partial n}\right]^\top$ holding the normal derivatives of the Green's function , c.f. Eq. (2.17).

To determine the source strengths, Eq. (2.32) is expanded to an equation system that evaluates all L surface nodes:

$$\boldsymbol{v}_n = \frac{1}{-j\omega\rho} \boldsymbol{H} \boldsymbol{q}, \tag{2.33}$$

with

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial G(r_{11})}{\partial n_1} & \dots & \frac{\partial G(r_{1K})}{\partial n_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial G(r_{L1})}{\partial n_L} & \dots & \frac{\partial G(r_{LK})}{\partial n_L} \end{bmatrix}. \tag{2.34}$$

If $\boldsymbol{H}$ is a regular matrix and $K = L$, i.e., the same amount of sources and surface nodes are used, the source strengths can be found directly via matrix inversion:

$$\boldsymbol{q} = -j\omega\rho\boldsymbol{H}^{-1}\boldsymbol{v}_n \tag{2.35}$$

If $K \neq L$, or $\boldsymbol{H}$ singular, the matrix inversion is not feasible and other solutions have to be found. The Moore-Penrose pseudo inverse constitutes a generalization of the matrix inverse [40, Sec. 5.6]

$$\boldsymbol{q} = -j\omega\rho\boldsymbol{H}^{\dagger}\boldsymbol{v}_n, \tag{2.36}$$

and solves the over-determined or under-determined equation systems so that

$$\underset{\boldsymbol{q}}{\operatorname{argmin}} ||\boldsymbol{H}\boldsymbol{q} - \boldsymbol{v}_n||. \tag{2.37}$$

If $L > K$ and $\boldsymbol{H}$ is of full rank, the equation system is over-determined. In this case, the unique optimal solution is the left inverse of $\boldsymbol{H}$

$$\boldsymbol{H}^{\dagger} = \left(\boldsymbol{H}^{\mathsf{H}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^{\mathsf{H}}, \tag{2.38}$$

$$\boldsymbol{H}^{\dagger}\boldsymbol{H} = \boldsymbol{I}. \tag{2.39}$$

If $K > L$ or $\boldsymbol{H}$ is singular, the system is under-determined and $\left(\boldsymbol{H}^{\mathsf{H}}\boldsymbol{H}\right)^{-1}$ cannot be computed. The pseudo inverse must fulfill an additional constraint here, where a common approach is to minimize the vector length [40, Sec. 5.7]

$$\underset{\boldsymbol{q}}{\operatorname{argmin}} ||\boldsymbol{q}||. \tag{2.40}$$

A solution to this approach is the right inverse [41, Sec. A.4]

$$\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{\mathsf{H}}\left(\boldsymbol{H}\boldsymbol{H}^{\mathsf{H}}\right)^{-1}, \tag{2.41}$$

$$\boldsymbol{H}\boldsymbol{H}^{\dagger} = \boldsymbol{I}. \tag{2.42}$$

Using Matlab, computation of the pseudo inverse can be done via the command *pinv()*. Solving the whole equation system while minimizing the norm of the least-squares solution can be done via *lsqminnorm()*. For computation of a non-singular system that is either fully determined or over-determined, the backslash operator \ is recommended [42, Sec. 2].

2.2.3 The condition number of the ESM

Several measures exist to assess the quality of the equation system [10], [12]. The matrix condition number is a measure to quantify the system's overall quality. The condition number measures how

near to singular the matrix is, i.e., how much the perturbations in the equation, e.g., due to limited numerical precision, are magnified for the sound-field reproduction [43, Sec. 2.9]. It can be computed via

$$\kappa(\boldsymbol{H}) = ||\boldsymbol{H}||\,||\boldsymbol{H}^{-1}||, \tag{2.43}$$

where $||\boldsymbol{H}||$ denotes a matrix norm, which is not strictly defined in this context. The so-called spectral condition number applies the 2-Norm, i.e., inserts the largest singular value for $||\boldsymbol{H}||$ and its reciprocal smallest singular value for $||\boldsymbol{H}^{-1}||$ [7, Sec. O.4.5.2], [43, Sec. 2.9]. Computing the 2-Norm, requires a singular value decomposition, which can be of high computational cost for large matrices. Other matrix norms, such as the Frobenius norm, do not require a singular value decomposition, but when applied to Eq. (2.43), they require explicit computation of the matrix inversion. In Matlab, the command `cond()`, takes account for computation of the condition number, while it is possible to specify the intended norm in the arguments of the function.

The condition number has a value of one for a perfectly conditioned and symmetric matrix, and it diverges for a singular matrix. It can be seen as a *"...relative error magnification factor. Changes in the right-hand side can cause changes κ times as large in the solution."*[43, P. 17]. An ill-conditioned matrix can be regularized, e.g., via truncated singular value decomposition [13]. The regularization assures that small singular values are ignored in the matrix inversion.

Surface reproduction error

Another quantity to assess the reproduction quality is the normalized surface reproduction error, which can be determined for a single position via

$$\epsilon_v = \frac{|v_{surf} - v_{rep}|}{|v_{surf}|}. \tag{2.44}$$

If the surface velocity is perfectly reproduced, $\epsilon_v = 1$. A large reproduction error, however, may still yield good simulation results in the far field [11].

While neither of these error measures can predict the overall quality of the simulation, they assess the quality of the equation system. Hence, they serve as indicators for an adequate or inadequate source positioning cf. Section 2.2.4.

2.2.4 Placing the equivalent sources: exploiting parallel surfaces and thin layout

Not every constellation of equivalent sources leads to an appropriate sound-field reproduction. The number, position, and source type influence the quality of the output [8]. While many applications of the ESM use a large number of normal-distributed sources and despite the existence of approaches for automatic optimization of the source positioning [14], [13], careful consideration of the underlying problem can lead to a well-suited result.

Several issues must be considered when choosing the number of sources, their positions, and their type. If the equivalent sources form a closed interior surface, internal resonances can cause a non-unique solution at resonance frequencies. A remedy can be the combination of single-layer and double-layer potentials, as described in [12]. Another option would be an alignment that does not form a closed surface at all or one that comprises only resonances at frequencies beyond the range of interest. Further, it has been shown that equivalent sources lying too close to a surface node lead to a lack of generalization for the surface reconstruction, i.e., the reproduction error is small when evaluated directly on the nodes but grows large in between the nodes [10]. Bai et al. [15] observes

an optimal retraction distance for equivalent sources in acoustic near-field holography and finds an optimal retraction distance of 0.4 to 0.5 times the source spacing for a planar source arrangement.

As for choosing the number of sources, the unavoidable cross-talk between sources and surface nodes leads to a trade-off. Especially the sound from sources of simple directivity shapes, such as monopoles and dipoles, will be received at more than one node and could thus lead to coupling in the equation system. A higher density of sources or surface nodes will thus lead to a higher condition number of the matrix. By contrast, a low number of sources might not be capable of reconstructing structural radiation patterns of higher complexity [13].

Regarding the source type, one should keep in mind that higher-order sources produce stronger near fields than lower-order ones. Thus, if surface nodes lie within the near fields of the sources, higher-order sources dominate in the matrix inversion. When regularization is employed, only the higher-order sources remain.

Central planar monopole and dipole layer in thin box

The above reasons necessitate trade-offs for the choice regarding the sources' number, order, and position, particularly when simulating bodies of small measures. In this thesis, the challenge was to find a well-suited source constellation to reproduce the structural radiation of a thin cuboid. An appropriate solution could be found by exploiting the parallel surfaces of the cuboid.

First, the number of source positions is set to equal the number of surface nodes for obtaining a full set of equations. The matrix condition for such a setup diverges quickly if the source constellation does not fit the velocity pattern and no regularization is applied. According to [12], a well-conditioned matrix can be obtained by assuring that every source position has a single node to which its Euclidean distance is smaller than to any other node, i.e.,

$$||\boldsymbol{r}_{e,i} - \boldsymbol{r}_{s,j}||\Big|_{j\neq i} > ||\boldsymbol{r}_{e,i} - \boldsymbol{r}_{s,j}||\Big|_{j=i}, \tag{2.45}$$

where $r_{e,i}$ is the position vector from the origin to the the i^{th} surface node and $r_{s,j}$ to the j^{th} equivalent source.

It is not always straightforward to find a source constellation that satisfies Eq. (2.45). Especially corners of a body can lead to difficulties, as illustrated in Fig. 2.5. For determining suitable source positions, consider the thin cuboid with measures $l_x > l_y \gg l_z$. Due to the small l_z, let us assume the sound radiated sideways can be considered negligible. For further simplification, assume the nodes of the upper and lower surface be equal in (x, y) coordinates. To satisfy Eq. (2.45), place one source to each respective (x, y) coordinate. The remaining degrees of freedom are the source type and z coordinate of the sources. Again, we remember that l_z is small and hence, try to maximize the distance between sources and surface nodes. However, retracting the sources towards the center is only possible up to some extent, as the matrix condition worsens with the upper and lower sources approaching each other. Avoiding the cross-talk between upper and lower sources while maximizing the distance to the surface nodes, we place a single layer of source positions at the center. Each source position holds a monopole and a dipole oriented normally to the (x, y) plane, as illustrated in Fig. 2.6.

Exploiting the even-symmetric radiation of monopoles and the odd-symmetric radiation of dipoles regarding the z coordinate, we decompose the upper and lower surface velocities into even-symmetric and odd-symmetric parts via

$$v_{n_+}(\boldsymbol{x})\Big|_u = v_e(\boldsymbol{x})\Big|_u + v_o(\boldsymbol{x})\Big|_u,$$

$$v_{n_-}(\boldsymbol{x})\Big|_l = v_e(\boldsymbol{x})\Big|_u - v_o(\boldsymbol{x})\Big|_u,$$

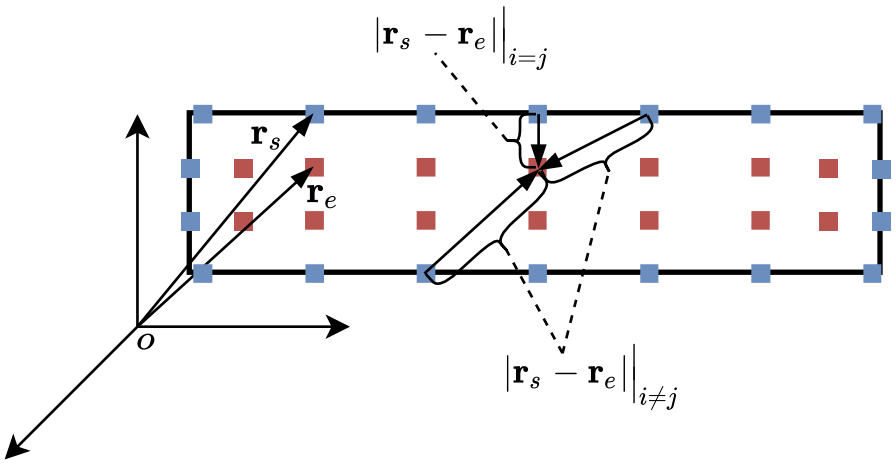


Figure 2.5 *Cross-section of a cuboid with surface nodes (blue) and equivalent sources (red), illustrating the problem of simulating a thin cuboid; the constellation does not satisfy Eq.* (2.45)*, as it has too many sources. The surface nodes at the corners would have to be left out, or additional sources in the corners would have to be placed. If $|\boldsymbol{r}_s - \boldsymbol{r}_e|$ is reduced, the generalization error grows. By contrast, the up-down cross-talk increases if the sources are retracted towards the center. A higher number of sources also increases the cross-talk.*

$$v_e(\boldsymbol{x})\Big|_u = \frac{v_{n_+}(\boldsymbol{x})\big|_u + v_{n_-}(\boldsymbol{x})\big|_l}{2},$$
$$v_o(\boldsymbol{x})\Big|_u = \frac{v_{n_+}(\boldsymbol{x})\big|_u - v_{n_-}(\boldsymbol{x})\big|_l}{2}, \tag{2.46}$$

where u stands for the z coordinate of the upper surface and l for the lower. We restrict the sources to emulate only their individual part of the radiation and obtain two separate, fully determined equation systems that can be solved as in Eq. (2.35):

$$\boldsymbol{q}_m = -j\omega\rho \boldsymbol{H}_m^{-1}\boldsymbol{v}_e, \tag{2.47}$$

$$\boldsymbol{q}_d = -j\omega\rho \boldsymbol{H}_d^{-1}\boldsymbol{v}_o, \tag{2.48}$$

where $\boldsymbol{H}_m$ is similar to Eq. (2.34). $\boldsymbol{H}_d$ holds the second-order derivatives of the Green's function, cf. Eq. (2.24)

$$\boldsymbol{H}_d = \begin{bmatrix} \frac{\partial^2 G(r_{11})}{\partial n\, \partial n_0} & \cdots & \frac{\partial^2 G(r_{1K})}{\partial n\, \partial n_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 G(r_{K1})}{\partial n\, \partial n_0} & \cdots & \frac{\partial^2 G(r_{KK})}{\partial n\, \partial n_0} \end{bmatrix}. \tag{2.49}$$

Note that separating the equation system as described does not increase its overall size. The symmetry relations allow solving the equations concerning only one of the surfaces. The boundary conditions on the other surface will inherently be satisfied.

Using the determined source strengths from Eqs. (2.47) and (2.48), the sound pressure at a single field point can be synthesized as a superposition of monopoles and dipoles

$$p(\boldsymbol{x}) = \boldsymbol{g}_m^\top \boldsymbol{q}_m - \boldsymbol{g}_d^\top \boldsymbol{q}_d, \tag{2.50}$$

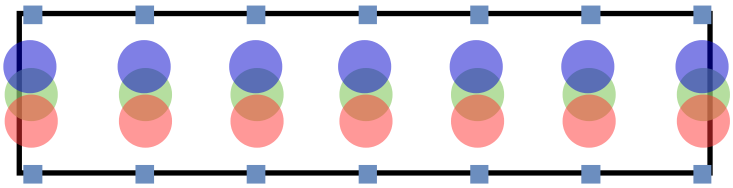


Figure 2.6 *A single layer of source positions inside the thin cuboid; centered green circles represent acoustic monopoles, and off-center blue and red circles represent dipoles. All source positions satisfy Eq.* (2.45) *since the surface nodes on the sides are ignored. The retraction distance is maximized while the up-down cross-talk is zero.*

where $\boldsymbol{g}_m$ holds the Green's functions for each source-receiver combination and $\boldsymbol{g}_d$ its normal derivatives with respect to the source coordinate:

$$\boldsymbol{g}_m = [G(r_1), \ldots, G(r_K)]^\top , \tag{2.51}$$

$$\boldsymbol{g}_d = \left[\frac{\partial G(r_1)}{\partial n_0}, \ldots, \frac{\partial G(r_K)}{\partial n_0}\right]^\top . \tag{2.52}$$

While the above-described source constellation leads to a reliable simulation output, one should remember that the formulation does not include the sound radiated sideways. Thus, it is only expected to be valid for thin bodies. The common approach to include the side surfaces would be to let the equivalent sources form an inner surface hull using a combination of monopoles and dipoles, as suggested in [12]. Another option would be to include the side-surface velocities as additional constraints for Eqs. (2.47) and (2.48) and solve the over-determined equation system as in Eqs. (2.36) and (2.38). However, these attempts were discarded, as none seemed to yield a good reproduction of the measured sound field, even when varying the relative number of sources or applying matrix regularization.

2.3 Rayleigh Integrals

When we consider a radiation scenario, the Kirchhoff-Helmholtz integral Eq. (2.28) represents a generic form to relate the boundary values of a notional body to the radiated sound pressure in a field. This universality, however, also comprises the drawback of computational complexity when the structure is of complex shape. Further, it is sometimes neither necessary nor desired to compute the sound radiation of a single body into a full, free space. The simulation of electroacoustic transducers would be an example, as they are often measured in large baffles where they radiate only into a half-space.

To simplify the Kirchhoff-Helmholtz integral, let the volume be an infinitely expanded plane in $[x, y, z = 0]^\top$ that is closed by an infinitely large hemisphere that reaches towards $z \to +\infty$. Further, let the normal derivatives head into the volume. The hemisphere's contribution to the sound field vanishes at finite observation points since the Green's functions fulfill Sommerfeld's radiation condition. The remaining equation is [44, Sec. 2.1.6]

$$p(\boldsymbol{x}) = \iint_{-\infty}^{\infty} \left(G(|||[x - x_0, y - y_0, z]|||) \frac{\partial p([x_0, y_0, z_0 = 0]^\top)}{\partial n_{z_0}} - \right.$$

$$\left. p([x_0, y_0, z_0 = 0]^\top) \frac{\partial G(|||[x - x_0, y - y_0, z]|||)}{\partial n_{z_0}} \right) \mathrm{d}x_0 \mathrm{d}y_0. \tag{2.53}$$

Note that the normal derivatives are denoted $\frac{\partial}{\partial n_{z_0}}$, as the plane is aligned horizontally and all other contributions vanish. The function arguments are written explicitly to indicate that all source positions are lying on the plane at $z_0 = 0$. Eq. (2.53) splits the infinitely large full space into two half spaces. While this would technically allow for half-room simulations, we need to provide both the sound pressure and its derivative continuously along with the entire (x, y) plane a valid result. Hence, we seek alternative Green's functions that allow eliminating one of the two quantities.

The so-called Rayleigh integrals can be derived from Eq. (2.53) correspondingly. For the derivation, the image source principle is exploited to obtain either a Neumann or a Dirichlet boundary condition for the Green's functions on the plane. A Neumann boundary generally prescribes values to the normal derivative of a boundary value. We refer to a homogeneous Neumann boundary when all values of the normal derivatives are to be zero, which represents an infinitely extended, rigid wall in the acoustic sense. By contrast, a Dirichlet boundary condition prescribes the boundary values directly instead of their derivatives. Again, we refer to a homogeneous Dirichlet boundary when all values are to be zero. In the acoustic sense, this results in a so-called pressure-release boundary. [34, Sec. 13.2, Sec. 13.6] [38, Sec. 9.3] [44, Sec. 2.1.6]

For a Green's functions with an inherent Neumann boundary on the (x, y) plane, let the boundary extend to infinity and separate two symmetrically aligned free-space Green's functions. Both are driven by the same weight. If we take the limit $\Delta z \to 0$ for the distance of the monopoles to the boundary, they will contribute equally to the sound field at any observation point. The Neumann Green's function $G_N(r)$ for the infinite plane will thus be the superposition of two collocated free-space Green's functions, i.e., the free-space Green's function times two. Considering the acoustic monopole Eq. (2.8) and its normal derivative Eq. (2.17), we obtain

$$G_N(r) = \lim_{\Delta z \to 0} \Big(G(r_+) + G(r_-) \Big) = 2G(r), \tag{2.54}$$

$$\left.\frac{\partial G_N(r)}{\partial n_z}\right|_0 = \left.\frac{\partial G(r_+)}{\partial n_{z_+}}\right|_0 + \left.\frac{\partial G(r_-)}{\partial n_{z_-}}\right|_0 = \left.\frac{z - z_0}{r}\frac{\partial G(r_+)}{\partial r_-}\right|_0 - \left.\frac{z - z_0}{r}\frac{\partial G(r_-)}{\partial r_-}\right|_0 = 0. \tag{2.55}$$

The derivatives' contributions cancel when evaluated at the (x, y) plane due to the opposite sign of derivation in the two half spaces. Using Eq. (2.54), we can form the first Rayleigh Integral, which allows for a sound-field synthesis by prescribing only the pressure derivative. Using the Euler equation Eq. (2.2), the pressure derivative can be replaced by the normal velocity on the plane

$$\begin{aligned} p([x, y, z > 0]^\top) &= 2 \iint_{-\infty}^{\infty} \frac{\partial p(\boldsymbol{x}_0)}{\partial n_{z_0}} G(r) \mathrm{d}x_0 \mathrm{d}y_0, \\ &= -2j\omega\rho \iint_{-\infty}^{\infty} v_z(\boldsymbol{x}_0) G(r) \mathrm{d}x_0 \mathrm{d}y_0. \end{aligned} \tag{2.56}$$

A Green's function with an inherent Dirichlet boundary can be obtained similarly. This time, we apply a minus sign to the lower monopole so that it radiates with an inverse phase. Due to the inversion, the direct superposition of the Green's functions vanishes when evaluated on the (x, y) plane. In contrast, their derivatives will add up due to the opposite directions of derivation. Hence, we can define the Dirichlet-Green's function

$$\begin{aligned} \frac{\partial G_D(r)}{\partial n_z} &= \lim_{\Delta z \to 0} \left(\frac{\partial G(r_+)}{\partial n_{z_+}} - \frac{\partial \left(G(r_-) \right)}{\partial n_{z_-}} \right), \\ &= \lim_{\Delta z \to 0} \left(\frac{z - z_0}{r}\frac{\partial G(r_+)}{\partial r_+} - \frac{-(z - z_0)}{r}\frac{\partial G(r_-)}{\partial r_-} \right) = 2\frac{\partial G(r)}{\partial n_z}, \end{aligned} \tag{2.57}$$

$$G_D(r)\Big|_0 = G(r_+)\Big|_0 - G(r_-)\Big|_0 = 0. \tag{2.58}$$

The second Rayleigh integral is then formed using Eq. (2.57):

$$p([x, y, z > 0]^\top) = -2 \iint_{-\infty}^{\infty} p(\boldsymbol{x}_0) \frac{\partial G(r)}{\partial n_{z_0}} \mathrm{d}x_0 \mathrm{d}y_0. \tag{2.59}$$

The two Rayleigh integrals allow for simulating the radiated sound pressure of an infinite planar radiator in the half space. Their advantage is that only one quantity, pressure or velocity, has to be known. The other quantity will be inherently correct. While it is possible to prescribe values on a finite region A on the (x, y) plane, the boundary values on the infinite complementary region $\bar{A}$ are inherently set to 0, as either the corresponding homogeneous Neumann or Dirichlet boundary condition holds.

We can numerically approximate the integrals by finite sums to compute the Rayleigh integrals. The approximation allows for computing the equations via matrix multiplications. We assume equally-spaced source positions on a rectangular grid, with position steps Δx, Δy. The discrete Rayleigh integrals for the upper half-space are then

$$\text{Rayleigh I:} \quad p([x, y, z > 0]^\top) = -2j\omega\rho \, \Delta x \Delta y \, \boldsymbol{g}^\top \boldsymbol{v}_0, \tag{2.60}$$

$$\text{Rayleigh II:} \quad p([x, y, z > 0]^\top) = -2 \, \Delta x \Delta y \, \boldsymbol{h}^\top \boldsymbol{p}_0, \tag{2.61}$$

where K is the number of source positions. $\boldsymbol{v}_0$ and $\boldsymbol{p}_0$ are $[K \times 1]$ vectors containing the discrete boundary values on the (x, y) plane. $\boldsymbol{g} = [G(r_1), \ldots, G(r_K)]^\top$ and $\boldsymbol{h} = \left[\frac{\partial G(r_1)}{\partial n_{z_0}}, \ldots, \frac{\partial G(r_K)}{\partial n_{z_0}}\right]^\top$ hold the Green's functions for each source-to-receiver combination and their normal derivatives in z direction, respectively.

2.4 A Combined Rayleigh-Integral Formulation

The Rayleigh integrals Eq. (2.56) and Eq. (2.59) are valid only in a half space. Whereas they can be deployed for both half spaces separately, no energy can travel through the boundary. Eq. (2.53) would allow for a full-space simulation, but we are required to provide both sound pressure and velocity on the boundary, which is not practical in most applications. This section introduces a formulation that combines the two Rayleigh integrals. The hope is to find a solution that enables energy transfer through the boundary plane by linking the plane's upper and lower vibrating velocities.

For this purpose, consider the boundary as an infinitely expanded planar radiator whose vibration velocities can be measured on both sides. Further, let the vibrating velocities $v_{z\pm}(\boldsymbol{x}_0)$ be the superposition of two layers of monopoles: one single-potential layer, which represents a Neumann boundary, and one double-potential layer, which forms a Dirichlet boundary as described, cf. Section 2.3. The velocities can thus be decomposed into an even-symmetric and an odd-symmetric part via Eq. (2.46). Due to the linearity of the Helmholtz equation, we can assume the sound pressure in the field as a superposition of the two Rayleigh integrals. The combined expression for the Rayleigh integrals is then

$$p(\boldsymbol{x}) = 2 \iint_{-\infty}^{\infty} \left(-j\omega\rho \, v_e(\boldsymbol{x}_0) G(r) - p_o(\boldsymbol{x}_0) \frac{\partial G(r)}{\partial n_{z_0}} \right) \mathrm{d}x_0 \mathrm{d}y_0, \tag{2.62}$$

which differs to Eq. (2.53) only in the prescribed values and the applied Euler equation. v_e stands for the even-symmetric velocity and p_o for the odd-symmetric sound pressure in the plane. For dis-

crete signals, we can again approximate the integrals as vector multiplications, assuming K source-positions:

$$p(\boldsymbol{x}) = 2\ \Delta x\ \Delta y\ \left(-j\omega\rho \boldsymbol{g}^\top \boldsymbol{v}_e - \boldsymbol{h}^\top \boldsymbol{p}_o\right). \tag{2.63}$$

Where Δx and Δy are the equally-spaced discrete steps between the source positions in the (x, y) plane. $\boldsymbol{v}_e$ and $\boldsymbol{p}_o$ are $[K \times 1]$ vectors containing the even-symmetric velocities and the odd-symmetric sound pressures respectively. $\boldsymbol{g}$ and $\boldsymbol{h}$ are as in Eqs. (2.60) and (2.61).

2.4.1 A Fourier-based approach to obtain the odd-symmetric sound pressure

The evaluation is based on velocity measurements available for the front and backside of the device. In Eq. (2.62), we can directly insert the even-symmetric velocities into the first Rayleigh integral. However, we still need to find a sound pressure distribution from the measured, odd-symmetric velocities for the second Rayleigh integral.

Sound pressure and particle velocity are related via the Euler equation Eq. (2.2). So the clue lies in finding the integral of the odd-symmetric velocity in the field concerning z. To get there, we take a detour into the so-called k space that can be used to model sound propagation. Let us first consider the Rayleigh integrals as 2D linear convolution operations:

$$p([x, y, z > 0]^\top) = -2j\omega\rho\left(v(\boldsymbol{x}_0) * *G(r)\right), \tag{2.64}$$

$$p([x, y, z > 0]^\top) = -2\left(p(\boldsymbol{x}_0) * *\frac{\partial G(r)}{\partial n_z}\right). \tag{2.65}$$

Next, we consider the Fourier-convolution theorem in 2D. The theorem states that a linear convolution simplifies to a multiplication in the Fourier domain (k space). For two functions $g(\boldsymbol{x})$ and $f(\boldsymbol{x})$, we can denote the convolution correspondingly as [45, Sec. 2.10] [46, Sec. 4.4]:

$$g(\boldsymbol{x}) * *f(\boldsymbol{x}) = \mathcal{F}_{x,y}^{-1}\Big\{\mathcal{F}_{x,y}\Big\{f(\boldsymbol{x})\Big\}\mathcal{F}_{x,y}\Big\{g(\boldsymbol{x})\Big\}\Big\}, \tag{2.66}$$

where $\mathcal{F}_{x,y}\Big\{\ \cdot\ \Big\}$ is the spatial 2D Fourier transform and $\mathcal{F}_{x,y}^{-1}\Big\{\ \cdot\ \Big\}$ its inverse.

We want to combine Eq. (2.64) and Eq. (2.65) so that we can temporarily get rid of the inverse Fourier transform and solve the equation in k space. For the velocity, we can directly Fourier transform the measured data. As for the Green's function, we find by comparison that the derivative $\frac{\partial}{\partial n_z}$ transforms to multiplication with jk_z. Considering the (Fourier-) synthesis equation for the k-space Green's function lying in the origin [37, Sec. 4.5]

$$G(||\boldsymbol{x}||) = \frac{1}{(2\pi)^3}\iiint_{-\infty}^{\infty} G(\boldsymbol{k})\mathrm{e}^{j\boldsymbol{k}^\top \boldsymbol{x}}\mathrm{d}k_x\mathrm{d}k_y\mathrm{d}k_z, \tag{2.67}$$

where the spatial wave numbers are defined as in Eq. (2.5). To find the k-space Green's function, Eq. (2.67) can be substituted into the Helmholtz equation Eq. (2.6)

$$(\Delta + k^2)G(\boldsymbol{k}) = -1, \tag{2.68}$$

which leads to

$$G(\boldsymbol{k}) = \frac{-1}{k^2 - k_x^2 - k_y^2 - k_z^2} = \frac{-1}{k^2 - \boldsymbol{k}^\top \boldsymbol{k}}. \tag{2.69}$$

The same approach leads to the formulation for the Green's function in $k_{x,y}$ space, i.e., that is dependent on the planar wave numbers k_x and k_y and the position on the z axis. The synthesis equation for $k_z \rightarrow z$ is

$$G(k_x, k_y, z) = \frac{1}{2\pi} \int_{-\infty}^{\infty} G(\boldsymbol{k}) \mathrm{e}^{jk_z z} \mathrm{d}k_z, \tag{2.70}$$

and points out that the normal derivative in z becomes

$$\frac{\partial G(k_x, k_y, z)}{\partial n_z} = jk_z G(k_x, k_y, z). \tag{2.71}$$

For the purpose of finding the sound pressure, we do not need to calculate Eq. (2.70) explicitly. This is getting more obvious when we combine the Fourier-transformed Eq. (2.64) and Eq. (2.65):

$$-2j\omega\rho \, v(k_x, k_y, z = 0) \, G(k_x, k_y, z) = -2jk_z \, p(k_x, k_y, z = 0) \, G(k_x, k_y, z). \tag{2.72}$$

The Green's functions in Eq. (2.72) cancel, and finally, we obtain a formulation for the desired sound pressure via Fourier synthesis:

$$p([x, y, z = 0]^\top) = \omega\rho \mathcal{F}_{x,y}^{-1} \left\{ \frac{\mathcal{F}_{x,y}\left\{ v([x, y, z = 0]^\top) \right\}}{k_z} \right\}. \tag{2.73}$$

2.4.2 Determine the outwards-normal wave number k_z

In a next step, we have to determine what is unknown in Eq. (2.73), the wave number in z-direction, k_z. To do so, we reformulate the characteristic equation Eq. (2.5), so that k_z is the dependent variable:

$$k_z = \pm\sqrt{k^2 - k_x^2 - k_y^2}. \tag{2.74}$$

Eq. (2.74) contains all mathematically possible solutions for k_z. However, not all solutions are valid for every physical situation; hence, we have to choose a subset of solutions. We have four situations: plane waves and evanescent waves, propagating either into the positive or the negative half space. The waves should propagate outwards from the plane and must fulfill the Sommerfeld condition, therefore also vanish at $r \rightarrow \pm\infty$. A supersonic bending wave, which corresponds to $k^2 \geq k_x^2 + k_y^2$, will result in a real-valued k_z [45, Sec. 2.6]. In this case, a harmonic plane wave with the exponent $\mathrm{e}^{-jk_z|z|}$ is being radiated from the plane. This wave must be directed outwards, which we have defined in a negative exponent. Hence, to obtain $\Im\{-jk_z|z|\} < 0$, the real part must be $\Re\{k_z\} \geq 0$. A subsonic bending wave, where $k^2 < k_x^2 + k_y^2$, causes k_z to be imaginary. Here, an evanescent wave with a real-valued exponent will be radiated. The exponent must satisfy $\Re\{-jk_z|z|\} < 0$ to vanish at infinity. The claim is here for the imaginary part $\Im\{k_z\} < 0$, so that the imaginary-valued $k_z = -j|k_z|$ forms a negative sign when multiplied by $-j$. We can conclude the only valid solutions as

$$k_z = \sqrt[*]{k^2 - k_x^2 - k_y^2}, \tag{2.75}$$

where $\sqrt[*]{\cdot}$ denotes the complex conjugate of the positive root, causing a positive real- and a negative imaginary-part. Note that Eq. (2.75) is only valid for $z \geq 0$. There is no need to specify k_z for $z < 0$ here, since Eq. (2.73) is evaluated only at $z = 0$.

2.4.3 Dealing with irregularities at tangential waves

Even though we are already able to compute the odd-symmetric sound pressure on the (x, y) plane using Eq. (2.73) and Eq. (2.75), there is another trouble spot we have to consider. The issue occurs when the characteristic equation Eq. (2.5) is fulfilled for waves that travel tangentially along the (x, y) plane without an outwards-normal component. Tangential waves occur at coincidence frequency, where a bending wave propagates with the speed of sound. In this case, $k^2 = k_x^2 + k_y^2$ so that $k_z = 0$, which causes the division by k_z in Eq. (2.73) to diverge.

An explanation for this phenomenon can be given considering the two Rayleigh integrals Eqs. (2.56) and (2.59). By substituting Eq. (2.73) into Rayleigh II, we try to emulate its radiated sound pressure using Rayleigh I. There is a directivity mismatch between the Dirichlet Green's function Eq. (2.58), representing an acoustic dipole, and the Neumann Green's function Eq. (2.54), representing a monopole. This mismatch is compensated via $G_N(k_x, k_y, z) = \frac{G_D(k_x,k_y,z)}{jk_z}$ in the $k_{x,y}$ space. $G_D(r)$ is zero when evaluated on the (x, y) plane, implying that also $G_D(k_x, k_y, z)\big|_0 = 0$. Hence, no strictly tangential waves can occur. As $G_N(r)$ is omnidirectional, it allows for tangential-wave propagation and thus is non-zero when the characteristic equation is fulfilled in the horizontal plane. The factor $\frac{1}{jk_z}$ must thus grow towards infinity to compensate the inequality.

A more physical argument for the incorrect modeling of tangential waves can be given, considering the (x, y) plane as a vibrating plate. The mathematical model assumes *"...the normal velocity of the plate to be continuous with the fluid velocity in contact with it. However, a plane wave traveling parallel to the plate has only a fluid velocity parallel to the plate (in the direction of travel) and zero normal velocity."*[45, P. 31]

The declared problem can be mitigated by introducing a lower barrier ϵ, allowing k_z only to shrink to a minimum value:

$$\hat{k}_z = \begin{cases} k_z & |k_z| \geq |\epsilon|, \\ \epsilon & |k_z| < |\epsilon|. \end{cases} \tag{2.76}$$

Replacing k_z by $\hat{k}_z$ is a rather simple remedy, but there remains a trade-off when choosing ϵ. Large values distort the result in a non-physical and non-linear manner, while small values are not capable of smoothing the peaks sufficiently.

Regularization of k_z

A more sophisticated approach to regularize a very similar problem is presented by Pagavino et al. [47, Sec. 2.4.1]. Pagavino treated the Fourier-based calculation of sound fields from prescribed velocities [45, Sec. 2.9]. Eq. (2.73) is the same equation but with a missing exponential term because we evaluate only at positions $z = 0$. This paragraph is devoted to briefly presenting Pagavino's regularization procedure applied for Eq. (2.73).

In the beginning, it is informative to write down the inverse Fourier transform. We denote $k_z(k_x, k_y)$ to emphasize the dependencies of the variable:

$$p([x, y, z = 0]^\top) = \frac{\omega\rho}{4\pi^2} \iint_{-\infty}^{\infty} v(k_x, k_y, z = 0) \frac{1}{k_z(k_x, k_y)} \mathrm{e}^{jk_x x + jk_y y} \mathrm{d}k_x \mathrm{d}k_y. \tag{2.77}$$

As we usually have to deal with discrete signals of finite length instead of infinite, continuous ones, Eq. (2.77) must be discretized. Assume $v[k_{x_n}, k_{y_m}, z = 0]$ being the 2D discrete Fourier transform

of the equally-spaced sampled velocities on the (x, y) plane. A discrete form of the integral is then

$$p([x, y, z = 0]^\top) = \frac{\omega\rho}{4\pi^2} \sum_n \sum_m v[k_{x_n}, k_{y_m}, z = 0] \frac{1}{k_z[k_{x_n}, k_{y_m}]} \mathrm{e}^{jk_{x_n}x + jk_{y_m}y} \, \Delta k_x \, \Delta k_y, \quad (2.78)$$

where $k_{x_n} = n\Delta k_x$ and $k_{y_m} = m\Delta k_y$ are the sampled wave number bins in the (k_x, k_y) plane.

The discretization, however, comprises a drawback. A singularity in Eq. (2.77) is, in theory, infinitely narrow since pure-tangential waves only occur exactly at the coincidence frequency. By contrast, the discretized wave-number bins in Eq. (2.78) might lead to $k_z \approx 0$ on a broader frequency band due to truncation of the values.

A remedy to prevent the occurrence of these near-singular values is to separate the discretization into two parts. For every other but the $\frac{1}{k_z}$ term, the sampling approach works fine as it does not lead to singularities. For $\frac{1}{k_z}$, an alternative discretization must be found.

The alternative discretization is done in two steps. First, the sampled $\frac{1}{k_z[k_{x_n}, k_{y_m}]}$ is interpolated, using a two-dimensional interpolation function. Second, the integral is evaluated analytically over the extent of the interpolation function. The integration assures that a sporadically occurring singular bin of $\frac{1}{k_z[k_{x_n}, k_{y_m}]}$ is smoothed before it can distort the outcome.

The interpolation function can be a two-dimensional rectangular function that is centered to the respective wave-number bin and has an extend of $l_{x,y}\Delta k_{x,y}$, i.e., $(l_{x,y} \in \mathbb{R}^+)$ times a wave-number step in both dimensions of the (k_x, k_y) plane:

$$\prod_{((n,m)-l_{x,y}))\Delta k_{x,y}}^{((n,m)+l_{x,y})\Delta k_{x,y}} (k_x, k_y) = \begin{cases} 1, & \text{for } ((n,m) - l_{x,y})\,\Delta k_{x,y} \leq k_{x,y} \leq ((n,m) + l_{x,y})\,\Delta k_{x,y}, \\ 0, & \text{otherwise.} \end{cases} \quad (2.79)$$

where we concluded the two dimensions into a comma-separated notation. Hence, (n, m) corresponds to the n^{th} and m^{th} bin of k_x and k_y. $k_{x,y}$ and $l_{x,y}$ represent the wave numbers and extent of the interpolation function in both dimensions. Including the interpolation function into Eq. (2.78) leads to

$$p([x, y, z = 0]^\top) = -\frac{j\omega\rho}{4\pi^2} \sum_n \sum_m v[k_{x_n}, k_{y_m}, z = 0] \mathrm{e}^{jk_{x_n}x + jk_{y_m}y} \ \ldots$$
$$\iint_{((n,m)-l_{x,y})\Delta k_{x,y}}^{((n,m)+l_{x,y})\Delta k_{x,y}} \frac{1}{-jk_z[k_{x_n}, k_{y_m}]} \mathrm{d}k_x \mathrm{d}k_y, \quad (2.80)$$

where a factor $\frac{-j}{-j}$ is added to comply Pagavino's notation.

An analytic solution for the remaining integral in Eq. (2.78) can be found by using polar coordinates so that the planar wave numbers are described by a radius r and a polar angle $\varphi(r)$. The integral can then be simplified, assuming $\varphi(r)$ to vary approximately linear within the region of two corners of the interpolation function. The transformed expression for a single bin in the (k_x, k_y) plane is then

$$\Upsilon = \int_{r_1}^{r_4} \frac{1}{-jk\sqrt[*]{1 - \left(\frac{r}{k}\right)^2}} r\Delta\varphi(\boldsymbol{r}) \mathrm{d}r, \quad (2.81)$$

where Υ is the regularized outcome of the integral and $r_{1\ldots4}$ are the four radii defining the edges of the interpolation function. $\Delta\varphi(\boldsymbol{r})$ is here the maximum angular change within the region of interpolation. $\boldsymbol{r} = [k_x, k_y]^\top$ is the position vector in the (k_x, k_y) plane and r its length.

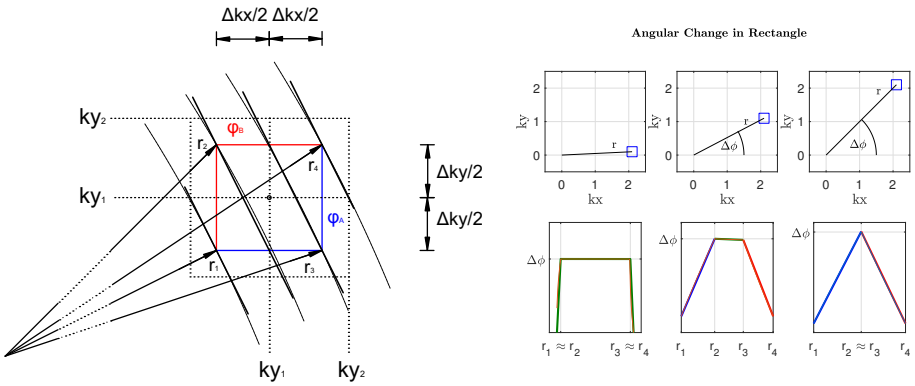


Figure 2.7 *Illustration of the four edge-radii and the approximated angular change, from [47, P. 45]; left: a segment in the lower-left quadrant of the $k_{x,y}$-plane, showing the radii defining the extent of the interpolation function. The skewed lines represent the linearized angular change. Right: Angular change as a function of the edge radii for three different positions in the $k_{x,y}$-plane.*

To solve Eq. (2.81) analytically, Pagavino decomposed the integral into three intervals. Each interval shows a characteristic behavior of $\Delta\varphi(\boldsymbol{r})$. In the interval r_1-r_2, $\Delta\varphi(\boldsymbol{r})$ grows steadily. For r_2 -r_3, it approximately remains constant and for r_3-r_4, it diminishes steadily. Fig. 2.7 shows a screenshot from [47, P. 45], illustrating radii and angular change in the (k_x, k_y) plane.

The analytic integration of the three parts leads to the following equations. The radii $r_{1\ldots4}$ and the maximum angular change $\Delta\varphi(\boldsymbol{r})$ are derived via geometrical considerations, as illustrated in Fig. 2.7. The radii are expressed as functions of the discrete wave-number steps. The maximum angular change is computed as the angle between the edge $\boldsymbol{r}_2$, and a spot $\boldsymbol{c}$, where the horizontal line $(\boldsymbol{r}_3 - \boldsymbol{r}_1)$ is cut by a circle that is drawn from the origin with a radius r_2. We obtain:

$$\begin{aligned} r_1 &= \sqrt{\left[\Delta k_x\,(n-l_x)\right]^2 + \left[\Delta k_y\,(m-l_y)\right]^2} \quad r_2 = \sqrt{\left[\Delta k_x\,(n-l_x)\right]^2 + \left[\Delta k_y\,(m+l_y)\right]^2} \\ r_3 &= \sqrt{\left[\Delta k_x\,(n+l_x)\right]^2 + \left[\Delta k_y\,(m-l_y)\right]^2} \quad r_4 = \sqrt{\left[\Delta k_x\,(n+l_x)\right]^2 + \left[\Delta k_y\,(m+l_y)\right]^2}, \end{aligned} \tag{2.82}$$

$$\begin{aligned} \Delta\varphi(\boldsymbol{r}) = \varphi_2 - \varphi_c &= \arctan\left(\frac{k_y(r_2)}{k_x(r_2)}\right) - \arctan\left(\frac{k_y(r_3)}{k_x(c)}\right) \\ &= \arctan\left(\frac{\Delta k_y\,(m+l_y)}{\Delta k_x\,(n-l_x)}\right) - \arctan\left(\frac{\Delta k_y\,(m-l_y)}{\sqrt{(r_2)^2 - \left(\Delta k_y\,(m-l_y)\right)^2}}\right), \end{aligned} \tag{2.83}$$

$$\Upsilon_{2,3} = \Delta\varphi(\boldsymbol{r})k\left\{\left(-\sqrt{\left(\frac{r_3}{k}\right)^2 - 1}\right) - \left(-\sqrt{\left(\frac{r_2}{k}\right)^2 - 1}\right)\right\}, \tag{2.84}$$

$$\begin{aligned} \Upsilon_{1,2} = \frac{k\Delta\varphi(\boldsymbol{r})}{(r_2 - r_1)}\Bigg\{\frac{k}{2}\Bigg[&\left(\frac{r_2}{k}\right)\left(-\sqrt{\left(\frac{r_2}{k}\right)^2 - 1}\right) - \left(\frac{r_1}{k}\right)\left(-\sqrt{\left(\frac{r_1}{k}\right)^2 - 1}\right) + \\ &\ln\left(\frac{\left(-\sqrt{\left(\frac{r_2}{k}\right)^2 - 1}\right) + \left(\frac{r_2}{k}\right)}{\left(-\sqrt{\left(\frac{r_1}{k}\right)^2 - 1}\right) + \left(\frac{r_1}{k}\right)}\right)\Bigg] - \end{aligned}$$

$$r_1\left[\left(-\sqrt{\left(\frac{r_2}{k}\right)^2-1}\right)-\left(-\sqrt{\left(\frac{r_1}{k}\right)^2-1}\right)\right]\Bigg\}, \tag{2.85}$$

$$\Upsilon_{3,4}=\frac{-k\Delta\varphi(\boldsymbol{r})}{(r_4-r_3)}\Bigg\{\frac{k}{2}\Bigg[\left(\frac{r_4}{k}\right)\left(-\sqrt{\left(\frac{r_4}{k}\right)^2-1}\right)-\left(\frac{r_3}{k}\right)\left(-\sqrt{\left(\frac{r_3}{k}\right)^2-1}\right)+$$
$$\ln\left(\frac{\left(-\sqrt{\left(\frac{r_4}{k}\right)^2-1}\right)+\left(\frac{r_4}{k}\right)}{\left(-\sqrt{\left(\frac{r_3}{k}\right)^2-1}\right)+\left(\frac{r_3}{k}\right)}\right)\Bigg]-$$
$$r_3\left[\left(-\sqrt{\left(\frac{r_4}{k}\right)^2-1}\right)-\left(-\sqrt{\left(\frac{r_3}{k}\right)^2-1}\right)\right]\Bigg\}, \tag{2.86}$$

$$\Upsilon=\frac{\Upsilon_{1,2}+\Upsilon_{2,3}+\Upsilon_{3,4}}{4\,l_x l_y\,\Delta k_x\Delta k_y}. \tag{2.87}$$

Note that Eqs. (2.82) to (2.87) differ from the results in [47]. One difference is that Pagavino used a constant factor $l=\frac{1}{2}$ to define the extent of the interpolation function. Further, the normalization term $\frac{1}{4\,l_x l_y\,\Delta k_x\Delta k_y}$ is added to Eq. (2.87), the complex-conjugate square roots are exchanged for the negative ones in Eqs. (2.84) to (2.86), and some formal typos are fixed. These modifications were necessary for the simulation to comply with the acoustic reference measurement.

Eq. (2.87) solves Υ only within the lower-left quadrant of the (k_x,k_y) space. Hence, we must apply some further modifications to obtain a complete solution. First, consider that Υ diverges at $(m,n)=(0,0)$. For this single wave-number bin, we can exchange the value by the non-regularized $\frac{1}{k_z}$ since there is no bending wave that could enforce a tangential wave in the air. Next, for finding a solution in the three other quadrants of the (k_x,k_y) plane, we can circumvent the explicit computation by exploiting the symmetries of the k space. k-space expressions for the Green's function or its derivative are always rotationally symmetric, i.e., they contain the same amplitude for any azimuthal plane wave of the same zenith propagation angle. Let us conclude the result using matrix notation

$$\boldsymbol{\Upsilon}_{tot}=\begin{bmatrix}\boldsymbol{\Upsilon} & \leftrightarrow(\boldsymbol{\Upsilon})\\ \updownarrow(\boldsymbol{\Upsilon}) & \circlearrowleft(\boldsymbol{\Upsilon})\end{bmatrix}, \tag{2.88}$$

where the first entry of the matrix is replaced by $\frac{1}{-jk_z[0,0]}$. Hereby, $\boldsymbol{\Upsilon}$ contains the solution for all bins up to the Nyquist bin: $(0,0)<(n,m)\leq\frac{(N,M)}{2}$. The $\leftrightarrow(\cdot)$ operator indicates a left-to-right flip of the matrix after deletion its zeroth column. $\updownarrow(\cdot)$ is an up-down flip of the matrix after deletion of its zeroth row. $\circlearrowleft(\cdot)$ indicates a rotation by $180°$, after deletion both the zeroth column and row.

The regularized equation for obtaining the desired sound pressure on the plane can finally be stated as an inverse 2D discrete Fourier transform

$$p([x,y,z=0]^\top)=-\frac{j\omega\rho}{4\pi^2}\sum_n\sum_m v[k_{x_n},k_{y_m},z=0]\Upsilon_{tot}[n,m]\mathrm{e}^{jk_{x_n}x+jk_{y_m}y}$$
$$=-j\omega\rho\,\mathrm{iFFT}_{k_x,k_y}\Big[\mathrm{FFT}_{x,y}\Big[v[x_n,y_m]\Big]\Upsilon_{tot}[n,m]\Big], \tag{2.89}$$

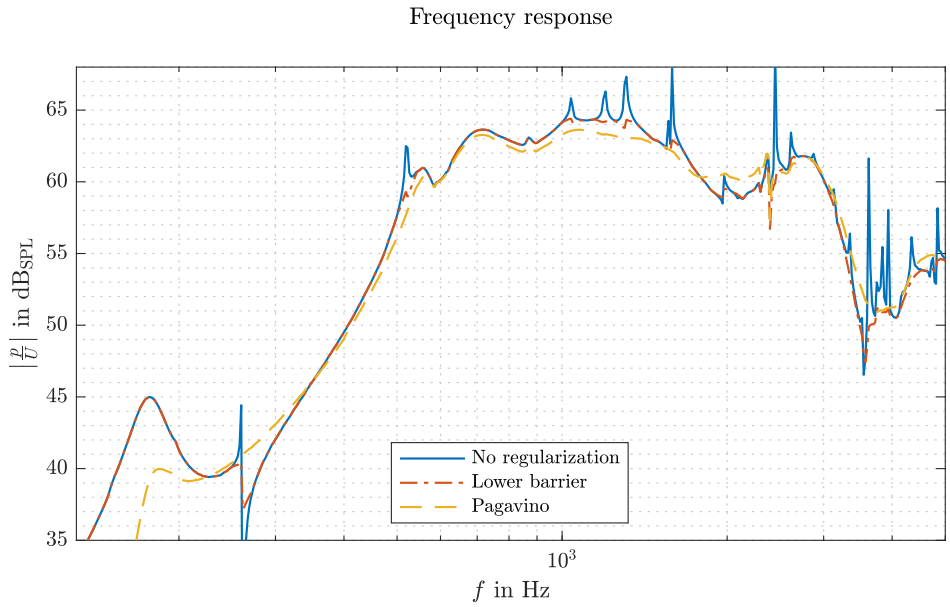


Figure 2.8 *Full-space frequency responses of the smartphone dummy, simulated without regularization of k_z, with a lower barrier $\epsilon = 0.3$, and with Pagavino's regularization using the parameters defined in Section 4.2.4. The frequency response is evaluated in* $10\ \mathrm{cm}$ *distance in the front-center of the device*

where $v[x_n, y_m]$ is the velocity function, sampled at the coordinates $(x_n, y_m) = (n\Delta x, m\Delta y)$. $\mathrm{FFT}_{x,y}$ denotes the 2D fast Fourier transform with respect to the (x, y) coordinate and iFFT_{k_x,k_y} its inverse.

The effect of regularizing k_z can be observed in Fig. 2.8, which shows frequency responses evaluated using the combined-Rayleigh formulation. Without regularization, the frequency response contains ripples that are not physical. Applying a lower barrier for k_z as defined in Eq. (2.76) results in a smoother curve that, however, still contains some of the ripples. By contrast, the curve that uses Pagavino's regularization does not yield such ripples and matches well with the acoustic measurement (cf. Chapter 5).

2.4.4 Approximating the linear convolution

In a final step, consider that the two Rayleigh integrals express a 2D linear convolution of infinite-length signals. In contrast, Eq. (2.89) forms a 2D circular convolution of finite-length signals [46, Sec. 4.4]. Hence, formulation suffers from the inherent periodicity of the discrete Fourier transform that replicates the signals to an infinite extent [48, Sec 8-3.1].

A common remedy is to pad the signal with zeros, pushing the replicas apart. For a linear convolution of finite-length signals, it would be sufficient to pad both signals to a length [48, Sec 5-4.3.2]

$$L(a * b) = L_a + L_b - 1. \tag{2.90}$$

In our case, the zero-padded and Fourier-transformed velocity function $v[k_{x_n}, k_{y_m}]$ determines the length of $k_z[k_{x_n}, k_{y_m}]$. Further, the inverse-Fourier-transformed $\frac{1}{-jk_z}$ does not yield zeros outside the vibrating region, but shows some exponential-decay bathtub shape. We thus need to add more zeros than stated in Eq. (2.90) for approximating the linear convolution appropriately. Fig. 2.9 depicts the inverse-transformed $\frac{1}{-jk_z}$ for various lengths of k_z, showing that a higher number of padded zeros broadens the function spatially.

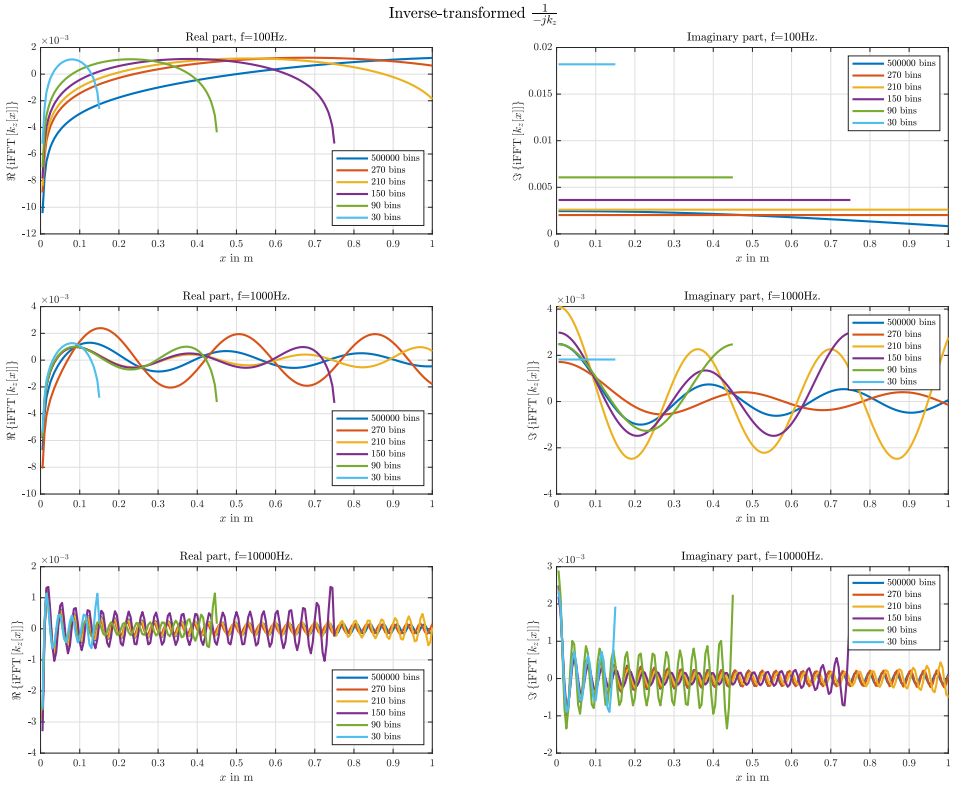


Figure 2.9 *Inverse-Fourier-transformed $\frac{1}{-jk_z}$ for a two-dimensional case along the x axis, with a spatial step size of $\Delta x = 5$ mm. The number of bins determines the spatial extend of the function.*

Here, too, a trade off concerning the number of padded zeros has to be made. While increasing the number enhances the approximation to a linear convolution, it also increases the computational cost. Additionally, the extent of the interpolation function must be adjusted so that the finer resolved $\frac{1}{-jk_z}$ is smoothed sufficiently.

A problem with zero padding signals is the discontinuity occurring at the borders between signal and padded zeros. The discontinuity represents a convolution with a rectangular window and impacts the resulting wave-number spectrum. The common remedy is multiplying the signal with another window function to smooth the transition. The topic of windowing is beyond the scope of this thesis. Detailed listings of window functions and correction factors can be found in [49] and [50].

While zero padding is a standard procedure in signal processing, it also comprises a drawback: the window introduces a loss of information in the primary signal, especially at high frequencies. A remedy here can be so-called wave-field extrapolation, i.e., extrapolating the signal with values other than zero [51]. The extrapolated function allows for applying a broader window that does not influence the primary signal, e.g., a Tukey window [50, Sec. 4.22] with an onset at the original limit of the signal. Common methods of wave-field extrapolation are so-called border padding, and linear-predictive border padding [52]. Border padding is very similar to zero padding, except that here signal is padded by the repeated boundary values. Linear-predictive border padding uses some method of linear prediction, e.g. a polynomial fit [53], [54]. Fig. 2.10 illustrates the principles of zero padding, border padding, and linear-predictive border padding for a one-dimensional case.

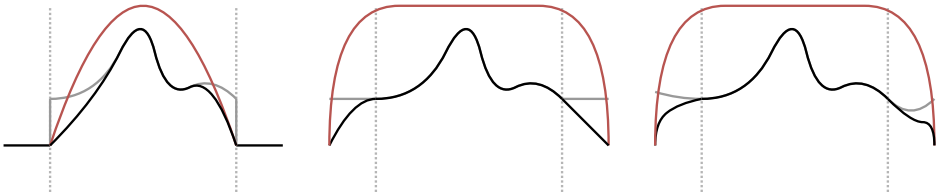


Figure 2.10 *Illustration of zero padding and wave-field extrapolation. Dashed gray lines represent the borders of the signal. Steady gray lines show the non-windowed signal and extrapolations, red lines illustrate window functions. Black lines show the resulting, extrapolated and windowed curves. From left to right: zero padding, border padding, linear-predictive border padding.*

2.5 Boundary Elements Method (BEM)

To begin with, the Kirchhoff-Helmholtz integral Eq. (2.28) is restated here, after replacing the pressure gradient via the Euler equation Eq. (2.2):

$$c(\boldsymbol{x})p(\boldsymbol{x}) = -j\omega\rho \oint_{\Gamma_0} v_n(\boldsymbol{x}_0)G(r)\mathrm{d}\Gamma_0 - \oint_{\Gamma_0} p(\boldsymbol{x}_0)\frac{\partial G(r)}{\partial n_0}\mathrm{d}\Gamma_0. \tag{2.91}$$

The idea behind the BEM in acoustics lies in solving Eq. (2.91) numerically, as analytical solutions can only be found for a limited set of simply shaped surfaces [30, Sec. 4]. The numerical integration requires a surface mesh, i.e., discretization of the boundary into smaller elements of simple shape, mostly triangular or quadrilateral elements. The BEM covers aspects that this thesis cannot cover, such as numerical integration and how to treat singular integrals. Thus, unlike the formerly described methods, additional software is employed for its core computation. The open-source software Salome [32] is used for computing the surface mesh, and the open-source Matlab toolbox OpenBem [31] takes care of loading/checking the mesh and computing the BEM matrices. This section discusses some key points of BEM theory required for understanding its application to the underlying exterior radiation problem. Interested readers are referred to [55], [29], and [30].

2.5.1 Discretizing the boundary

The first step for the numerical computation is discretizing the shape of the body and discretizing the prescribed velocities and sound pressures. The discretization can be done via the collocation method of weighted residuals [55, Sec. 1.3], which approximates the equation with a function that satisfies the boundary values at a set of discrete positions, i.e., the collocation points. The discretization is termed isoparametric if the boundary values and the surface are treated equally [30, Sec. 4.4.3].

The discretization of a three-dimensional body decomposes its surface into a mesh of smaller elements. The closed surface integral then breaks up into a sum of integrals over the surface elements

$$c(\boldsymbol{x})p(\boldsymbol{x}) = \sum_{l=1}^{L}\left(-j\omega\rho\int_{\Gamma_l} v_n(\boldsymbol{x}_l)G(r_l)\,\mathrm{d}\Gamma_l\right) - \sum_{l=1}^{L}\left(\int_{\Gamma_l} p(\boldsymbol{x}_l)\frac{\partial G(r_l)}{\partial n_0}\,\mathrm{d}\Gamma_l\right), \tag{2.92}$$

whereas l indicates the l^{th} surface element. In the simplest case, assuming constant surface elements, the continuous functions of the boundary values can be directly pulled out of the integrals, and we can thus rewrite Eq. (2.92) as vector products,

$$c\,p = \boldsymbol{m}^\top \boldsymbol{v}_n - \boldsymbol{d}^\top \boldsymbol{p}. \tag{2.93}$$

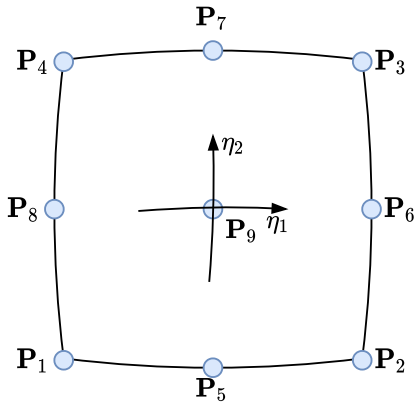


Figure 2.11 *Illustration of a quadratic quadrilateral element, showing its nine surface nodes. The image is drawn in the style of [29, P. 93, Fig. 3].*

Where the $\boldsymbol{v}_n$ and $\boldsymbol{p}$ contain the grid points for each surface element and $\boldsymbol{m}^\top$ and $\boldsymbol{d}^\top$ comprise the integrals of the monopole and dipole terms in the Kirchhoff-Helmholtz integral, respectively:

$$m_l = -j\omega\rho \int_{\Gamma_l} G(r_l)\mathrm{d}\Gamma_l, \quad d_l = \int_{\Gamma_l} \frac{\partial G(r_l)}{\partial n_0}\, \mathrm{d}\Gamma_l. \tag{2.94}$$

Taking account of non-constant quantities over a single surface element can be done by approximating the continuous functions via a sum of weighted basis functions [29, Sec. 2.2]:

$$f(\boldsymbol{x}) \approx \sum_{k=1}^{K} \phi_k(\boldsymbol{x}) f_k. \tag{2.95}$$

Where $f(\boldsymbol{x})$ is a place holder for the sound pressure or velocity and $f_{1\ldots K}$ are the extracted values on the collocation points. $\phi_k(\boldsymbol{x})$ are basis functions.

For surface elements of non-constant shape, e.g., linear or quadratic elements, the elements can be defined by a superposition of Lagrangian polynomials:

$$\boldsymbol{x} = \sum_{k=1}^{K} N_k(\eta_1, \eta_2)\boldsymbol{P}_k, \tag{2.96}$$

where the positions $\boldsymbol{P}_{1\ldots K}$ hold the coordinates of the so-called surface nodes of a single element. The Lagrangian polynomials $N_k(\eta_1, \eta_2)$ have a value of one only at a single node and zero at all other nodes [29, Sec. 5.2]. They are often termed 'shape functions' when used in this context [30, Sec. 4.7]. Fig. 2.11 illustrates definition of the surface nodes and the coordinates (η_1, η_2) of a quadratic quadrilateral element.

2.5.2 Calibrating the sound pressure boundary values

In many practical applications only one quantity is known, either the normal velocity or the pressure on the surface. The other quantity is inherently determined by the nature of the Kirchhoff-Helmholtz integral, c.f. Section 2.1.6. Hence, the boundary-elements procedure is usually done in two steps: calibration of the dependent quantity and subsequent computation of the desired problem.

In this thesis, we consider a radiation problem with prescribed normal velocities. For the calibration, we evaluate Eq. (2.93) on all collocation points, so that the $\boldsymbol{m}$ and $\boldsymbol{d}$ vectors expand to $L \times L$ matrices

of the form

$$\boldsymbol{M} = \begin{bmatrix} m_{1,1} & \dots & m_{1,L} \\ \vdots & \ddots & \vdots \\ m_{L,1} & \dots & m_{L,L} \end{bmatrix}, \boldsymbol{D} = \begin{bmatrix} d_{1,1} & \dots & d_{1,L} \\ \vdots & \ddots & \vdots \\ d_{L,L} & \dots & d_{L,L} \end{bmatrix}. \tag{2.97}$$

c expands to an $L \times L$ diagonal matrix, holding the position-dependent constants for the surface, c.f. Eq. (2.29). The surface pressures are then determined via matrix inversion,

$$\boldsymbol{p} = (\boldsymbol{C} + \boldsymbol{D})^{-1} \boldsymbol{M} \boldsymbol{v}_n. \tag{2.98}$$

The sound pressure on a single point in the field can finally be determined via Eq. (2.93) using the determined set of values.

2.5.3 Non-uniqueness and CHIEF points

When evaluated on the (smooth) surface, the interior and exterior formulation of the Kirchhoff-Helmholtz integral comprise the same integral terms for monopoles and dipoles, except for the inverted sign in the dipole part. This relationship leads to a non-physical behavior in the BEM when computing an exterior radiation problem, which is known as the non-uniqueness problem [29, Sec. 7.1]. Internal resonances for a Dirichlet ($\boldsymbol{p} = \boldsymbol{0}$) or a Neumann ($\boldsymbol{v}_n = \boldsymbol{0}$) boundary lead to near-singular entries in the $\boldsymbol{m}$ and $\boldsymbol{d}$ vectors of Eq. (2.93) for the interior formulation, distorting the exterior radiation problem since the resonances do not occur in the outer field. The frequencies at which the non-uniqueness problem occurs are often termed irregular frequencies.

A simple remedy to the non-uniqueness problem is the Combined Helmholtz Integral Equation Formulation (CHIEF), first proposed by [27]. The clue is to add so-called CHIEF points to the exterior radiation problem. The CHIEF points are located in the interior volume and serve as additional constraints for the equation system, forcing the sound pressure to zero at their location. The resulting over-determined equation system can then be solved using a least-squares approach [29, Sec. 7.2], i.e., minimizing the norm of the equation

$$\begin{aligned} \boldsymbol{p} &= \left((\boldsymbol{C} + \boldsymbol{D})^{\mathsf{H}} (\boldsymbol{C} + \boldsymbol{D}) \right)^{-1} (\boldsymbol{C} + \boldsymbol{D})^{\mathsf{H}} \boldsymbol{M} \boldsymbol{v}_n, \\ &= (\boldsymbol{C} + \boldsymbol{D})^{\dagger} \boldsymbol{M} \boldsymbol{v}_n, \end{aligned} \tag{2.99}$$

where $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudo inverse, and $(\cdot)^{\mathsf{H}}$ is the Hermitian-transposed of a matrix.

Application of the CHIEF method comprises a major drawback: the constraints hold only for the discrete positions of the CHIEF points. Hence, there remains the possibility of a non-zero field aside from the CHIEF points. The major difficulties lie, therefore, in *"...the determination of the number of good CHIEF points needed to ensure a unique solution, ...and how to choose good CHIEF points."* [30, P. 116]. A good CHIEF point adds *"...an additional constraint to the original system of equations..."*[30, P. 116], i.e., it does not coincide with a nodal surface of internal resonance.

The conditions are often determined via a trial-and-error procedure, while there exist indicators for sufficiently many good CHIEF points, such as the condition number of the pseudo inverse or the interior velocity potential, as suggested by [28]. Thus, careful consideration of the positioning can save time and workload in the process of finding an acceptable solution. As Seybert et al. find:

"As the nodal surfaces of an arbitrary body do not have any regular pattern (e.g.,the nodal surfaces will not, in general, be equally spaced), it seems that a uniform pattern of CHIEF points is appropriate. On the other hand, a body with parallel surfaces ... has equally spaced nodal surfaces; in this case, the CHIEF points should not be spaced at regular intervals." [28, Sec. V]

3 Data Acquisition

Before being able to run the simulation, we need to acquire the vibration velocities. Additionally, we need to verify the validity of the simulations and hence, must conduct acoustic reference measurements. This chapter describes the measurement processes for obtaining the surface velocities and the reference sound pressure. Beforehand, the tested device is introduced. All measurements took place at Sound Solutions in Vienna. The company has provided the infrastructure, including the devices under test, appropriate mountings, and adapters.

3.1 Tested Devices

The device under test was a smartphone dummy with vibrating actuators attached to the inner side of its display and side lengths of $15.55\ \mathrm{cm} \times 7.26\ \mathrm{cm} \times 1.1\ \mathrm{cm}$. The components of the dummy are

- a front display,
- a transparent back cover,
- an aluminum frame,
- an aluminum block to dampen interior vibrations,
- two miniature loudspeakers in enclosures (not used in the project),
- two vibrating actuators of type 'SiDiAc 102, TR280', with resonance frequency at $180\ \mathrm{Hz}$,
- wiring that ends in an RJ-45 bucket outside.

Photographs of the dummy are shown in Fig. 3.2.

As a reference case, additional measurements of a miniature voice-coil loudspeaker of type 'Manticore HX617' were conducted. These measurements provide an alternative boundary condition to verify the half-space simulation more broadly and serve as a cross-check, as they are expected to be less prone to errors compared to the half-space measurements of the smartphone dummy. The Manticore has a size of $1.5\ \mathrm{cm} \times 1.1\ \mathrm{cm} \times 0.2\ \mathrm{cm}$. Fig. 3.1 shows a photograph of the Manticore loudspeaker.

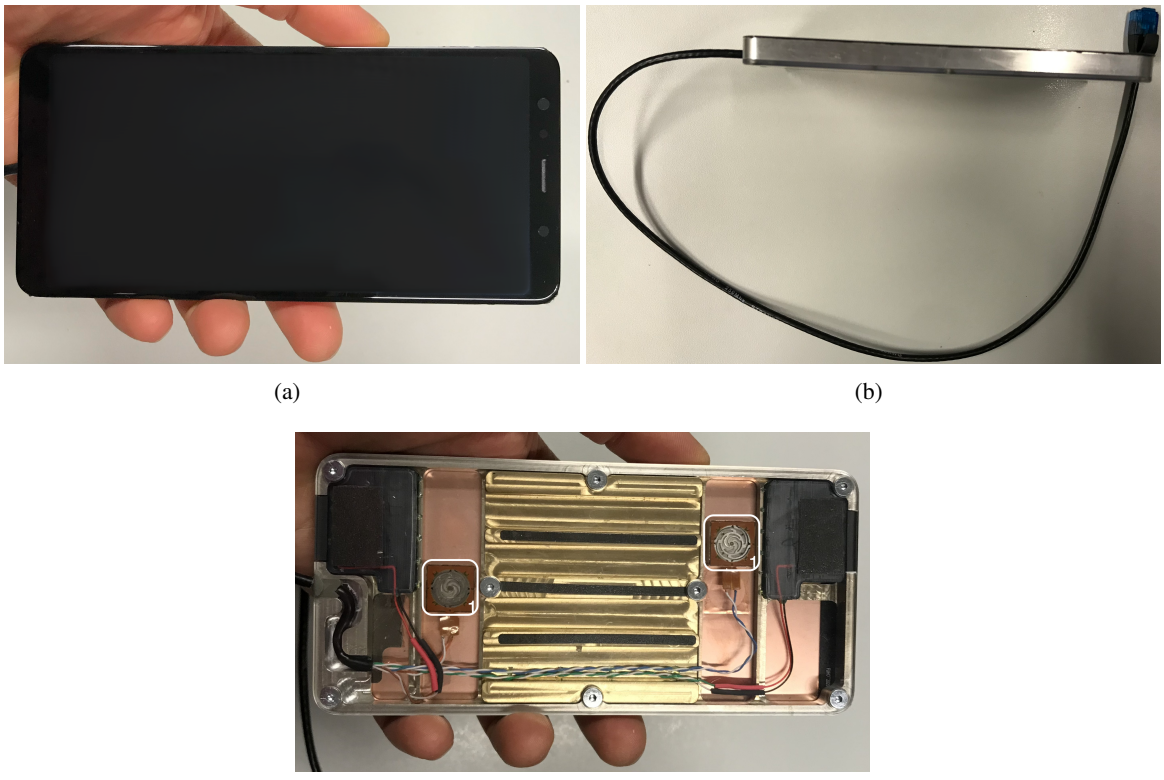
3.2 Scanning the Surface Vibration

We used a Polytec PSV-I-680 QTec Laser-Doppler Vibrometer with the proprietary Polytec data acquisition software to scan the surface velocities. Pictures of the measurement setups are in the appendix Fig. A.1, Fig. A.2 and Fig. A.3.

Two potentially impairing factors had to be considered. The first factor was the translucency of surfaces. The display consists of multiple, variably translucent layers, which could induce errors in the measurements since the laser beam could shine through some of the layers. For the transparent back cover, the laser beam shined through the entire surface. Hence, a layer of adhesive paper had to be applied to both surfaces.

The second factor was the mechanical coupling between the ground and the smartphone dummy.

Figure 3.1 *Frontal view of the* $1.5\ \mathrm{cm} \times 1.1\ \mathrm{cm} \times 0.2\ \mathrm{cm}$ *Manticore loudspeaker that was used as an alternative verification for the half-space simulations.*



(a) (b)

(c)

Figure 3.2 *Photographs of the* $15.55\ \mathrm{cm} \times 7.26\ \mathrm{cm} \times 1.1\ \mathrm{cm}$ *smartphone dummy. (a): Front view, showing the display; (b): right frame and cable; (c): backside view; the white markers highlight the vibrating actuators.*

The actuators induce vibration in all surfaces, as the dummy forms a closed box whose surfaces are coupled mechanically and volumetrically. The coupling can become problematic when a passively vibrating surface connects to another rigid surface. Sound Solutions provided a mounting that places the dummy onto two elastic rubber bands. This elastic suspension, with a resonance frequency of about $12\,\mathrm{Hz}$, assured that the coupling does not distort the measurements in the frequencies of interest. The mounting is depicted in Fig. A.1 (b).

Additional rubber bands were used to clamp the dummy to an upright position for scanning the side surfaces, as depicted in Fig. A.2. Pretests yield that the frame velocities were about $20\,\mathrm{dB}$ lower than those of the display and back cover. Hence, the effect of coupling was considered negligible when measuring the frame velocities.

The surfaces were sampled into meshes of equidistant, quadrilateral elements. The display and the back cover were measured at 31×17 positions. The long and short frames were measured at 31×3 and 17×3 positions. Considering the measures of the dummy, the spatial resolution was about $0.5\,\mathrm{cm} \times 0.43\,\mathrm{cm} \times 0.37\,\mathrm{cm}$. Screenshots from the built-in camera of the Polytec laser during the scans are in the appendix, Figure A.2. Note that the meshes include some invalid positions, which had to be treated in the subsequent data handling, cf. Chapter 4.

As a preparation for the data handling, it was also important to define axes for the dummy so that every scanned position could be assigned to the correct position in the model later on. An abstraction of the dummy's alignment for each surface scan is depicted in Fig. 3.3. All settings for the scans were set in the Polytec data-acquisition software. As excitation signals, periodic chirps that range from $50\,\mathrm{Hz}$ to $25\,\mathrm{kHz}$ were used. The amplitude of the chirp was set to $0.5\,\mathrm{V}$. The data as complex frequency data was saved in '.fbd'-files. For the Fourier transform, 12775 frequency bins and rectangular windows were set. No averaging was used. Screenshots of the settings can are in the appendix, Figs. B.1 and B.2.

For the alternative verification case with the Manticore voice-coil loudspeaker, 13×7 positions were used to scan the front plate, excluding the torus. The chirp was set to an amplitude of $0.1\,\mathrm{V}$, while all other settings remained as in Figs. B.1 and B.2. An appropriate mounting with an open back volume was used. Pictures of the measurement are shown in Fig. A.3. Fig. 3.4 shows the mean of absolute amplitudes for normalized frequency spectra of the vibrating velocities on the phone's surfaces and the Manticore.

A proprietary software tool of Sound Solutions called Dive was used for transferring the '.fbd' files to Matlab. The tool allows opening the .fbd-files, selecting scan points and channels, and applying a frequency-bin reduction to pre-selected frequency scales. There are options for export in various data formats, such as '.xls' or '.csv'. Dive is compiled from a Matlab source code, which allowed for adaptation for exporting the data in '.mat'-file format. In the Dive settings, the bin reduction was set to an 'R320', a logical continuation of the Renard-series-based ISO-R frequency scales that contains 320 bins per decade. The data of each surface scan was exported to four files containing the absolute value and phase of the vibrometer-velocity channel and the voltage reference channel, listed for the coordinates of each scan position.

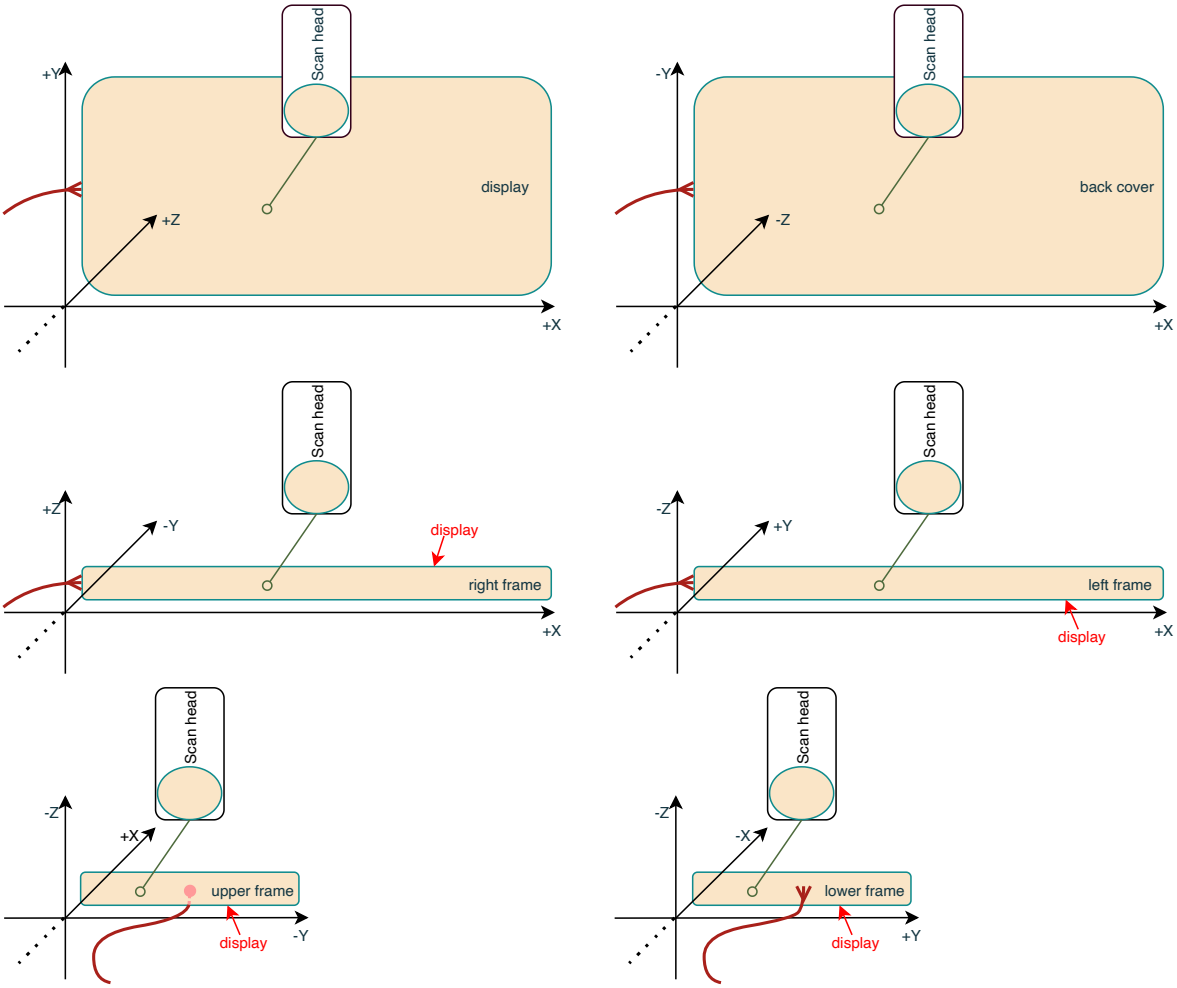


Figure 3.3 *Illustration of the phone's placement for each surface scan, showing a plan view; scanned surfaces are: display (upper-left), back cover (upper-right), right frame (mid-left), left frame (mid-right), upper frame (lower-left), and lower frame (lower-right image); The axes show the defined coordinate system for the phone's dimensions: height in X, width in Y, and depth in Z;*

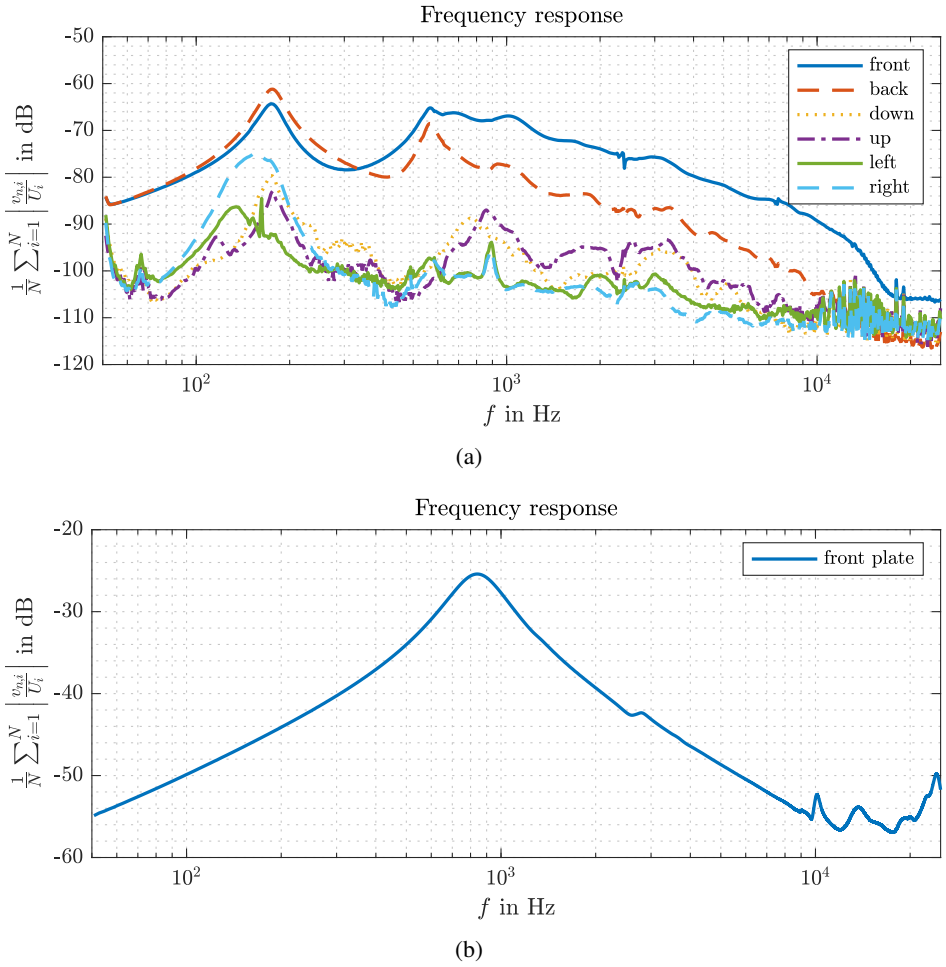


Figure 3.4 *Frequency response of the overall surface vibration as the average of absolute velocities, normalized by the input voltage reference in dB; (a) surfaces of the smartphone dummy; (b) front plate of the voice-coil Manticore loudspeaker used as an alternative verification case for the half-space simulations;*

3.3 Acoustic Measurements

To provide reference curves to assess the simulated sound pressures, we performed measurements of the acoustic frequency response in three different scenarios:

1. the smartphone dummy in half space,
2. the Manticore coice-coil loudspeaker in half space as an alternative verification case,
3. the smartphone dummy in full space.

The first two setups were measured in laboratory 1. Two different baffles were used, a mobile and a fixed baffle. The mobile baffle provides a pivoted arm as a microphone holding, allowing for measurements at off-center positions. By contrast, the fixed baffle has an attached guide for the microphone, keeping the recording position to the acoustic center of the tested device. Both baffles had measures of $71.8\ \mathrm{cm} \times 76.2\ \mathrm{cm}$. Exemplary photographs during the measurements are shown in Fig. 3.5.

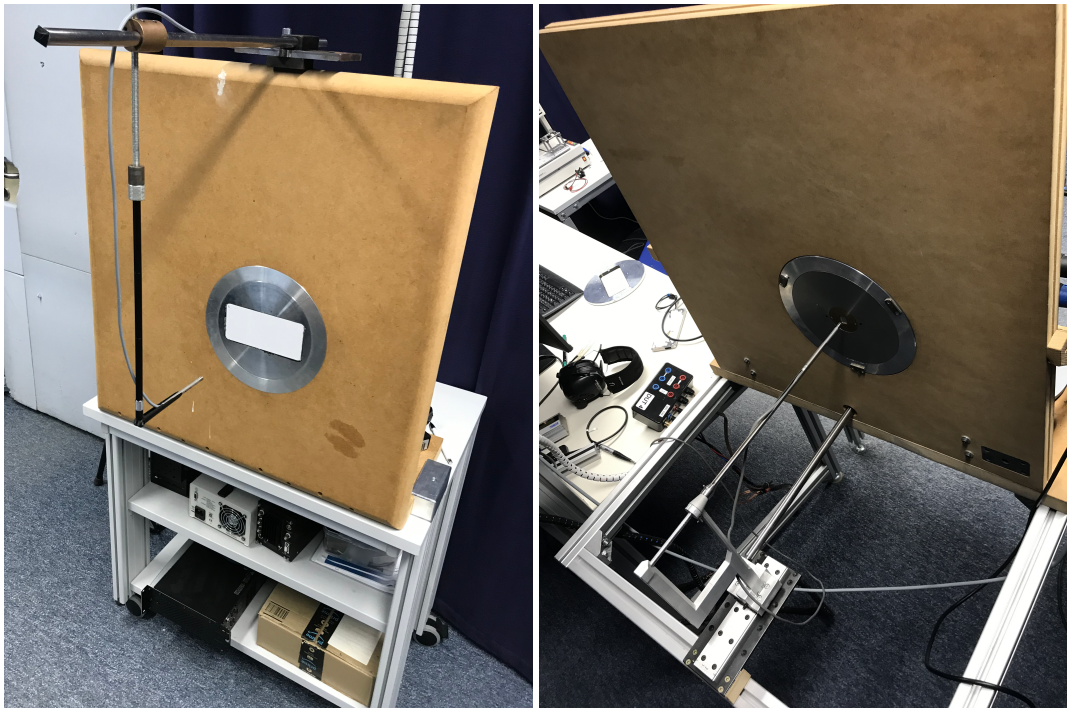
The scenario of the smartphone dummy in half space has mainly been observed using the mobile baffle. There were two factors that could potentially influence the result and therefore had to be considered for choosing the microphone positions. First, the distance between the center of the dummy and the closest edge of the baffle is $30.3\ \mathrm{cm}$. Therefore, it is likely that diffracted waves could impair the recording. Secondly, as described in the previous section, the vibrating actuators induce vibration not only to the display but into all surfaces of the dummy. Since the dummy is clamped into the baffle via an adapter, there is a chance that radiating bending waves are being excited in the baffle. Due to these factors, we chose the microphone position to be in the acoustic center of the device, at $10\ \mathrm{cm}$ distance. This position allowed for a cross check with the fixed baffle.

The alternative verification case of the Manticore voice-coil loudspeaker has been measured in the fixed baffle, in distances of $1\ \mathrm{cm}$, $2\ \mathrm{cm}$,$3.16\ \mathrm{cm}$, $5\ \mathrm{cm}$, and $10\ \mathrm{cm}$. The fixed baffle is designed to be used for measuring miniature loudspeakers, such as the Manticore, and has been optimized by Sobtzick [56]. This scenario could hence be considered being less prone to errors.

The smartphone dummy has been suspended for the third scenario, hanging down vertically from a frame. The suspension consisted of two duct tapes that were attached to the frame. This positioning aimed to reduce the mechanical coupling between mounting and phone. To gain an overview of the all-around radiation of the dummy, we used a turntable. The turntable was set to rotate between $270\ ^\circ$ and $90\ ^\circ$ and the phone was re-positioned for measuring the back side. The measurement chamber here was laboratory 3, which has a room height of $2.8\ \mathrm{m}$. The farthest-possible distance between microphone and a reflecting surface was $1.3\ \mathrm{m}$. To prevent possible impairments due to room reflections, a microphone distance of $10\ \mathrm{cm}$ was chosen. We repeated the procedure three times to check for reproducibility. Photographs during the measurement are shown in Fig. 3.6.

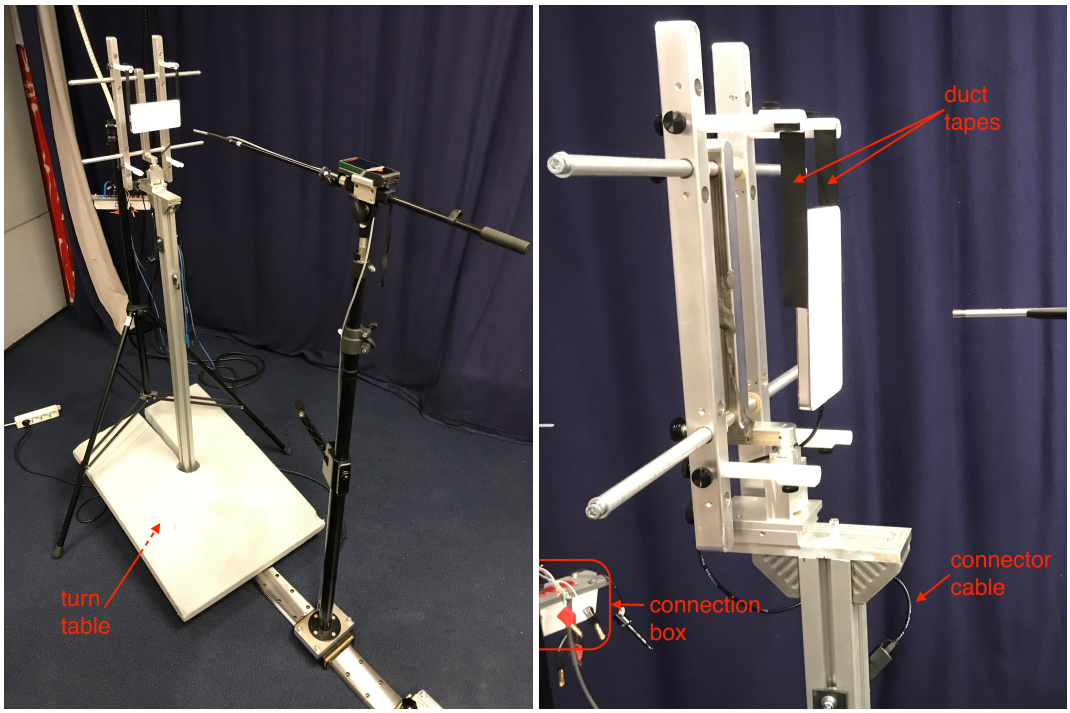
The installed APX-500 measurement systems were used for the data acquisition, combined with Brüel and Kjaer Type 2670 microphone pre-amplifiers and corresponding 1/4-inch microphone capsules. As excitation signals served the built-in logarithmic sweep of the APX-500 system. The sweep was set to range from $20\ \mathrm{Hz}$ to $20\ \mathrm{kHz}$, in a duration of $2\ \mathrm{s}$. Averaging was set to 3 sweeps per measurement. The voltage setting of the sweep was set to $250\ \mathrm{mV_{RMS}}$, $200\ \mathrm{mV_{RMS}}$, and $400\ \mathrm{mV_{RMS}}$, for the dummy in the baffle, the manticore in the baffle, and the dummy in a free field scenario. The exported impulse responses were saved in a '.mat' format.

The impulse responses were windowed using a two-sided Tukey tapered cosine window with a cosine fraction of 0.5. The on-set time was set to $-2\ \mathrm{ms}$ and the off-set time to $80\ \mathrm{ms}$. As the window length of time T_W corresponds to a minimal applicable frequency difference $\Delta f = \frac{1}{T_W}$, it was chosen to be large enough for resolving the audible frequency spectrum: $\Delta f = \frac{1}{80\ \mathrm{ms}} = 12.5 Hz$. Filtering out all



(a) (b)

Figure 3.5 *Exemplary photographs during half-space measurements; (a): smartphone dummy, mounted in the mobile baffle; (b): alternative verification case with the Manticore voice-coil loudspeaker in the installed baffle;*



(a) (b)

Figure 3.6 *Exemplary photographs during the full-space measurements; (a): full setup; (b): closer photograph showing the smartphone dummy hanging upright in the mounting;*

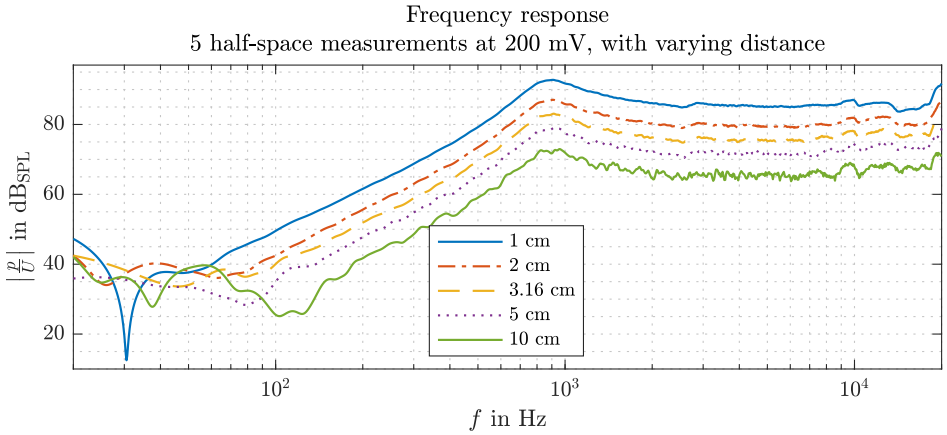


Figure 3.7 *Frequency response of the alternative verification case with the Manticore voice-coil loudspeaker, as sound pressure normalized by the input voltage in* $\mathrm{dB_{SPL}}$*; the microphone position is in the acoustic center of the loudspeaker.*

reflected sound waves would have required a window shorter than $T_W = \frac{2.6\ \mathrm{m} - 0.1\ \mathrm{m}}{343\ \frac{\mathrm{m}}{\mathrm{s}}} = 7.3\ \mathrm{ms}$ [57], whereas the resulting frequency resolution $\Delta f = 137\ \mathrm{Hz}$ does not resolve the actuators' resonance frequency $f_{res} = 180\ \mathrm{Hz}$ correctly. As the measurements did not show any comb-filter shapes as would be typical for occurring reflections, it seemed reasonable to choose the larger window. The on-set time was set to include the on-set transient. The cosine fraction was chosen arbitrarily.

The windowed impulse responses were Fourier-transformed into the frequency domain, using 51552 FFT-bins. The same processing was applied to the voltage reference. The reference was then used for normalizing the sound pressure via complex division in the frequency domain. The normalized frequency spectra were smoothed using a Savitzky-Golay filter [58] with a window size of 50 bins. Finally, the frequency bins were reduced using spline interpolation to match the frequency scale of the exported velocities. As a result, frequency responses of the Manticore are depicted in Fig. 3.7. Fig. 3.8 shows the minimum, maximum, and mean curves of repeated measurements for exemplary measurement setups of the phone.

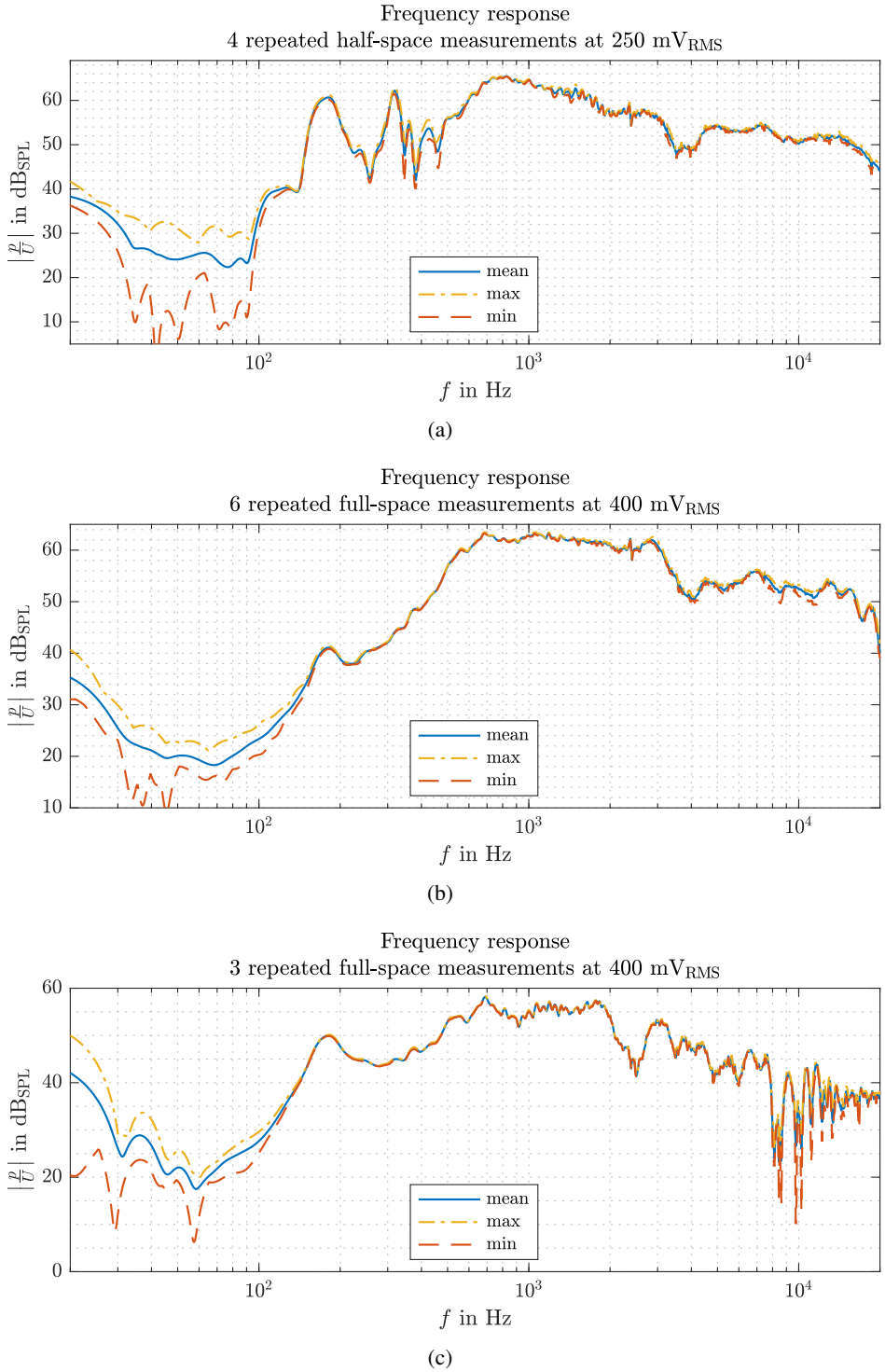


Figure 3.8 *Frequency response of the smartphone dummy, as sound pressure normalized by the input voltage in* $\mathrm{dB_{SPL}}$*; shown are the mean, minimum and maximum curves of repeated measurements. The microphone position is in the acoustic center of the device, at* 10 cm *distance. (a): half-space measurement; (b): full-space measurement in front of the device; (c): full-space measurement at the device's back side.*

4 Implementation of the Software

This chapter discusses how the simulation models are implemented in Matlab and roughly explains their usage. In the beginning, the pre-processing of the acquired velocities and the structure of the extracted data are described. Next, auxiliary functions used in the models and for the data handling are outlined, followed by ESM, Rayleigh integrals, and BEM implementation. The final section is about implementing custom plotting functions that aim to ease the analysis of the device's radiation behavior.

4.1 Preprocessing the Velocity Data

In a first step, the .mat files containing velocities and voltages of each surface scan must be loaded into Matlab and prepared so that every scanned position assigns to its corresponding positions on the model. Three functions do the main work of preparing the data:

- *prepare_polytec_data()*
- *create_box()*
- *extract_and_interpolate()*

prepare_polytec_data() is the high-level function that takes the scan names, the measures of the device, and the number of scan positions as arguments and calls the other two functions with hard-coded arguments so that the data is assigned correctly if the smartphone is measured as suggested in Fig. 3.3. The function contains options for plotting the scan positions and saving the prepared data into a struct. Listing 4.1 shows the function call of *prepare_polytec_data()* with the plot and save flags set to zero. Note that it is possible to specify the surface scans of only the first, the first and second, or all surfaces. The data is created in the same way for all three options, but the velocities of not specified surfaces are set to zero. The complete function is listed in the appendix, in Listing C.1.

create_box() creates a number of equidistant positions for the six surfaces of a box and returns their positions and corresponding outwards-normal vectors. The box is centered with respect to the (x, y) coordinate, and the surface corresponding to the display is at $z = 0$. All other surfaces are at $z \leq 0$. A scatter plot of such a box created for the scans used in this thesis is depicted in Fig. 4.1.

extract_and_interpolate() does the actual work of assigning the data and is called once for each surface. Here, the .mat files of velocity and voltage are loaded (cf. Section 3.2) and assembled into a matrix of complex-valued, normalized velocities. The scan positions from the Polytec measurements are extracted and mapped using linear equations to fit the box's coordinates. The resulting grid still deviates in some fraction of a millimeter from the exactly equidistant grid on the box's surface. Hence, the velocities are then interpolated to the box positions, which is necessary for the symmetry assumptions we have to take in the following sections. Finally, all outlying or left-out scan points that result from the equidistant measurement of a device with rounded corners are treated (cf. Fig. A.2). The velocities of the outliers are set to the mean of their two nearest neighbors, as shown in Fig. 4.2. The left-out scan points on the frame are extrapolated using Matlab's *interp2()* function after inserting auxiliary positions with zero velocity at a farther distance, as illustrated in Fig. 4.3. The complete function is listed in the appendix, in Listing C.2.

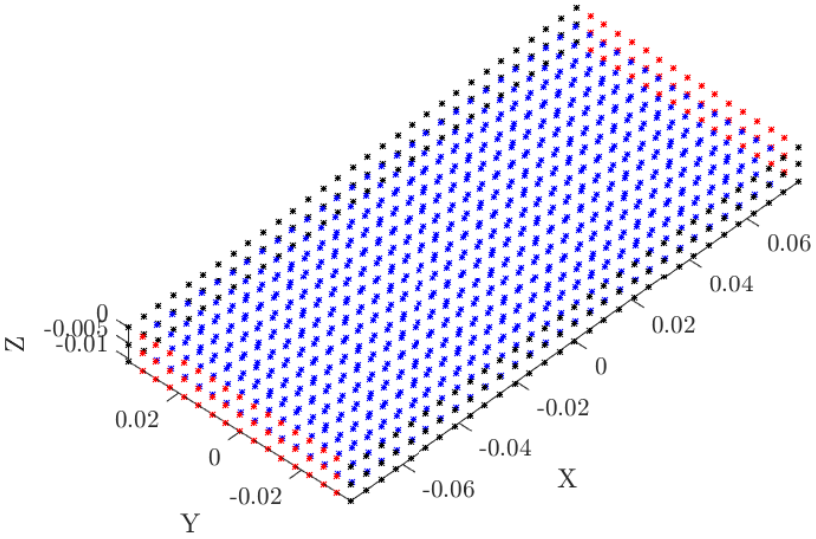


Figure 4.1 *Scatter plot created with create_box(); the box has the measures of the smartphone dummy described in Section 3.1. Blue dots represent the front/back cover in* (31×17) *positions; red dots the upper/lower frame in* (17×3) *positions; black dots the left/right frame in* (31×3) *positions. The axes are defined as depicted in Fig. 3.3 and specified in* m.

Listing 4.1 *Function call of prepare_polytec_data()*

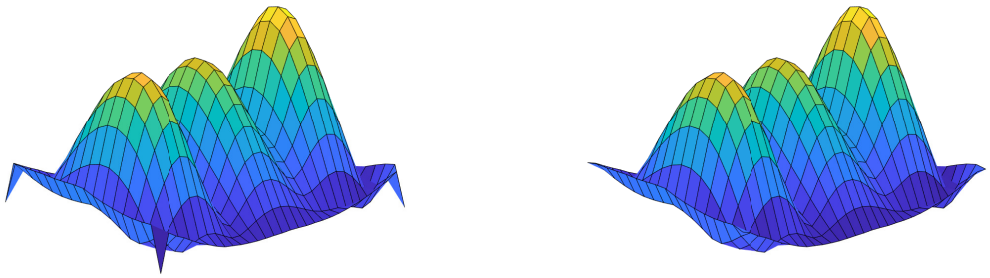
```
% measures of the phone
height = 0.156;
width = 0.0725;
depth = 0.011;
measures = [height,width,depth];
% how many positions have been measured in each axis
n_sources = [31,17,3];
% load scans
scannames = ["scan_display";"scan_back";"scan_frame_low";
"scan_frame_up";"scan_frame_left";"scan_frame_right"];
[Q,n_vecs,grids,V,f] = prepare_polytec_data(scannames,measures,...
n_sources,0,0,0);
```

Data structure

Finally, variables that are commonly used and uniformly named throughout the code examples should be explained briefly. We start with the variables returned by *prepare_polytec_data()*:

- *Q* is a six-element cell array. Each cell holds a $(3 \times N_{Q_i})$ matrix holding the three-dimensional position coordinates, whereas N_{Q_i} is the number of scan positions for the i^{th} surface.
- *n_vecs* is a six-element cell array, also holding matrices of size $(3 \times N_{Q_i})$. The matrices contain the three-dimensional coordinates of the outwards-normal vectors of each position.
- *grids* is a (2×6)-element cell array, containing meshgrids for each surface.
- *V* is a six-element cell array. Each cell holds a $(N_f \times N_{Q_i})$ matrix with the complex-valued velocities data for each position *Q* and frequency in *f*.
- *f* is a $(N_f \times 1)$ matrix holding the frequency scale of the extracted data. The frequency scale must equal for all exported surface scans.

Each cell consistently corresponds to one surface:



(a) (b)

Figure 4.2 *Surface plot of the absolute surface velocities on the front display of the smartphone dummy at $f = 1$ kHz. (a): outliers in the corners not treated; (b): outliers set to the mean of their two nearest neighbors.*

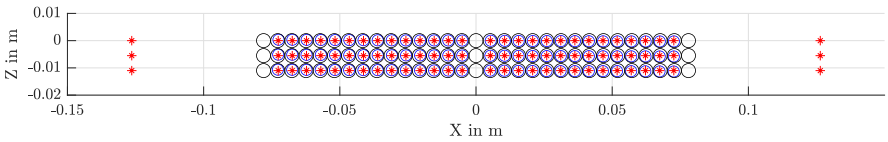


Figure 4.3 *Scatter plot of the mapped positions corresponding to the right frame of the smartphone dummy. Black circles indicate the assigned positions on the box surface; blue circles are the mapped positions of the velocity scan; red stars indicate the auxiliary positions that allow for extrapolating the surface vibration to the outer edges of the box. The axes are defined as in Fig. 3.3.*

- cell 1: front cover (display),
- cell 2: back cover,
- cell 3: lower frame,
- cell 4: upper frame,
- cell 5: left frame,
- cell 6: right frame;

Other commonly used variables are:

- *P* as a $(3 \times N_P)$ matrix holding the three-dimensional coordinates of all receiver positions,
- *measures* as a three-element vector holding the measures of the device in the (x, y, z) coordinates,
- *n_sources* as a three-element vector specifying the number of scan positions in (x, y, z) coordinates,
- *c, ρ* as scalars holding the speed of sound and the air density, respectively, and
- *p* as a $(N_f \times N_P)$ matrix holding the resulting sound pressures for each frequency and receiver position.

4.1.1 Auxiliary functions

Four auxiliary functions evolved during this thesis. These functions are either incorporated into the simulation models or utilized for the analysis. Hence, they should be briefly discussed in this subsection.

First is the above described *create_box()*, which can also be called independently and allows for quick and flexible creation of a box with an equidistant rectangular grid. The function supports specifying indents for the grids so that they do not reach the edges of the box. Further, a retraction distance can be specified, which can be used for creating a smaller box. The retraction distance was used in the ESM to observe equivalent sources' positioning. An additional flag 'centerandcalcdA' causes *create_box()* to specify the mid-point coordinates of each quadrangle of the grids and to return the area of each quadrangle as well. A flag 'plotbox' creates a scatter plot of the box. A grid representing the scan positions on the smartphone dummy's display and the corresponding mid-point positions computed using *create_box()* with the 'centerandcalcdA' flag is shown in Fig. 4.4. The usefulness of *create_box()* emphasizes when it is used in combination with another auxiliary function *interpolate_box()*, which interpolates the velocities assigned to the surface grids of one box to the surface grids of another box. *interpolate_box()* exploits Matlab's *interp2()* function and the Modified Akima interpolation method (Makima) [42, Chap. 8]. Points on the new grid that lies outside the old grid are extrapolated by adding points with zero velocity outside the old grid and interpolating between the old grid and the added points (cf. Fig. 4.3). The distance is specified with the optional 'extrapdist' parameter. The two functions serve to adjust the grid density without remeasuring. Additionally, they are used to interpolate the measured scan points to the mid-point positions and compute the area of the quadrangle surface elements when computing the Rayleigh integrals. Exemplary calls for both functions are shown in Listing 4.2. The complete functions are listed in the appendix, Listings C.3 and C.4.

Another auxiliary function, *evaluate_G_dGdn_d2Gdn2()*, computes the free-space Green's function and its first-order and second-order derivatives for an arbitrary number of source-receiver combinations. The first-order derivative is computed concerning the source coordinate and the second-

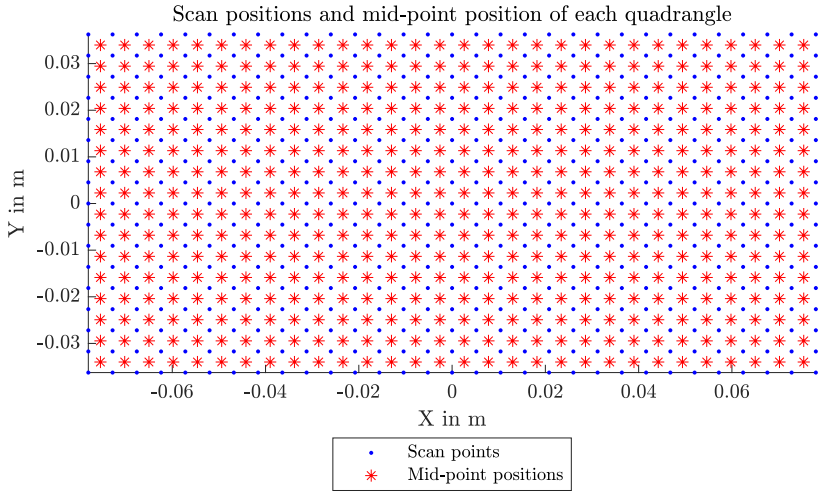


Figure 4.4 *Scatter plot of the* (31×17) *scan positions of at the display (blue) and the* (30×16) *mid-point positions computed with create_box() and the 'centerandcalcdA' flag.*

order derivative concerning the receiver coordinate. *evaluate_G_dGdn_d2Gdn2()* is based on the function *evaluate_G_dGdn()* written by Franz Zotter and Hannes Pomberger. It is adapted to support evaluating the first-order derivative with respect to the source coordinate and the second-order derivative with respect to the receiver coordinate. All models, except for the BEM, rely on *evaluate_G_dGdn_d2Gdn2()* for computing the Green's function and its derivatives. An exemplary function call is shown in Listing 4.3. The function is listed in the appendix, Listing C.5.

As the analysis tools described in Section 4.3 require evaluating the sound field at specific coordinates, the fourth auxiliary function *defineReceivers()* was written to simplify the assignment of receiver positions. The function supports modes for a spherical and a hemispherical receiver alignment, as used for the full-space and half-space balloon plots, as well as cross-sectional alignments as required for the field plots. The receiver alignment is directly plotted in a scatter plot with the additional' show' flag. An exemplary function call is shown in Listing 4.4. The corresponding scatter plot is depicted in Fig. 4.5. The complete function is in the appendix, Listing C.6.

Listing 4.2 *Function calls of create_box() and interpolate_box() that creates a new box and interpolates the old velocities to the new grid.*

```
% no indents to the edges
indents = [0,0,0];
% no retraction from the surfaces
indents = [0,0,0];
% compute mid-point positions and the area of the quadrangle elements
centerandcalcdA = 1;
% scatter plot of the assigned positions
plotbox = 1;
[Q_new,n_vecs_new,grids_new,dA] = create_box(measures,n_sources,indents,retract,...
centerandcalcdA,plotbox)
% interpolate
[V_new] = interpolate_box(grids,grids_new,V)
```

Listing 4.3 *Excemplary function call for evaluate_G_dGdn_d2Gdn2().*

```
% compute Green's functions and derivatives from the front surface to some pre-defined receiver coordinates P.
[G,dGdn,d2Gdn2] = evaluate_G_dGdn_d2Gdn2(Q{1}.',P.',w/c,n_vecs{1}.',n_vecs_P.');
```

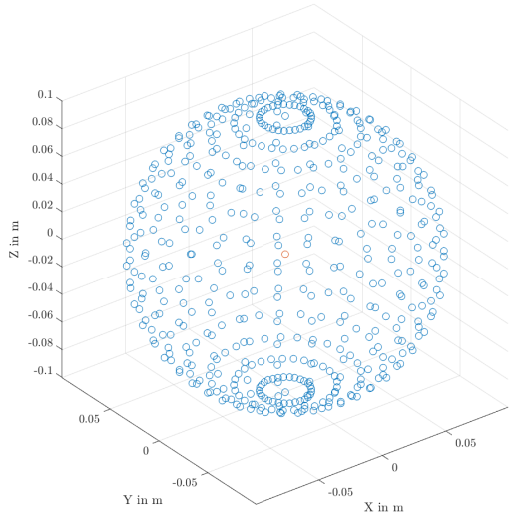


Figure 4.5 *Scatter plot of the spherical receiver arrangement created in Listing 4.4; the sphere has a radius of $r = 0.1$ m and consists of 20 horizontal rings with 30 receiver positions at each ring; blue circles represent the receiver positions; the red circle indicates the origin.*

Listing 4.4 *Function call of defineReceivers() for creating a spherical receiver arrangement.*

```
% number of receivers in a single horizontal ring
n_rec_hor = 30;
% number of horizontal rings
n_rec_vert = 20;
% 'omni' for sphere
mode = 'omni';
% plot
show = 1;
% radius of the sphere
r = 0.1;
P = defineReceivers(r,n_rec_hor,n_rec_vert,mode,show);
```

4.2 Implementation of the Simulation Models

The simulations are packed into functions so that a single function call computes the frequency response over the specified frequency range and for multiple receiver positions. This section provides an insight into how the five approaches are implemented.

4.2.1 Half-space ESM

The half-space ESM is computed using the function *calcESM()*. The function call is shown in Listing 4.5. The complete function is listed in the appendix, Listing C.7. The algorithm involves the following steps:

1. Assign the positions of the equivalent-source layer. The (x, y) coordinates are set to equal the scan positions on the upper surface. The retraction distance to the upper surface is set to 0.5-times the average source spacing in (x, y), which is based on the findings in [15].

2. Compute the Green's function's first-order derivatives via *evaluate_G_dGdn_d2Gdn2()*, using the equivalent-source positions as source coordinate and the positions at the upper surface as receiver coordinate. Solve Eq. (2.35) using Matlab's backslash operator for obtaining the source strengths.
3. Compute the Green's functions via *evaluate_G_dGdn_d2Gdn2()*, using the equivalent-source positions as source coordinate and the field points in *P* as receiver coordinates. Solve Eq. (2.30) for obtaining the field sound pressures.

Listing 4.5 *Function call for the half-space ESM.*

```
p = calcESM(P,Q,n_vecs,V,f,c,rho,measures,n_sources);
```

4.2.2 Full-space ESM

The full-space ESM is computed using the function *calcESM_full()*. The function call is shown in Listing 4.5. The complete function is listed in the appendix, Listing C.8. The algorithm involves the following steps:

1. Assign the positions of the equivalent-source layer. The (x, y) coordinates are set to equal the scan positions on the upper surface. The retraction distance is set to 0.5-times the device's depth.
2. Separate the even-symmetric and odd-symmetric parts of the surface velocities via Eq. (2.46).
3. Compute the Green's function's first-order and second-order derivatives via *evaluate_G_dGdn_d2Gdn2()*, using the equivalent-source positions as source coordinate and the positions at the upper surface as receiver coordinate. Solve Eqs. (2.47) and (2.48) using Matlab's backslash operator for obtaining the source strengths.
4. Compute the Green's functions and first-order derivatives via *evaluate_G_dGdn_d2Gdn2()*, using the equivalent-source positions as source coordinate and the field points in *P* as receiver coordinates. Solve Eq. (2.50) for obtaining the field sound pressures.

Listing 4.6 *Function call for the full-space ESM.*

```
p = calcESM_full(P,Q,n_vecs,V,f,c,rho,measures);
```

4.2.3 Rayleigh I integral

The function call for computing the half-space radiation using the Rayleigh I integral is shown in Listing 4.7. The complete function is listed in the appendix, Listing C.9. The algorithm involves the following steps:

1. Create a grid containing the mid-points of each quadrangle using *create_box()* with the 'centerandcalcdA' flag.
2. Interpolate the velocities to the new grid using *interpolate_box()*.
3. Compute the Green's functions via *evaluate_G_dGdn_d2Gdn2()*, using the positions at the upper surface as source coordinate and the field points in *P* as receiver coordinates. Solve Eq. (2.60) for obtaining the field sound pressures.

Listing 4.7 *Function call for the Rayleigh I integral.*

```
p = calcRayleigh(P,V,f,c,rho,grids,measures,n_sources);
```

4.2.4 Combined-Rayleigh formulation

Investigating the combined-Rayleigh formulation led to a function *calcReileigh_full()* that includes three modes of extrapolation and an option for regularizing k_z either using Pagavino's method or the lower-barrier method (cf. Section 2.4.3). An exemplary call of *calcReileigh_full()* is shown in Listing 4.8. The complete function is listed in the appendix, Listing C.10. The algorithm for computing the combined-Rayleigh formulation involves the following steps:

1. Create a grid containing the mid-points of each quadrangle using *create_box()* with the 'centerandcalcdA' flag.
2. Interpolate the velocities to the new grid using *interpolate_box()*.
3. Separate the even-symmetric and odd-symmetric parts of the surface velocities via Eq. (2.46).
4. Extend the odd-symmetric velocities to at an appropriate length, either using zero padding or wave-field extrapolation (cf. Section 2.4.4).
5. Transform the extended odd-symmetric velocities into the (k_x, k_y) space via 2D FFT.
6. Compute the wave-number bins $(k_{x_1} \dots k_{x_N}, k_{y_1} \dots k_{y_M})$:

$$k_{x,y} = \frac{2\pi(n,m)}{(\hat{N},\hat{M})\Delta x},$$
$$(n,m) \in \left\{0, 1, \dots, \frac{(\hat{N},\hat{M})}{2}, -\frac{(\hat{N},\hat{M})}{2}+1, -\frac{(\hat{N},\hat{M})}{2}+2, \cdots -1\right\}, \tag{4.1}$$

   where $(\hat{N},\hat{M})$ are the FFT lengths in (x,y).
7. Compute k_z, as in Eq. (2.75).
8. Apply a regularization:
   - If the lower-barrier method is used: limit k_z as in Eq. (2.76) and solve Eq. (2.73) using the inverse 2D FFT.
   - If Pagavino's integral is used:
   - Compute $\boldsymbol{\Upsilon}_{tot}$:
     a) compute Eq. (2.82) - Eq. (2.87),
     b) truncate and flip $\boldsymbol{\Upsilon}$ and unite to $\boldsymbol{\Upsilon}_{tot}$, as in Eq. (2.88).

     Solve Eq. (2.89) using the inverse 2D FFT.
9. Crop the result to keep only the original bins $\rightarrow$ obtain the odd-symmetric pressures at the surface.
10. Compute the combined-Rayleigh Integral, as in Eq. (2.63):
    a) Compute the Green's functions via *evaluate_G_dGdn_d2Gdn2()*, using the positions at the upper surface as source coordinates and the field points in *P* as receiver coordinates. Solve Eq. (2.60) for obtaining the field sound pressures of the even-symmetric part.
    b) Compute the first-order derivatives of the Green's functions via *evaluate_G_dGdn_d2Gdn2()*, using the positions at the upper surface as source coordinate and the field points in *P* as receiver coordinates. Solve Eq. (2.61) for obtaining the field sound pressures of the odd-symmetric part.
    c) Superimpose the two Rayleigh integrals.

Several degrees of freedom had to be considered for the implementation: the FFT length in both dimensions, the type of extrapolation, the window, which regularization to use, and the parameters of the regularization, i.e., the lower barrier's value or the extent of the interpolation function. A well-working model could be found using Pagavino's method and frequency-dependent zero padding that adjusts the number of padded bins for a higher and a lower frequency region. For frequencies below 400 Hz, the grid is padded to the next-higher power of two than five times the device's measures, i.e., (256×128) bins in (x, y). Above 400 Hz, the next-higher power of two than three times the device's measures, i.e., (128×64) bins in (x, y), yielded better results. The extent $l_{x,y}$ of the interpolation function is set to adapt to the FFT length, keeping a constant ratio of two to the device's measures:

$$2\Delta(x, y)\ (N, M) = \frac{2\pi}{\Delta k_{x,y}\ 2l_{x,y}}, \tag{4.2}$$

$$l_{x,y} = \frac{\pi}{2\ \Delta k_{x,y}\ \Delta(x, y)\ (N, M)}, \tag{4.3}$$

where $\Delta(x, y)\ (N, M)$ are the device's measures in (x, y). A rectangular window is applied in this model since other window functions seemed to worsen the results.

Listing 4.8 *Function call for the combined Rayleigh formulation.*

```
p = calcRayleigh_full(P,V,f,c,rho,grids,measures,n_sources);
```

4.2.5 BEM

The open-source Matlab toolbox OpenBem [31] has been used for the core computation of the BEM. This subsection discusses the application of OpenBem for the simulations in this thesis. The procedure of computing the BEM involves the following steps:

1. Create a mesh.
2. Compute the BEM matrices, i.e., the matrices used for solving the sound pressures on the surface.
3. Assign the boundary values to the mesh.
4. Add CHIEF points to the BEM matrices and compute the surface pressures.
5. Compute the BEM field matrices, i.e., the matrices used to solve the field's sound pressures.
6. Compute the sound pressures in the field.

Operations 2-5 must be done for each frequency separately, which makes the standard BEM inefficient when a high frequency range is desired.

OpenBem does not support three-dimensional mesh creation, so the mesh must be created using additional software. In this thesis, the open-source platform Salome [32] was used to create a model with the exact measures of the smartphone dummy. The model was meshed using the 'Netgen 2D' algorithm in Salome. A mesh with 1263 linear elements and 548 nodes seemed to be a reasonable trade-off between performance and accuracy. A screenshot of the mesh is depicted in Fig. 4.6. The selected properties for the meshing algorithm and more detailed properties of the created mesh can be found in the appendix, Fig. B.3.

The created mesh must be transferred to Matlab. OpenBem supports the import of meshes in the Gmsh ('.msh') file format. However, Salome does not support the Gmsh format, and thus, the mesh needs to be converted. The conversion has been done via the open-source meshing software Gmsh

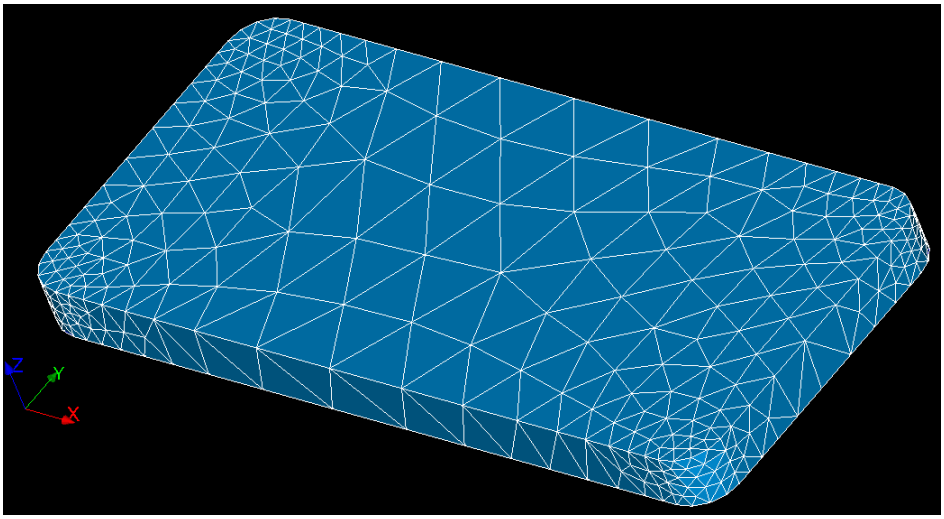


Figure 4.6 *Screenshot of the mesh used for the BEM simulations. The mesh has* 1263 *linear elements and* 547 *nodes.*

[59]. The .msh file is then read via *readgeomGMSH()*, checked for errors via *meshcheck()*, and its outwards-normal vectors are computed via *normals()*, as shown in Listing 4.9.

After the mesh is loaded, the BEM matrices can be computed using the function OpenBem's *TriQuadEquat()* function. As this must be done for every frequency separately, the computation time can grow quickly to an impracticable amount if a high frequency range should be computed. Computing the matrices for the mesh in Fig. 4.6 and for 862 frequencies took more than ten hours on a modern laptop. Thus, it is reasonable to save the matrices after computation into .mat files. Using the '-v7.3' flag allows storing files larger than 2 GB. Listing 4.10 shows how the matrices are computed and stored.

In the next step, the BEM matrices are used to compute the surface sound pressures, which are then used along with the velocities to compute the sound pressure in the field. These steps have been concluded to a single function *calc_BEM_radiation()*, allowing for computing the field sound pressures for all frequencies in a single code line. The function call of textitcalc_BEM_radiation() is shown in Listing 4.11. The complete function is listed in the appendix, Listing C.11. The algorithm involves the following steps:

1. Create the coordinates of 20 CHIEF points. Their positions are randomly distributed inside the device using Matlab's *rand()* function and with an indent of 2 mm to the borders.
2. Assign the surface velocities to the OpenBem model. This is done for each surface separately, using Matlab's *interp2()* function and the Makima interpolation method.
3. Compute the additional matrix entries for the CHIEF points via OpenBem's *point()* function and add them to the BEM matrices.
4. Solve for the sound pressure, as in Eq. (2.99) using Matlab's backslash operator.
5. Determine the BEM field matrices via *point()* and compute the sound pressures, as in Eq. (2.93).

Note that no seed for creating a pseudo-random distribution that sustains over multiple function calls was used. The 20 CHIEF points were sufficient for avoiding non-uniqueness over multiple function calls.

Listing 4.9 *Load the .msh mesh file into Matlab using OpenBem. The syntax is partially adopted from the OpenBem tutorial scripts.*

```
% specify mesh file
msh = 'path_to_mesh.msh';
% read mesh file
[nodesTMP,elementsTMP]=readgeomGMSH(msh);
% check for errors in the mesh
[nodesb,topologyb]=meshcheck(nodesTMP,elementsTMP,0,0);
% extract number of nodes
M=size(nodesb,1);
% extract number of faces
N=size(topologyb,1);
clear nodesTMP elementsTMP;
% compute outwards-normal vectors
[nvect]=normals(nodesb,topologyb,'n'); % 'n' for not plotting the mesh
```

Listing 4.10 *Compute and store the BEM matrices using OpenBem.*

```
% initialize cell arrays to store the BEM matrices for each frequency
A_cell = cell(size(f));
B_cell = cell(size(f));
% loop through every frequency
for ii = 1:length(f)
% compute current wave number
k = 2*pi*f(ii)/c;
% Calculate the BEM matrices for solving the pressures on the surface
[A,B]=TriQuadEquat(nodesb,topologyb,k,1);  % 1 for dealing with near-singular integrals
A_cell{ii} = A;
B_cell{ii} = B;
end
% store the results using .mat-file version 7.3, which allows for storing files larger than 2GB
save('path_to_store_A.mat','A_cell','-v7.3');
save('path_to_store_B.mat.mat','B_cell','-v7.3');
```

Listing 4.11 *Function call for computing the sound field using OpenBem.*

```
p = calc_BEM_radiation(A_cell,B_cell,nodesb,topologyb,grids,measures,V,P,f,c,rho);
```

4.3 Analysis Tools

Three plotting functions have been implemented to evaluate the simulation models and investigate the device's overall radiation behavior. The functions are briefly explained in this section.

plotFR() plots the frequency response of multiple curves and includes three subplots. The subplots show the sound pressure's linear magnitude in Pa, its logarithmic magnitude in $\mathrm{dB_{SPL}}$, and its angle in rad. The function allows for specifying custom limits for the frequency axis and the magnitude in $\mathrm{dB_{SPL}}$, whereas the limits in the linear-magnitude plot adjust automatically to the limits specified in $\mathrm{dB_{SPL}}$. Additionally, the function allows for specifying custom legend entries. An exemplary function call that plots the frequency response of two sound-pressure curves is shown in Listing 4.12. The corresponding frequency-response plot is depicted in Fig. 4.7. The complete function can be found in the appendix, Listing C.12.

The function *balloonPlot()* creates a balloon plot for observing the three-dimensional directivity patterns of the device, where the radius of the balloon plot corresponds to the logarithmic magnitude of the sound pressure radiated in that direction. *balloonPlot()* is designed to be used along with the spherical or hemispherical receiver positions created in *defineReceivers()*. It is possible to plot the balloon either in $\mathrm{dB_{SPL}}$, or in dB normalized to the maximum magnitude. For the normalized plot, an offset can be specified for the origin so that only the specified dynamic range is plotted and lower values are truncated. The coloration of the balloon plot can indicate either the magnitude, or the angle of the radiated sound pressure in that direction. For the angle coloration, the front-centered position is used as the reference angle of $0°$. Exemplary function calls that plot balloon plots in two configurations are shown in Listing 4.13. The corresponding plots are depicted in Fig. 4.8. The complete function can be found in the appendix, Listing C.13.

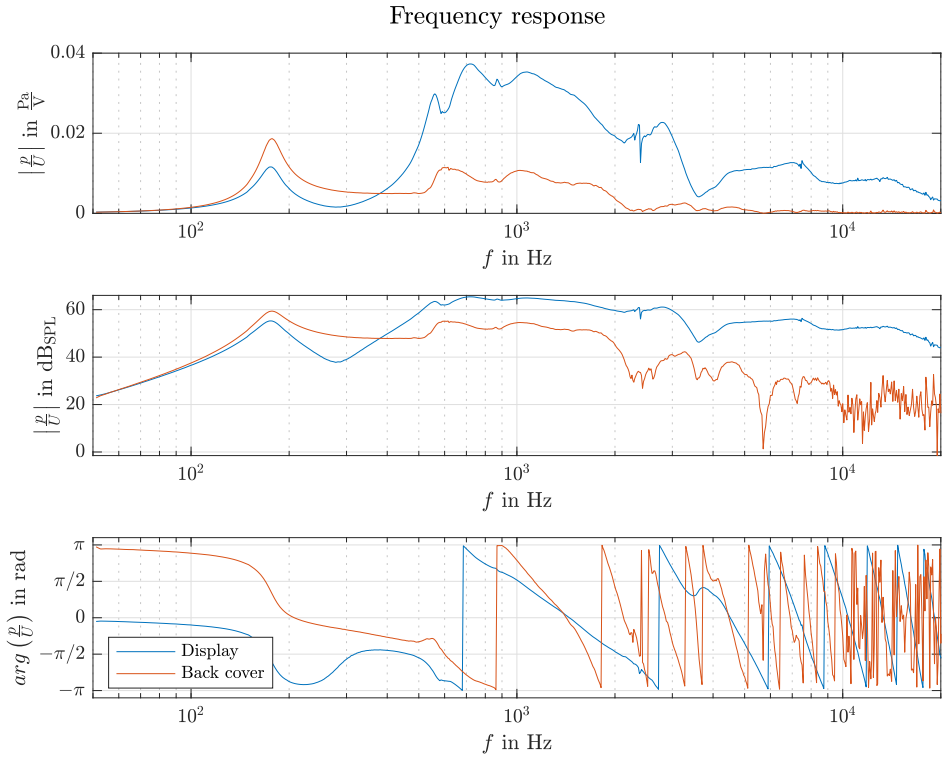
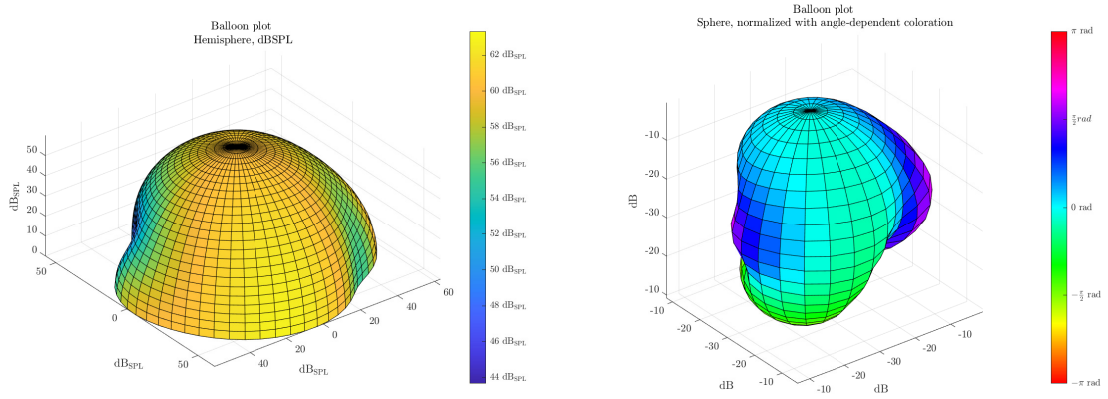


Figure 4.7 *Frequency response plot created with plotFR() that corresponds to the function call in Listing 4.12. The curves show the simulated half-space radiation of the display and the back cover, evaluated at a centered position in* 10 cm *distance.*

With *fieldPlot()*, cross-section plots of the sound field are possible in two modes. The mode 'p', which is the default, plots an image plot of the sound pressure in dB_{SPL}. The mode 'I' computes the sound intensity and plots its real part in dB_{I}, plus an overlay quiver plot that indicates the direction of the effective energy flow. *fieldPlot()* allows for specifying custom titles and the range for the coloration in dB. It is designed to work with the receivers created by *defineReceivers()* with the 'fieldx' or 'fieldy' mode. An exemplary function call that creates a sound-pressure field plot and an intensity field plot is shown in Listing 4.14. The corresponding plots are depicted in Fig. 4.9. The complete function can be found in the appendix, Listing C.14.

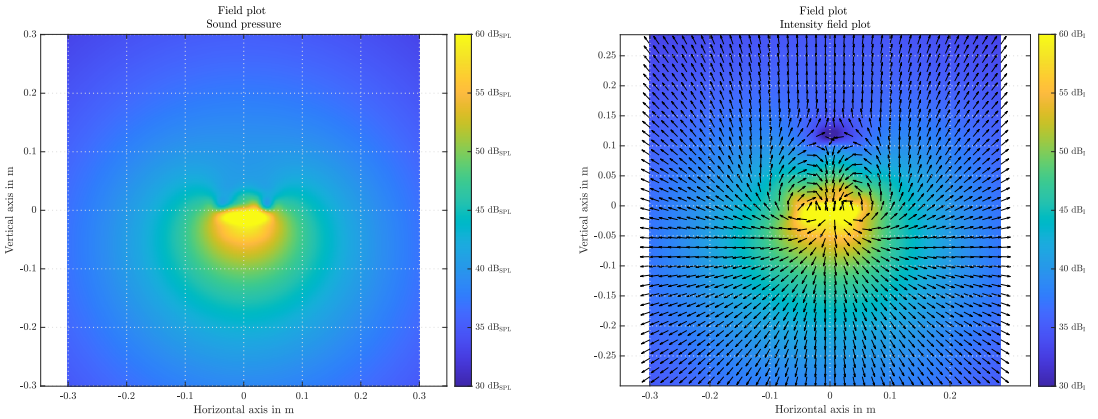
Listing 4.12 *Exemplary function call for plotting frequency response curves with plotFR().*

```
% define receiver
P = [0;0;0.1];
% compute radiation
p1 = calcRayleigh(P,V(1),f,c,rho,grids,measures,n_sources);
p2 = calcRayleigh(P,V(2),f,c,rho,grids,measures,n_sources);
% frequency limits
xlims = [100,20000];
% magnitude limits in dBSPL
ylims = [10,68];
% legends for two curves
legends = ['front cover'; 'back cover'];
% plot p1 and p2 over the frequency scale in f
plotFR(f,[p1,p2],legends,xlims,ylims);
```



(a) (b)

Figure 4.8 *Balloon plots created with balloonPlot() that correspond to the function call in Listing 4.13; (a): hemispherical balloon plot in* $\mathrm{dB_{SPL}}$*; (b): spherical balloon plot that is normalized to its maximum magnitude and has a phase-indicating coloration.*



(a) (b)

Figure 4.9 *Field plots created with fieldPlot() that correspond to the function call in Listing 4.14; (a): cross-section field plot of the sound pressure level in* $\mathrm{dB_{SPL}}$*; (b): cross-section field plot of the intensity in* $\mathrm{dB_I}$ *with overlaid quiver plot to indicate the intensity's direction at every receiver coordinate.*

Listing 4.13 *Exemplary function calls for creating balloon plots with balloonPlot().*

```
% choose a single frequency bin
ff = f(510); % =2000Hz
% extract velocities for this frequency
v = {V{1}(f==ff,:),V{2}(f==ff,:),...
    V{3}(f==ff,:),V{4}(f==ff,:),...
    V{5}(f==ff,:),V{6}(f==ff,:)};

% define receivers...
% evaluation distance
r_eval = 0.10;
% number of receivers for each elevation step
n_rec_hor = 30;
% number of horizontal rings
n_rec_vert = 25;
% create a hemisphere
P_hemi = defineReceivers(r_eval, n_rec_hor, n_rec_vert, 'hemi');
% create a and a full sphere
P_full = defineReceivers(r_eval, n_rec_hor, n_rec_vert, 'omni');

% compute the sound pressures for the full sphere, using combined Rayleigh
p_raylf = calcRayleigh_full(P_full,v,ff,c,rho,grids,measures,n_sources);
% and the ones on the hemisphere, using the simple Rayleigh
p_raylh = calcRayleigh(P_hemi,v,ff,c,rho,grids,measures,n_sources);

% example balloon plots
balloonPlot(p_raylh, P_hemi, n_rec_hor, 0,0,0,...
"Hemisphere, dBSPL");
balloonPlot(p_raylf, P_full, n_rec_hor,1,30,1,...
"Sphere, normalized with angle-dependent coloration");
```

Listing 4.14 *Exemplary function calls for plotting a sound-pressure field plot and an intensity field plot with fieldPlot().*

```
% choose a single frequency bin
ff = f(174); % =180Hz
% extract velocities for this frequency
v = {V{1}(f==ff,:),V{2}(f==ff,:),...
  V{3}(f==ff,:),V{4}(f==ff,:),...
  V{5}(f==ff,:),V{6}(f==ff,:)};

% define receivers...
% number of receivers in horizontal axis
n_rec_hor = 40;
% number of receivers in vertical axis
n_rec_vert = 40;
% extent of the field plot in +-m
r = 0.3;
% create receiver matrix in a cross section along the y axis
P = defineReceivers(r,n_rec_hor,n_rec_vert,'fieldy');

% compute sound pressures using some model
p = calcRayleigh_full(P,v,ff,c,rho,grids,measures,n_sources);

% choose custom coloration in dB
rng = [30,60];

% sound pressure field plot in dBSPL
fieldPlot(p,P,n_rec_vert,"Sound-pressure field plot",rng)
% intensity field plot in dBI
fieldPlot(p,P,n_rec_vert,"Intensity field plot",rng,'I',2*pi*ff,rho)
```

5 Discussion of the Simulation Results

This chapter is devoted to assessing the quality of the full-space ESM, the combined-Rayleigh formulation, and the BEM as the three full-space simulation models, as well as the half-space ESM and the Rayleigh I integral as the two half-space models. The chapter is divided into three sections. The first section concerns the full-space simulations, where simulated frequency responses are compared to acoustic reference measurements at four positions. The full-space models are further investigated by observing their radiated intensity fields and three-dimensional directivity patterns at selected frequencies. Section two assesses the half-space simulations by comparing the simulated and measured frequency responses of the smartphone dummy and, as an alternative verification case, the Manticore voice-coil loudspeaker. In section three, the computational cost of all five models is compared in terms of the elapsed computation time for a reference problem.

5.1 Full-Space Simulations

The first scenario to be assessed is the smartphone dummy in full space. The full-space sound field is more complex than the half-space one due to wave diffraction effects that occur at the edges of the device, which raises the question if the three models can reproduce these effects. Hence, this scenario is investigated in more depth than the half-space one.

5.1.1 Comparison of measured and simulated frequency responses

The frequency responses are assessed at four receiver positions: 1) centered in front of the device, 2) centered at its backside, 3) $60°$ rotated around the y axis, and 4) $60°$ rotated around the x axis. All positions are in $10\ \mathrm{cm}$ distance to the device's center. Their coordinates are shown in Table 5.1. As already mentioned in Chapter 3, the acquired sound pressures of the acoustic measurement and the velocity scan are normalized by the driving voltage of the corresponding measurements. Hence, the resulting frequency responses are comparable despite their different excitation signals. Further, the ρ and c constants were computed to match the environment of the acoustic measurement using a function of the OpenBem toolbox, *amb2prop()*. The values used in the simulations are $\rho = 1.18\ \frac{\mathrm{kg}}{\mathrm{m}^3}$ and $c = 346.7\ \frac{\mathrm{m}}{\mathrm{s}}$. The frequency responses are compared in terms of their logarithmic magnitude in $\mathrm{dB_{SPL}}$ and the phase mismatch

$$\Delta\phi = \phi_{meas} - \phi_{sim}, \tag{5.1}$$

where ϕ_{meas} and ϕ_{sim} are the unwrapped angles of the measured and simulated frequency responses. The overall frequency responses of all four positions can be found in the appendix Figs. D.1 to D.4. More detailed graphics are presented and discussed in the following paragraphs.

The magnitude responses at the front-center position 1 are shown in Fig. 5.1 (a), where all three models show good overall compliance to the acoustic reference. The logarithmic magnitude deviates less than $\pm 1.5\ \mathrm{dB}$ from $237\ \mathrm{Hz}$–$3.45\ \mathrm{kHz}$. The deviation is less than $\pm 2.5\ \mathrm{dB}$ from $173\ \mathrm{Hz}$–$12.1\ \mathrm{kHz}$ and grows with increasing frequency above $12.1\ \mathrm{kHz}$. The acoustic reference shows more ripples in the high-frequency region above $12.1\ \mathrm{kHz}$, where ESM and Rayleigh seem to keep track of the reference. The BEM curve plunges above $12.1\ \mathrm{kHz}$. Below $173\ \mathrm{Hz}$, the ESM stays within $\pm 2\ \mathrm{dB}$

Table 5.1 *Evaluated positions for comparing the frequency responses of full-space measurements and simulations.*

Nr.	**Position**	**Cartesian coordinates** (in m)		
		x	y	z
1	front, center	0	0	0.1
2	back, center	0	0	-0.111
3	60° rotated y axis	0.087	0	0.05
4	60° rotated x axis	0	-0.087	0.05

down to 61 Hz. In this low-frequency region, the BEM appears to underestimate the magnitude by about -3 dB. The combined-Rayleigh formulation shows the highest deviation in this region at 142 Hz, but converges towards the BEM at low frequencies < 80 Hz.

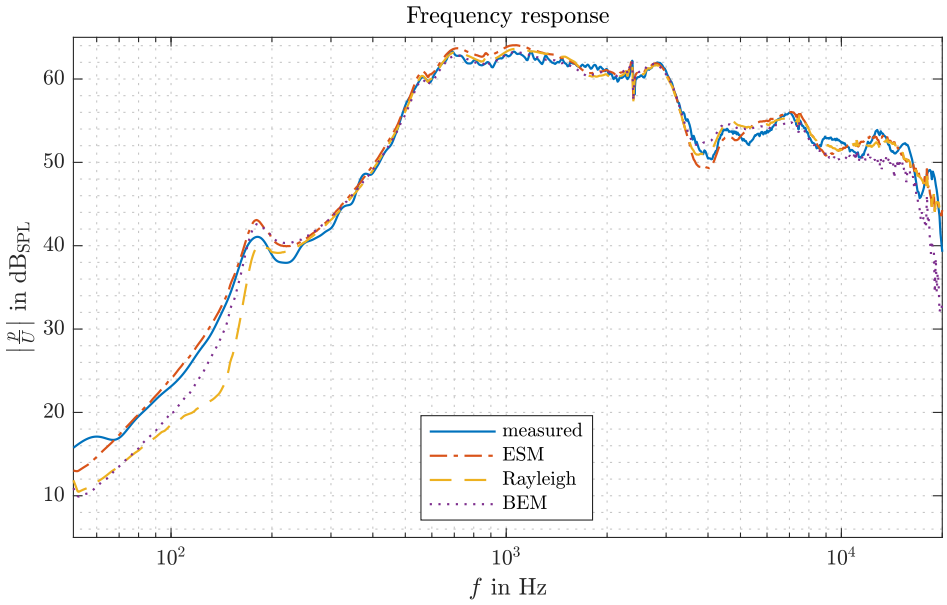
Observing the phase mismatch in Fig. 5.2 (a) yields that all three models deviate less than $\pm\frac{\pi}{8}$ rad from 200 Hz–3.55 kHz. The absolute phase mismatch constantly grows towards higher frequencies but stays within $\pm\frac{\pi}{4}$ rad until 11.2 kHz for all models. Below 200 Hz, the ESM stays within the $\pm\frac{\pi}{8}$ rad bound. BEM and Rayleigh show their negative peaks at 169 Hz, whereas the BEM converges towards the ESM at lower frequencies, and the combined-Rayleigh formulation exhibits a larger phase shift.

The magnitude responses of the back-side position 2 are depicted in Fig. 5.1 (b). The acoustic reference of position 2 shows the most ripples of all four measurements, especially at higher frequencies > 3 kHz. The three models keep track of the measurement up to 2 kHz, deviating by less than ± 2 dB. Their absolute deviation increases above 2 kHz, and the simulations appear to lose the measurement's track above 3.5 kHz. At low frequencies, the combined-Rayleigh formulation underestimates the measurement and exceeds the ± 2 dB bound below 167 Hz. ESM and BEM stay within the ± 2 dB bound down to 86 Hz.

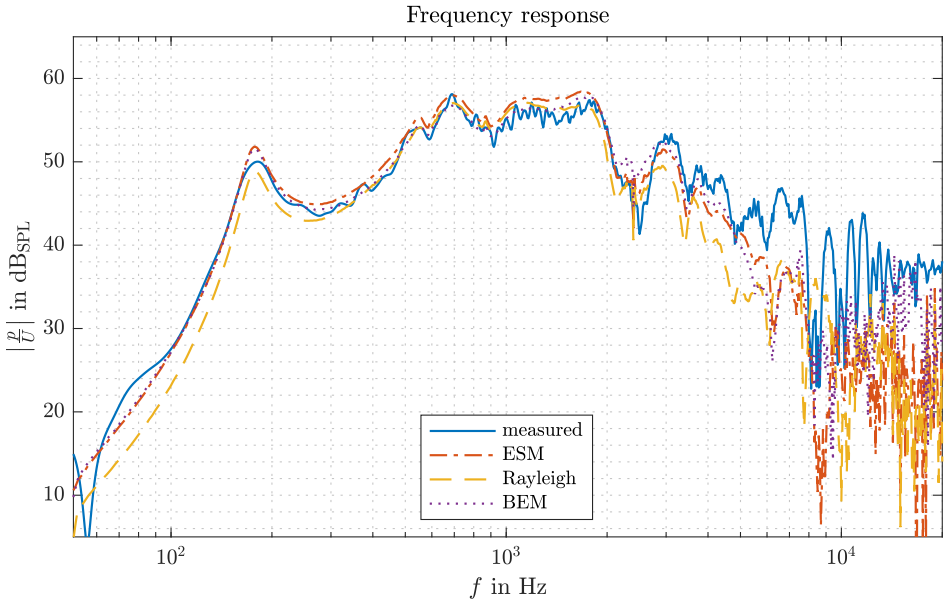
The phase mismatch of position 2 is shown in Fig. 5.2 (b). The three models stay within $\pm\frac{\pi}{8}$ rad up to 1.01 kHz and within $\pm\frac{\pi}{4}$ rad up to 5 kHz. Towards lower frequencies, the combined-Rayleigh formulation yields a higher phase mismatch than ESM and BEM and exceeds the $\pm\frac{\pi}{8}$ rad below 137 Hz.

Fig. 5.3 (a) shows the magnitude response of position 3. The three models stay within a ± 2 dB bound from 230 Hz–2.7 kHz. The models do not completely resolve the attenuation pattern in the measurement at frequencies after the peak/notch pattern, between 2.2 kHz and 3 kHz. The BEM differs here by $+5$ dB, Rayleigh and ESM differ by $+3.5$ dB. All three models follow the comb-filter shape that is in the measurements above 3 kHz. Again, the BEM appears to follow the measured curve less accurately than the other two. The BEM again plunges towards higher frequencies, starting at 14 kHz. Interestingly, the negative peaks of the notches are slightly shifted towards lower frequencies in the simulations. For example, the notch at 6.8 kHz in the Rayleigh and ESM curves compared to 7.05 kHz in the acoustic measurement. At lower frequencies < 230 Hz, the combined-Rayleigh formulation stays within the ± 2 dB bound down to 166 Hz. ESM and BEM differ by $+3$ dB at the 180 Hz peak. Below that peak, the BEM follows the measurement down to 100 Hz.

The corresponding phase mismatch of position 3 can be observed in Fig. 5.4 (a). All three models stay within $\pm\frac{\pi}{8}$ rad from 186 Hz–2.46 kHz, and within $\pm\frac{\pi}{4}$ rad from 86 Hz–8.13 kHz. An exception is one notch around 7 kHz, where the frequency shift of the negative peak causes a large phase mismatch. All three models drift off towards high frequencies > 8.13 kHz. At low frequencies, < 315 Hz, the



(a)



(b)

Figure 5.1 *Measured and simulated full-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage in* $\mathrm{dB_{SPL}}$*; evaluated at two positions in* 10 cm *distance; (a): centered in front of the device (position 1); (b): centered at the back side (position 2);*

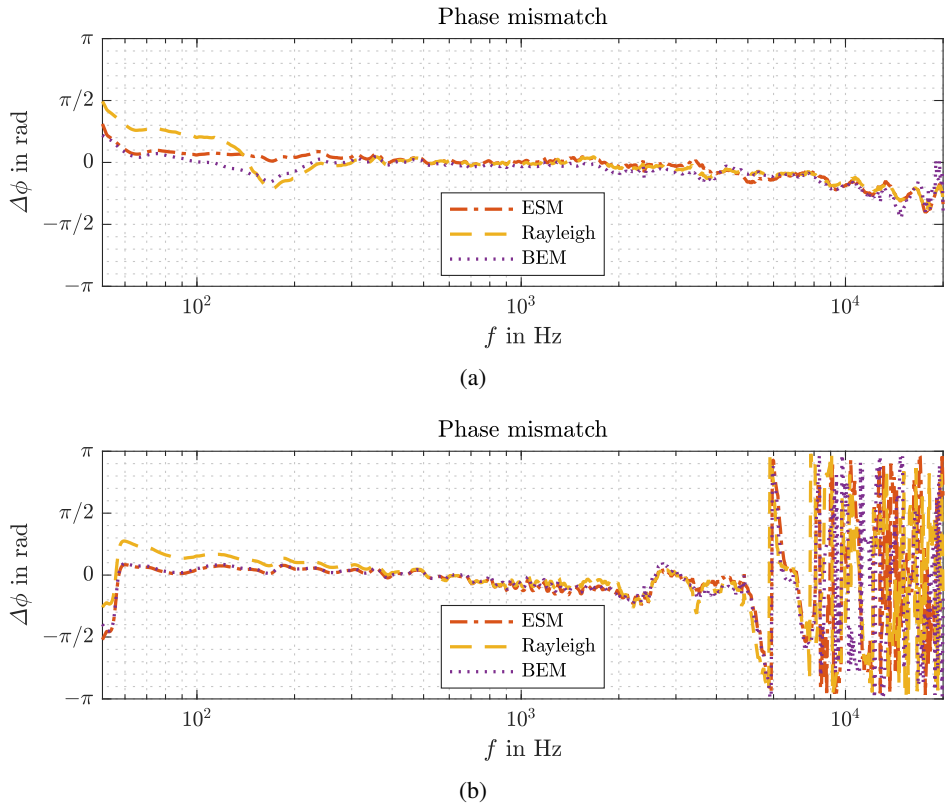


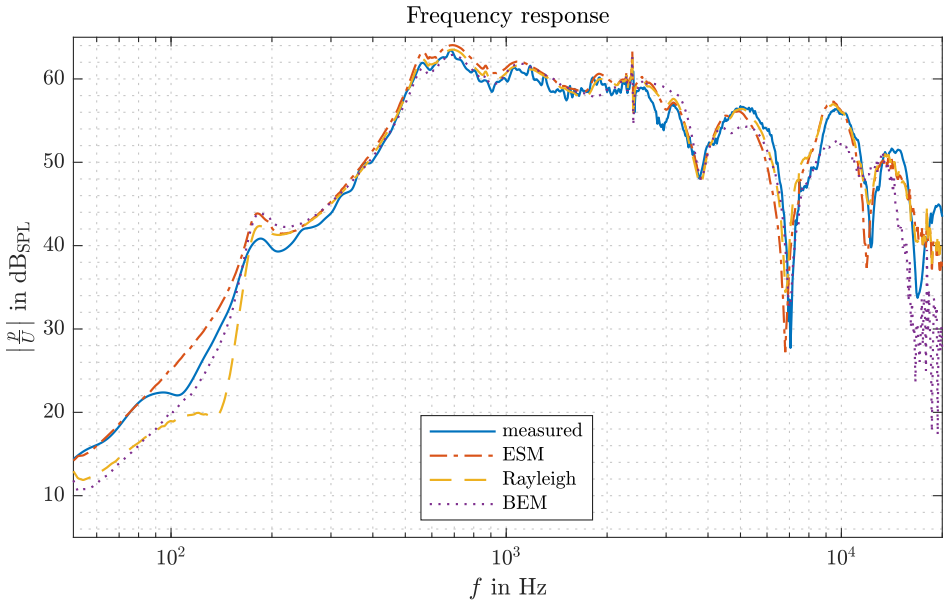
Figure 5.2 *Phase mismatch of the full-space simulations and acoustic reference measurements, evaluated at four positions in* 10 cm *distance; (a): centered in front of the device (position 1); (b): centered at the back side (position 2);*

ESM appears to reproduce the measured phase most accurately.

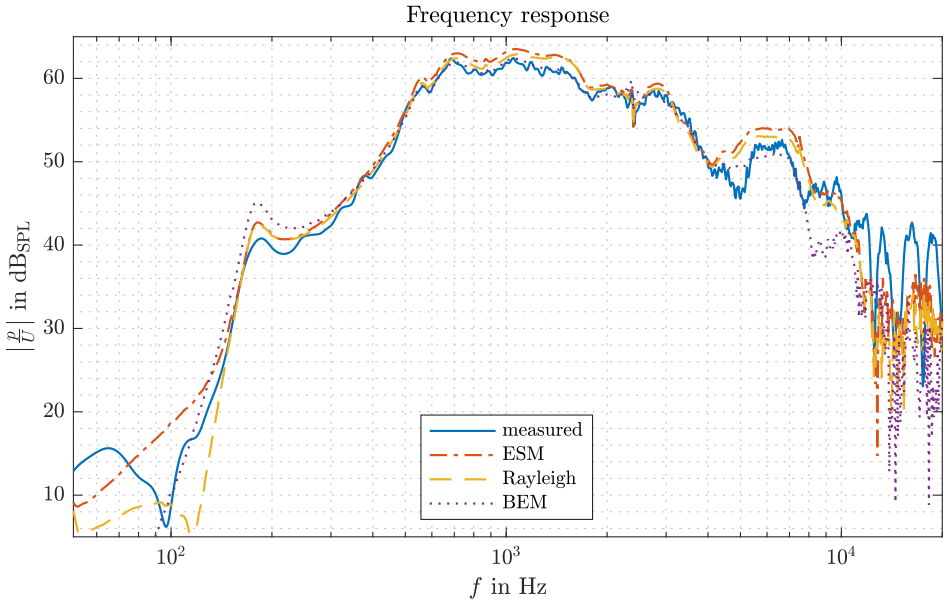
Finally, the magnitude responses of position 4 are shown in Fig. 5.3 (b). Here, ESM and Rayleigh stay within ± 2 dB deviation to the reference from 142 Hz–4.25 kHz. The BEM yields a higher deviation around the 180 Hz peak, with the maximum deviation of 4.6 dB at 175 Hz. The simulations appear to follow the ripples of the acoustic reference that occurs at frequencies > 4.25 kHz until about 8 kHz. Here, the BEM stays within ± 4 dB and the other two within ± 6 dB deviation. All models seem to lose the track above 8 kHz.

Position 4's phase mismatch is depicted in Fig. 5.4 (b). The frequency range of the $\pm\frac{\pi}{8}$ rad bound is 158 Hz–3.3 kHz for the ESM, 172 Hz–3.3 kHz for the combined-Rayleigh formulation, and 172 Hz–2.06 kHz for the BEM. The $\pm\frac{\pi}{4}$ rad bound is up to 4.5 kHz for all models. Again, the absolute mismatch continuously increases towards high frequencies.

To sum up, a good overall sound-field reproduction is obtained from about 230 Hz–2 kHz, which is the region where all three models deviate by less than ± 2 dB from the acoustic reference at all four positions. The reproduction of the back-side sound field is not accurate above 2 kHz and fails above 3.5 kHz. However, the front side seems to be sufficiently reproduced to at least 4 kHz at position 4. Here, the reproduction fails above 10 kHz with ESM and Rayleigh and above 8 kHz with the BEM. Positions 1 and 3 are well reproduced to at least 14 kHz with the ESM and Rayleigh, whereas the BEM seems to fail to reproduce high frequencies above 10 kHz. Regarding low frequencies < 230 Hz, the peak at the actuator resonance frequency 180 Hz is resolved within ± 2 dB deviation, except for the BEM at position 4 with $+4.6$ dB and BEM and ESM at position 3, with $+3$ dB. The



(a)



(b)

Figure 5.3 *Measured and simulated full-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage in* $\mathrm{dB_{SPL}}$*; evaluated at two positions in* 10 cm *distance; (a): in front of the device with* 60° *rotation angle around the* y *axis (position 3); (b): in front of the device, with* 60° *rotation angle around the* x *axis (position 4);*

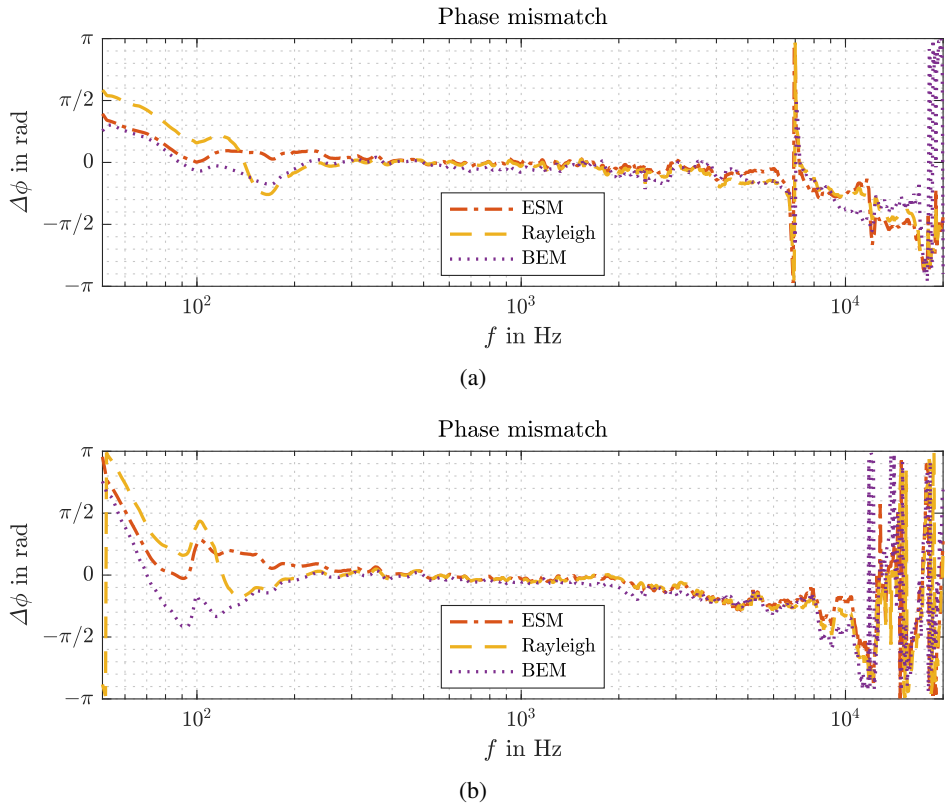


Figure 5.4 *Phase mismatch of the full-space simulations and acoustic reference measurements, evaluated at four positions in* 10 cm *distance; (a): in front of the device with* 60° *rotation angle around the* y *axis (position 3); (b): in front of the device, with* 60° *rotation angle around the* x *axis (position 4);*

deviation of all models generally grows below that resonance peak.

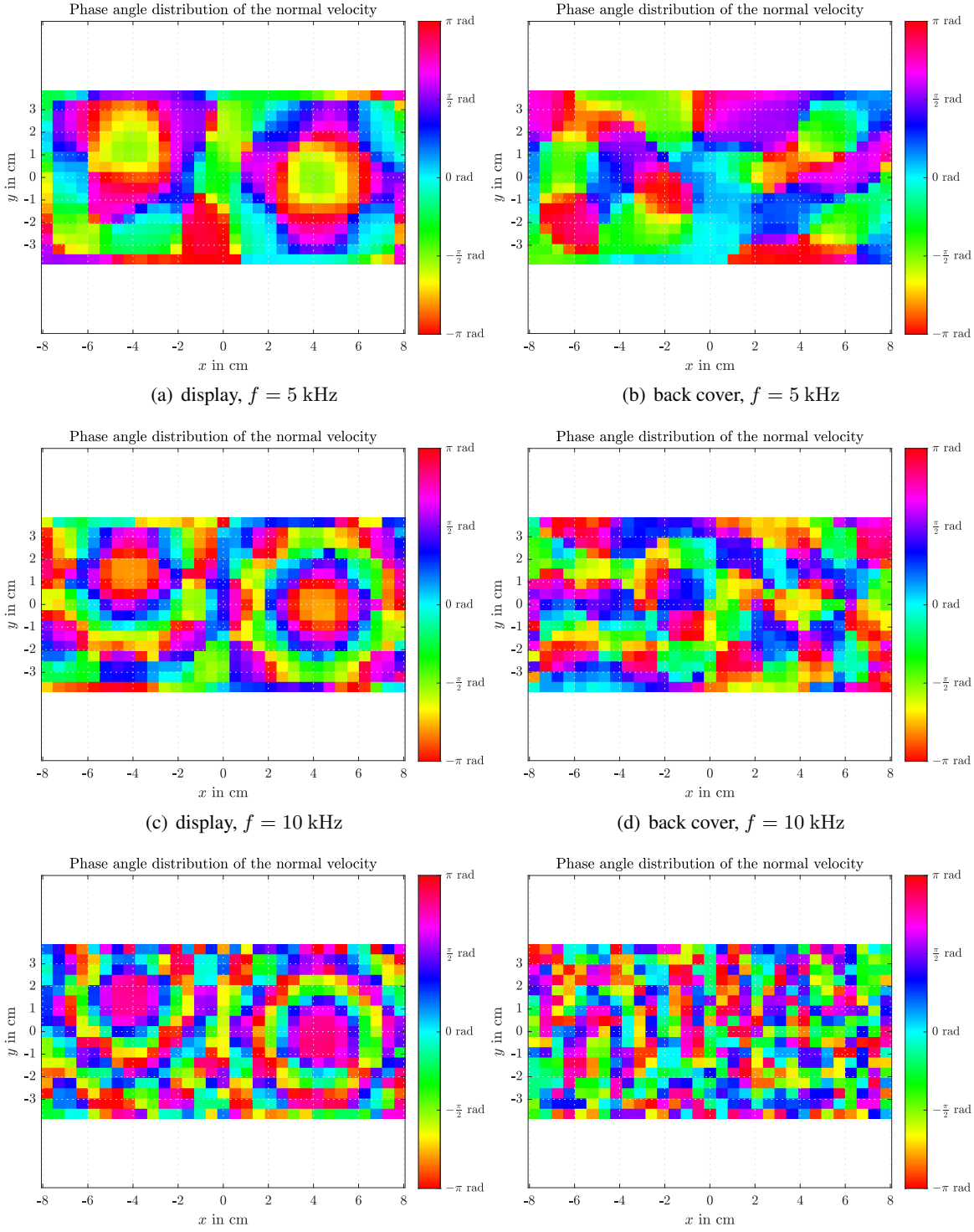
The mismatch towards low frequencies might be partially explained by observing Fig. 3.8. The graph reveals that the acoustic measurements are not well reproducible below 150 Hz. Further, note that the measurements were repeated without changing the position in the room. The room influence could thus be even larger than observed in Fig. 3.8.

Another impairing factor in the low-frequency region might be acoustic leakage in the smartphone dummy. An acoustic leak causes parts of the interior sound field to escape to the exterior. As the interior sound field is responsible for the volumetric coupling between the display and the back cover, the leak results in lower vibration amplitudes at the back cover. In such a case, the escaped acoustic energy contributes to the total sound field outside. However, as the simulation models only rely on the surface velocities, their computed sound fields might differ. A leak between the frame and the back cover of the smartphone dummy was found during the development of this thesis. The leak caused a stronger destructive interference at the actuator's resonance in the acoustic measurements while causing a weaker interference in the simulations due to the less-vibrating back cover. This effect resulted in a large deviation of 17 dB at 180 Hz. Even though that leak could be identified and sealed for the final evaluation, the presence of smaller acoustic leaks in the device cannot be ruled out.

Towards high frequencies, the quality of the simulations seems to decrease with the amount of back-surface radiation involved. Position 2 yields the worst results, followed by position 4, which is at

a greater distance to the device's edges than position 3. Position 1 yields the best results. For an explanation, let us consider Fig. 3.4 (a). The graph shows the averaged absolute sums of the device's surfaces, where the back cover and the frames pass into a noise-like shape above 10 kHz. This might have two causes: 1) the vibration amplitudes are too low to be captured by the laser; 2) the wavelengths are too short to be captured accurately with the scanned grid.

Investigating the measured velocity distribution in terms of its phase at the back cover revealed that, indeed, the back cover vibrates in more complex patterns than the display. The more complex patterns might stem from the different materials used in the back cover, which is made out of a single layer of plastic compared to the multi-layer structure of a display, or from the different fixation applied for the two surfaces. While the display is merely glued to the device's frame, the back cover is additionally fixated with eight screws, as observable in Fig. 3.2. The screws might form bending-wave nodes at their location, splitting the plate modes into smaller wavelengths. The spatial resolution of the velocity scan might thus not be sufficient for resolving the waveforms at high frequencies. After the Nyquist/Shannon theorem, two elements per wave are sufficient for resolving a wave in the Fourier transform. However, one might require more than two elements to obtain a solution within certain error bounds. This is demonstrated by Marburg [25] for the BEM and might partially apply to Rayleigh and ESM. Hence, the spatial resolution might be insufficient to resolve the back-cover vibration even below 10 kHz. Further, consider that the BEM mesh in Fig. 4.6 has variably sized elements that grow towards the surface's center. The elements partially exceed the scanned mesh size, which might explain the BEM's inferior performance towards high frequencies. Fig. 5.5 shows the scanned phase angle distribution of the normal velocity of the front display and the back cover, at 5 kHz, 10 kHz, and 15 kHz.



(a) display, $f = 5$ kHz

(b) back cover, $f = 5$ kHz

(c) display, $f = 10$ kHz

(d) back cover, $f = 10$ kHz

(e) display, $f = 15$ kHz

(f) back cover, $f = 15$ kHz

Figure 5.5 *Image plots showing the smartphone dummy's phase angle distribution of the normal velocity at its display and back cover, illustrating the bending waves occurring in the surfaces.*

5.1.2 Investigating the spatial sound fields

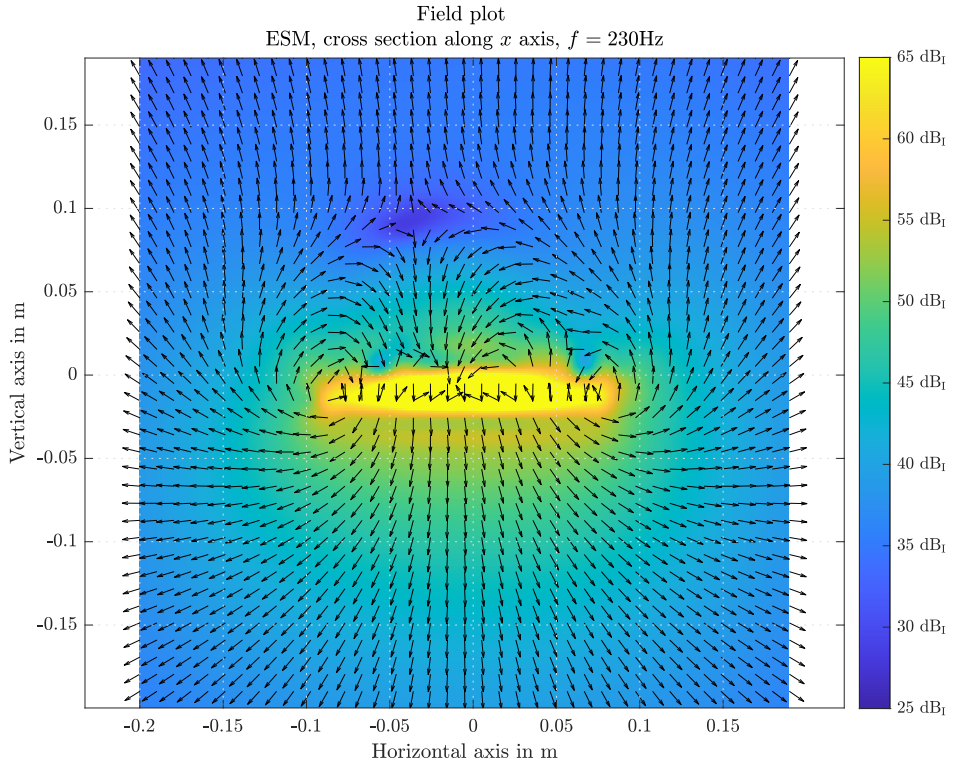
This subsection provides a deeper insight into the spatial sound field of the smartphone dummy by evaluating balloon plots and intensity-field plots created with *balloonPlot()* and *fieldPlot()* (cf. Section 4.3). One central point is to investigate whether the models can reproduce wave diffraction to link the upper and the lower half-space radiation. Even though the BEM, as the more sophisticated and more approved method, should be capable of reproducing wave diffraction, this cannot be granted for the here employed single-source-layer ESM and the combined-Rayleigh formulation. The following paragraphs discuss the spatial sound fields at the valid frequency limits found in Section 5.1.1: $230\ \mathrm{Hz}$ and $2\ \mathrm{kHz}$. Balloon plots and intensity-field plots at other frequencies are shown in the appendix, Figs. D.7 to D.24.

Figs. 5.6 to 5.8 show intensity-field plots and balloon plots at $230\ \mathrm{Hz}$ for the ESM, the combined-Rayleigh formulation, and the BEM, respectively. Considering the intensity fields, the circumferential-shaped turbulences formed by the arrows represent the energy flow between the upper and the lower half space. All three models appear to allow energy transferring through the (x, y) boundary plane. The turbulences are more pronounced at lower frequencies, cf. Figs. D.7 to D.12, which might be due to an increasing phase shift between the upper and lower surface vibration and the longer wavelengths at lower frequencies. One can observe that the extent of the circumferential turbulences is generally shorter in the combined-Rayleigh formulation than in ESM and BEM, which might be a hint that the wave diffraction is less resolved in the Rayleigh field. The balloon plots in Figs. 5.6 to 5.8 show that the ESM yields the largest phase shift between front-side and back-side radiation. The combined-Rayleigh formulation shows the lowest phase shift. The phase shifts correspond to the turbulences in the intensity fields, where the ESM yields the most intensive turbulences at $r = 10\ \mathrm{cm}$ distance origin.

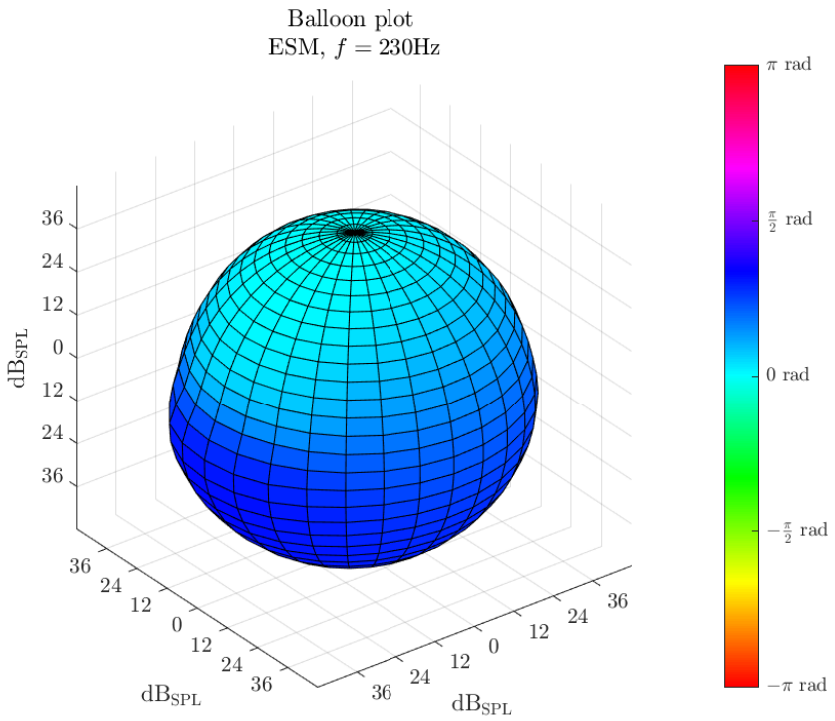
The intensity-field plots and balloon plots at $2\ \mathrm{kHz}$ are depicted in Figs. 5.9 to 5.11. The field plots of ESM and Rayleigh reveal a higher similarity, whereas the BEM appears to produce smoother transitions in the intensity field. All three models yield a drop in magnitude towards the lower-right corner and towards the central-left side edge. These drops in magnitude are also mapped at the lower-right and central-left sides of the balloon plots, where the highest phase shifts and the lowest magnitudes are observable. The balloon plot of the BEM yields a smaller phase shift at its lower-right side, which corresponds to the smoother field plot in this region. However, a smoother sound field of the BEM cannot be generally assumed, as plots at higher frequencies Figs. D.16 to D.24 indicate.

To sum up, the spatially radiated sound fields of all three models are comparable within the valid frequency range of $200\ \mathrm{Hz}$–$2\ \mathrm{kHz}$. The graphs in the appendix show that ESM and combined-Rayleigh formulation maintain similar sound fields up to $15\ \mathrm{kHz}$. The BEM increasingly deviates above $2\ \mathrm{kHz}$, which might be explained by insufficient spatial sampling at the surfaces, as already described in Section 5.1.1. At low frequencies $\leq 200\ \mathrm{Hz}$, the turbulences in the BEM and ESM fields appear more similar than in the combined-Rayleigh formulation.

A reason for the smaller turbulences in the combined-Rayleigh formulation might be its inherent assumption of a planar source. The display and the back cover radiate from $z = 0$ in this formulation, resulting in $180°$ edge angles at the surface boundaries. By contrast, the BEM accurately models the device's depth of $1.1\ \mathrm{cm}$ and thus maintains the $90°$ edge angles at the upper and lower surface. The ESM represents a special case. Even though the single source layer is a planar radiator containing monopoles and dipoles similar to the combined-Rayleigh formulation, the model includes the depth of the device. The equivalent sources sit $z = -0.55\ \mathrm{cm}$ and mimic the display and back-cover velocities at their original positions in z. Since the frame velocity is not modeled in this ESM, a clear definition of an edge angle cannot be stated. However, we may assume that the three models imply different conditions at the edges, which could determine their performance in resolving wave diffraction.

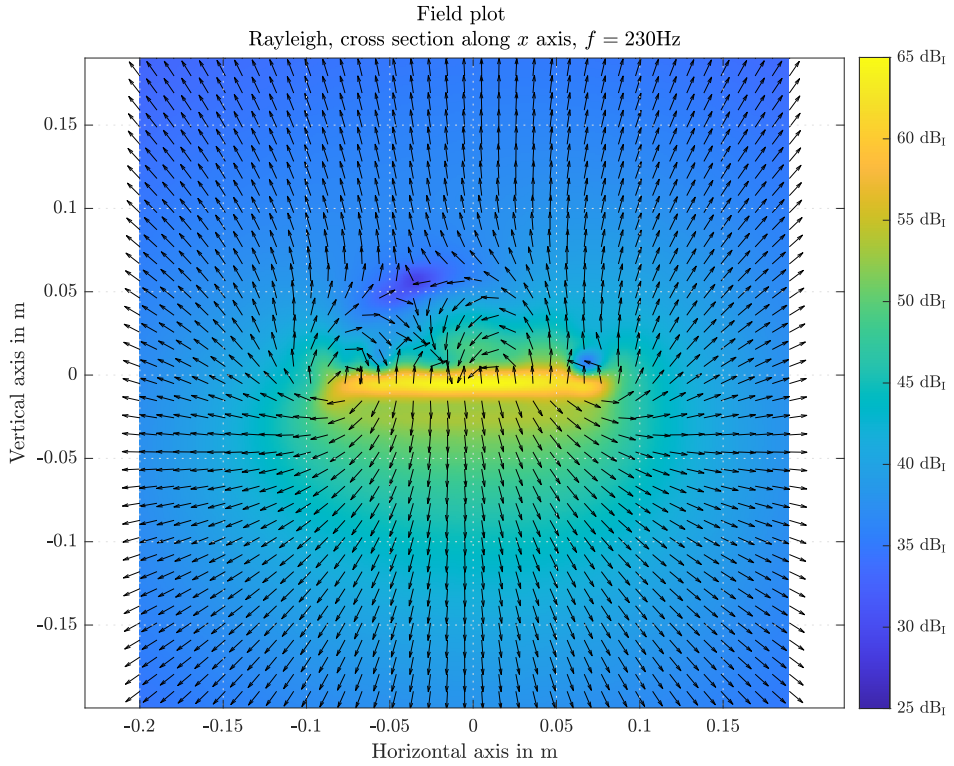


(a)

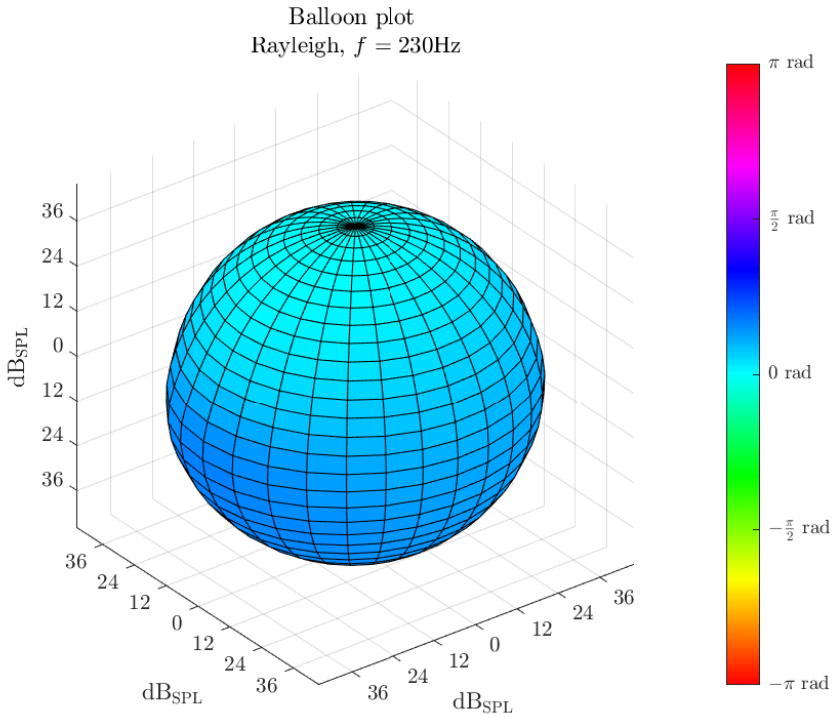


(b)

Figure 5.6 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 230 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

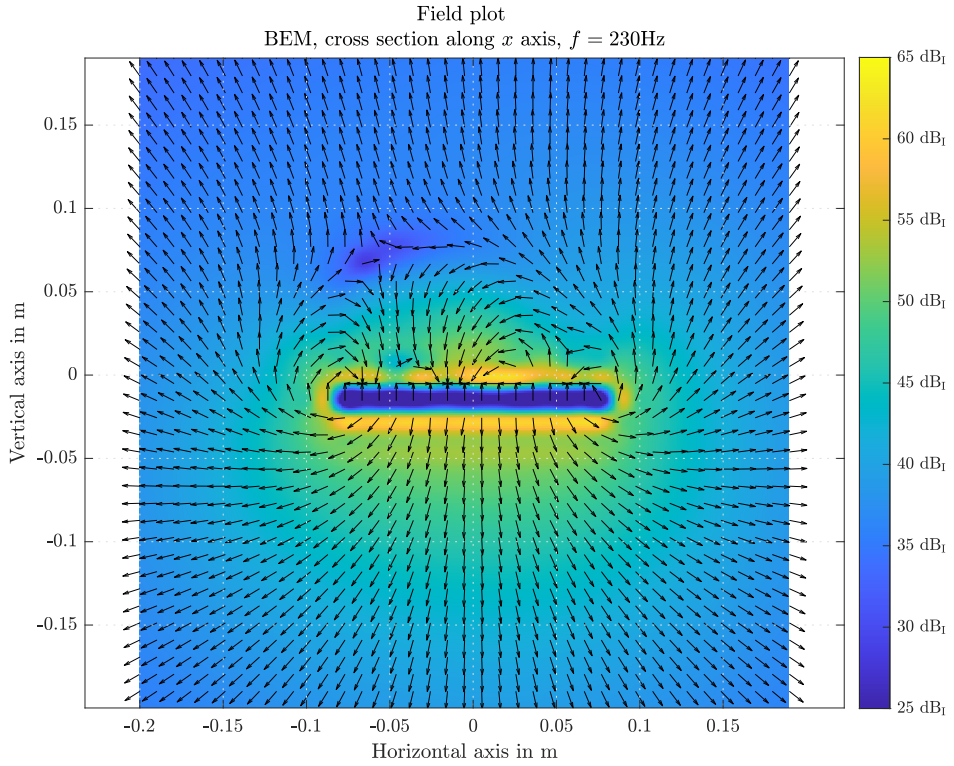


(a)

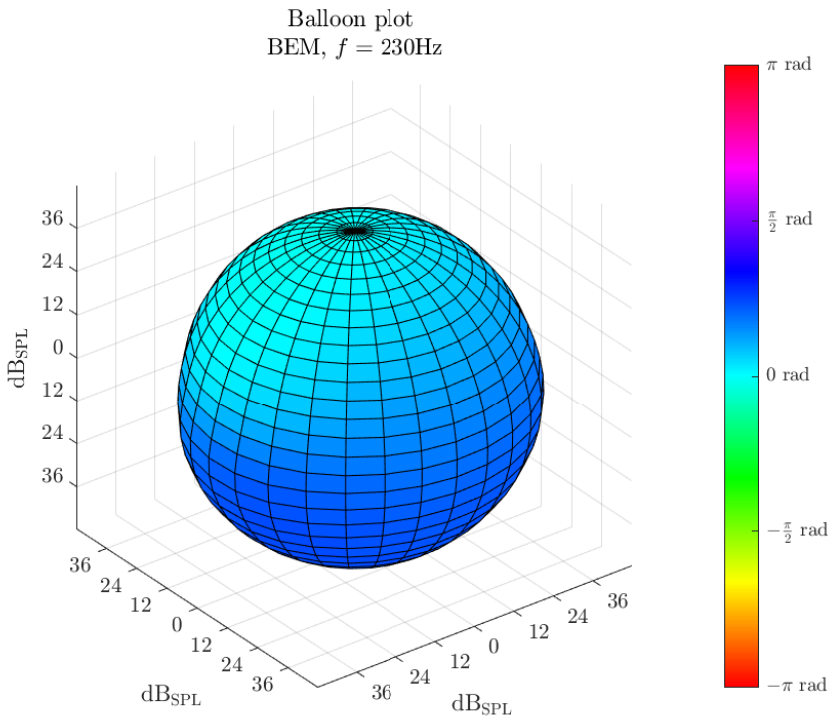


(b)

Figure 5.7 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 230 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

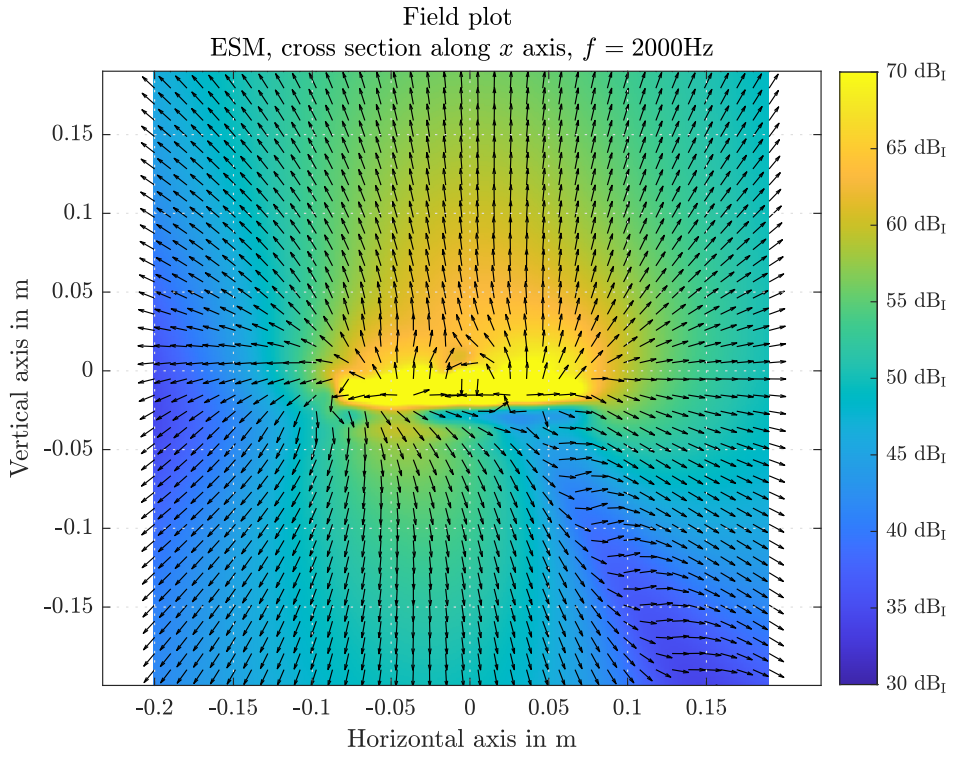


(a)

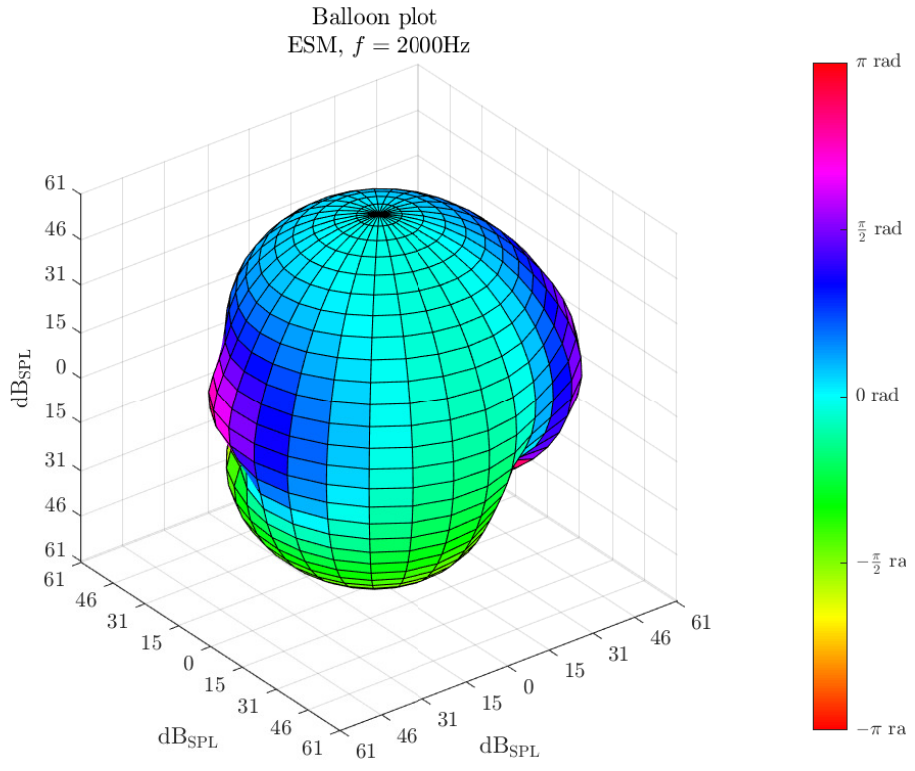


(b)

Figure 5.8 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 230 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

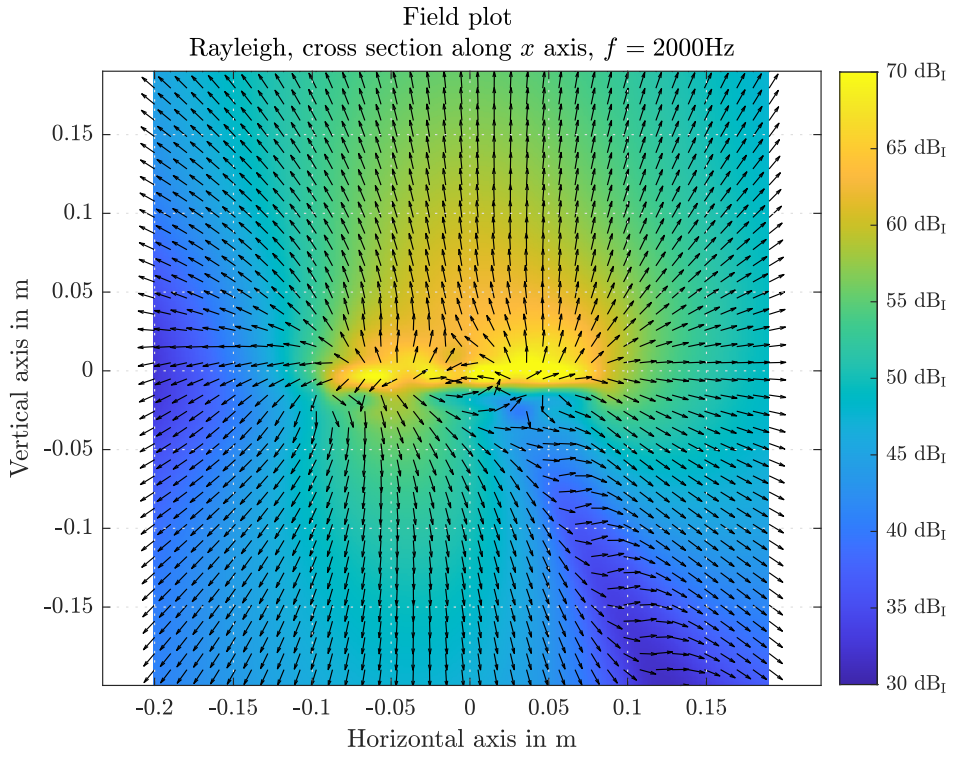


(a)

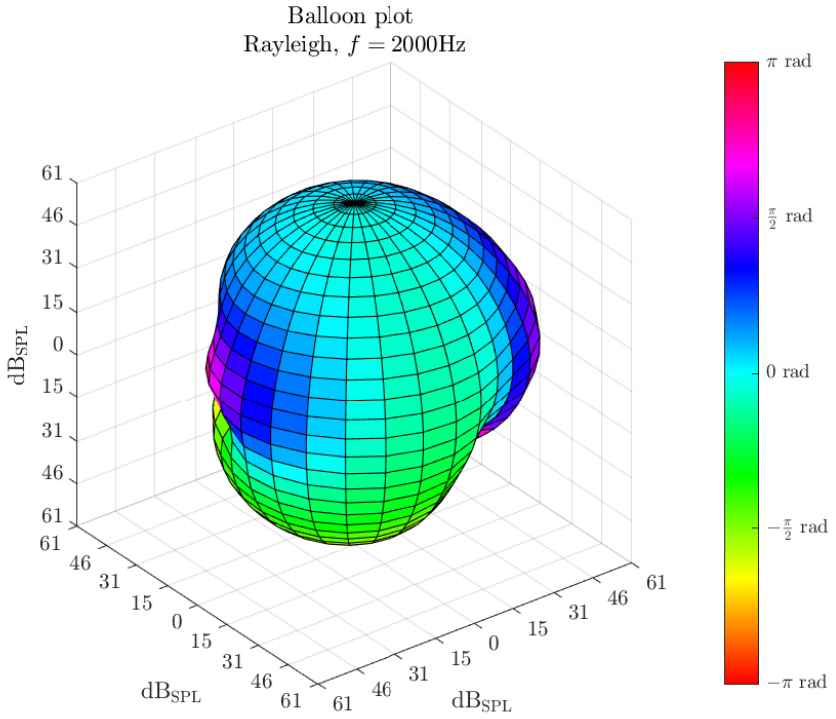


(b)

Figure 5.9 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 2 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

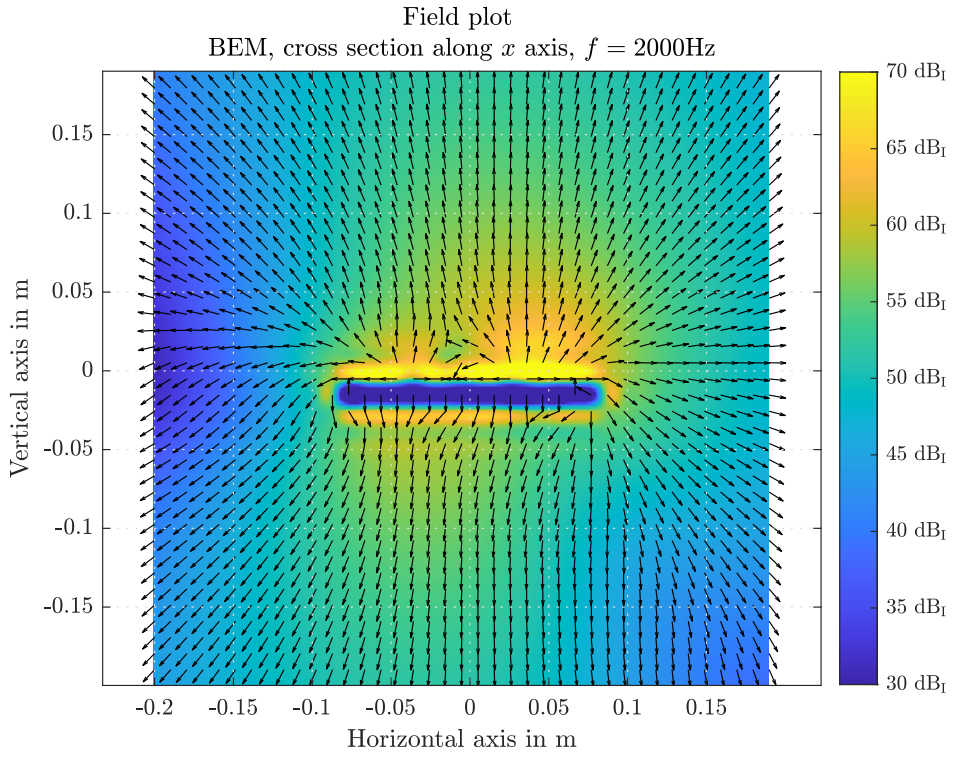


(a)

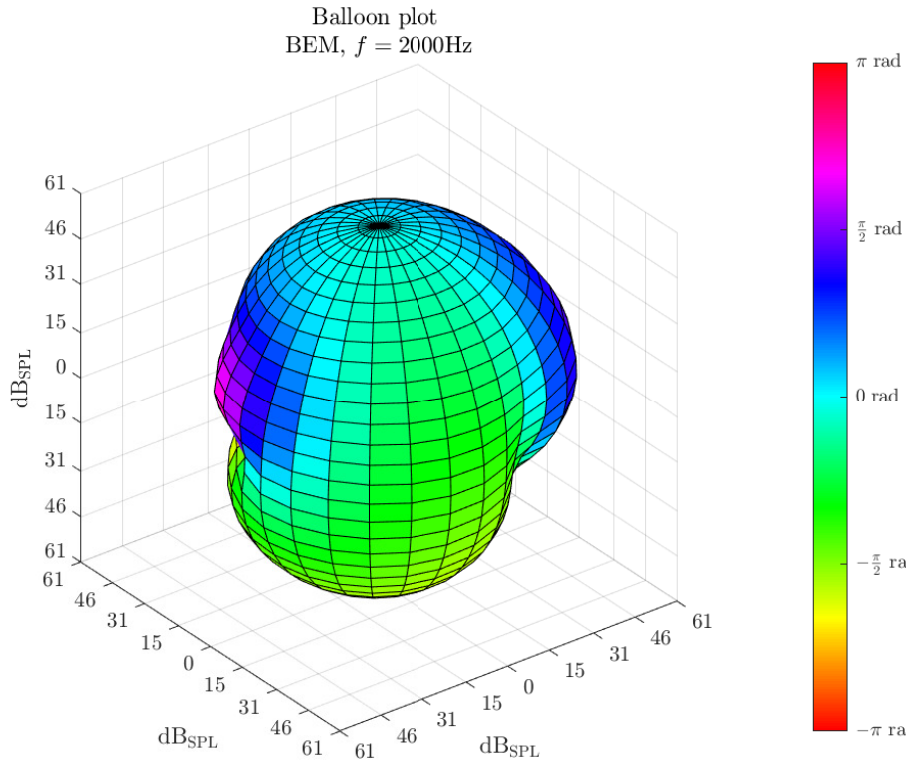


(b)

Figure 5.10 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 2 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*



(a)



(b)

Figure 5.11 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 2 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

5.2 Half-Space Simulations

The computed frequency responses of the ESM and the Rayleigh I integral are compared to the acoustic reference measurements for assessing the half-space simulations. The comparison is made for two scenarios: 1) the smartphone dummy in $10\ \mathrm{cm}$ distance, and 2) the Manticore voice-coil loudspeaker as an alternative verification case in $3.16\ \mathrm{cm}$ distance. The receiver is positioned in the center of the devices for both scenarios. As in Section 5.1, the frequency responses are normalized by the driving voltage and the constants used in the simulations are $\rho = 1.18\frac{kg}{m^3}$ and $c = 346.7\frac{m}{s}$. The following paragraphs discuss the deviations between the simulations and the measurements in terms of their magnitude response, as depicted in Fig. 5.12, and their phase mismatch according to Eq. (5.1), depicted in Fig. 5.13. The overall frequency responses of the two scenarios, plotted with the function *plotFR()*, can be found in the appendix: Figs. D.5 and D.6.

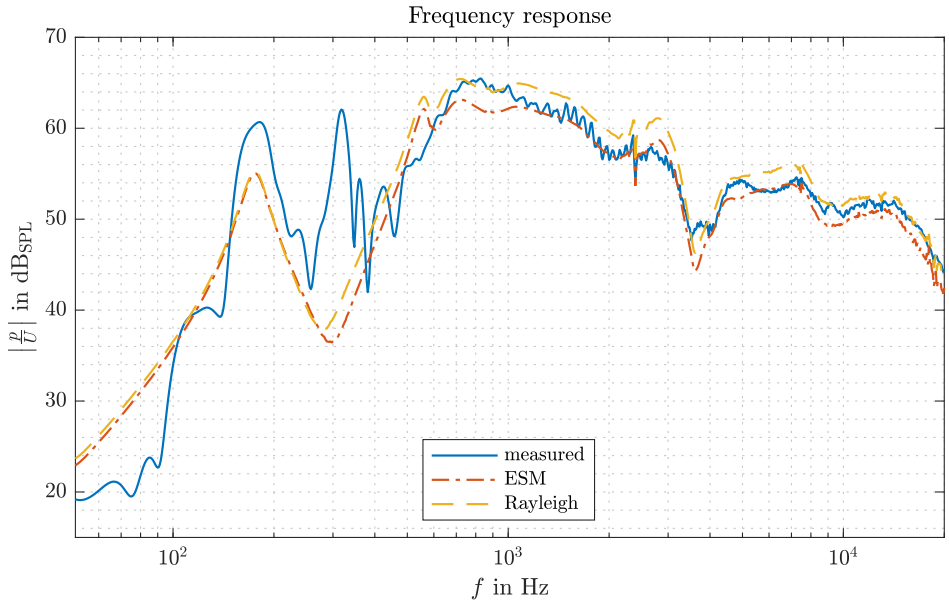
Considering the magnitude response of the smartphone dummy in Fig. 5.12 (a) reveals large deviations between the simulations and the acoustic measurement below $585\ \mathrm{Hz}$. In this lower frequency region, the acoustic reference yields ripples that are mostly not followed by the simulated curves, e.g., the peaks at $320\ \mathrm{Hz}$, $362\ \mathrm{Hz}$, and $422\ \mathrm{Hz}$. The actuator-resonance peak at $180\ \mathrm{Hz}$ is resolved by the simulations but appears $+5.7\ \mathrm{dB}$ louder in the acoustic reference. At $565\ \mathrm{Hz}$, the simulations show a peak that does not occur in the reference. The simulations then stay within a $\pm 3.5\ \mathrm{dB}$ bound to the acoustic reference from $585\ \mathrm{Hz}$–$3.53\ \mathrm{kHz}$. From $3.8\ \mathrm{kHz}$–$18.75\ \mathrm{kHz}$, the deviation stays within $\pm 2.5\ \mathrm{dB}$. The notch around $3.63\ \mathrm{kHz}$ is more distinct in the simulations, where the ESM reaches $-4.9\ \mathrm{dB}$ and the Rayleigh I integral $-2.8\ \mathrm{dB}$ below the reference.

The phase mismatch in of the smartphone dummy in Fig. 5.13 (a) remains within $\pm\frac{\pi}{8}\ \mathrm{rad}$ from $607\ \mathrm{Hz}$–$3.68\ \mathrm{kHz}$ and within $\pm\frac{\pi}{4}\ \mathrm{rad}$ from $453\ \mathrm{Hz}$–$7.4\ \mathrm{kHz}$. The absolute mismatch continuously increases towards higher frequencies, which indicates that the microphone might have been slightly displaced in the acoustic measurement. At low frequencies $< 453\ \mathrm{Hz}$, the mismatch grows large, which is in accordance with the high deviations in the magnitude plot.

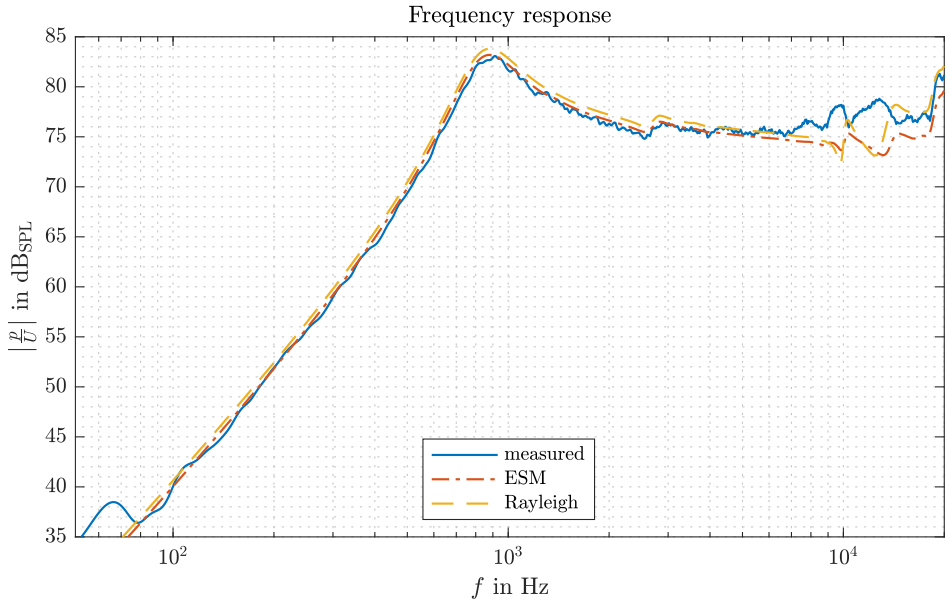
Observing the alternative verification case with the Manticore voice-coil loudspeaker in Figs. 5.12 and 5.13 (b), the magnitude response of the simulations stay within $\pm 1.5\ \mathrm{dB}$ deviation to the acoustic measurement from $75\ \mathrm{Hz}$–$7.45\ \mathrm{kHz}$. From $7.45\ \mathrm{kHz}$–$14.5\ \mathrm{kHz}$, the simulations underestimate the reference by up to $-5.7\ \mathrm{dB}$. The deviation then stays within $\pm 2.5\ \mathrm{dB}$ from $14.5\ \mathrm{kHz}$–$20\ \mathrm{kHz}$. The phase mismatch also shows good results here, remaining within $\pm\frac{\pi}{8}\ \mathrm{rad}$ from $70\ \mathrm{Hz}$–$7.75\ \mathrm{kHz}$.

Note that the acoustic half-space measurement of the smartphone dummy is more prone to errors than that of the Manticore, as the baffle and the measurement chamber are optimized for the measurement of miniature voice-coil loudspeakers (cf. Section 3.3). The smaller measures of the Manticore allow for a shorter measurement distance, which reduces the influence of reflected or diffracted waves occurring in the room and at the baffle edges. Further, the adapter plate that integrates the Manticore into the baffle was optimized by Sobtzick [56] to attenuate bending waves and is more advanced than that of the smartphone dummy. While the Manticore adapter plate is a three-layer composition of aluminum and bitumen, the adapter plate used for the smartphone dummy is a simple aluminum plate. Hence, the smartphone dummy might transmit bending waves more easily into the baffle than the Manticore. In a subsequent investigation, the Polytec vibrometer was used to observe the vibration of the smartphone dummy clamped in its adapter plate without the baffle. The fundamental plate resonance was identified around $320\ \mathrm{Hz}$, which corresponds to the strong peak in Fig. 5.12 (a).

Regarding the significant deviation from $7.45\ \mathrm{kHz}$–$14.5\ \mathrm{kHz}$ in the Manticore magnitude response in Fig. 5.12 (b), a consultation with the acoustic engineers at Sound Solutions revealed that the higher magnitude in the reference measurement is likely due to pressure accumulation that occurs between the microphone capsule and the loudspeaker. The simulation models do not cover this phenomenon.



(a)



(b)

Figure 5.12 *Measured and simulated half-space frequency responses as sound pressure normalized by the input voltage in* $\mathrm{dB_{SPL}}$*; measured at centered positions in front of the devices; (a): smartphone dummy measured in* 10 cm*; (b): Manticore voice-coil loudspeaker as alternative verification case, in* 3.16 cm *distance;*

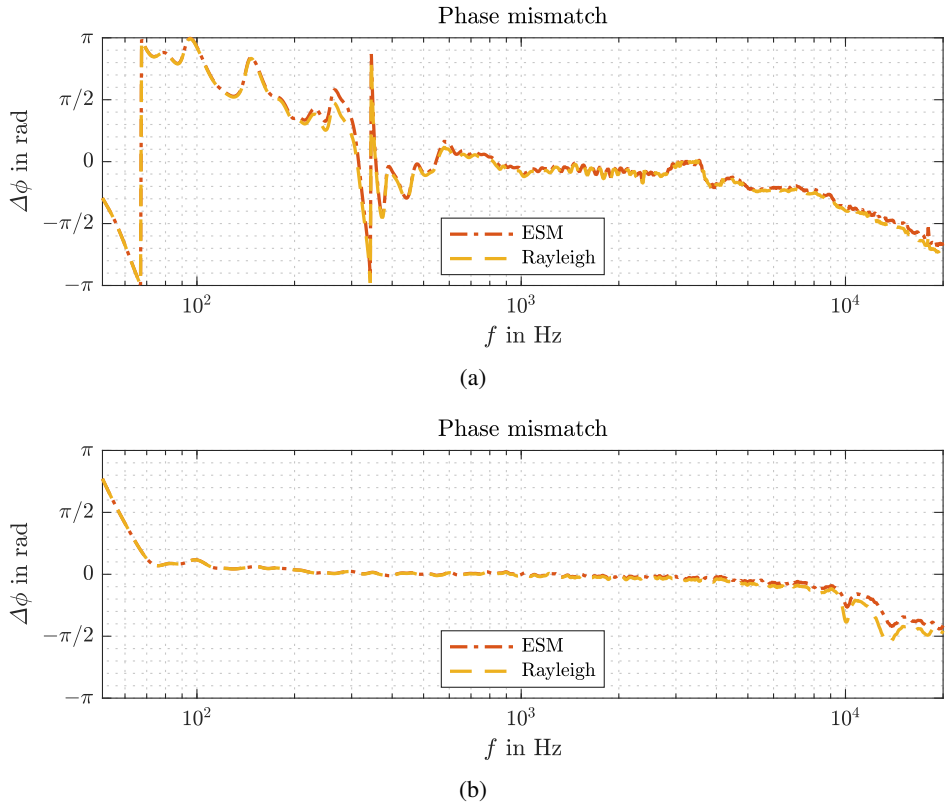


Figure 5.13 *Phase mismatch of the half-space simulations and acoustic reference measurements, evaluated at centered positions in front of the devices; (a): smartphone dummy measured in* 10 cm*; (b): Manticore voice-coil loudspeaker as alternative verification case, in* 3.16 cm *distance;*

5.3 Assessing the Computational Cost

Finally, this section discusses the computational cost in terms of the retarded time for computing the sound radiation of the smartphone dummy. For the experiment, the sound field is evaluated at ten receiver coordinates on a horizontal ring in the upper half space, at 1 m distance. The models are evaluated at ten frequencies, reaching from 51 Hz–20 kHz. Evaluating multiple frequencies seemed necessary due to the matrix inversions in the BEM and the ESM, as the computation time of matrix-inversion algorithms can vary with the condition of the matrix. The computation times of BEM and ESM might thus be frequency-dependent, as the inverted matrices contain Green's functions. The experiment was repeated five times to avoid outliers that could occur due to the complex resource allocation in modern computers.

The median retarded times of the five repetitions are plotted over the ten evaluated frequencies in Fig. 5.14. The graph shows that all models vary in their efficiency over frequency. However, the frequency-dependent variation is small compared to the absolute computation times: ± 7.3 ms for the combined-Rayleigh formulation, ± 14.3 ms for the full-space ESM, ± 8.8 ms for the BEM, ± 1.7 ms for the Rayleigh I integral, and ± 15.3 ms for the half-space ESM.

Considering the small frequency dependence, further discussing the average computation time for a single frequency is reasonable. Fig. 5.14 reveals that the two Rayleigh-integral methods outperform ESM and BEM. As expected, the Rayleigh I integral is with an average of 7.3 ms the most efficient model, as it does not contain matrix inversions and is less complex than the combined-Rayleigh formulation. The half-space ESM, on average, requires 78 ms for a single frequency, i.e., 10.7 times the Rayleigh I integral. As for the full-space models, the ESM computation lasts on average 106 ms, which is 4.9 times the combined-Rayleigh formulation, which takes 21.6 ms. The BEM is by far the least efficient and lasts on average 531 ms, which is 24.6 times the combined-Rayleigh formulation or 5 times the full-space ESM.

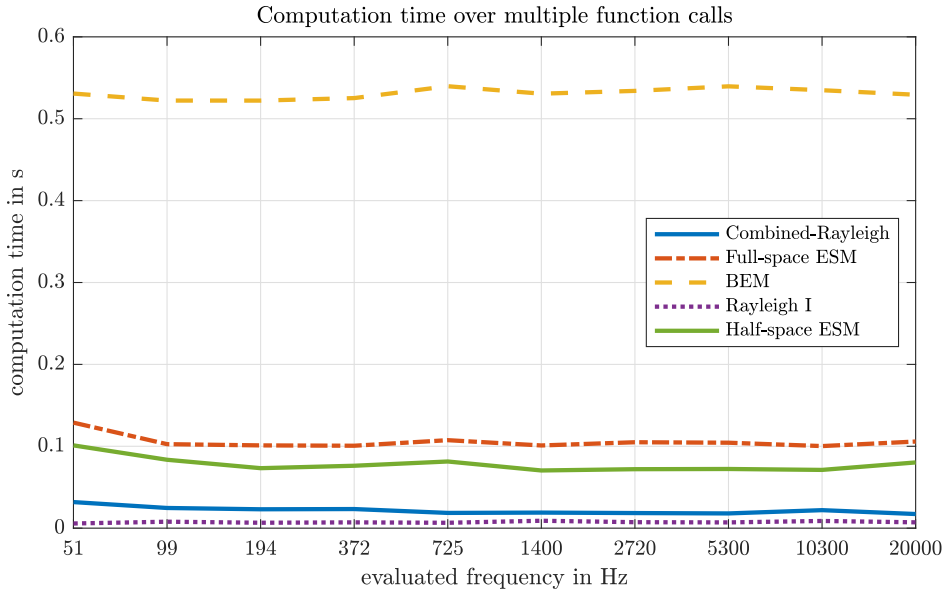


Figure 5.14 *Retarded time of the five simulation models for computing the radiated sound pressures of the smartphone dummy at ten receiver positions; the curves show the median retarded time of five repeated computations, over multiple frequencies.*

6 Conclusion and Outlook

This thesis successfully developed and implemented five models for simulating the radiated sound field of a hand-held device with a vibrating-display loudspeaker. It recalled the underlying theory required for the implementation and application of the simulation models, and it discussed the source code and code examples that demonstrate their application. Further, it implemented three custom plotting functions for analyzing the radiation behavior of the device under test: a frequency-response plot, a balloon plot, and a cross-section field plot function. The models and the plotting functions were implemented in the scientific programming language Matlab and rely on typical boundary-velocity-based simulation approaches, the Rayleigh integrals, the Equivalent Source Method (ESM), and the Boundary Elements Method (BEM).

The Rayleigh I integral and the ESM with a single, flat layer of monopoles were exploited to simulate the half-space radiation scenario of the display in an infinite baffle. For simulating the full-space sound field that considers the device's front-side and back-side surface, two custom models that exploit the parallel surfaces and the thin structure of the device were suggested: 1) the ESM with a single source layer of monopoles and dipoles that emulate the even-symmetric and odd-symmetric vibration patterns of the upper and lower surface vibration; 2) a combined-Rayleigh formulation that is novel to the author's knowledge; the formulation uses the Rayleigh I integral to emulate the even-symmetric patterns and exploits Fourier-based methods to insert the odd-symmetric vibration patterns into the Rayleigh II integral. As the third and more generic full-space model, a surface mesh of the device under test was created and simulated with the BEM while exploiting the open-source Matlab toolbox OpenBem for the core implementation.

The thesis described the tested devices, a smartphone dummy with two vibrating actuators attached to its display and a miniature voice-coil loudspeaker that provides an alternative boundary condition/verification case for the half-space simulations. Further, the thesis documented the measurement procedure for acquiring the surface velocities and the acoustic reference frequency responses of the devices. The models were assessed by comparing simulated frequency responses to acoustic reference measurements.

The comparison of frequency responses revealed that all models succeeded in producing valid results at a certain frequency range. The two half-space models were evaluated and compared at a single position in the front-center of the smartphone dummy, where the curves deviated by less than $\pm 3.5\,\mathrm{dB}$ from $585\,\mathrm{Hz}$–$3.53\,\mathrm{kHz}$ and by less than $\pm 2.5\,\mathrm{dB}$ from $3.8\,\mathrm{kHz}$–$18.75\,\mathrm{kHz}$. A higher deviation at lower frequencies could be partially linked to insufficient damping of bending-wave transmission to the baffle in the acoustic measurement. The alternative verification case using the Manticore voice-coil loudspeaker could confirm the validity of both half-space models towards lower frequencies, where the simulations deviated by less than $\pm 1.5\,\mathrm{dB}$ to the reference from $75\,\mathrm{Hz}$–$7.45\,\mathrm{kHz}$.

The three full-space models were assessed at four positions: three in the front and one at the back side of the device. Concluding all four positions, the three models deviated by less than $\pm 2\,\mathrm{dB}$ in a range from $230\,\mathrm{Hz}$–$2\,\mathrm{kHz}$. At higher frequencies, the front-side simulations appeared to match better the reference than the back-side simulations, which could be partially reasoned by more complex and low-amplitude vibration patterns occurring in the back cover, combined with a possibly too low spatial resolution in the velocity scan. The frequency responses at the centered-front position, for example, stayed within a range of $\pm 1.5\,\mathrm{dB}$ from $237\,\mathrm{Hz}$–$3.45\,\mathrm{kHz}$ and $\pm 2.5\,\mathrm{dB}$ from $173\,\mathrm{Hz}$–$12.1\,\mathrm{kHz}$. The

combined-Rayleigh formulation and the ESM approach even revealed acceptable results at the front-center position above 12.1 kHz. The BEM failed to reproduce the curve above 12.1 kHz, possibly due to the partially larger elements used in the BEM mesh.

The full-space models were further evaluated by investigating the balloon plots of the sound pressure and the acoustic-intensity field plots. The three models yielded comparable spatial sound fields within the overall valid frequency range of 200 Hz–2 kHz. Above 2 kHz, the ESM and the combined-Rayleigh formulation remained similar, whereas the BEM increasingly deviated from the other two models. Observing low frequencies ≤ 200 Hz revealed more similarities between BEM and ESM. All simulation models were identified to allow energy transfer between the upper and the lower half-space. This energy transmission is especially intriguing for the combined-Rayleigh formulation, as it suggests that the formulation might be capable of reproducing the correct boundary conditions outside the region of prescribed values in the horizontal plane.

The computational cost of the five simulation models was demonstrated in terms of the retarded time for computing the sound pressures of a predefined setup. The Rayleigh-integral-based models outperformed the others: the combined-Rayleigh formulation appeared 4.9 times faster than the full-space ESM and 24.6 times faster than the BEM. The Rayleigh I integral was 10.7 times faster than the half-space ESM.

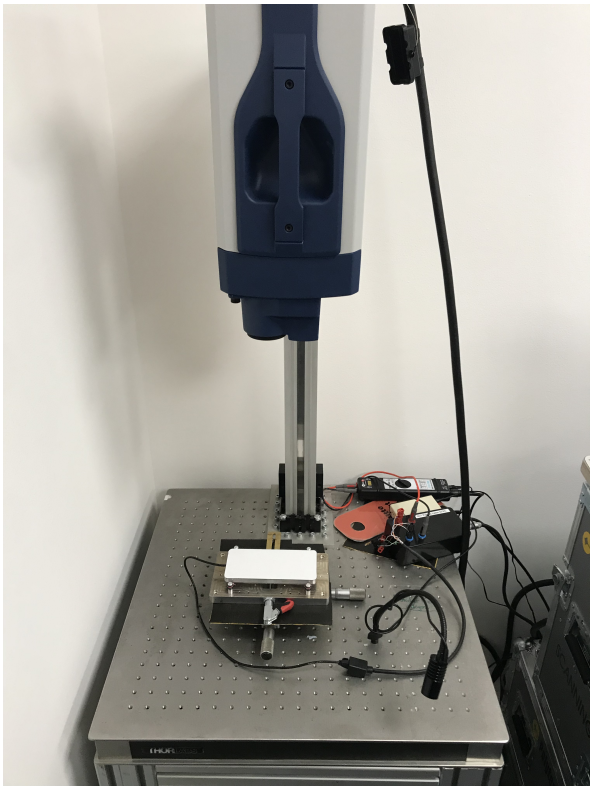
Outlook

It has been demonstrated that the combined-Rayleigh formulation and the ESM with a single source layer of monopoles and dipoles are both capable of simulating the full-space sound radiation from a hand-held device with a vibrating-display loudspeaker. Presumably, the two approaches could also be applied to other thin structures, whereas they are expected to comprise different characteristics.

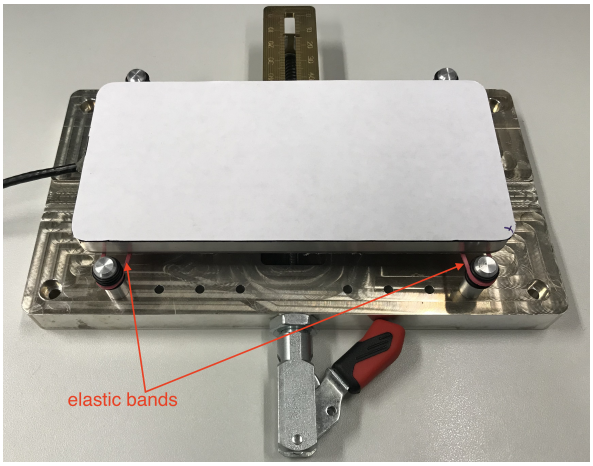
The ESM with the described source constellation might be applicable to simulating shapes with parallel surfaces that are non-flat, provided that the orientation of the dipoles is adjusted accordingly. Further studies could observe the application of this equivalent-source constellation to such non-flat objects. Another characteristic of this constellation is that it is expected to produce sufficiently exact simulations only for devices/vibrating objects with a certain depth. While thin surfaces might cause distortions due to the singularities at the equivalent source's origin, thick surfaces might worsen the condition of the ESM matrix. Investigating these thickness limits and linking them to the required number of sources/reference points could thus enhance the applicability of this ESM.

The combined-Rayleigh formulation is expected only to perform well simulating flat vibrating objects. Presumably, it performs best when employed on an infinitely thin object. Here, a possible investigation would concern the maximum depth of a vibrating object that still produces valid results with the combined-Rayleigh formulation. Further, converting the odd-symmetric velocities into surface sound pressure requires setting multiple parameters, such as the number of zeros padded, the window used, the kind of wave-field extrapolation, or the size of the interpolation function for the regularization. Future studies could investigate these parameters and find rules that allow for a more generic application of the combined-Rayleigh formulation. Additionally, future work could implement more advanced numerical integration techniques to enhance the numerical accuracy of this formulation.

Appendix A Photographs and Screenshots

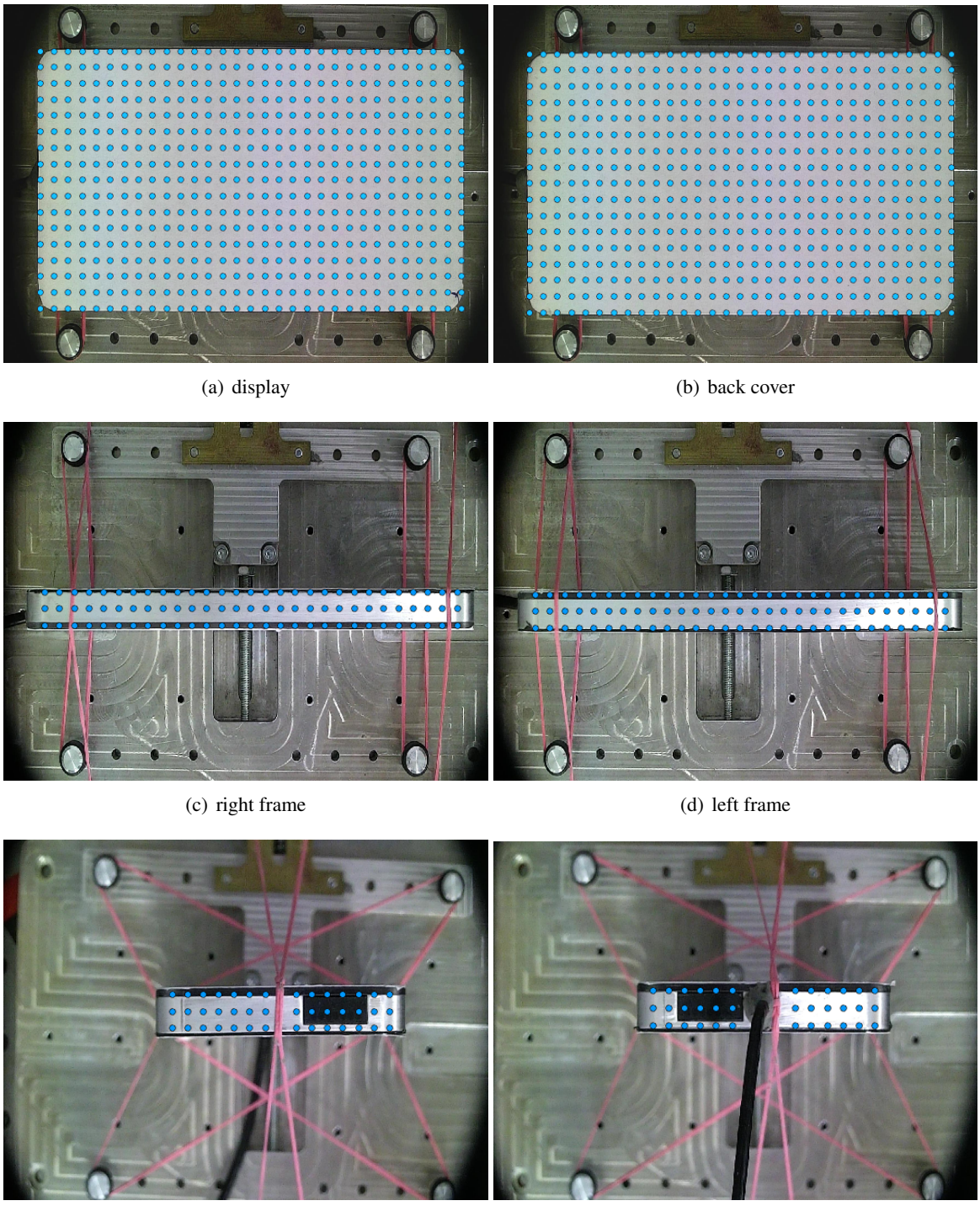


(a)



(b)

Figure A.1 *Photographs of the measurement setup for the velocity scans; (a): measuring table; (b): smartphone dummy with applied adhesive paper on mounting;*

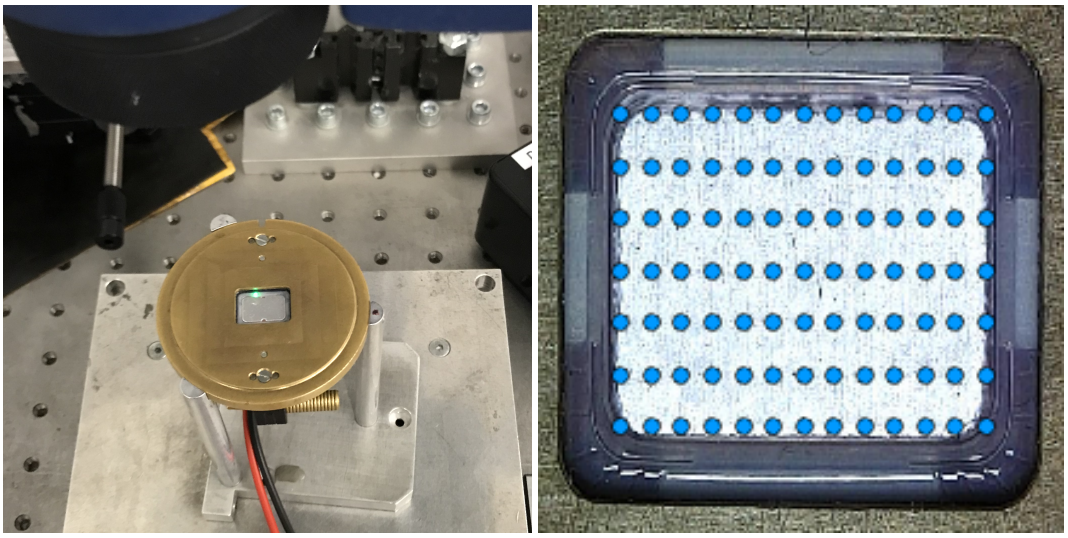


(a) display (b) back cover

(c) right frame (d) left frame

(e) upside frame (f) downside frame

Figure A.2 *Screenshots during the velocity scans of the smartphone dummy; An adhesive paper is applied to the display and the back cover. Due to the equidistant mesh and the rounded corners of the device, some scan points have to be extrapolated later on: (a)-(b): outlying scan points in the corners; (c)-(f): scan points at the edges, the cable, and the rubber bands are left out.*



(a) (b)

Figure A.3 *Pictures during a velocity scan of the additional verification case using the Manticore voice-coil loudspeaker. Only the plate is scanned. Scan positions at the translucent torus are left out. (a): Manticore loudspeaker in mounting; (b): screenshot from the Polytec system;*

Appendix B Settings

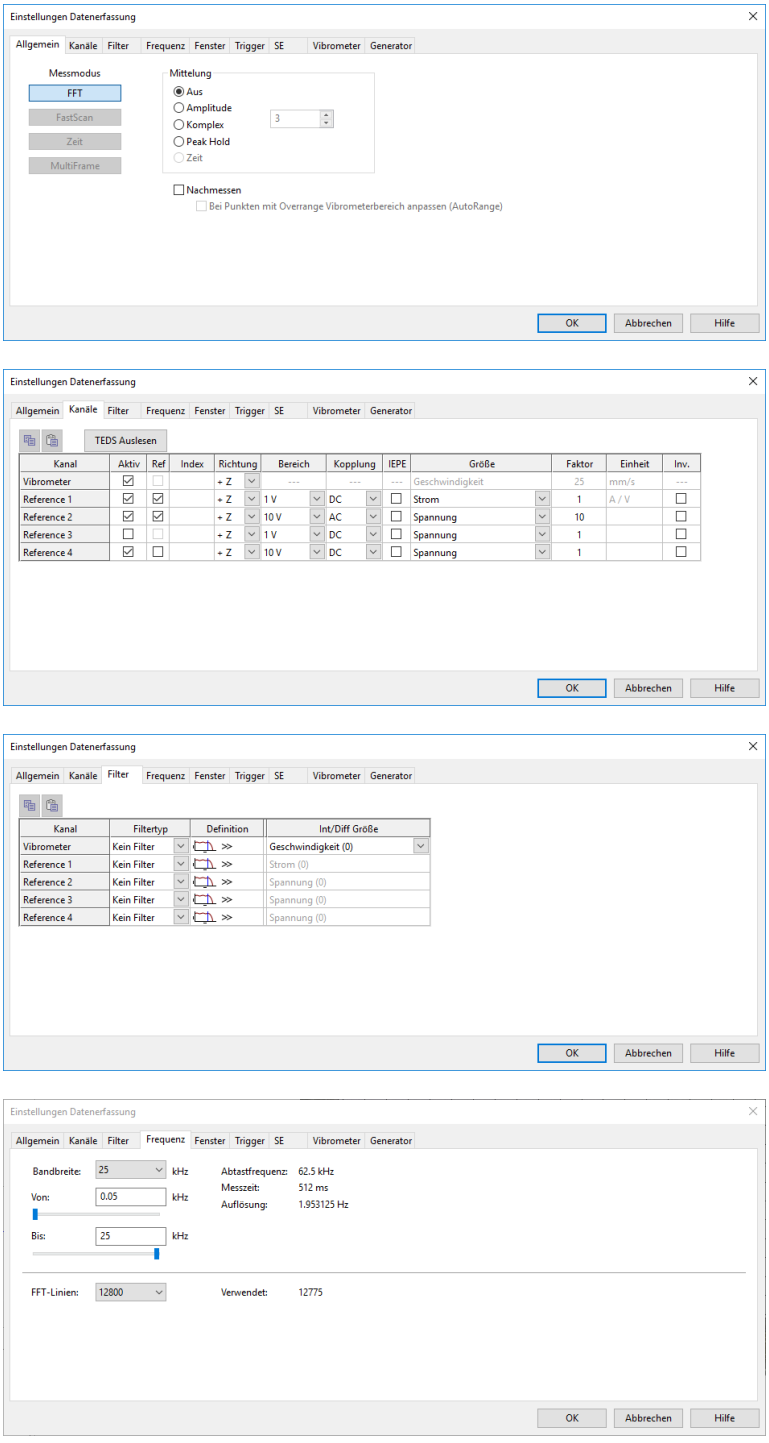
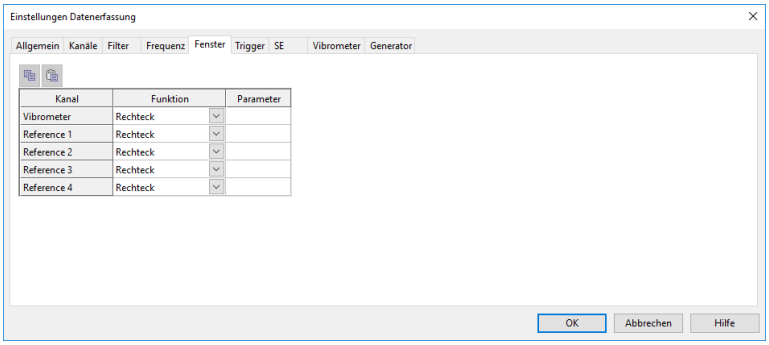
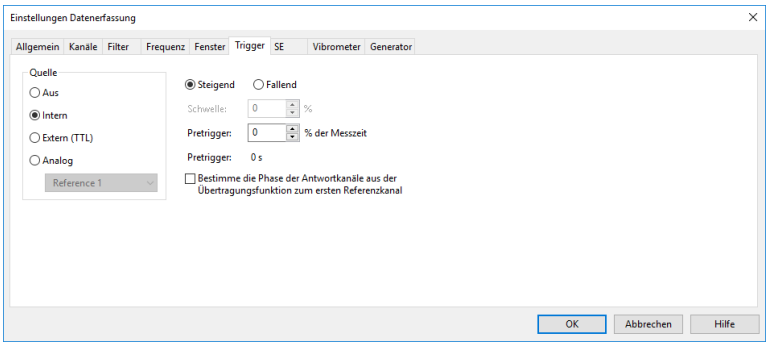
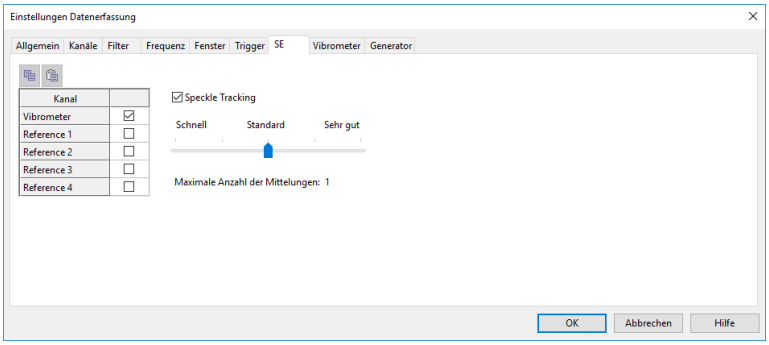
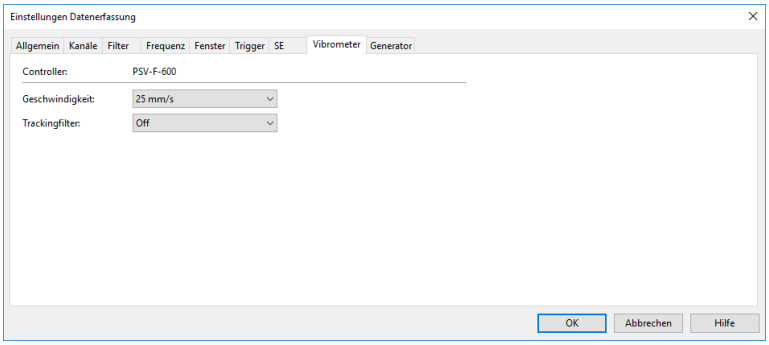


Figure B.1 *Settings screenshots in the Polytec data-acquisition software, part 1;*









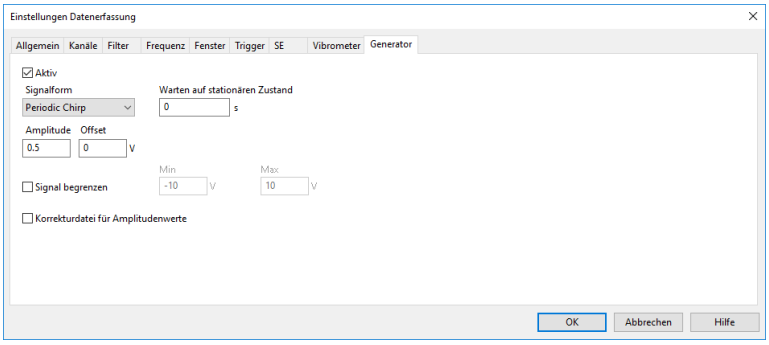
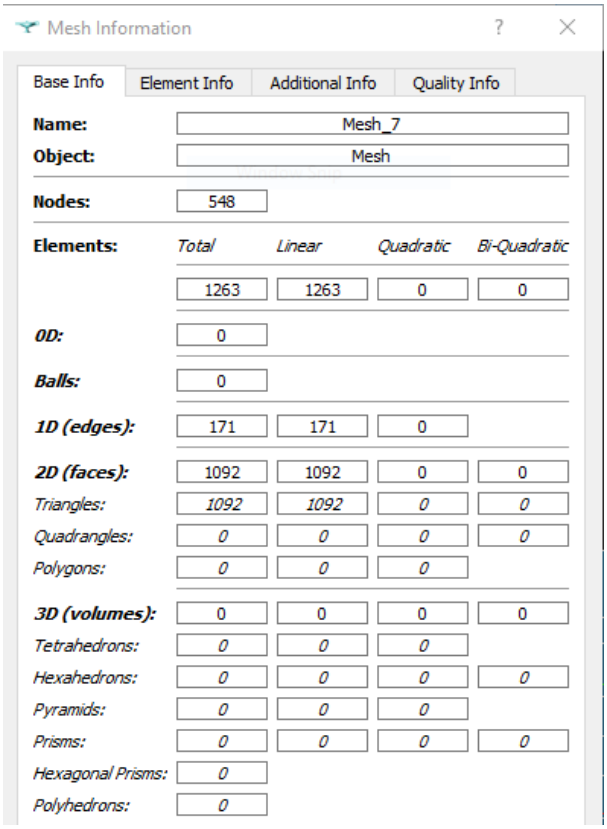
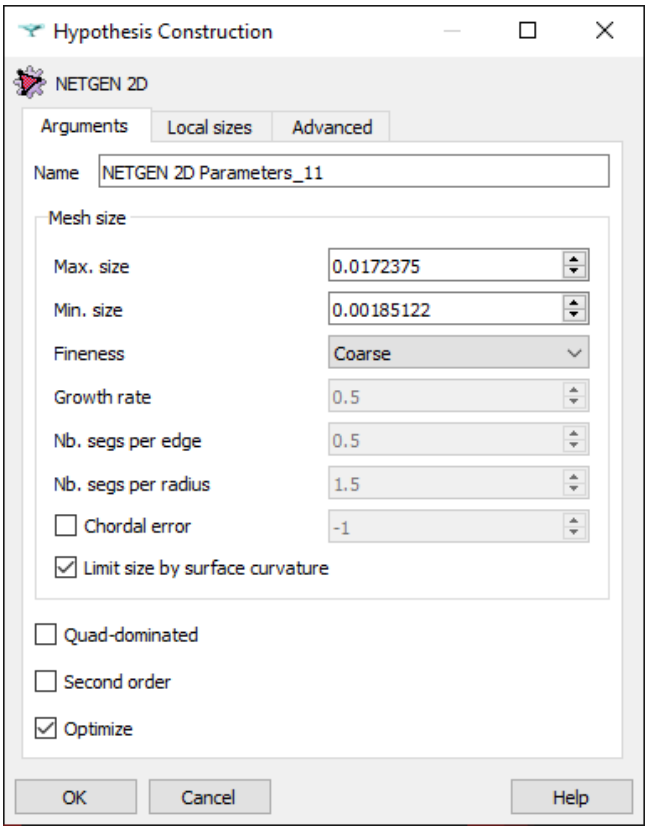


Figure B.2 *Settings screenshots in the Polytec data-acquisition software, part 2;*



(a)



(b)

Figure B.3 *Properties of the surface mesh used for the BEM computation; (a): mesh info; (b): construction settings;*

Appendix C Listings

Listing C.1 *Listing of the function prepare_polytec_data().*

```
function [Q,n_vecs,grids,V,f] = prepare_polytec_data(scannames,measures,...
n_sources_meas,save_data,check_scanpoints,check_index,meas_indents)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [Q,n_vecs,grids,V,f] = prepare_polytec_data(scannames,measures,...
%   n_sources_meas,save_data,check_scanpoints,check_index,meas_indents)
%
% function that extracts and prepares the measurements, in a way that the
% data can be handled by the acoustic simulation tools. It does so by
% extracting the data from the dive-exported '.mat' files and interpolating
% the data to querry points that lie on the surface of a box with the
% chosen measures. The box is created by calling the function 'create_box'.
% the data is being extracted by calling the function 'extract_and_interpolate()'.
% This function connects the two embedded functions and takes care of the
% correct function calls for each surface, so that every measured position
% gets assigned to the correct position on the box.
% the measurement of the surfaces follows the convention:
% - cover front: display -> up;             cable -> left
% - cover back:  display -> down;           cable -> left
% - frame down:  display -> to engineer;    cable -> down
% - frame up:    display -> to engineer;    cable -> up
% - frame left:  display -> to engineer;    cable -> left
% - frame right: display -> to wall;        cable -> left
%
% Inputs:
% scannames   ...   names of the '.mat' files of the dive-export, as string
%                   array. The order is: [front;back;down;up;left;right]
%                   The frame is set to have 0 velocity on all querry
%                   points and all frequencies, if only one or two strings
%                   are in scannames. If only one string is in scannames,
%                   the back cover is also set to be 0;
% measures   ...   measures of the device in order [height,width,depth]
% n_sources_meas   ...   number of measurement points for each surface in
%                        [n_height,n_width,n_depth];
% save_data   ...   optional bool. If 1, the function additionally saves
%                   the data to a .mat file
% check_scanpoints, check_index   ...   optional bool variables that allow
%                                       for plotting the scan positions and
%                                       indexing and normal vectors.
% meas_indents   ...   optional (n_measurements x 2) bool matrix,
%                      specifying if the surface has been measured to the
%                      edges (0) or with an indent of one mesh distance
%                      (1). The rows specify the surface, the first column
%                      specifies the indent of the longer axis, the second
%                      column specifies the indent of the shorter axis of
%                      the surface.
% Outputs:
% Q   ...   6-element cell array, containing positions for each surface in
%           (3 x n_Q) matrices.
% n_vecs   ...   6-element cell array, containing outwards-normal vectors
%                for each surface in (3 x n_Q) matrices.
% grids   ...   {2x6}-element cell array, containing 2D meshgrids of
%               positions for each surface
% V   ...   6-element cell array, containing velocity data for each point
%           in Q and frequency in f (n_f x n_Q) matrices.
% f   ...   frequency scale as (n_f x 1) matrix. Must be the same for each
%           surface.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if nargin < 7
  meas_indents = [0,0;0,0;1,0;1,0;1,0;1,0];
end
if nargin < 6
  check_index = 0;
end
if nargin < 5
  check_scanpoints = 0;
end
if nargin < 4
  save_data = 0;
end

% measures of the phone - approximate indent of measurement
height = measures(1);
width = measures(2);
depth = measures(3);
% indents of the interpolated querry points
indent_q = [0.00,0.00,0.00];
% create box with specified sources and outwards facing normal vectors
[Q,n_vecs,grids] = create_box(measures,n_sources_meas,...
```

```
        indent_q,[0,0,0],0);

  if length(scannames)<3
    frame_is_zero = 1; % if 1, assumes frame to be 0. if 0, takes measured data
  else
    frame_is_zero = 0;
  end
  %% front cover
  %%%%%%%%% set name of scan
  scanname = scannames{1};
  %%%%%%%%%
  %%%%%%%%% state, how many sources are measured for each column
  sources_per_column = n_sources_meas(2);
  %%%%%%%%%
  %%%%%%%%% state, if the measurement has indents
  indent_long = meas_indents(1,1);
  indent_short = meas_indents(1,2);
  corners_not_valid = 1;
  %%%%%%%%%
  %%%%%%%%% state, if Y has to be flipped
  flip_x = 0;
  flip_y = 0;
  %%%%%%%%% if not symmetrically around 0
  offset_y = 0;
  %%%%%%%%%
  %%%%%%%%%
  interpolate_left_out = 0;
  %%%%%%%%%
  [Vint, fex] = extract_and_interpolate(scanname,...
        sources_per_column,height,width,offset_y,grids{1,1},grids{2,1},flip_x,flip_y,...
        indent_long,indent_short,corners_not_valid,...
        interpolate_left_out, check_scanpoints);
  V{1} = Vint;
  f{1} = fex;

  % only if there is data for the frame provided, extract...
  % if not, set querry points to 0
  if length(scannames)>=2
    %% back cover
    %%%%%%%%% set name of scan
    scanname = scannames{2};
    %scanname = 'phone_scan_back_500mV_adhesive';
    %%%%%%%%%
    %%%%%%%%% state, how many sources are measured for each column
    sources_per_column = n_sources_meas(2);
    %%%%%%%%%
    %%%%%%%%% state, if the measurement has indents
    indent_long = meas_indents(2,1);
    indent_short = meas_indents(2,2);
    corners_not_valid = 1;
    %%%%%%%%%
    %%%%%%%%% state, if Y has to be flipped
    flip_x = 0;
    flip_y = 1;
    %%%%%%%%% if not symmetrically around 0
    offset_y = 0;
    %%%%%%%%%
    [Vint, fex] = extract_and_interpolate(scanname,...
          sources_per_column,height,width,offset_y,grids{1,2},grids{2,2},flip_x,flip_y,...
          indent_long,indent_short,corners_not_valid,...
          interpolate_left_out,check_scanpoints);
    V{2} = Vint;
    f{2} = fex;
    else
    V{2} = zeros(size(f{1},1),n_sources_meas(1)*n_sources_meas(2));
    f{2} = f{1};
  end

  % only if there is data for the frame provided, extract...
  % if not, set querry points to 0
  if length(scannames)>=3
    %% frame, down side
    %%%%%%%%% set name of scan
    scanname = scannames(3);
    %%%%%%%%%
    %%%%%%%%% state, how many sources are measured for each column
    sources_per_column = n_sources_meas(3);
    %%%%%%%%%
    %%%%%%%%% state, if the measurement has indents
    indent_long = meas_indents(3,1);
    indent_short = meas_indents(3,2);
    corners_not_valid = 0;
    %%%%%%%%%
    %%%%%%%%% state, if Y has to be flipped
    flip_x = 0;
    flip_y = 1;
    %%%%%%%%% if not symmetrically around 0
    offset_y = -depth/2;
    %%%%%%%%%
    [Vint, fex] = extract_and_interpolate(scanname,...
          sources_per_column,width,depth,offset_y,grids{1,3},grids{2,3},flip_x,flip_y,...
          indent_long,indent_short,corners_not_valid,...
          interpolate_left_out,check_scanpoints);
    if frame_is_zero == 1
      Vint = zeros(size(Vint));
```

```
    end
    V{3} = Vint;
    f{3} = fex;

    %% frame, up side
    %%%%%%%%% set name of scan
    scanname = scannames(4);
    %%%%%%%%%
    %%%%%%%%% state, how many sources are measured for each column
    sources_per_column = n_sources_meas(3);
    %%%%%%%%%
    %%%%%%%%% state, if the measurement has indents
    indent_long = meas_indents(4,1);
    indent_short = meas_indents(4,2);
    corners_not_valid = 0;
    %%%%%%%%%
    %%%%%%%%% state, if Y has to be flipped
    flip_x = 1;
    flip_y = 1;
    %%%%%%%%% if not symmetrically around 0
    offset_y = -depth/2;
    %%%%%%%%%
    [Vint, fex] = extract_and_interpolate(scanname,...
          sources_per_column,width,depth,offset_y,grids{1,4},grids{2,4},flip_x,flip_y,...
          indent_long,indent_short,corners_not_valid,...
          interpolate_left_out,check_scanpoints);
    if frame_is_zero == 1
      Vint = zeros(size(Vint));
    end
    V{4} = Vint;
    f{4} = fex;

    %% frame, left side
    %%%%%%%%% set name of scan
    scanname = scannames(5);
    %%%%%%%%%
    %%%%%%%%% state, how many sources are measured for each column
    sources_per_column = n_sources_meas(3);
    %%%%%%%%%
    %%%%%%%%% state, if the measurement has indents
    indent_long = meas_indents(5,1);
    indent_short = meas_indents(5,2);
    corners_not_valid = 0;
    %%%%%%%%%
    %%%%%%%%% state, if Y has to be flipped
    flip_x = 0;
    flip_y = 0;
    %%%%%%%%% if not symmetrically around 0
    offset_y = -depth/2;
    %%%%%%%%%
    [Vint, fex] = extract_and_interpolate(scanname,...
          sources_per_column,height,depth,offset_y,grids{1,5},grids{2,5},flip_x,flip_y,...
          indent_long,indent_short,corners_not_valid,...
          interpolate_left_out,check_scanpoints);
    if frame_is_zero == 1
      Vint = zeros(size(Vint));
    end
    V{5} = Vint;
    f{5} = fex;
    %% frame, right side
    %%%%%%%%% set name of scan
    scanname = scannames(6);
    %%%%%%%%%
    %%%%%%%%% state, how many sources are measured for each column
    sources_per_column = n_sources_meas(3);
    %%%%%%%%%
    %%%%%%%%% state, if the measurement has indents
    indent_long = meas_indents(6,1);
    indent_short = meas_indents(6,2);
    corners_not_valid = 0;
    %%%%%%%%%
    %%%%%%%%% state, if Y has to be flipped
    flip_x = 0;
    flip_y = 1;
    %%%%%%%%% if not symmetrically around 0
    offset_y = -depth/2;
    %%%%%%%%%
    [Vint, fex] = extract_and_interpolate(scanname,...
          sources_per_column,height,depth,offset_y,grids{1,6},grids{2,6},flip_x,flip_y,...
          indent_long,indent_short,corners_not_valid,...
          interpolate_left_out,check_scanpoints);
    if frame_is_zero == 1
      Vint = zeros(size(Vint));
    end
    V{6} = Vint;
    f{6} = fex;
  else
    V{3} = zeros(size(f{1},1),n_sources_meas(2)*n_sources_meas(3));
    V{4} = zeros(size(f{1},1),n_sources_meas(2)*n_sources_meas(3));
    V{5} = zeros(size(f{1},1),n_sources_meas(1)*n_sources_meas(3));
    V{6} = zeros(size(f{1},1),n_sources_meas(1)*n_sources_meas(3));
    f{3} = f{1};
    f{4} = f{1};
    f{5} = f{1};
    f{6} = f{1};
```

```
  end

  if save_data == 1
    % assign data to a struct
    data.Q = Q;
    data.n_sources = n_sources_meas;
    data.measures = measures;
    data.n_vecs = n_vecs;
    data.grids = grids;
    data.indents = indent_q;
    data.V = V;
    data.f = f{1};
    save('./data.mat','data')
  end
  try if any(diff(sum(cell2mat(f),1),1,2)>0)
    warning('not all measurements have same frequency range!')
  end
  catch
    warning('not all measurements have same frequency range!')
  end
  f = f{1};
  %%
  % check scanpoints and indexing
  if check_index == 1
    figure(101)
    sides = ["Front","Back","Down","Up","Left","Right"];
    cols = ["b","b","r","r","k","k"];
    for ii = 1:length(Q)
      scatter3(Q{ii}(1,:),Q{ii}(2,:),Q{ii}(3,:), ...
            5,cols(ii))
      hold on
      quiver3(Q{ii}(1,1),Q{ii}(2,1),Q{ii}(3,1), ...
            n_vecs{ii}(1,1)/100,n_vecs{ii}(2,1)/100,n_vecs{ii}(3,1)/100,...
            'Color',cols(ii))
      text(Q{ii}(1,floor(length(Q{ii})/2))+n_vecs{ii}(1,1)/200,...
            Q{ii}(2,floor(length(Q{ii})/2))+n_vecs{ii}(2,1)/200,...
            Q{ii}(3,floor(length(Q{ii})/2))+n_vecs{ii}(3,1)/200, ...
            sides(ii),'Color',cols(ii))
      text(Q{ii}(1,1)+n_vecs{ii}(1,1)/200,...
            Q{ii}(2,1)+n_vecs{ii}(2,1)/200,...
            Q{ii}(3,1)+n_vecs{ii}(3,1)/200, ...
            'start','Color',cols(ii))
      text(Q{ii}(1,end)+n_vecs{ii}(1,1)/200,...
            Q{ii}(2,end)+n_vecs{ii}(2,1)/200,...
            Q{ii}(3,end)+n_vecs{ii}(3,1)/200, ...
            'end','Color',cols(ii))
    end
    title('querry positions with indexing and normal vectors')
    grid on
    axis equal
  end
  end
```

Listing C.2 *Listing of the function extract_and_interpolate().*

```
function [Vint, f] = extract_and_interpolate(scanname, sources_per_column,...
size_x,size_y,offset_y,Xq,Yq,flip_x,flip_y,indent_x,indent_y,...
corners_not_valid,interpolate_left_out, check_scanpoints)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [Vint, f] = extract_and_interpolate(scanname, sources_per_column,...
%    size_x,size_y,offset_y,Xq,Yq,flip_x,flip_y,indent_x,indent_y,...
%    corners_not_valid,interpolate_left_out, check_scanpoints)
%
% Extracts data from polytec measurement, maps positions to fit the
% specified size and interpolates it to specified querry points.
% The mat-files containing the velocity and voltage data should be arranged
% in folders
% Inputs:
% scanname  ... name of the polytec scan
% sources_per_column    ... how many measured points are in each column
% size_x    ... size in x-dimension
% size_y    ... size in y-dimension
% offset_y  ... if the measured points should not be interpolated to querry
%               points that are symmetrically around 0: states the offset
%               in y-axis
% Xq,Yq     ... grids of querry points
% flip_x,flip_y    ...  flips the respective axis
% indent_x,indent_y ... takes 1, if the measurement of the respective axis
%                       has an indent. Assumes to be the indent of the same
%                       size as the mean distance between measured points
%                       in the respective direction. Assumes values of 0
%                       for the velocity at edge points.
% corners_not_valid ... takes 1, if the 4 measured points in the corners
%                       are not valid e.g. lie out of the object's
%                       dimensions
% check_scanpoints  ... if 1, plots a scatterplot of the mapped measurement
%                       positions, the position grid and the interpolated
%                       positions
%
% Outputs:
% Vint   ...   (n_f x n_Q) Matrix, containing velocity data for the all
%              positions on the surface in for each frequency f.
% f   ...   (n_f x 1) frequency vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% load data from relative folder positions
filename = "./Data/Velocity/"+scanname+"_abs_";
velocity_data_abs = load(filename).data2write;
filename = "./Data/Velocity/"+scanname+"_phase_";
velocity_data_phase = load(filename).data2write;
filename = "./Data/Voltage/"+scanname+"_abs_";
voltage_data_abs = load(filename).data2write;
filename = "./Data/Voltage/"+scanname+"_phase_";
voltage_data_phase = load(filename).data2write;

% extract frequency, complex velocities and source positions
% frequencies
f = velocity_data_abs(4:end,1);
% complex velocities and voltages
velocity = velocity_data_abs(4:end,2:end).*exp(1i*velocity_data_phase(4:end,2:end));
voltage = voltage_data_abs(4:end,2:end).*exp(1i*voltage_data_phase(4:end,2:end));
% velocity over voltage; multiply by -1 because Polytech has a default
% normal direction inwards
V = velocity./voltage;
% positions
pos_x = velocity_data_abs(2,2:end);
pos_y = velocity_data_abs(3,2:end);
% make grids valid for interpolation, by setting to mean x- and
% y-positions and insert edge positions into grids for interpolation
Xgrid = buffer(pos_x,sources_per_column);
Ygrid = buffer(pos_y,sources_per_column);

% take account for left out scanpoints by adding artificial ones
if interpolate_left_out
  thresholdx = abs(mean(mean(diff(Xgrid)))/2);
  thresholdy = abs(mean(mean(diff(Ygrid)))/2);
  while any(abs(diff(Xgrid))>thresholdx,'all') &&...
        any(abs(diff(Ygrid.').').')>thresholdy,'all')
    indx = find(abs(diff(Xgrid))>1E-4,1)+sources_per_column;
    pos_x = [pos_x(1:indx-1),pos_x(indx-1),pos_x(indx:end)];
    indy = find(abs(diff(Ygrid.').').')>1E-4,1)+sources_per_column;
    pos_y = [pos_y(1:indy-1),...
          pos_y(indy-sources_per_column),...
          pos_y(indy:end)];
    Xgrid = buffer(pos_x,sources_per_column);
    Ygrid = buffer(pos_y,sources_per_column);
    V = [V(:,1:indy-1),zeros(size(V,1),1),V(:,indy:end)];
    disp('attention: left-out scanpoints are set to zero before interpolation')
    %Vgrid = buffer(V(1,:),sources_per_column);
  end
end

sources_per_row = size(Xgrid,2);
```

```
  if indent_x == 1
    indent_x = abs(mean(diff(Xgrid(1,:))));
  end
  if indent_y == 1
    indent_y = abs(mean(diff(Ygrid(:,2))));
  end
  % take median for robustness outliers
  validXgrid = repmat(median(Xgrid,1),sources_per_column,1);
  validYgrid = repmat(median(Ygrid,2),1,sources_per_row);
  % scale positions to fit actual size
  kx = size_x/(max(validXgrid(:))-min(validXgrid(:))+2*indent_x);
  dx = size_x/2-kx*(max(validXgrid(:)+indent_x));
  validXgrid = validXgrid*kx+dx;
  ky = size_y/(max(validYgrid(:))-min(validYgrid(:))+2*indent_y);
  dy = (size_y/2+offset_y)-ky*(max(validYgrid(:))+indent_y);

  validYgrid = validYgrid*ky+dy;
  % take account for indents
  % add points in far distance, so that extrapolation does not force to 0
  if indent_x ~= 0
    factor = 10;
    validXgrid = [validXgrid(:,1)-factor*indent_x*kx,validXgrid,...
          validXgrid(:,end)+factor*indent_x*kx];
    validYgrid = [validYgrid(:,1),validYgrid,validYgrid(:,end)];
  end
  if indent_y ~= 0
    factor = 10;
    validXgrid = [validXgrid(1,:);validXgrid;validXgrid(end,:)];
    validYgrid = [validYgrid(1,:)+factor*indent_y*ky;validYgrid;...
          validYgrid(end,:)-factor*indent_y*ky];
  end
  % flip if needed
  if flip_y == 1
    validYgrid = flipud(validYgrid);
  end
  if flip_x == 1
    validXgrid = fliplr(validXgrid);
  end

  % interpolate...
  Vint = zeros(length(f),length(Xq(:)));
  for ii = 1:length(f)
    Vgrid = reshape(V(ii,:),sources_per_column,sources_per_row);
    if corners_not_valid == 1
      % set corner points to mean of next points, to avoid big
      % discontinuities
      Vgrid(1,1) = (Vgrid(1,2)+Vgrid(2,1))/2;
      Vgrid(sources_per_column,1) = (Vgrid(sources_per_column,2)+...
            Vgrid(sources_per_column-1,1))/2;
      Vgrid(1,sources_per_row) = (Vgrid(2,sources_per_row)+...
            Vgrid(1,sources_per_row-1))/2;
      Vgrid(sources_per_column,sources_per_row) = ...
            (Vgrid(sources_per_column-1,sources_per_row)+...
            Vgrid(sources_per_column,sources_per_row-1))/2;
    end
    % set far-away points to zero and let interp do the rest...
    if indent_x ~= 0
      Vgrid = [zeros(size(Vgrid,1),1),Vgrid,...
            zeros(size(Vgrid,1),1)];
    end
    if indent_y ~= 0
      Vgrid = [zeros(1,size(Vgrid,2));Vgrid;...
            zeros(1,size(Vgrid,2))];
    end
    Vgridint = interp2(validXgrid,validYgrid,Vgrid,Xq,Yq,'makima',0);
    Vint(ii,:) = Vgridint(:);
  end
  if check_scanpoints == 1
    figure
    scatter3(pos_x*kx+dx,pos_y*ky+dy,zeros(size(pos_x)),10,'b')
    scatter(Xgrid(:)*kx+dx,Ygrid(:)*ky+dy,10,'b')
    hold on
    text(pos_x*kx+dx+0.001,pos_y*ky+dy+0.001,zeros(size(pos_x)),string(1:length(pos_x(:))),'Color','b')
    scatter3(validXgrid(:),validYgrid(:),-0.05*ones(size(validXgrid(:))),10,'r')

     ↪ text(validXgrid(:)+0.001,validYgrid(:)+0.001,-0.05*ones(size(validXgrid(:))),string(1:length(validXgrid(:))),'Color','r')
    scatter3(Xq(:),Yq(:),-0.1*ones(size(Xq(:))),10,'k')
    text(Xq(:)+0.001,Yq(:)+0.001,-0.1*ones(size(Xq(:))),string(1:length(Xq(:))),'Color','k')
    grid minor
    axis equal
    legend("original","valid","querry")
  end
  end
```

Listing C.3 *Listing of the function create_box().*

```
function [pos,n_vecs,grids,dA] = create_box(size_box,n_srcs,ind,d,...
centerandcalcdA,plotbox)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [pos,n_vecs,grids,dA] = create_box(size_box,n_srcs,ind,d,...
%   centerandcalcdA,plotbox)
%
% creates a box with equally spaced positions on each surface and assigns
% normalvectors in outward direction. With centerandcalcdA, this function
% assumes the equally spaced positions as nodes of quadrilateral surface
% elements. It outputs then also the area of each surface element and
% sets the positions to the mid point of the elements.
%
% Inputs:
% size_box   ...   size of the box as 1x3 matrix for (x,y,z)
% n_srcs   ...   number of sources as 1x3 matrix for (x,y,z)
% ind   ...   indents of sourcepositions as 1x3 matrix for (x,y,z)
% d   ...   retraction distance, to make the box smaller. 1x3 matrix for (x,y,z)
% centerandcalcdA   ...   (optional) assumes the positions as nodes of
%                         quadrilateral elements, sets the output
%                         positions to the center of the elements and
%                         calculates the area of each element.
% plotbox   ...   (optional) scatter plot of the positions
%
% Outputs:
% Qe   ...   1x6-cell array with source positions for: top,bottom,left,
%            right,up,down side. (front,back,down,up,left,right on the phone)
% n_vecs_Qe   ...   1x6-cell array containing respective normal-vectors
% grids   ...   2x3-cell array containing the grids of each surface of the
%               box. the first row of cells contains the longer axes,
%               the second row contains the shorter axes
% dA   ...   if centerandcalcdA = 1, outputs the area for each
%            quadrilateral element in a 1x6-cell array
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

size_x = size_box(1);
size_y = size_box(2);
size_z = size_box(3);

n_x = n_srcs(1);
n_y = n_srcs(2);
n_z = n_srcs(3);

d_x = d(1);
d_y = d(2);
d_z = d(3);

ind_x = ind(1);
ind_y = ind(2);
ind_z = ind(3);

pos = cell(1,6);
n_vecs = cell(1,6);
grids = cell(2,6);

pos_x = linspace(-size_x/2+d_x+ind_x,size_x/2-d_x-ind_x,n_x);
pos_y = linspace(-size_y/2+d_y+ind_y,size_y/2-d_y-ind_y,n_y);
pos_z = linspace(-size_z+d_z+ind_z,0-d_z-ind_z,n_z);

% front, back
[X,Y] = meshgrid(pos_x,pos_y);
Y = flipud(Y);

if centerandcalcdA == 1
  % interpolate to mid-values and calculate dA values
  Xm = ([zeros(size(X,1),1),X]+[X,zeros(size(X,1),1)])/2;
  Xm = Xm(2:end,2:end-1);
  Ym = ([zeros(1,size(Y,2));Y]+[Y;zeros(1,size(Y,2))])/2;
  Ym = Ym(2:end-1,2:end);
  dX = diff(X, 1, 2);
  dX(end,:) = [];
  dY = diff(Y, 1, 1);
  dY(:,end) = [];
  dA_grid = abs(dX.*dY);
  dA{1} = dA_grid(:);
  dA{2} = dA_grid(:);
  X = Xm;
  Y = Ym;
end
pos{1} = [X(:),Y(:),zeros(size(X(:)))-d_z].';
pos{2} = [X(:),Y(:),-size_z*ones(size(X(:)))+d_z].';
n_vecs{1} = [0;0;1].*ones(size(pos{1}));
n_vecs{2} = -[0;0;1].*ones(size(pos{2}));
grids{1,1} = X;
grids{2,1} = Y;
grids{1,2} = X;
grids{2,2} = Y;
clear X Y Xm Ym

```

```
  % down, up
  [Y,Z] = meshgrid(pos_y,pos_z);
  Z = flipud(Z);
  if centerandcalcdA == 1
    % interpolate to mid-values and calculate dA values
    Ym = ([zeros(size(Y,1),1),Y]+[Y,zeros(size(Y,1),1)])/2;
    Ym = Ym(2:end,2:end-1);
    Zm = ([zeros(1,size(Z,2));Z]+[Z;zeros(1,size(Z,2))])/2;
    Zm = Zm(2:end-1,2:end);
    dY = diff(Y, 1, 2);
    dY(end,:) = [];
    dZ = diff(Z, 1, 1);
    dZ(:,end) = [];
    dA_grid = abs(dY.*dZ);
    dA{3} = dA_grid(:);
    dA{4} = dA_grid(:);
    Y = Ym;
    Z = Zm;
  end
  pos{3} = [(-size_x/2+d_x)*ones(size(Y(:))),Y(:),Z(:)].';
  pos{4} = [(+size_x/2-d_x)*ones(size(Y(:))),Y(:),Z(:)].';
  n_vecs{3} = -[1;0;0].*ones(size(pos{3}));
  n_vecs{4} = [1;0;0].*ones(size(pos{4}));
  grids{1,3} = Y;
  grids{2,3} = Z;
  grids{1,4} = Y;
  grids{2,4} = Z;
  clear Y Z Ym Zm

  % left, right
  [X,Z] = meshgrid(pos_x,pos_z);
  Z = flipud(Z);
  if centerandcalcdA == 1
    % interpolate to mid-values and calculate dA values
    Xm = ([zeros(size(X,1),1),X]+[X,zeros(size(X,1),1)])/2;
    Xm = Xm(2:end,2:end-1);
    Zm = ([zeros(1,size(Z,2));Z]+[Z;zeros(1,size(Z,2))])/2;
    Zm = Zm(2:end-1,2:end);
    dX = diff(X, 1, 2);
    dX(end,:) = [];
    dZ = diff(Z, 1, 1);
    dZ(:,end) = [];
    dA_grid = abs(dX.*dZ);
    dA{5} = dA_grid(:);
    dA{6} = dA_grid(:);
    X = Xm;
    Z = Zm;
  end
  pos{5} = [X(:),(size_y/2-d_y)*ones(size(X(:))),Z(:)].';
  pos{6} = [X(:),(-size_y/2+d_y)*ones(size(X(:))),Z(:)].';
  n_vecs{5} = [0;1;0].*ones(size(pos{5}));
  n_vecs{6} = -[0;1;0].*ones(size(pos{6}));
  grids{1,5} = X;
  grids{2,5} = Z;
  grids{1,6} = X;
  grids{2,6} = Z;
  clear X Z Xm Zm

  if nargin >= 6
    if plotbox == 1
      figure
      for ii = 1:length(pos)
        scatter3(pos{ii}(1,:),pos{ii}(2,:),pos{ii}(3,:),'b','.')
        hold on
        axis equal
      end
    end
  end
  end
```

Listing C.4 *Listing of the function interpolate_box().*

```
function [V_new] = interpolate_box(grids_old,grids_new,V_old,extrapdist)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [V_new] = interpolate_box(grids_old,grids_new,V_old,extrapdist)
%
% interpolates the velocities of from the old positions to new positions on
% a box of the same size. Extrapolates the surface functions, if new
% positions lie outside the old ones.
%
% Inputs:
% grids_old   ...   2x3-cell array containing the grids of each surface of
%                   the box, for the old positions. The first row of cells
%                   contains the longer axes, the second row contains the
%                   shorter axes.
% grids_new   ...   2x3-cell array containing the grids of each surface of
%                   the box, for the new positions. The first row of cells
%                   contains the longer axes, the second row contains the
%                   shorter axes.
% V_old   ...    cell array, containing velocity data for the old
%                positions, for each frequency f. Each cell contains a
%                (n_f x n_Q) matrix. The cell array can contain 1-6
%                elements, containing the values for the surfaces in order:
%                front,back,down,up,left,right. If only n<6 entries are
%                provided, the output will conatin 0-values for the
%                non-provided surfaces.
% extrapdist   ...   (optional) parameter that has an influence on the
%                    extrapolation. Extrapolation is done inserting
%                    far-distant positions with a velocity of 0. extrapdist
%                    is an integer specifying how far (in times of a mean
%                    distance) this positions are apart from the edge
%                    positions. By default, extrapdist=4
%
% Outputs:
% V_new   ...   6-element cell array, containing velocity data for the new
%               positions, for each frequency f. Each cell contains a
%               (n_f x n_Q) matrix. The surface order is: front,back,down,
%               up,left,right.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

V_new = cell(1,6);
if nargin < 4
  extrapdist = 0;
end
for ii = 1:length(V_old)
  dx = extrapdist*mean(mean(abs(diff(grids_old{1,ii},1,2))));
  dy = extrapdist*mean(mean(abs(diff(grids_old{2,ii},1,1))));
  nx = size(grids_old{1,ii},1);
  ny = size(grids_old{2,ii},2);
  minx = min(min(grids_old{1,ii}));
  maxx = max(max(grids_old{1,ii}));
  miny = min(min(grids_old{2,ii}));
  maxy = max(max(grids_old{2,ii}));
  for iii = 1:size(V_old{ii},1)
    v_old = V_old{ii}(iii,:);
    v_old = reshape(v_old,size(grids_old{1,ii}));
    if extrapdist == 0
      X = grids_old{1,ii};
      Y = grids_old{2,ii};
    else
      % add far-distant zeros for extrapolation
      X = [ones(nx+2,1)*(-dx+minx),...
          [grids_old{1,ii}(1,:);grids_old{1,ii};grids_old{1,ii}(end,:)],...
      ones(nx+2,1)*(dx+maxx)];
      Y = [ones(1,ny+2)*(dy+maxy);...
          [grids_old{2,ii}(:,1),grids_old{2,ii},grids_old{2,ii}(:,end)];...
      ones(1,ny+2)*(-dy+miny)];
      v_old = [zeros(1,size(v_old,2)+2);
          [zeros(size(v_old,1),1),v_old,zeros(size(v_old,1),1)];
      zeros(1,size(v_old,2)+2)];
      end
      vn = interp2(X,Y,v_old,grids_new{1,ii},grids_new{2,ii},"makima",0);
      V_new{ii}(iii,:) = vn(:);
  end
end
end
```

Listing C.5 *Listing of the function evaluate_G_dGdn_d2Gdn2(). The function is based on Zotter's and Pomberger's evaluate_G_dGdn().*

```
function [G,dGdn,d2Gdn2,cosphi0,r]=evaluate_G_dGdn_d2Gdn2(Xsrc,Xeval,k,n_vec_src, n_vec_eval)
% [G,dGdn,d2Gdn2,cosphi0,r]=evaluate_G_dGdn_d2Gdn2(Xsrc,Xeval,k,n_vec_src, n_vec_eval)
% evaluates Green's function for the 3D Helmholtz equation in free space
% as well as it's first and (optionally also) second derivative along
% arbitrary direction vectors for source and receiver.
%
% The first derivative is being computed as 'derivative of the source' and
% the second as 'derivative of the receiver'. As the radius is
% r=sqrt((x-x0)^2+(y-y0^2)+(z-z0)^2), n_vec_src specifies the direction of
% derivation for x0,y0,z0 and n_vec_eval for x,y,z
%
% Inputs:
% Xsrc          ...   Qx3, containing the source positions
% Xeval         ...   Lx3, contatining the points of observation
% k             ...   wavenumber = omega/c
% n_vec_src     ...   Qx3, normal vectors of all sources
% n_vec_eval    ...   Lx3, normal vectors of all receivers (second
%                     derivative direction)
%
% Outputs:
% G             ...   QxL Green's functions
% dGdn          ...   QxL first derivatives of Green's function, evaluated
%                     along the vectors n_vec_src
% d2Gdn2        ...   QxL second derivatives, the first evaluated along
%                     n_vec_src and the second
% cosphi0       ...   QxL source angles
% r             ...   QxL radii between sources and receivers
%
% Franz Zotter, Hannes Pomberger, 2015
% Edit by Patrick Heidegger 2021

n_src = size(Xsrc, 1);
n_eval = size(Xeval, 1);

% Components of the position difference vector:
DX = repmat(Xeval(:,1),1,n_src)-repmat(Xsrc(:,1).',n_eval,1);
DY = repmat(Xeval(:,2),1,n_src)-repmat(Xsrc(:,2).',n_eval,1);
DZ = repmat(Xeval(:,3),1,n_src)-repmat(Xsrc(:,3).',n_eval,1);
% vectorized into L*Q x 3
R = [DX(:) DY(:) DZ(:)];
clear DX DY DZ

% distances between all L points of observation and all source points
r = sqrt(sum((R).^2,2));

% calculate G, dGdr and d2Gdr2
G = exp(-1i*k.*r)./(4*pi.*r);
if nargin > 3
  dGdr = -(1i*k + 1./r).*G;
  d2Gdr2 = (-(k^2) + (2*1i*k)./r + 2./(r.^2)).*G;
  % calculate dGdn...
  % normalizing the L*Q x 1 position difference vectors by the distance
  eR = R./repmat(r,1,3); % normalize coordinates to vectorlength 1
  % take the vectorlength only along the normalvector axis
  nvsx = repmat(n_vec_src(:,1).',n_eval,1);
  nvsy = repmat(n_vec_src(:,2).',n_eval,1);
  nvsz = repmat(n_vec_src(:,3).',n_eval,1);
  % vectorize
  nvs = [nvsx(:),nvsy(:),nvsz(:)];
  % compute angle
  cosphi0 = dot(nvs,eR,2);
  dGdn = -cosphi0.*dGdr;
  if nargin > 4
    % calculate d2Gdn2...
    % find directivity angles between normalvectors as well as radial
    % directions
    cosphi = dot(repmat(n_vec_eval,n_src,1),eR,2);
    coseta = dot(repmat(n_vec_eval,n_src,1),nvs,2);
    d2Gdn2 = -coseta./r.*dGdr + cosphi.*cosphi0.*(dGdr./r - d2Gdr2);
    d2Gdn2 = reshape(d2Gdn2.',n_eval, n_src).';
  end
  % reshape to (Q x L) matrix
  dGdn = reshape(dGdn.',n_eval, n_src).';
  r = reshape(r.',n_eval, n_src).';
  cosphi0 = reshape(cosphi0.',n_eval, n_src).';
end
G = reshape(G.',n_eval, n_src).';
end
```

Listing C.6 *Listing of the function defineReceivers().*

```
function [P] = defineReceivers(r,n_rec_hor,n_rec_vert,mode,show,dirspan)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [P] = defineReceivers(r,n_rec_hor,n_rec_vert,mode,show,dirspan)
%
% Creates a set of positions (sphere, hemisphere, a single point with
% surrounding positions aligned as a part of a sphere, or a cross-section).
%
% Inputs:
% r   ...   radius of the sphere (assuming the sphere is centered to the
%           origin) as a scalar
% n_rec_hor   ...   number of receiver positions along a horizontal ring as
%                   a scalar
% n_rec_vert   ...   number of horizontal rings as a scalar
% mode   ...   can be either:
%              'dir'   ...   either a single position at [0;0;r] or
%                            multiple receivers aligned around [0;0;r]
%              'omni'   ...   a full sphere
%              'hemi'   ...   hemisphere in z>=0
%              'fieldx'   ...   creates n_rec_hor points in x and
%                               n_rec_vert points in z, at y=0 and for a
%                               region of -r to r
%              'fieldy'   ...   creates n_rec_hor points in y and
%                               n_rec_vert points in z, at x=0 and for a
%                               region of -r to r
% show   ...   if 1 plots a scatter plot to show the receivers. Adds a red dot
%              for the origin
% dirspan   ...   (optional) the directional span of the receivers in
%                 degree. Allows for creating only receivers on a part of a
%                 sphere or hemisphere. Only working for mode='dir'.
%
% Outputs:
% P   ...   (3 x n_rec) matrix containing the three-dimensional coordinates
%           of all receivers
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if strcmp(mode,'dir')
  if dirspan == 0
    P = [0;0;r];
    return
  end
  if nargin < 6
    dirspan = 0;
  else
    dirspan = dirspan*pi/180;
  end
  phi = 0:1/n_rec_hor*pi:pi-pi/n_rec_hor;
  phi_mtx = repmat(phi.',1,n_rec_vert);
  theta = linspace(pi/2-dirspan,pi/2+dirspan, n_rec_vert);
  theta_mtx = repmat(theta,n_rec_hor,1);
elseif strcmp(mode,'omni')
  theta = linspace(-pi/2,pi/2, n_rec_vert);
  theta_mtx = repmat(theta,n_rec_hor,1);
  phi = linspace(-pi,pi,n_rec_hor);
  phi_mtx = repmat(phi.',1,n_rec_vert);
elseif strcmp(mode,'hemi')
  theta = linspace(0,pi, n_rec_vert);
  theta_mtx = repmat(theta,n_rec_hor,1);
  phi = linspace(0,pi,n_rec_hor);
  phi_mtx = repmat(phi.',1,n_rec_vert);
elseif strcmp(mode,'fieldx')
  px = linspace(-r,r,n_rec_hor);
  pz = linspace(-r,r,n_rec_vert);
  [Px,Pz] = meshgrid(px,pz);
  Pz = flipud(Pz);
  P = [Px(:),zeros(size(Px(:))),Pz(:)].';
  if nargin >= 5 && show == 1
    figure
    scatter3(P(1,:),P(2,:),P(3,:))
    hold on
    scatter3(0,0,0)
    axis equal
  end
  return
elseif strcmp(mode,'fieldy')
  py = linspace(-r,r,n_rec_hor);
  pz = linspace(-r,r,n_rec_vert);
  [Py,Pz] = meshgrid(py,pz);
  Pz = flipud(Pz);
  P = [zeros(size(Py(:))),Py(:),Pz(:)].';
  if nargin >= 5 && show == 1
    figure
    scatter3(P(1,:),P(2,:),P(3,:))
    hold on
    scatter3(0,0,0)
    axis equal
  end
  return
else
  error('unknown mode!')
```

```
end
x = cos(phi_mtx).*cos(theta_mtx);
y = sin(phi_mtx).*cos(theta_mtx);
z = sin(theta_mtx);
P = r.*[x(:),y(:),z(:)].';
if nargin >= 5 && show == 1
  figure
  scatter3(P(1,:),P(2,:),P(3,:))
  hold on
  scatter3(0,0,0)
  axis equal
  xlabel('X in m', 'Interpreter', 'Latex')
  ylabel('Y in m', 'Interpreter', 'Latex')
  zlabel('Z in m', 'Interpreter', 'Latex')
end
end
```

Listing C.7 *Listing of the function calcESM() that computes the half-space sound radiation with the ESM approach.*

```
function [p] = calcESM(P,Q,n_vecs,V,f,c,rho,measures,n_sources)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [p,q] = calcESM_full(P,Q,n_vecs,V,f,c,rho,measures)
%
% computes the sound pressure for a half-space simulation using the
% equivalent source method. The equivalent sources are
% aligned as a single layer containing monopoles. The retraction distance
% is computed as the mean of the grid distance in x and y.
%
% Inputs:
% P   ...   (3 x n_rec) matrix, containing the coordinates of all
%           receiver positions
% Q   ...   (2-6)-element cell array, containing the source positions for all
%           surfaces. Only the upper (first entry) and lower (second
%           entry) are needed for full-space simulation.
% n_vecs   ...   (1-6)-element cell array containing the normal vectors of
%                measurement source positions. Only one entry (upper
%                surface) is needed
% V   ...   (2-6)-element cell array, containing velocity data for each point
%           in Q and frequency in f (n_f x n_Q) matrices.
% f   ...   frequency scale as (n_f x 1) matrix
% c   ...   speed of sound
% rho   ...   air density
% measures   ...   measures of the device in order [height,width,depth]
% n_sources   ...   number of positions for each surface in
%                   [n_height,n_width,n_depth] (only the first two entries
%                   are needed;
%
% Outputs:
% p   ...   (n_f x n_rec) matrix containing the results for each frequency
%           and receiver.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if length(f) ~= size(V{1},1)
  error('V must have the same number of entries as f!')
end

% extract values of first surface
Q = Q{1};
V = V{1};
n_vecs = n_vecs{1};

% compute equivalent source positions. Set layer distance to ~0.5 times the
% grid distance
griddist = measures./n_sources;
d = (griddist(1)+griddist(2))/2;

Qe = Q-repmat([0;0;0.5*d],1,size(Q,2));
n_vec_Qe = n_vecs;

% initialize sound pressure
p = zeros(length(f),size(P,2));
% loop through frequencies
for ii = 1:length(f)
  disp(['Calculating frequency response: f=', num2str(f(ii))]);
  f_curr = f(ii);
  w = 2*pi*f(ii);
  v = V(f==f_curr,:).';

  % green's functions to front plate
  [a,dGdn_front] = ...
    evaluate_G_dGdn_d2Gdn2(Qe.',Q.',w/c,n_vec_Qe.');

  % evaluate source strengths
  q = (-1i*w*rho)*(dGdn_front.'\v);

  % calculate resulting sound pressures for all points in P
  [G,dGdn] = evaluate_G_dGdn_d2Gdn2(Qe.', P.',w/c,n_vec_Qe.');
  p(ii,:) = G.'*q;
end
end
```

Listing C.8 *Listing of the function calcESM_full() that computes the full-space sound radiation with the ESM apprach.*

```
function [p] = calcESM_full(P,Q,n_vecs,V,f,c,rho,measures)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [p,q] = calcESM_full(P,Q,n_vecs,V,f,c,rho,measures)
%
% computes the sound pressure using the equivalent source method,
% exploiting parallel surfaces of a thin cuboid. The equivalent sources are
% aligned as a single layer, containing monopoles to emulate the
% even-symmetric and dipoles to emulate the odd-symmetric sound pressure.
%
%
% Inputs:
% P   ...   (3 x n_rec) matrix, containing the coordinates of all
%           receiver positions
% Q   ...   (2-6)-element cell array, containing the source positions for all
%           surfaces. Only the upper (first entry) and lower (second
%           entry) are needed for full-space simulation.
% n_vecs   ...   (1-6)-element cell array containing the normal vectors of
%                measurement source positions. Only one entry (upper
%                surface) is needed
% V   ...   (2-6)-element cell array, containing velocity data for each point
%           in Q and frequency in f (n_f x n_Q) matrices.
% f   ...   frequency scale as (n_f x 1) matrix
% c   ...   speed of sound
% rho   ...   air density
% measures   ...   measures of the device in order [height,width,depth]
%
% Outputs:
% p   ...   (n_f x n_rec) matrix containing the results for each frequency
%           and receiver.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if length(f) ~= size(V{1},1)
  error('V must have the same number of entries as f!')
end

% compute equivalent source positions
Qe = Q{1}-repmat([0;0;measures(3)/2],1,size(Q{1},2));
n_vec_Qe = n_vecs{1};

% initialize sound pressure
p = zeros(length(f),size(P,2));
% loop through frequencies
for ii = 1:length(f)
  disp(['Calculating frequency response: f=', num2str(f(ii))]);
  f_curr = f(ii);
  w = 2*pi*f(ii);
  v = {V{1}(f==f_curr,:).',V{2}(f==f_curr,:).',...
      V{3}(f==f_curr,:).',V{4}(f==f_curr,:).',...
      V{5}(f==f_curr,:).',V{6}(f==f_curr,:).'};
  % split into in-phase and contra-phase part of velocities of front and
  % back surface
  v_inphase= (v{1}+v{2})/2;
  v_contraphase = (v{1}-v{2})/2;

  % green's functions to front plate
  [a,dGdn_front,d2Gdn2_front] = ...
    evaluate_G_dGdn_d2Gdn2(Qe.',Q{1}.',w/c,n_vec_Qe.',n_vecs{1}.');

  q_monop = (-1i*w*rho)*(dGdn_front.'\v_inphase);
  q_dip = (-1i*w*rho)*(d2Gdn2_front.'\v_contraphase);

  % calculate resulting sound pressures for all points in P
  [G,dGdn] = evaluate_G_dGdn_d2Gdn2(Qe.', P.',w/c,n_vec_Qe.');
  p(ii,:) = G.'*q_monop-dGdn.'*q_dip;
end
end
```

Listing C.9 *Listing of the function calcRayleigh() that computes the half-space sound radiation using the Rayleigh I integral.*

```
function [p] = calcRayleigh(P,V,f,c,rho,grids,measures,n_sources)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [p] = calcRayleigh(P,V,f,c,rho,grids,measures,n_sources)
%
% Calculation of the Rayleigh I integral (with neumann boundary conditions),
% for each frequency f.
%
% Inputs:
% P   ...   (3 x n_rec) matrix containing the three-dimensional coordinates
%           of all receiver positions
% V   ...   cell array with at least 1 cells, containing the velocity data
%           for the upper (first entry) surface in (n_f x n_Q) Matrix form,
%           containing velocity data for the all positions on the surface
%           in for each frequency f.
% f   ...   (n_f x 1) frequency vector
% c   ...   speed of sound
% rho   ...   air density
% grids   ...   {2x1}-{2x6}-element cell array, containing 2D meshgrids of
%               positions for each surface (only the first column is
%               needed)
% measures   ...   measures of the device in order [height,width,depth]
% n_sources   ...   number of measurement points for each surface in
%                   [n_height,n_width,n_depth]
%
% Outputs:
% p   ...   (n_f x n_rec) matrix containing the results for each frequency
%           and receiver.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if length(f) ~= size(V{1},1)
  error('V must have the same number of entries as f!')
end

% set source positions to the center of nodes and calculate dA elements.
[Q,n_vecs,gridsint,dA] = create_box(measures,n_sources,...
[0,0,0],[0,0,0],1,0);
% only use values on the upper surface of the box
n_vecs = n_vecs{1};
Q = Q{1};
dA = dA{1};

% interpolate velocities of the upper surface
Vint = interpolate_box(grids(:,1),gridsint(:,1),V(1),0);
V = Vint;
clear gridsint Vint

% initialize p
p = zeros(length(f),size(P,2));
for ii = 1:length(f)
  disp(['Calculating frequency response: f=', num2str(f(ii))]);
  w = 2*pi*f(ii);
  v = V{1}(f==f(ii), :).';

  % compute Green's functions
  G = evaluate_G_dGdn_d2Gdn2(Q.',P.',w/c,n_vecs.');

  % compute sound pressures in the field
  p(ii,:) = -2*1i*w*rho*G.'*(v.*dA);
end
```

Listing C.10 *Listing of the function calcRayleigh_full() that computes the full-space sound radiation using the combined-Rayleigh formulation.*

```
function [p] = calcRayleigh_full(P,V,f,c,rho,grids,measures,n_sources,mode,tol)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [p] = calcRayleigh_full(P,V,f,c,rho,grids,measures,n_sources,mode,tol)
%
% Computes the combined Rayleigh integral, using front- and back-side
% velocities of two parallel (ideally overlaying at z=0) surfaces.
% Comprises three different modes for computation of the odd-symmetric
% sound pressure on the surfaces
%
% Inputs:
% P   ...   (3 x n_rec) matrix containing the three-dimensional coordinates
%           of all receiver positions
% V   ...   cell array with at least 2 cells, containing the velocity data
%           for the upper (first entry) and lower (second entry) surfaces
%           in (n_f x n_Q) Matrix form, containing velocity data for the all
%           positions on the surface in for each frequency f.
% f   ...   (n_f x 1) frequency vector
% c   ...   speed of sound
% rho   ...   air density
% grids   ...   {2x2}-{2x6}-element cell array, containing 2D meshgrids of
%               positions for each surface (only the first two are used
% measures   ...   measures of the device in order [height,width,depth]
% n_sources   ...   number of measurement points for each surface in
%                   [n_height,n_width,n_depth]
% mode   ...   specifies the kind of zero padding / wavefield
%              extrapolation.
%              mode='' ...  no zero padding / wavefield extrapolation
%              (not recommended)
%              mode='lpbp' ...  linear predictive border padding
%              mode='zp' ...  zero padding
%              mode='azp' ...  adaptive zero padding. Switches the
%              amount of zeros at 400 Hz. seems to yield the best results
%              (recommended)
% tol   ...   if specified, the function uses tol as a lower bound for kz
%             and does not regularize using Pagavino's integral.
%
% Outputs:
% p   ...   (n_f x n_rec) matrix containing the results for each frequency
%           and receiver.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if length(f) ~= size(V{1},1)
  error('V must have the same number of entries as f!')
end

% set adaptive zero padding as default
if nargin < 9
  mode = 'azp';
end

% create a new box with interpolated positions so that every position is in
% the center of a quadrilateral element...

% assure an odd number of nodes in each dimension, so that the number of
% source positions will be even (needed for mode lpbp)
if strcmp(mode,'lpbp')
  n_sources = ceil(n_sources/2)*2-1;
end
% set source positions to the center of nodes and calculate dA elements.
[Q,n_vecs,gridsint,dA] = create_box(measures,n_sources,...
[0,0,0],[0,0,0],1,0);

% we only need the coordinates of the upper surface
Q = Q{1};
% interpolate velocities to the newly assigned positions
Vint = interpolate_box(grids(:,1:length(V)),gridsint(:,1:length(V)),V,0);
V = Vint;
grids = gridsint;
clear gridsint Vint

% extract the number of bins in both dimensions and set the desired
% size for wavefield extrapolation or zero padding
lenx = size(grids{1,1},2);
leny = size(grids{1,1},1);
if strcmp(mode,'')
  lenfftx = lenx;
  lenffty = leny;
end

if strcmp(mode,"zp") || strcmp(mode,"azp")
  padfactor = 5; % times the original size added for wavefield extrapolation
  lenfftx = 2^(nextpow2(lenx*padfactor));
  lenffty = 2^(nextpow2(leny*padfactor));
  % window for zero padding
  winx = hann(lenx).';
  winy = hann(leny).';
  win = winy'* winx;
```

```
  % apparently, no window works better.....
  win = ones(size(win));
  % pre assign zero-padded function
  v_contpad = zeros(lenffty,lenfftx);
  % compute correction factor
  nowin = v_contpad;
  nowin(1:leny,1:lenx) = 1;
  winc = nowin;
  winc(1:leny,1:lenx) = win;
  corrfac = rms(nowin,'all')/rms(winc,'all');
end

if strcmp(mode,"lpbp")
  % for linear-predictive border padding as wavefield extrapolation...
  % create new grids with 15 times the size
  padfactor = 1; % times the original size added for wavefield extrapolation
  lenpadx = leny*padfactor;
  lenpady = lenx*padfactor;
  lenfftx = lenx+2*lenpadx;
  lenffty = leny+2*lenpady;
  % determine the extrapolation size so that the extrapolated field is
  % larger than the zero-padded one
  extra_size = ceil(max((lenx+lenpadx*padfactor*2)./[n_sources(1),n_sources(2)]));
  [Qpad,n_vecspad,gridspad] = create_box(extra_size*measures,...
  (extra_size*padfactor+1)*n_sources,[0,0,0],...
  [0,0,0],1,0);
  % we only need the grids for the front side
  gridspad = gridspad(1:2,1);
  clear Qpad n_vecspad

  % create tukey window for extrapolated part
  mididx = [floor(lenffty/2+1),floor(lenfftx/2+1)];
  frac = 1/2;
  winx = create_window(-mididx(2)+(1:lenfftx),...
  -floor(lenx/2),floor(lenx/2),frac,1);
  winy = create_window(-mididx(1)+(1:lenffty),...
  -floor(leny/2),floor(leny/2),frac,1);
  win = winy'*winx;
  % energy window correction so that the energy is equal to the
  % zero-padded, non-windowed signal
  nowin = zeros(lenffty,lenfftx);
  nowin(lenpady+1:lenpady+leny,lenpadx+1:lenpadx+lenx) = 1;
  corrfac = rms(nowin,'all')/rms(win,'all');
end

% initialize p
p = zeros(length(f),size(P,2));
% constant for switch of zero-pad size in adaptive zero padding
aaa = 1;
% loop through every frequency in f
for ii = 1:length(f)
  disp(['Calculating frequency response: f=', num2str(f(ii))]);
  % assign values for current frequency
  w = 2*pi*f(ii);
  k = w/c;
  v = {V{1}(f==f(ii),:).',V{2}(f==f(ii),:).',...
    V{3}(f==f(ii),:).',V{4}(f==f(ii),:).',...
    V{5}(f==f(ii),:).',V{6}(f==f(ii),:).'};

  % compute in-phase and contra-phase part of velocities of front and
  % back surface
  v_inphase= (v{1}+v{2})/2;
  v_contraphase = (v{1}-v{2})/2;

  % compute radiation from in-phase part via Rayleigh I
  [G,dGdn] = evaluate_G_dGdn_d2Gdn2(Q.',P.',w/c,n_vecs{1}.');
  p_inphase = -2*1i*w*rho*G.'*(v_inphase.*dA{1});
  %---------
  % the rest of this function is to obtain the pressures on the surface
  % by filtering the contraphase velocities in (k_x,k_y) space...
  % reshape the velocities to form the 2D grid
  v_cont = reshape(v_contraphase,size(grids{1,1}));

  if strcmp(mode,"lpbp")
    % linear-predictive border padding as wavefield extrapolation...
    % extrapolate
    [Vext] = interpolate_box(grids(1:2,1),gridspad,{v_cont(:).'},...
      floor((n_sources(2)+n_sources(1))));
    vext = reshape(Vext{1},size(gridspad{1,1}));
    % truncate to size of padded v
    mididx = [floor(size(vext,1)/2+1),floor(size(vext,2)/2+1)];
    truncidxx = mididx(2)+(-floor(lenx/2)+1-lenpadx:...
      floor(lenx/2)+lenpadx);
    truncidxy = mididx(1)+(-floor(leny/2)+1-lenpady:...
      floor(leny/2)+lenpady);
    vexttrunc = vext(truncidxy,truncidxx);
    % window and apply correction factor
    vexttruncwin = win.*vexttrunc*corrfac;
    V_cont = fft2(vexttruncwin);
  end

  if strcmp(mode,'zp')
    % zero padding only...
    % window the signal
    v_cont = v_cont.*win*corrfac;
```

```
    % 2D zero-pad
    v_contpad(1:leny,1:lenx) = v_cont;
    V_cont = fft2(v_contpad);
  end

  if strcmp(mode,'azp')
    % adaptive zero padding, switches padfactor at 400 Hz
    if aaa == 1
      if f(ii) > 400
        % padfactor is a paramater describing the amount of zeros added. E.g., a
        % padfactor=10 means zero-padding to the closest higher power of 2 to 10
        % times the signal in x and y.
        padfactor = 3;
        lenfftx = 2^(nextpow2(lenx*padfactor));
        lenffty = 2^(nextpow2(leny*padfactor));

        % window for zero padding
        winx = hann(lenx).';
        winy = hann(leny).';
        win = winy'* winx;
        % apparently, no window works better...
        win = ones(size(win));
        % pre assign zero-padded function
        v_contpad = zeros(lenffty,lenfftx);
        % compute correction factor
        nowin = v_contpad;
        nowin(1:leny,1:lenx) = 1;
        winc = nowin;
        winc(1:leny,1:lenx) = win;
        corrfac = rms(nowin,'all')/rms(winc,'all');
        aaa = 0;
      end
    end
    % window the signal
    v_cont = v_cont.*win*corrfac;
    % 2D zero-pad
    v_contpad(1:leny,1:lenx) = v_cont;
    V_cont = fft2(v_contpad);
  end

  if strcmp(mode,'')
    % no wavefield extrapolation or zero padding
    V_cont = fft2(v_cont);
  end

  % calculate (kx,ky) axes
  dx = abs(diff(grids{1,1},1,2));
  dx = dx(1,1);
  kx = 2*pi/lenfftx/dx*[(0:lenfftx/2),(-lenfftx/2+1:-1)];
  dy = abs(diff(grids{2,1},1,1));
  dy = dy(1,1);
  ky = 2*pi/lenffty/dy*[(0:lenffty/2),(-lenffty/2+1:-1)];
  [Kx,Ky] = meshgrid(kx,ky);

  % if tol is specified, use lower bound as regularization
  if nargin > 9
    % compute kz...
    Kz = conj(sqrt(1-(Kx.^2+Ky.^2)/k^2));
    phiKz = angle(Kz);
    AKz = abs(Kz);
    AKz(AKz<tol) = tol;
    Kz = k*AKz.*exp(1i*phiKz);
    % negative sign so that we obtain a form (-1i*Kz) in denominator
    Ps = -1i*w*rho.*V_cont./(-1i*Kz);
    ps_direct = ifft2(Ps);
    % direct for comparison/debugging of regularization
    ps_direct = ps_direct(1:leny,1:lenx);
    p_contraphase = -2*dGdn.'*(ps_direct(:).*dA{1});

  % if tol is not specidied, regularize using Pagavino's approach
  else
    % compute kz...
    Kz = conj(sqrt(1-(Kx.^2+Ky.^2)/k^2));
    phiKz = angle(Kz);
    AKz = abs(Kz);
    Kz = k*AKz.*exp(1i*phiKz);
    % negative sign so that we obtain a form (-1i*Kz) in denominator
    Ps = -1i*w*rho.*V_cont./(-1i*Kz);

    % regularize kz using a rectangle interpolator...
    % wavenumber bins
    lx = 0:1:lenfftx/2;
    ly = 0:1:lenffty/2;
    [Lx,Ly] = meshgrid(lx,ly);
    % differential (kx,ky)
    dkx = diff(kx(1:2));
    dky = diff(ky(1:2));

    % set the side lengths of the rectangle interpolator to be include
    % an area that with side lengths two times the device's measures
    regsizex = pi/(dkx*dx*lenx)/2;
    regsizey = pi/(dky*dy*leny)/2;

    % compute radii of the corners of the interpolator for each
    % wavenumber bin
```

```
    r1 = sqrt((dkx*(Lx-regsizex)).^2+(dky*(Ly-regsizey)).^2);
    r2 = sqrt((dkx*(Lx-regsizex)).^2+(dky*(Ly+regsizey)).^2);
    r3 = sqrt((dkx*(Lx+regsizex)).^2+(dky*(Ly-regsizey)).^2);
    r4 = sqrt((dkx*(Lx+regsizex)).^2+(dky*(Ly+regsizey)).^2);
    % compute maximum angular change for each wavenumber bin
    dphi = atan2(dky*(Ly+regsizey),dkx*(Lx-regsizex)) - ...
          atan2(dky*(Ly-regsizey),sqrt(r2.^2-(dky*(Ly-regsizey)).^2));
    % analytic solutions for the three parts of the integral
    I12 = k.*dphi./(r2-r1).*(...
          k/2*((r2/k).*(-sqrt((r2/k).^2-1))-...
          (r1/k).*(-sqrt((r1/k).^2-1))+...
          log(((-sqrt((r2/k).^2-1))+(r2/k))./...
          ((-sqrt((r1/k).^2-1))+(r1/k))))...
          -r1.*((-sqrt((r2/k).^2-1))-(-sqrt((r1/k).^2-1))));
    I23 = dphi.*k.*((-sqrt((r3/k).^2-1))-(-sqrt((r2/k).^2-1)));
    I34 = k.*dphi./(r4-r3).*(...
          -k/2*((r4/k).*(-sqrt((r4/k).^2-1))-...
          (r3/k).*(-sqrt((r3/k).^2-1))+...
          log(((-sqrt((r4/k).^2-1))+(r4/k))./...
          ((-sqrt((r3/k).^2-1))+(r3/k))))...
          +r4.*((-sqrt((r4/k).^2-1))-(-sqrt((r3/k).^2-1))));
    % combine to total regularized function...
    Itot = I12+I23+I34;
    Itot(1,:) = I23(1,:);
    Itot(:,1) = I12(:,1)+I34(:,1);
    % mirror and combine to matrix
    Itotcombined = [Itot,                    fliplr(Itot(1:end,2:end-1));
    flipud(Itot(2:end-1,1:end)),    rot90(Itot(2:end-1,2:end-1),2)];
    % normalize
    Itotcombined = Itotcombined./(4*regsizex*regsizey*dkx*dky);

    % compute the sound pressures on the surface (in k space)
    Psreg = -1i*w*rho.*V_cont.*Itotcombined;
    % for kx=ky=0 take results without regularization
    Psreg(1,1) = Ps(1,1);
    % inverse fft to obtain the surface pressures
    ps = ifft2(Psreg);
    % extract the only valid values
    ps = ps(1:leny,1:lenx);
    % compute rayleigh integral from resulting ps and include
    p_contraphase = -2*dGdn.'*(ps(:).*dA{1});
  end
  % adding resulting pressures
  p(ii,:) = p_inphase+p_contraphase;
end
end
```

Listing C.11 *Listing of the function calc_BEM_radiation() that computes the sound radiation using OpenBem. The syntax is partially adopted from the OpenBem tutorial scripts.*

```
function [p,p_surface,v_surface] = calc_BEM_radiation(A_cell,B_cell,...
nodesb,topologyb,grids,measures,V,P,f,c,rho)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [p,p_surface,v_surface] = calc_BEM_radiation(A_cell,B_cell,nodesb,...
%                                topologyb,grids,measures,V,P,f,c,rho)
% function that computes the radiated sound pressure for the dummy phone,
% using OpenBem (http://www.openbem.dk).
%
% Inputs:
% A_cell, B_cell   ...   A and B-matrices, pre-computed using the openBEM
%                       function
%                       '[A,B,CConst]=TriQuadEquat(nodesb,topologyb,k,nsingON)'
% nodesb, topologyb   ...   mesh nodes and topology of the loaded mesh,
%                           pre-computed using the openBEM functions
%                           '[nodesTMP,elementsTMP,elementsQUAD]=readgeomGMSH([fileGEOMsphere '.msh']);
%                            [nodesb,topologyb,segms]=meshcheck(nodesTMP,elementsTMP,0,0);'
% grids   ...   cell-array with surface grids of all surfaces, containing points of
%               measured or interpolated velocities. Surface order:
%               {front,back,down,up,left,right}
% measures   ...   (1 x 3) vector containing measures of the phone in
%                  (height,width,depth)
% V   ...   cell-array with complex vibrating velocities for each surface.
%           Each cell contains (n_f x n_points) matrix, for each frequency
%           and point.
% P   ...   (3 x n_rec) matrix containing 3D-coordinates of all receiver
%           positions
% f   ...   (n_f x 1) frequency vector
% c   ...   speed of sound
% rho   ...   air density
%
% Outputs:
% p   ...   (n_f x n_rec) calculated pressures for each frequency and
%           receiver points
% p_surface   ...   surface pressures, as used for radiation calculation
% v_surface   ...   interpolated surface velocities for the mesh, as used
%                   for radiation calculation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if length(f) ~= size(V{1},1)
  error('V must have the same number of entries as f!')
end

% create 20 randomly distributed CHIEF points inside the interior domain:
ind = 0.002;
nchief = 20;
CHIEFpoint=[(measures(1)-2*ind)*rand(nchief,1)-measures(1)/2+ind, ...
            (measures(2)-2*ind)*rand(nchief,1)-measures(2)/2+ind,...
            (measures(3)-2*ind)*rand(nchief,1)-measures(3)+ind];
M=size(nodesb,1);

v_surface = zeros(M,length(f));
p_surface = zeros(M,length(f));
p = zeros(length(f),size(P,2));
for ii = 1:length(f)
  % assign surface velocities
  disp(['Assigning velocities to mesh nodes: f=', num2str(f(ii))]);
  % assign interpolated values from measurement
  v = zeros(M,1);
  % reverse order since so that points of front and back side are not overritten
  for iii = size(grids,2):-1:1
    switch iii
      % front
      case 1
        node_idx = find(round(nodesb(:,3),3)==0);
        querrynodes = nodesb(node_idx,1:2);
      % back
      case 2
        node_idx = find(round(nodesb(:,3),3)==round(-measures(3),3));
        querrynodes = nodesb(node_idx,1:2);
      % down
      case 3
        node_idx = find(round(nodesb(:,1),3)==round(-measures(1)/2,3));
        querrynodes = nodesb(node_idx,2:3);
      % up
      case 4
        node_idx = find(round(nodesb(:,1),3)==round(measures(1)/2,3));
        querrynodes = nodesb(node_idx,2:3);
      % left
      case 5
        node_idx = find(round(nodesb(:,2),3)==round(-measures(2)/2,3));
        querrynodes = [nodesb(node_idx,1),nodesb(node_idx,3)];
      % right
      case 6
        node_idx = find(round(nodesb(:,2),3)==round(measures(2)/2,3));
        querrynodes = [nodesb(node_idx,1),nodesb(node_idx,3)];
      end
      vnodes = interp2(grids{1,iii},grids{2,iii},...
                       reshape(V{iii}(f==f(ii),:),size(grids{1,iii})),...
```

```
                        querrynodes(:,1),querrynodes(:,2),'makima',0);
        v(node_idx) = vnodes;
      end
      v_surface(:,ii) = v;

      %% calculate surface pressures
      disp(['Calculating surface pressures: f=', num2str(f(ii))]);
      %f_idx = find(f==f(ii));
      k = 2*pi*f(ii)/c;
      % Include CHIEF points:
      [Aex,Bex]=point(nodesb,topologyb,k,CHIEFpoint);
      % Extend A and B matrices to include CHIEF points' coefficients
      A=[A_cell{ii};Aex];
      B=[B_cell{ii};Bex];
      % Solve for surface pressures
      B=1i*k*rho*c*B;
      ps=A\(B*v_surface(:,ii));
      p_surface(:,ii) = ps;

      %% calculate pressures on field points
      disp(['Calculating frequency response: f=', num2str(f(ii))]);
      k = 2*pi*f(ii)/c;
      % Calculate field points
      [Afp,Bfp,Cfp]=point(nodesb,topologyb,k,P.',1);
      p(ii,:)=(Afp*p_surface(:,ii)-1i*k*rho*c*Bfp*v_surface(:,ii)).*Cfp/(4*pi)^2;
    end
  end
```

Listing C.12 *Listing of the function plotFR().*

```
function [fig] = plotFR(f, p, legends,xlims,ylims)
% [fig] = plotFR(f, p, legends,xlims,ylims)
%
% plots the frequency response in Pa, dBspl and angle.
%
% Inputs:
% f    ...    (n_f x 1) frequency vector
% p    ...    (n_f x n_rec) vector containing the sound pressures for one
%             n_rec receivers
% legends    ...    (n_rec x 1) string array containing the legend entries,
%                   interpreted as LaTex syntax
% xlims    ...    (optional) limits for the x axes in Hz
% ylims    ...    (optional) limits for the y axes, in dBspl. Only affecting
%                 the first two subplots
%
% Outputs:
% fig    ...    figure handle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin < 4
  xlims = 'auto';
  ylims = 'auto';
elseif nargin < 5
  ylims = 'auto';
else
  ylims = 10.^(ylims./20)*20E-6;
end

fig = figure;
sgtitle('Frequency response','Interpreter','latex')
subplot(3,1,1)
semilogx(f, abs(p))
ylabel('$\left|\frac{p}{U}\right|$ in $\frac{\mathrm{Pa}}{\mathrm{V}}$','Interpreter','latex')
xlabel('$f$ in Hz','Interpreter','latex')
grid on
ylim(ylims)
ylims = ylim;
xlim(xlims)
xlims = xlim;
subplot(3,1,2)
semilogx(f, 20*log10(abs(p)/20E-6))
ylabel('$\left|\frac{p}{U}\right|$ in $\mathrm{dB_{SPL}}$','Interpreter','latex')
xlabel('$f$ in Hz','Interpreter','latex')
grid on
ylim(20*log10(abs(ylims)./20E-6))
xlim(xlims)
subplot(3,1,3)
semilogx(f, angle(p))
ylim([-pi-pi/10,pi+pi/10]);
yticks([-pi, -pi/2, 0, pi/2, pi])
set(gca,'TickLabelInterpreter','latex');
yticklabels({'$- \pi$'; '$- \pi/2$'; '$0$'; '$\pi/2$'; '$\pi$'})
ylabel('$arg\left(\frac{p}{U}\right)$ in rad','Interpreter','latex')
xlabel('$f$ in Hz','Interpreter','latex')
grid on
legend(legends, 'Location', 'southwest','Interpreter','latex')
xlim(xlims)
end
```

Listing C.13 *Listing of the function balloonPlot().*

```
function [fig] = balloonPlot(p, P, n_rec_hor, full,offset,phasecolor,titleaddon)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [fig] = balloonPlot(p, P, n_rec_hor, full,offset,phasecolor,titleaddon)
%
% creates a 3D balloon surface plot for each position in P. The radius of
% the baloon in each direction is assigned to the respective dB_spl of the
% sound pressures in p.
%
% Inputs:
% p    ...    (n_f x n_rec) matrix containing the sound pressures for each
%             frequency and receiver
% P    ...    (3 x n_rec) matrix containing the 3D coordinates of the
%             receiver positions
% n_rec_hor    ...    number of receiver positions along a horizontal ring as
%                     a scalar (must be the same as used for
%                     defineReceivers())
% full    ...    scalar value. If 1, a full sphere is plotted. Otherwise, a
%                hemisphere is plotted
% offset    ...    when offset>0, the balloon plot is normalized to a
%                  maximum of 0dB. The value of offset then specifies the
%                  offset of the origin, in dB. E.g., offset=30 scales the
%                  balloon so that -30dB lie in the origin and all values
%                  below -30dB are not considered.
% phasecolor    ...    allows for coloration according to the angle, instead
%                      of amplitude. The 0 angle is set to the front center.
% titleaddon    ...    (optional) adds an additional text to the title, is
%                      interpreted as LaTex syntax
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fig = figure;

% scale each receiver in distance, representing the resulting sound pressure
angp = angle(p);
% scaling for dBspl
p = 20*log10(abs(p)/20E-6);

% add titleaddon
if nargin <= 6
  titleaddon = '';
end

% manage limits
if offset > 0
  p = p - max(p) + offset;
  p(p<0) = 0;
  if full == 1
    rng = [-offset,offset];
  else
    rng = [0,offset];
  end
else
  if full == 1
    rng = [-max(p(:)),max(p(:))];
  else
    rng = [0,max(p(:))];
  end
end

% create the radius of the balloon
dist = sqrt(P(1,:).^2+P(2,:).^2+P(3,:).^2);
P_plot = P.*repmat(p,3,1)./repmat(dist,3,1);

% assign colorvalues according to pressure
if phasecolor==1
  % if the phasecolor is plotted, normalize so that 0 phase is in the
  % front center
  P_front = [0;0;dist(1)];
  P_dist = sqrt(sum((P_front-P).^2,1));
  P_idx = find(P_dist==min(P_dist),1);
  C_ref = angp(1,P_idx);
  C = angp-C_ref;
  % wrap phase
  C(C<-pi) = 2*pi+C(C<-pi);
  C(C>pi) = -2*pi+C(C>pi);
  % reshape for plot
  C = buffer(C, n_rec_hor);
else
  C = buffer(p, n_rec_hor);
end
% surface plot
X = buffer(P_plot(1,:), n_rec_hor);
Y = buffer(P_plot(2,:), n_rec_hor);
Z = buffer(P_plot(3,:), n_rec_hor);
surf(X, Y, Z, C)
% set title
title(['Balloon plot ',titleaddon],'Interpreter','latex')
axis equal
% set limits
```

```
xlim([-rng(2), rng(2)]);
ylim([-rng(2), rng(2)]);
zlim([-abs(rng(1)), rng(2)]);

if offset > 0
  tcks = round(linspace(-rng(2),rng(2),9));
  xticks(tcks);
  yticks(tcks);
  zticks(tcks);
  xtcks = abs(xticks)-offset;
  ytcks = abs(yticks)-offset;
  ztcks = abs(zticks)-offset;
  xticklabels(string(xtcks));
  yticklabels(string(ytcks));
  zticklabels(string(ztcks));
else
  tcks = round(linspace(-rng(2),rng(2),9));
  xticks(tcks);
  yticks(tcks);
  zticks(tcks);
  xtcks = abs(tcks);
  ytcks = abs(tcks);
  ztcks = abs(tcks);
  xticklabels(string(xtcks));
  yticklabels(string(ytcks));
  zticklabels(string(ztcks));
end
% add origin
hold on
scatter3(0,0,0)
hold off
% define color range and colorbar
if phasecolor==1
  colormap('hsv')
  caxis([-pi,pi]);
  colorbar('Ticks', [-pi,-pi/2,0,pi/2,pi],...
  'Ticklabels',{'$-\pi$ rad','$-\frac{\pi}{2}$ rad','$0$ rad',...
    '$\frac{\pi}{2}$ rad','$\pi$ rad'},'TickLabelInterpreter', 'latex')
elseif offset > 0
  caxis([0,offset]);
  c = colorbar;
  c.TickLabelInterpreter = 'latex';
  c.TickLabels = string((-flip(c.Ticks)))+" dB";
else
  c = colorbar;
  c.TickLabelInterpreter = 'latex';
  c.TickLabels = string((c.Ticks)) + ...
        " dB$\mathrm{_{SPL}}$";
end

if offset == 0
  xlabel('dB$\mathrm{_{SPL}}$','Interpreter', 'latex')
  zlabel('dB$\mathrm{_{SPL}}$','Interpreter', 'latex')
  ylabel('dB$\mathrm{_{SPL}}$','Interpreter', 'latex')
else
  xlabel('dB','Interpreter', 'latex')
  zlabel('dB','Interpreter', 'latex')
  ylabel('dB','Interpreter', 'latex')
end
end
```

Listing C.14 *Listing of the function fieldPlot().*

```
function [figs] = fieldPlot(p,P,n_rec_vert,titleaddon,col_range,mode,w,rho)
% [figs] = fieldPlot(pfield,Pfield,n_rec_vert,titleaddon,col_range)
%
% Image plot in the cross section of the sound field. The sound field is
% interpolated to a mesh size of 1mm, using spline interpolation. Two
% options are possible: plotting the sound-pressure magnitude in dBSPL or,
% with mode='I', plotting the real part of the acoustic intensity and a
% corresponding quiver plot to show the direction of the effective energy
% flow
%
% Inputs:
% p   ...   (1 x n_rec) matrix containing sound pressures corresponding to
%           the positions in P
% P   ...   (3 x n_rec) matrix containing the three-dimensional coordinates
%           of the receivers
% n_rec_vert   ...   number of positions in the vertical axis. Must be the
%                    same as specified in defineReceivers(). Needed to
%                    create a grid
% titleaddon   ...   (optional) adds an additional text to the title, is
%                    interpreted as LaTex syntax
% colrange   ...   (optional) specifies the range of coloration in dB.
%                  can be left out or specified as 'auto' for automatic
%                  coloration, or stated as a 2-element vector holding the
%                  lower and upper limits for the coloration.
% mode   ...   (optional) can be either 'p', for the default sound-pressure
%              magnitude plot. Or 'I', for the intensity plot.
% w   ...   angular frequency, must be specified if mode is 'I'
% rho   ...   air density, must be specified if mode is 'I'
% Outputs:
% figs   ...   figure handle
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Author: Patrick Heidegger; heidegger.patrick@gmail.com
% Company: Institute of Electronic Music and Acoustics, Graz &
%          Sound-Solutions Austria, Vienna
% Year: 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


if nargin < 4
  titleaddon = '';
end
% create grids
P(P(:,1)==0,:) = [];
X = buffer(P(1,:),n_rec_vert);
Z = buffer(P(2,:),n_rec_vert);
% calculate dBSPL and create grid
if nargin < 6 || strcmp(mode,'p')
  % grid of sound pressures in dBSPL
  plotfun = buffer(20*log10(abs(p)/20E-6),n_rec_vert);
  mode = 'p';
elseif strcmp(mode,'I')
  % compute the acoustic intensity
  pgrid = buffer(p,n_rec_vert);
  stepx = 2*max(max(X))/size(X,2);
  stepy = 2*max(max(Z))/size(X,1);
  vgridx = diff(pgrid,1,2)/stepx/(-1i*w*rho);
  vgridy = diff(pgrid,1,1)/stepy/(-1i*w*rho);
  Igridx = pgrid(:,1:end-1).*conj(vgridx);
  Igridx = Igridx(1:end-1,:);
  Igridy = pgrid(1:end-1,:).*conj(vgridy);
  Igridy = Igridy(:,1:end-1);
  % get rid of the elements for which the gradient is not computed
  X = X(1:end-1,1:end-1);
  Z = Z(1:end-1,1:end-1);
  %
  Igridlen = sqrt(real(Igridx).^2+real(Igridy).^2);
  plotfun = 10*log10((Igridlen/1E-12));
end
% interpolate to a resolution of 1mm
[Xint,Zint] = meshgrid(min(X(:)):0.001:max(X(:)),min(Z(:)):0.001:max(Z(:)));
Zint = flipud(Zint);
plotfunint = interp2(X,Z,plotfun,Xint,Zint,'spline');

figs = figure;
if nargin < 5
  imagesc([min(Xint(:)),max(Xint(:))],[min(Zint(:)),max(Zint(:))],...
          plotfunint)
elseif strcmp(string(col_range), 'auto')
  imagesc([min(Xint(:)),max(Xint(:))],[min(Zint(:)),max(Zint(:))],...
          plotfunint)
else
  % define color range
  crange = [col_range(1),col_range(2)];
  imagesc([min(Xint(:)),max(Xint(:))],[min(Zint(:)),max(Zint(:))],...
          plotfunint,crange)
end
title(['Field plot ',titleaddon],'Interpreter','latex')
c = colorbar;
c.TickLabelInterpreter = 'latex';
xlabel('Horizontal axis in m','Interpreter','latex')
ylabel('Vertical axis in m','Interpreter','latex')
grid on
yticklabels(string(yticks*-1))
```

```
  xticklabels(string(xticks*1))
  axis equal
  if strcmp(mode,'I')
    % custom ticklabels for colorbar
    c.TickLabels = string((c.Ticks)) + ...
    " dB$\mathrm{_{I}}$";
    hold on
    % scale arrows
    Igridxquiv = flipud(real((Igridx))./Igridlen);
    Igridyquiv = flipud(real((Igridy))./Igridlen);
    % quiver plot
    q = quiver(X,Z,Igridxquiv,Igridyquiv,3/4);
    q.Color = 'black';
  else
    % custom ticklabels for colorbar
    c.TickLabels = string((c.Ticks)) + ...
    " dB$\mathrm{_{SPL}}$";
  end
  end
```

Appendix D Additional Simulation Results

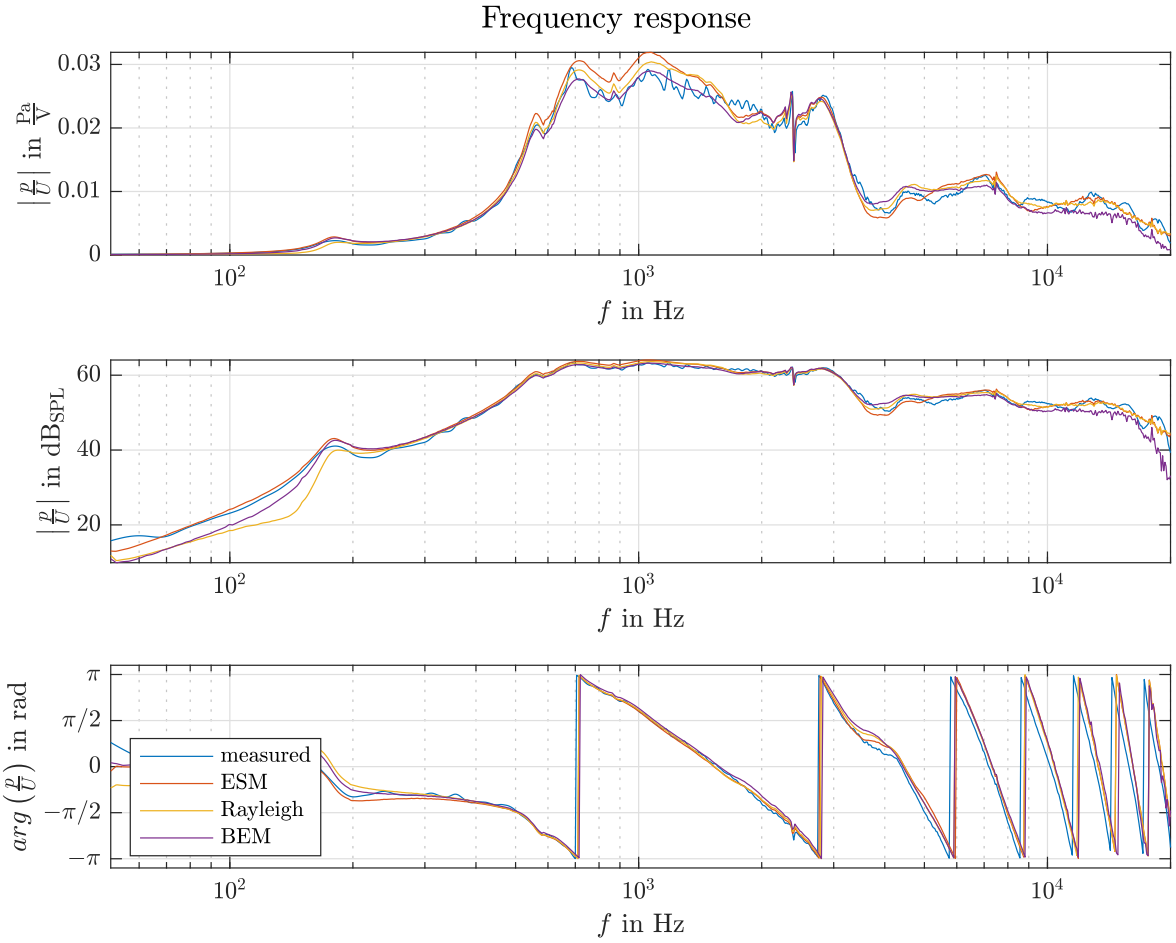


Figure D.1 *Overall measured and simulated full-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage; the receiver is in the front-center of the device, in* 10 cm *distance (position 1 in Table 5.1);*

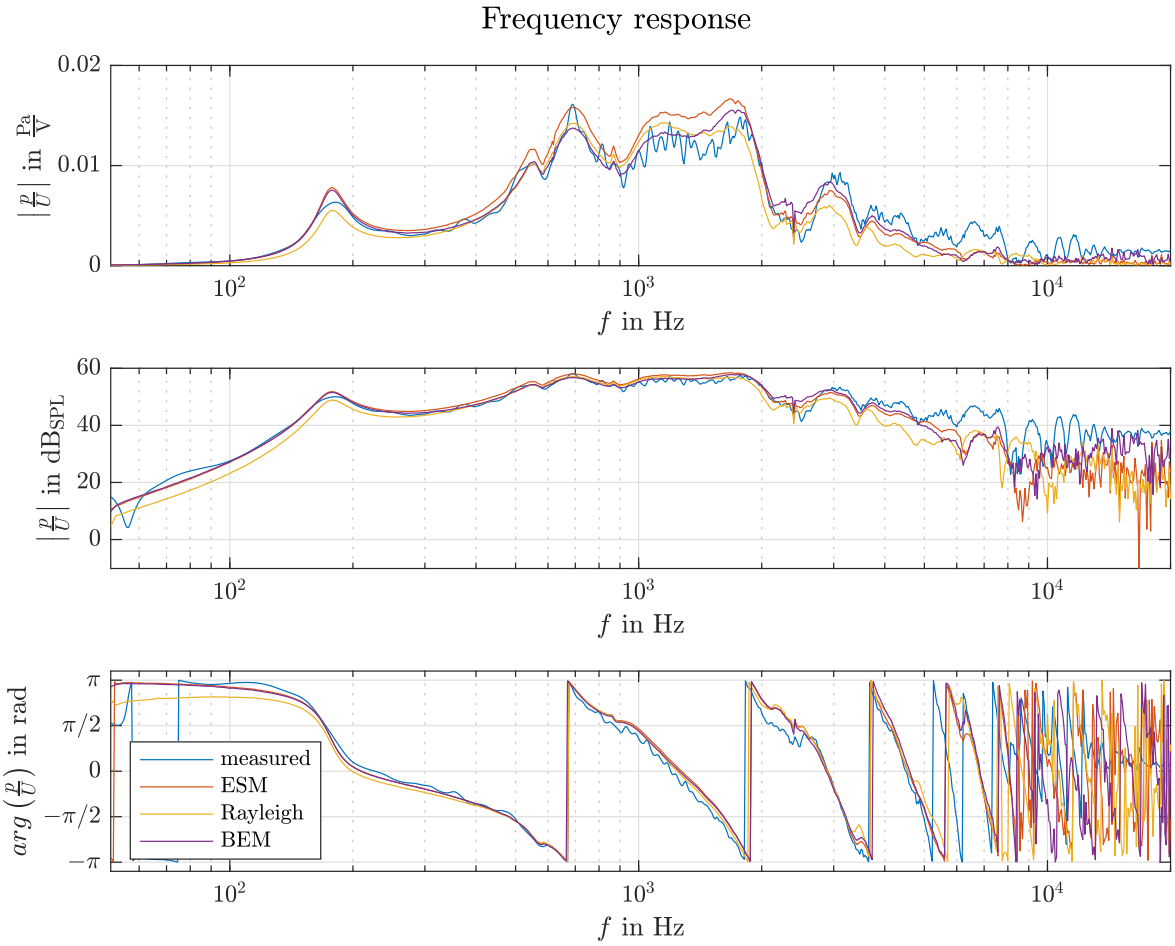


Figure D.2 *Overall measured and simulated full-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage; the receiver is in the back-center of the device, in* 10 cm *distance (position 2 in Table 5.1);*

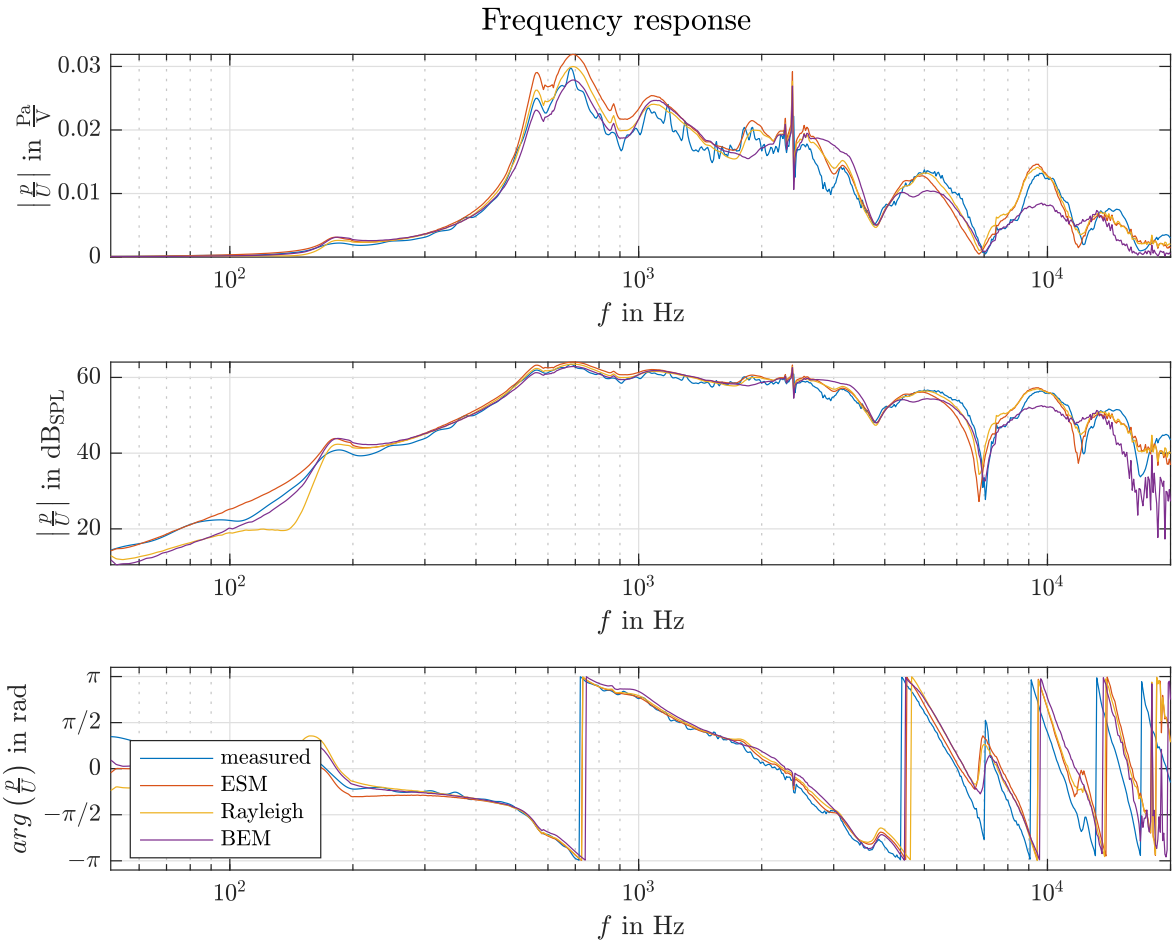


Figure D.3 *Overall measured and simulated full-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage; the receiver is in front of the device with* 60° *rotation angle around the* y *axis, in* 10 cm *distance (position 3 in Table 5.1);*

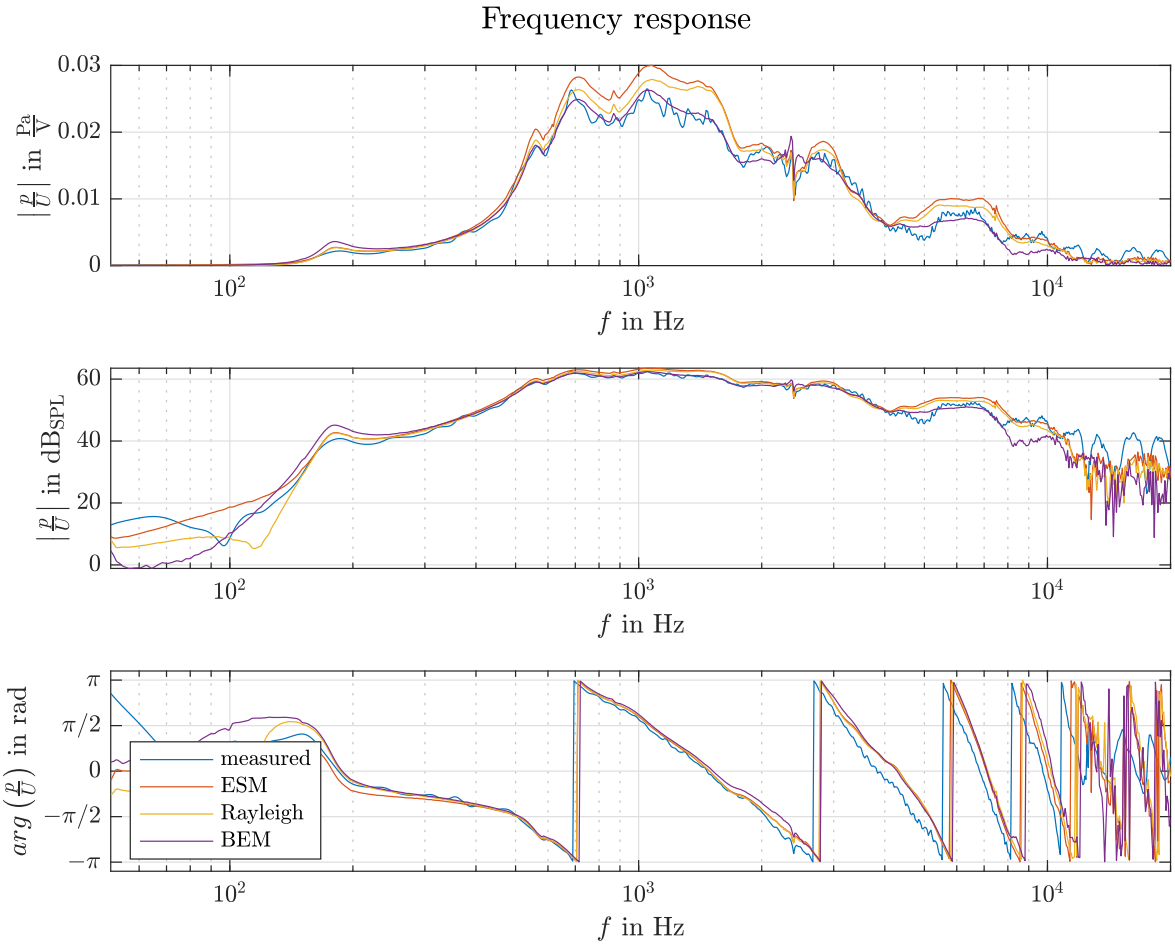


Figure D.4 *Overall measured and simulated full-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage; the receiver is in front of the device with* $60°$ *rotation angle around the* x *axis, in* 10 cm *distance (position 4 in Table 5.1);*

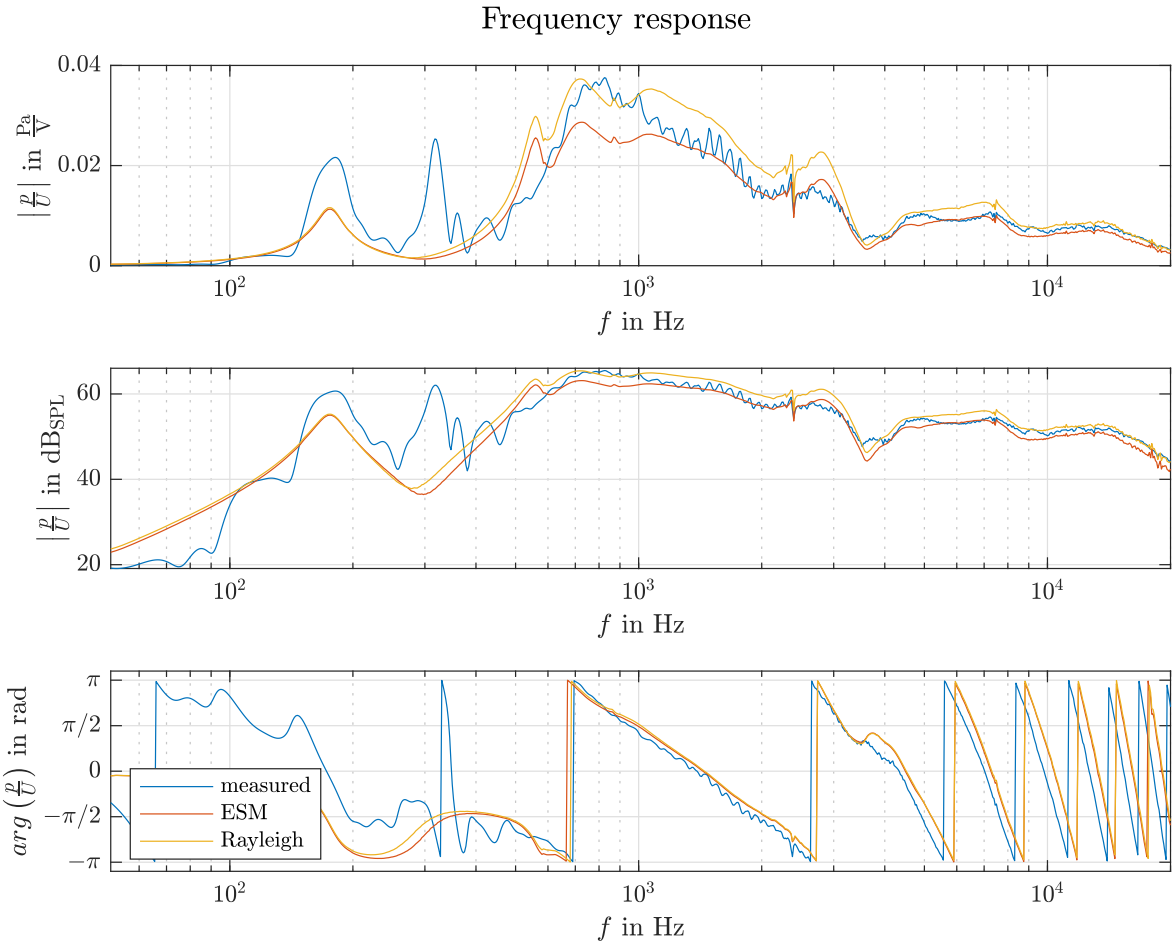


Figure D.5 *Overall measured and simulated half-space frequency responses of the smartphone dummy, as sound pressure normalized by the input voltage; the receiver is in the front-center of the device, in* 10 cm *distance (position 1 in Table 5.1);*

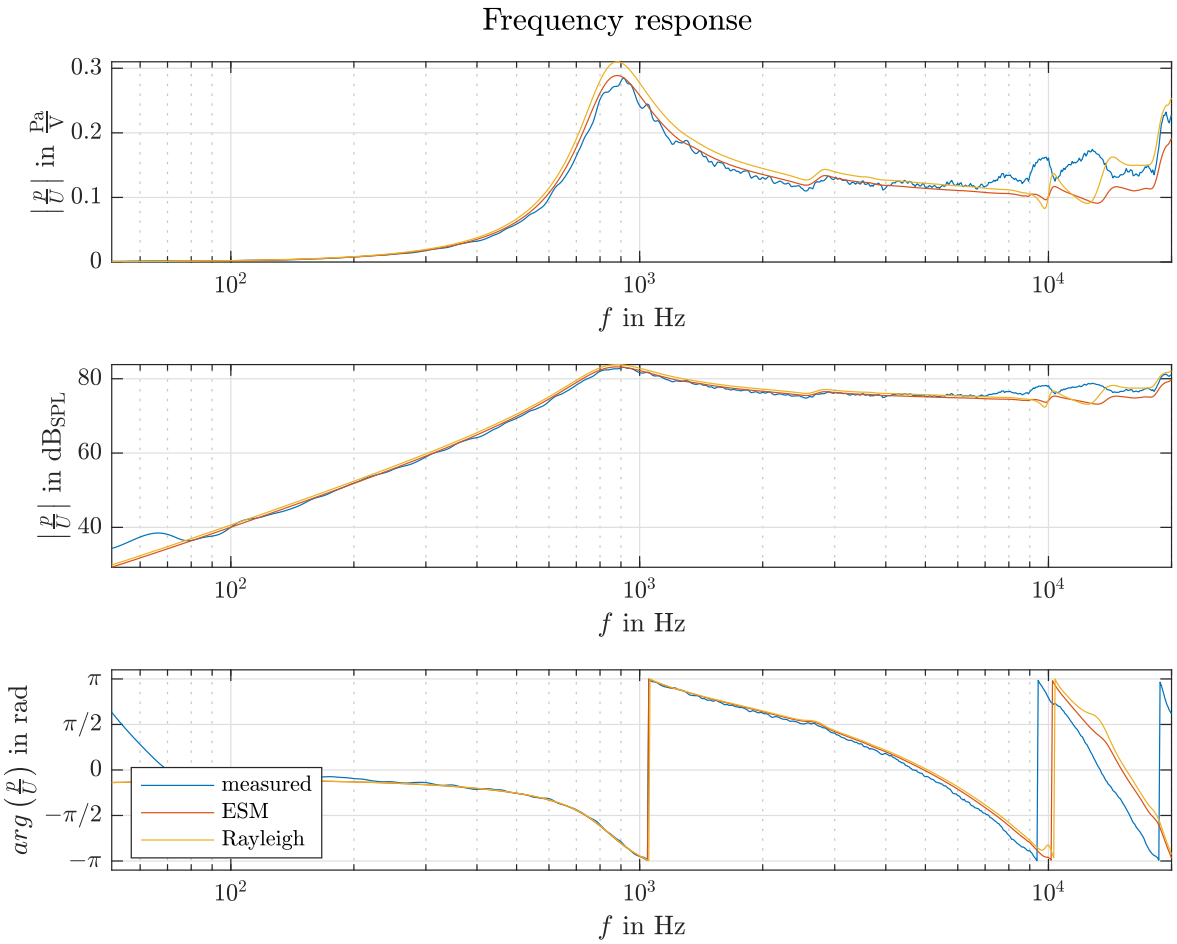
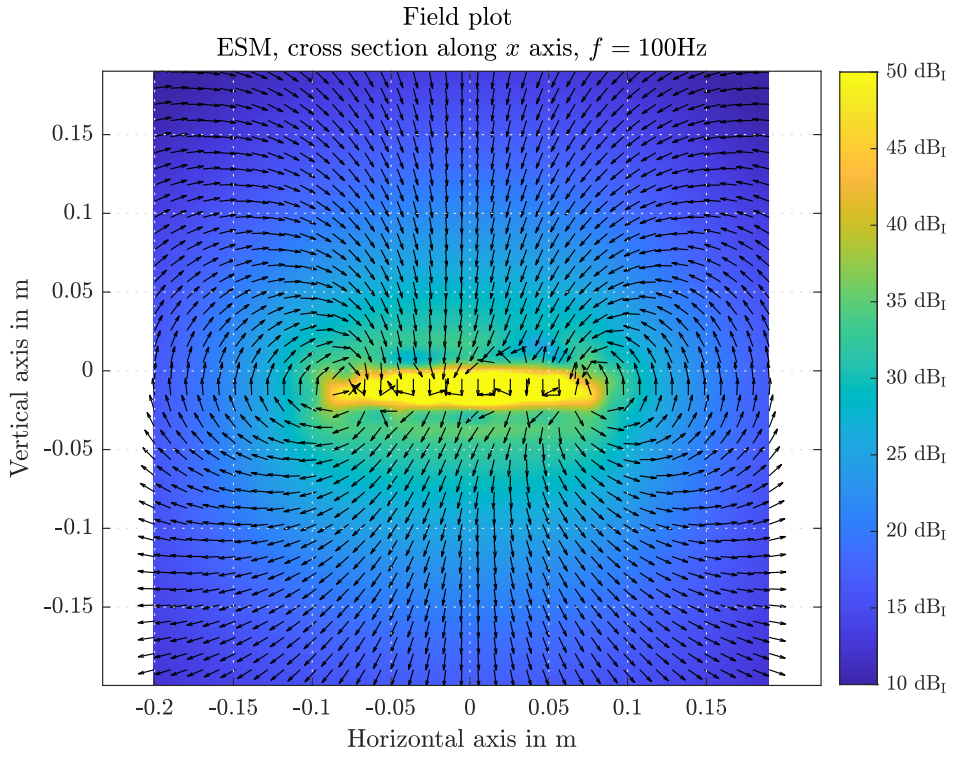
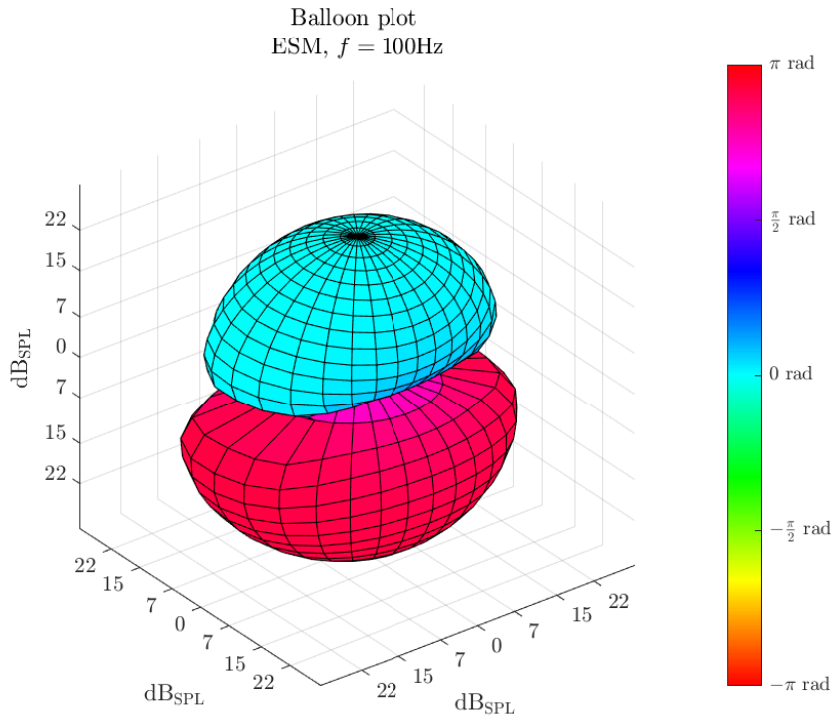


Figure D.6 *Overall measured and simulated half-space frequency responses of the alternative verification case using the Manticore voice-coil loudspeaker, as sound pressure normalized by the input voltage; the receiver is in the front-center of the device, in* 3.16 cm *distance;*

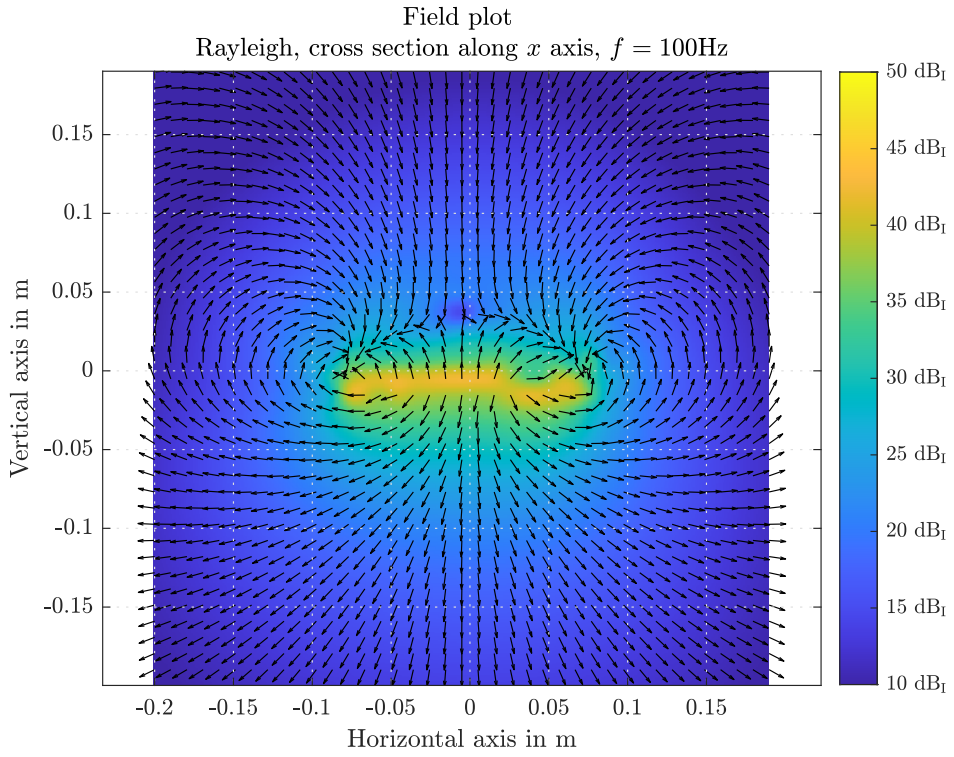


(a)

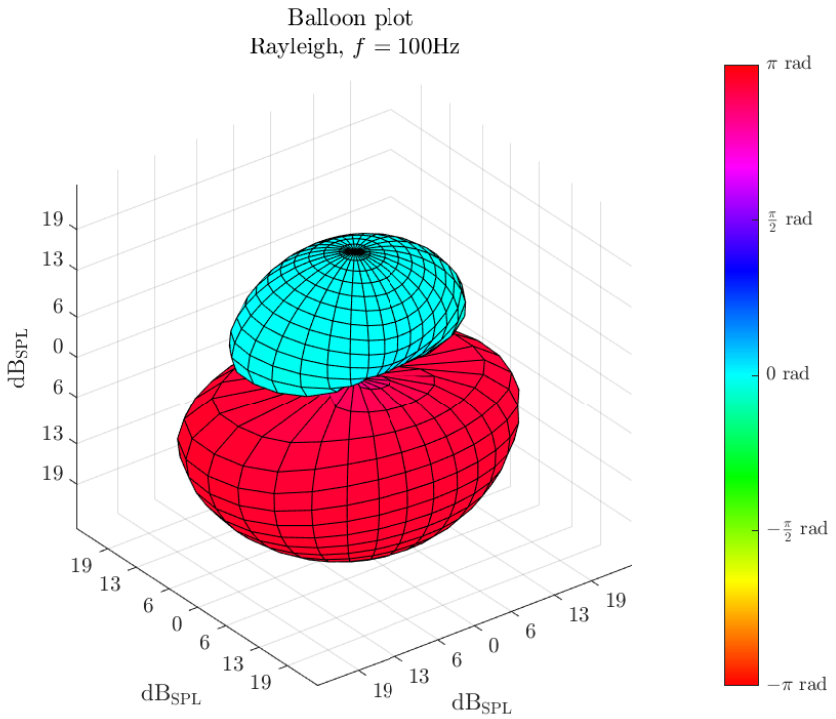


(b)

Figure D.7 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 100 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

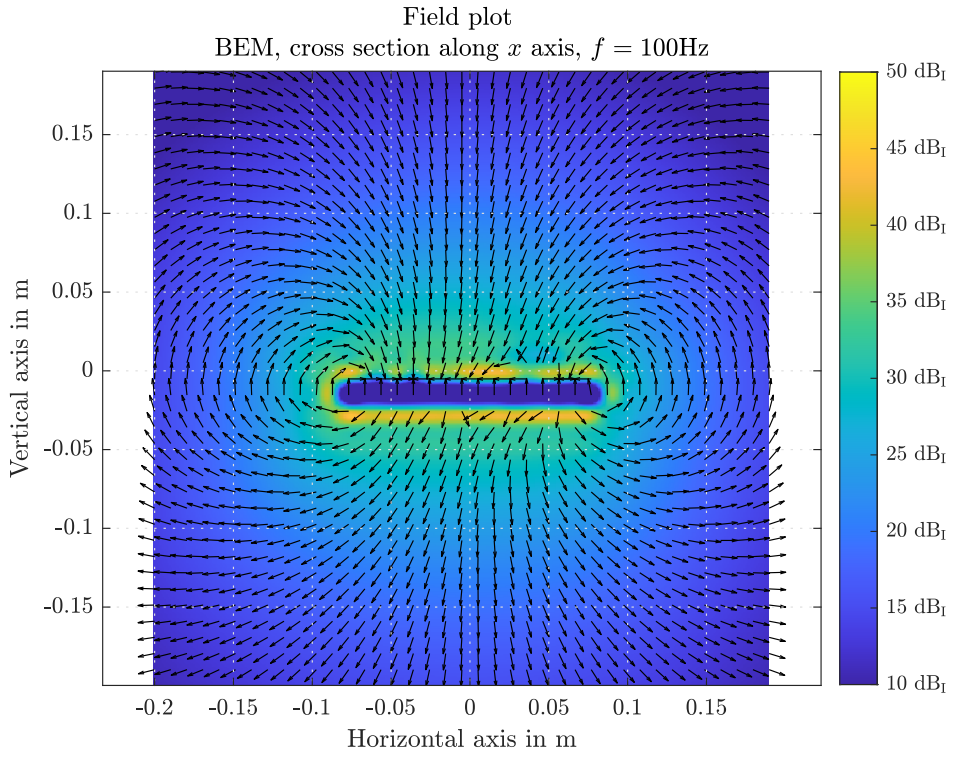


(a)

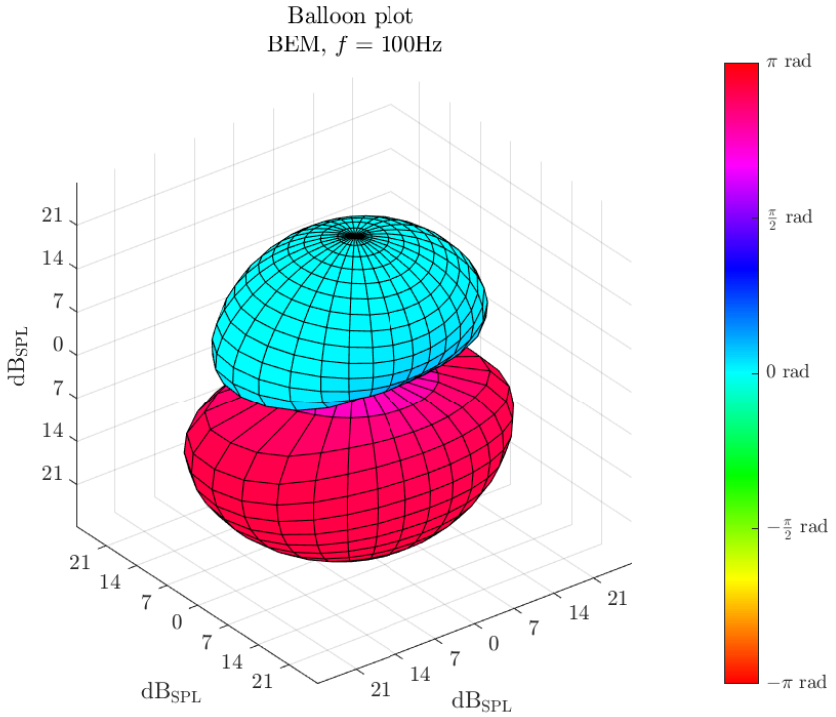


(b)

Figure D.8 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 100 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

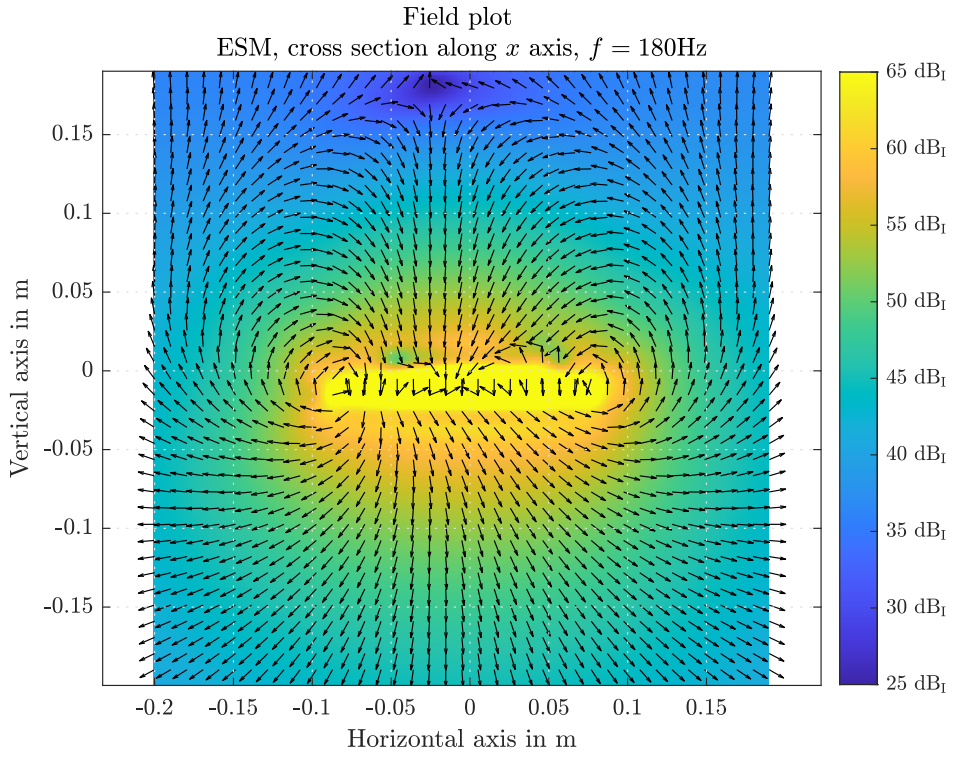


(a)

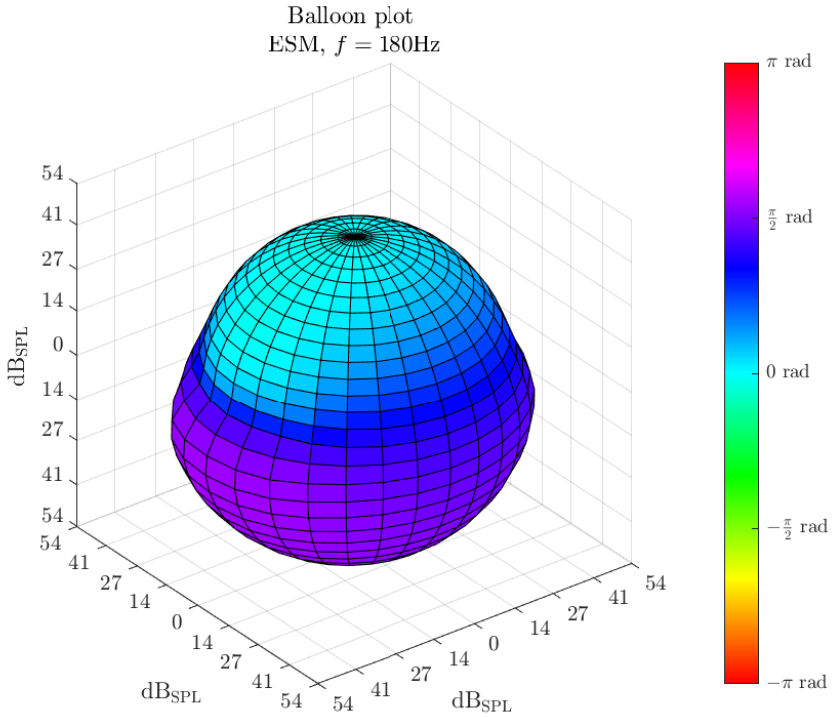


(b)

Figure D.9 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 100 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

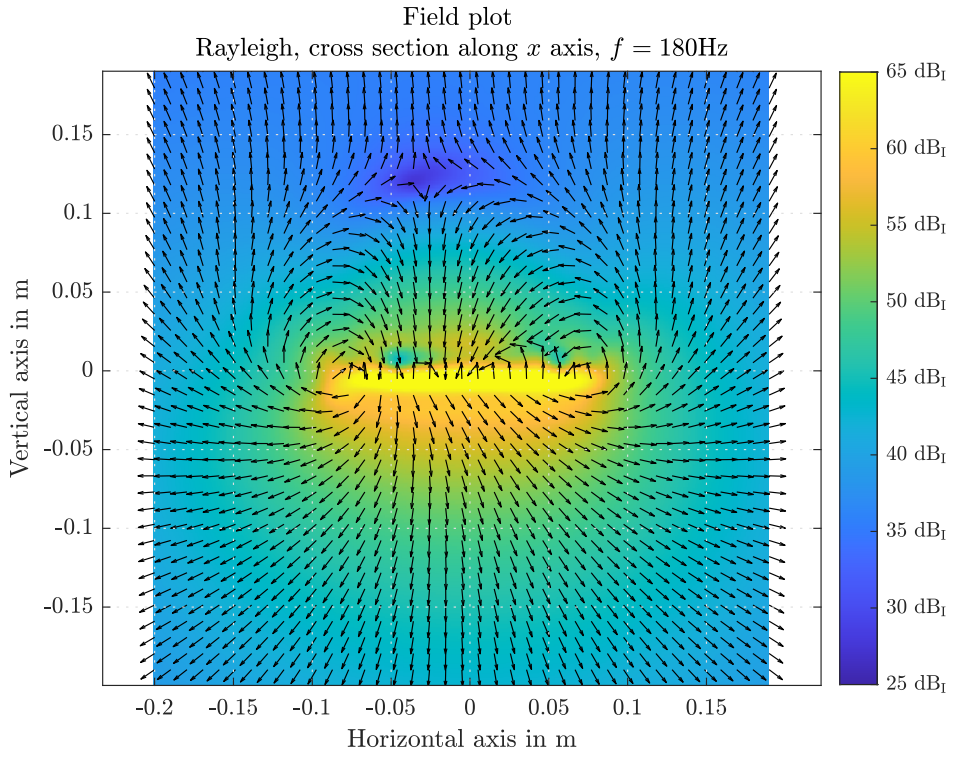


(a)

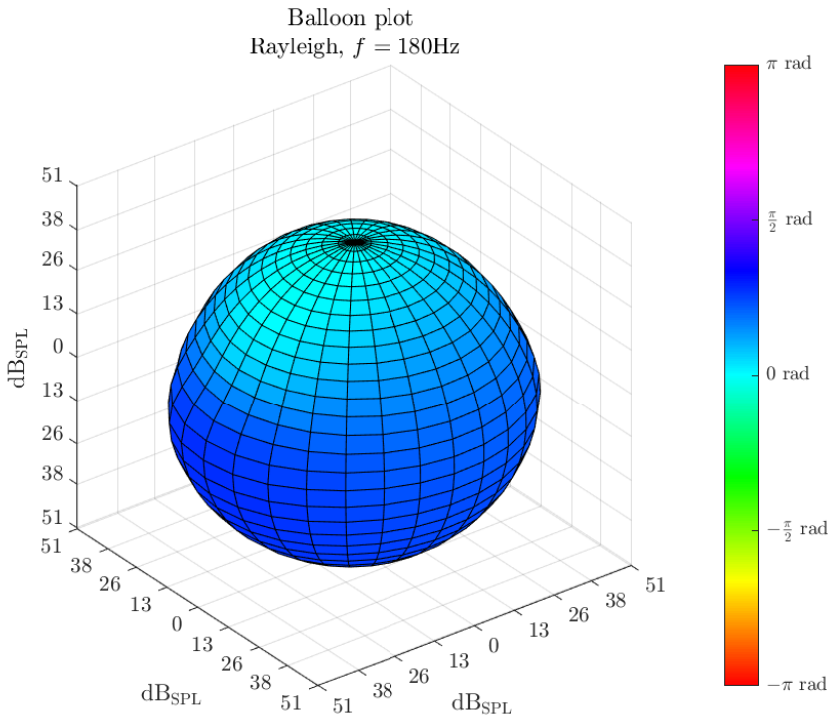


(b)

Figure D.10 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 180 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

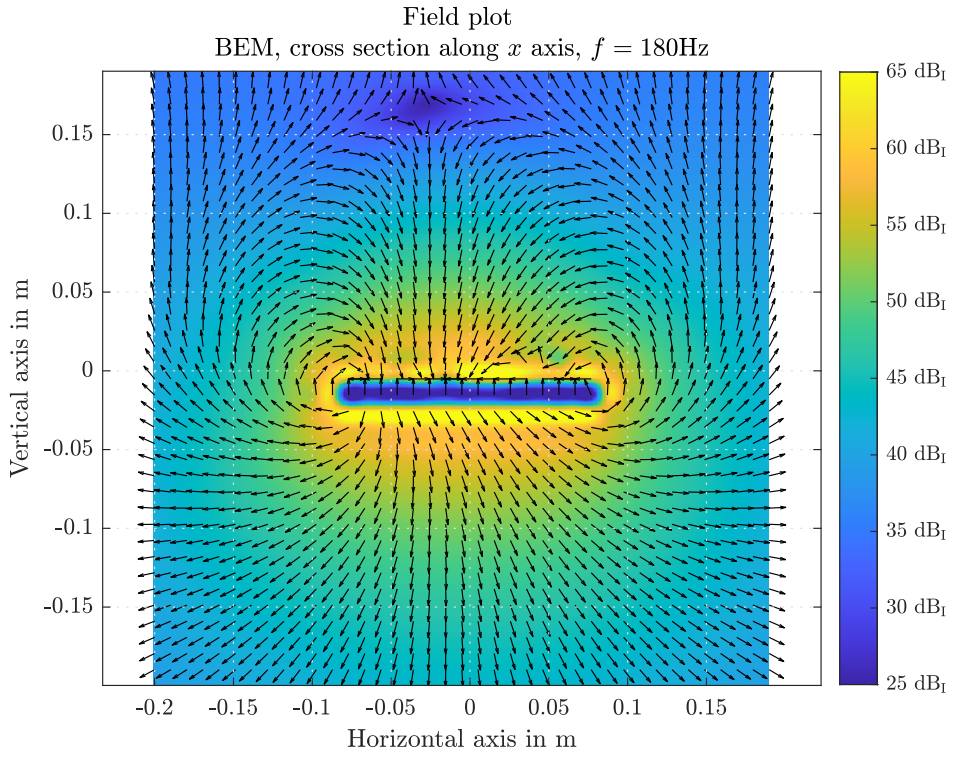


(a)

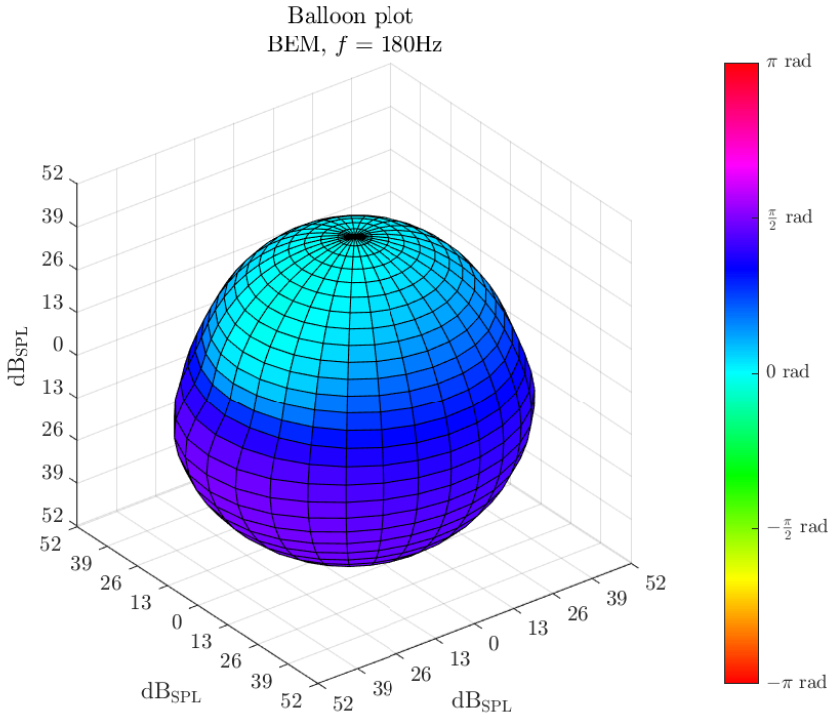


(b)

Figure D.11 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 180 Hz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

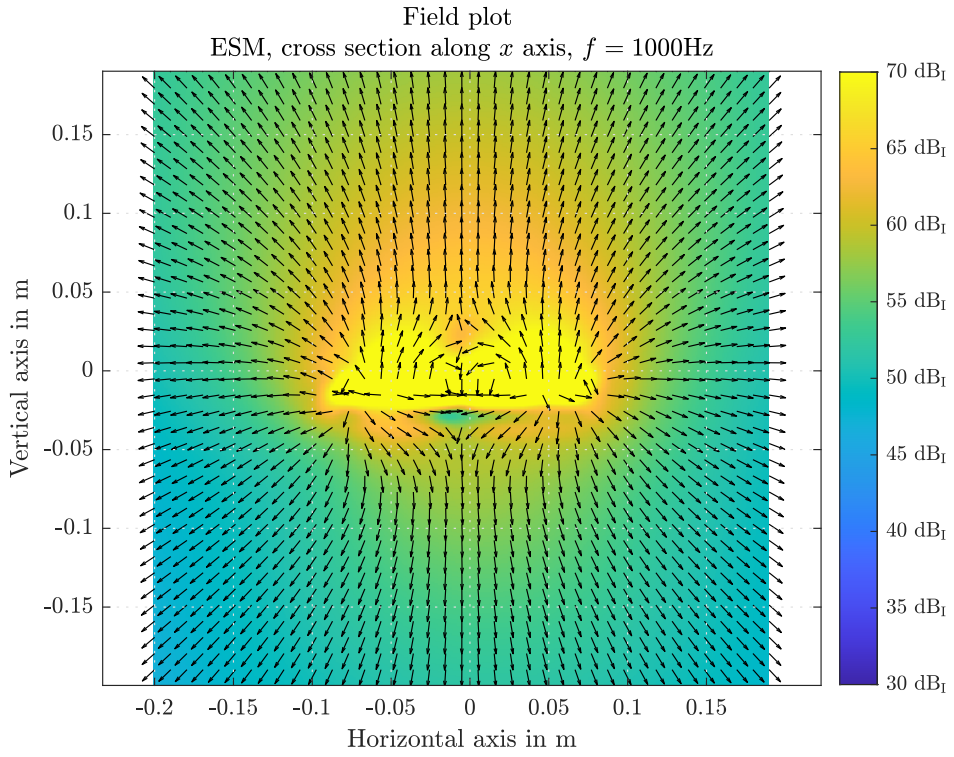


(a)

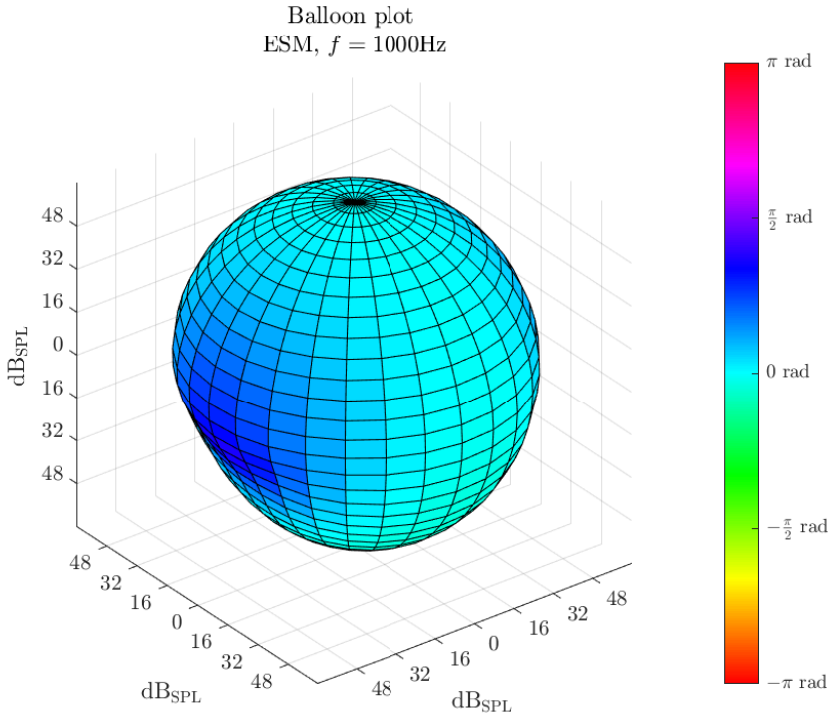


(b)

Figure D.12 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 180 Hz. *(a): acoustic intensity field of the cross-section along the* x *axis, at* $y = 0$. *The field is evaluated at a* (40×40) *grid in a range of* ± 20 cm. *(b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at* 30×30 *positions, at* $r = 0.1$ m *distance to the device's center. The axes of the plot are (left to right):* (z, y, x).

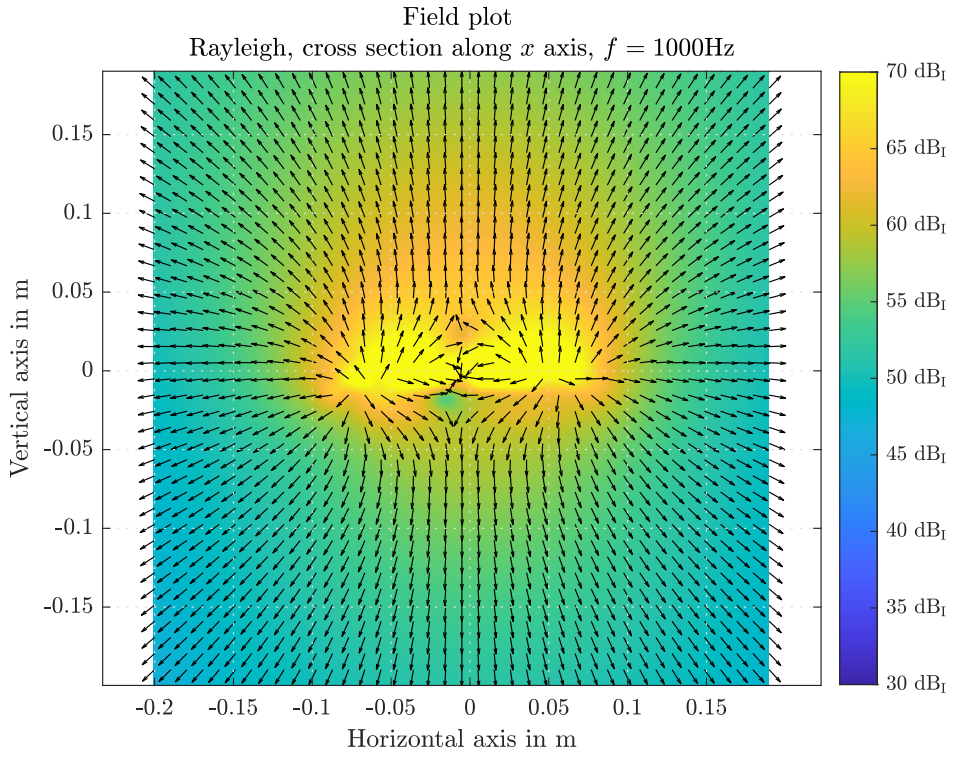


(a)

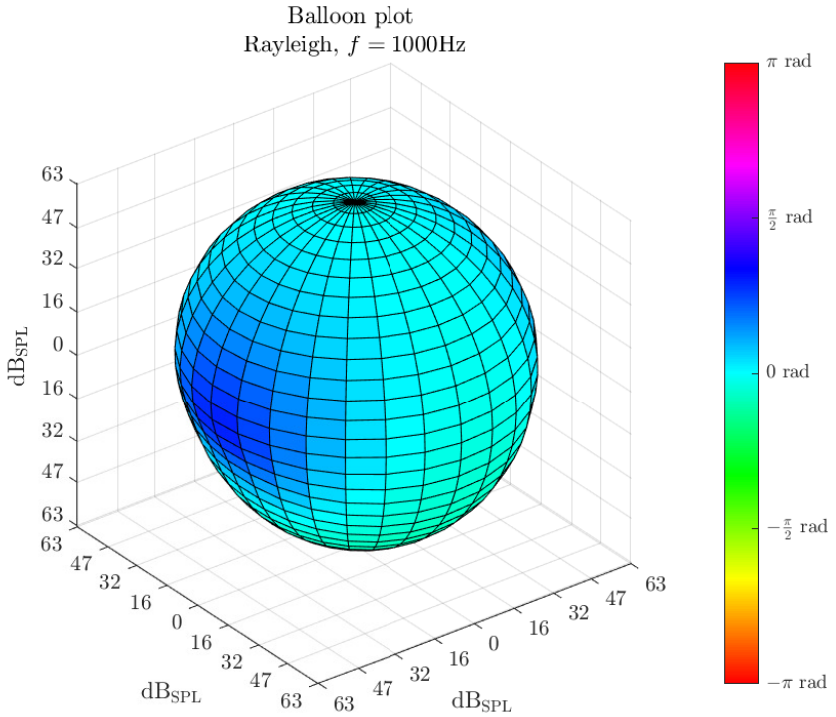


(b)

Figure D.13 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 1 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

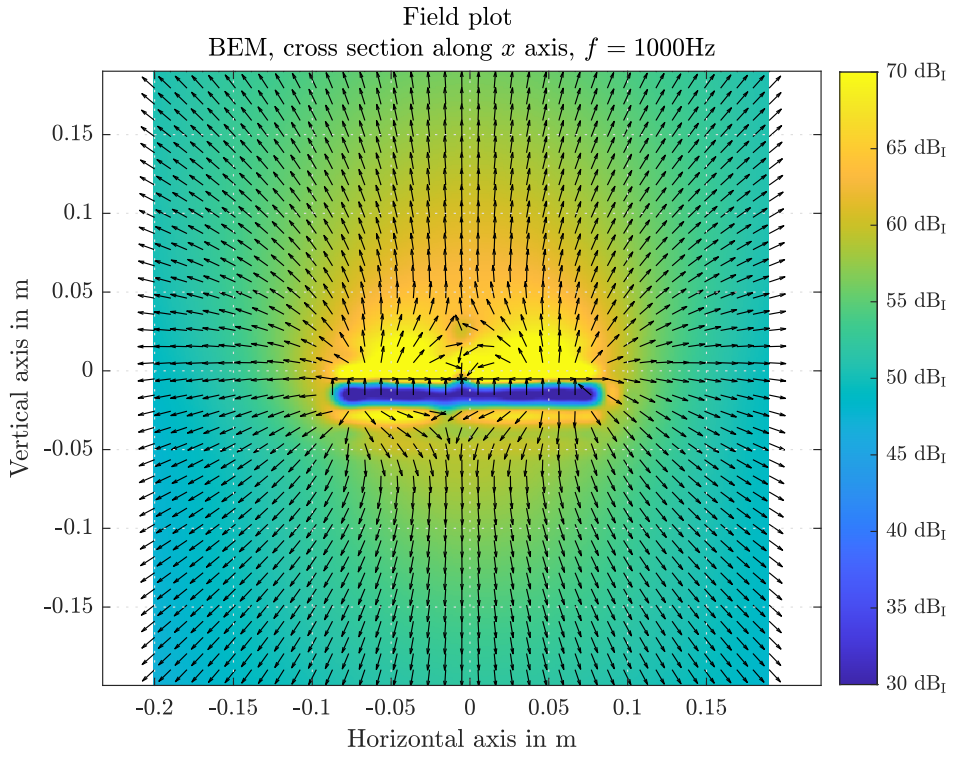


(a)

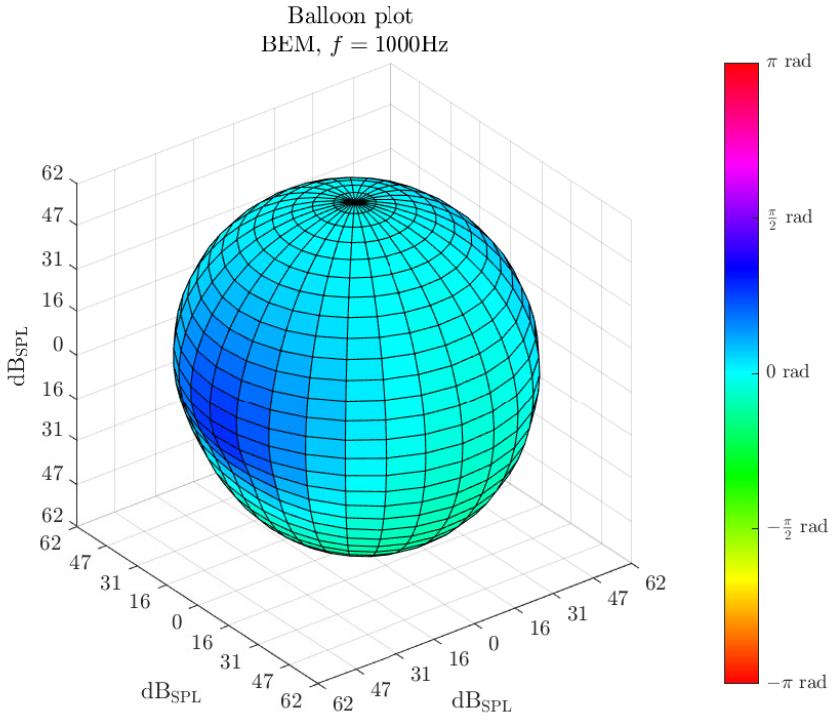


(b)

Figure D.14 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 1 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

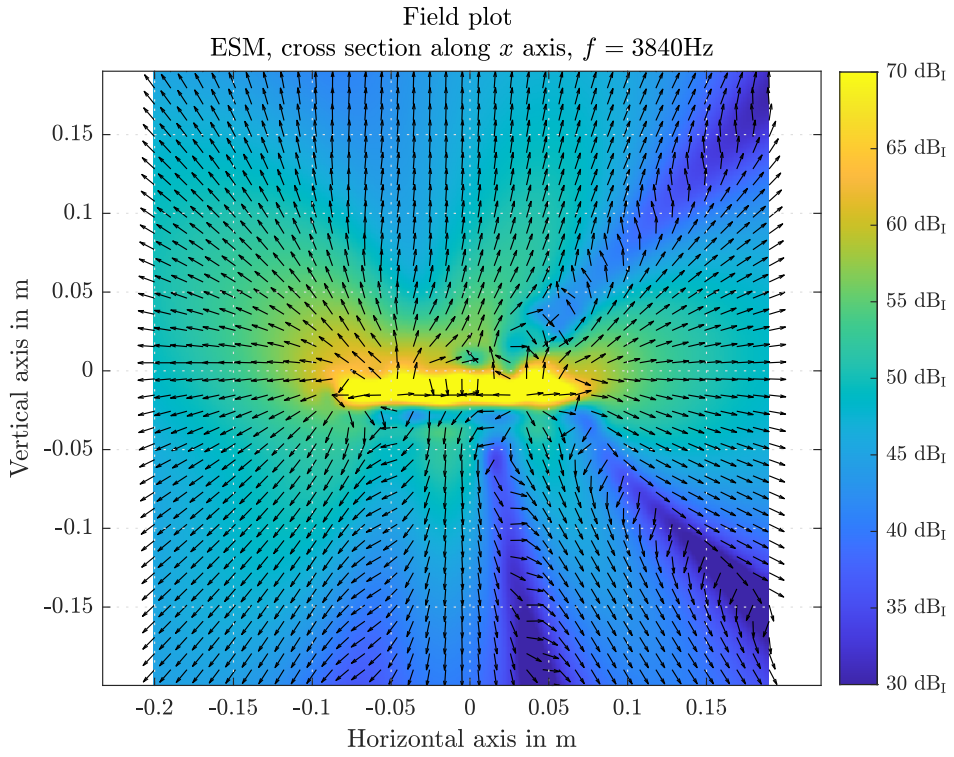


(a)

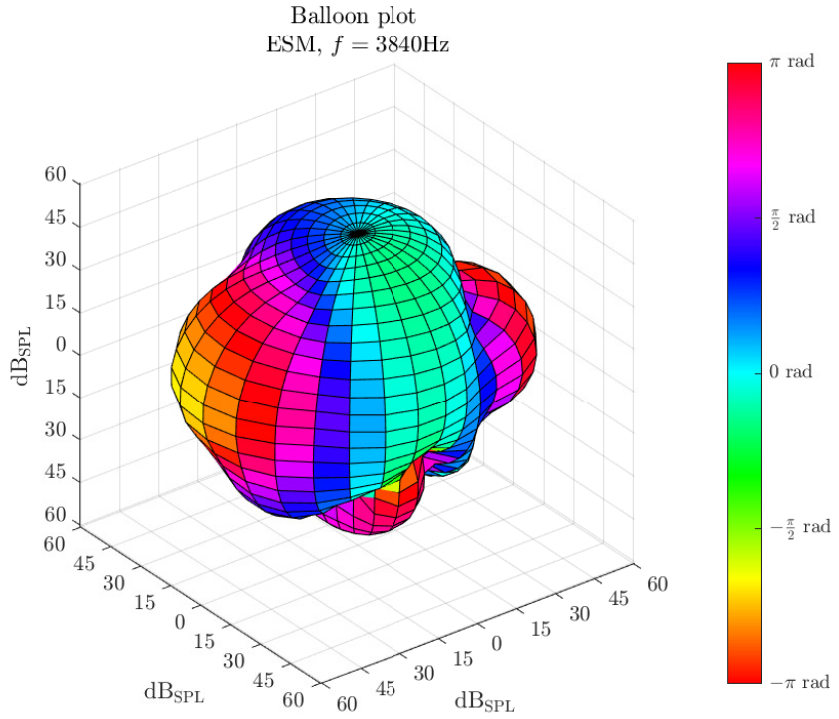


(b)

Figure D.15 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 1 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

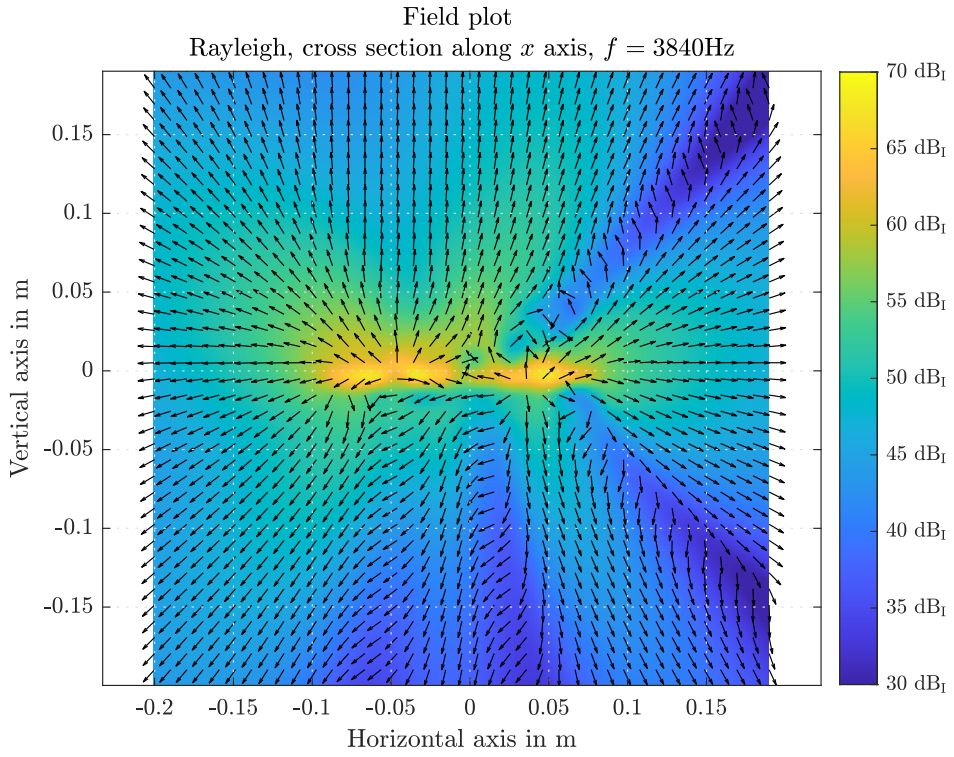


(a)

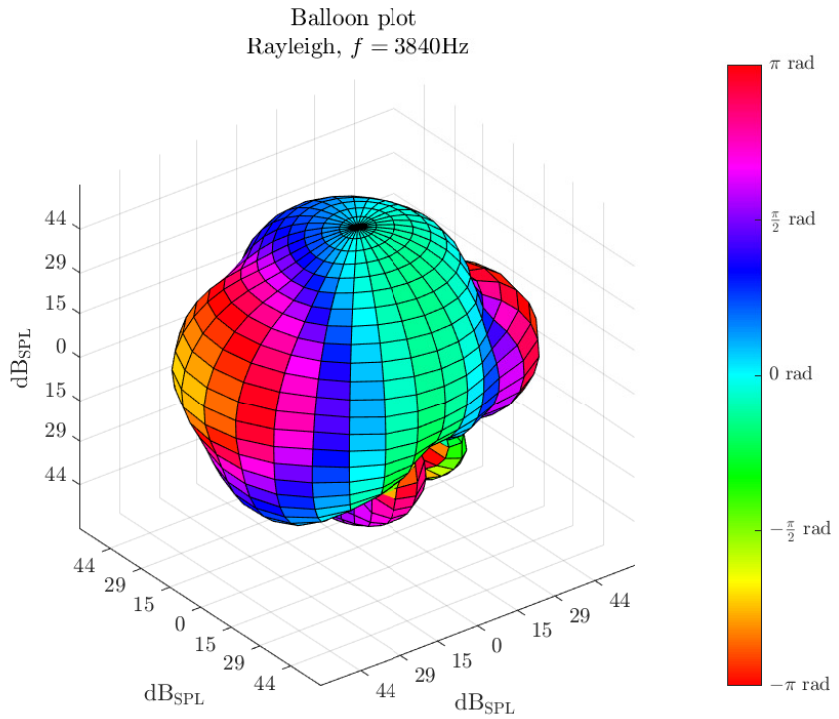


(b)

Figure D.16 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 3.84 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

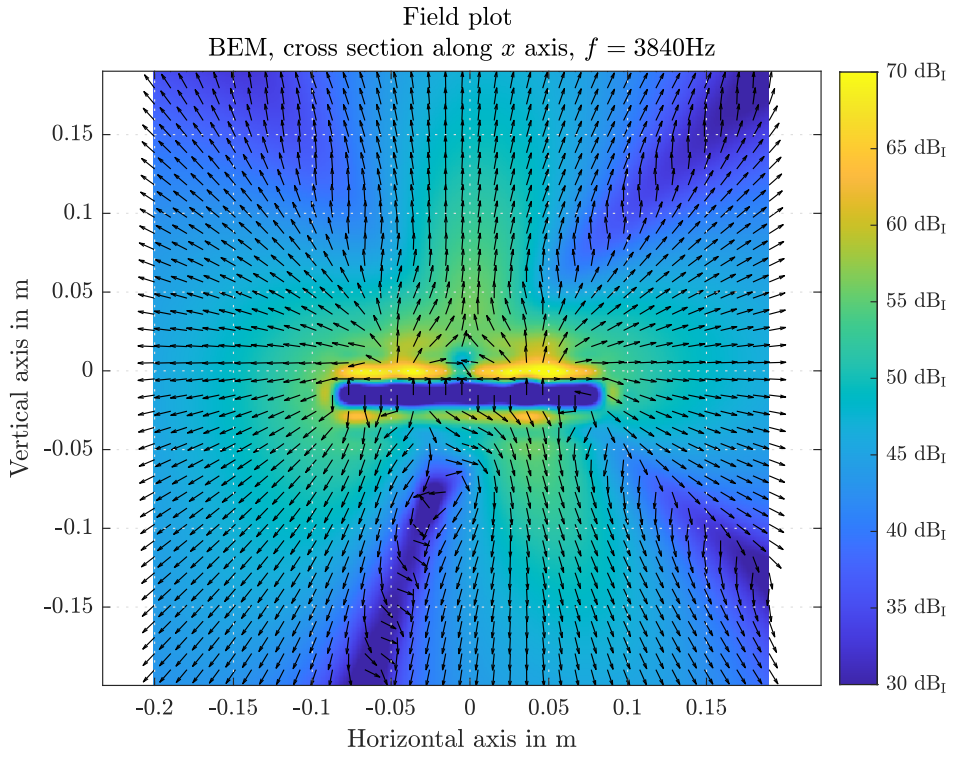


(a)

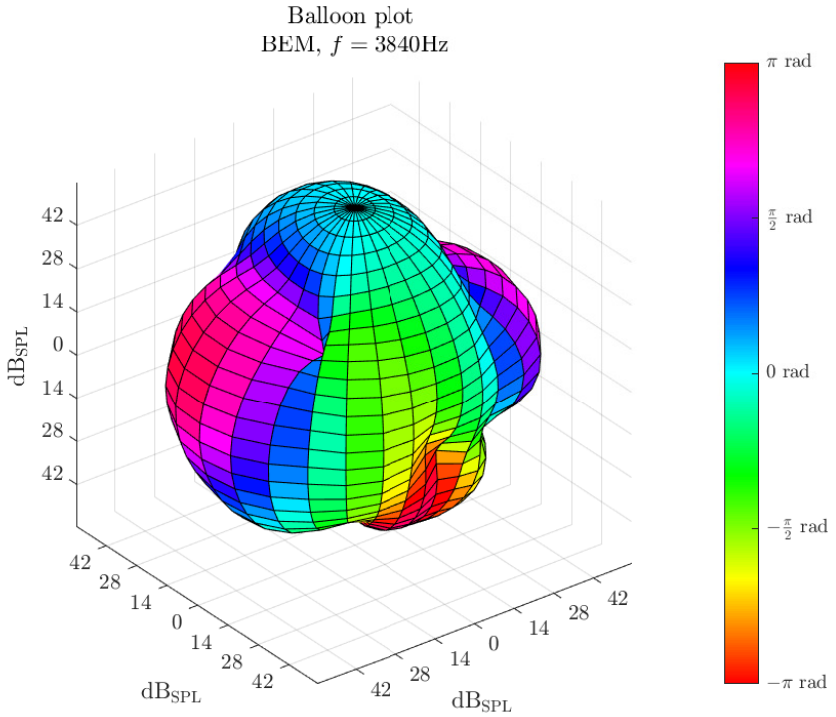


(b)

Figure D.17 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 3.84 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of* ± 20 cm. *(b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at* 30×30 *positions, at* $r = 0.1$ m *distance to the device's center. The axes of the plot are (left to right):* (z, y, x).

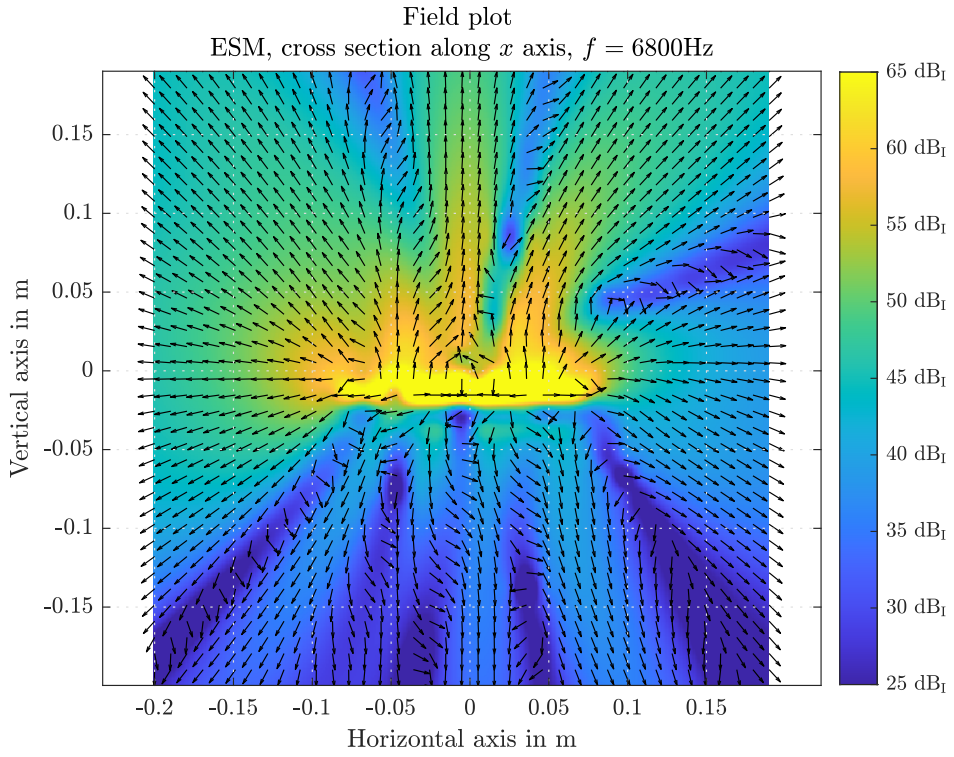


(a)

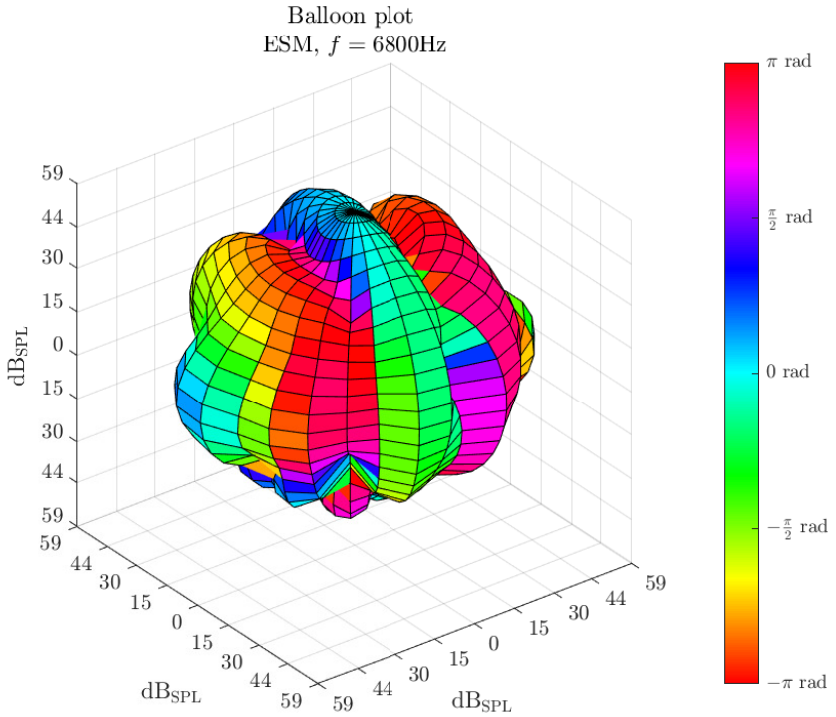


(b)

Figure D.18 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 3.84 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of* ± 20 cm. *(b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at* 30×30 *positions, at* $r = 0.1$ m *distance to the device's center. The axes of the plot are (left to right):* (z, y, x).

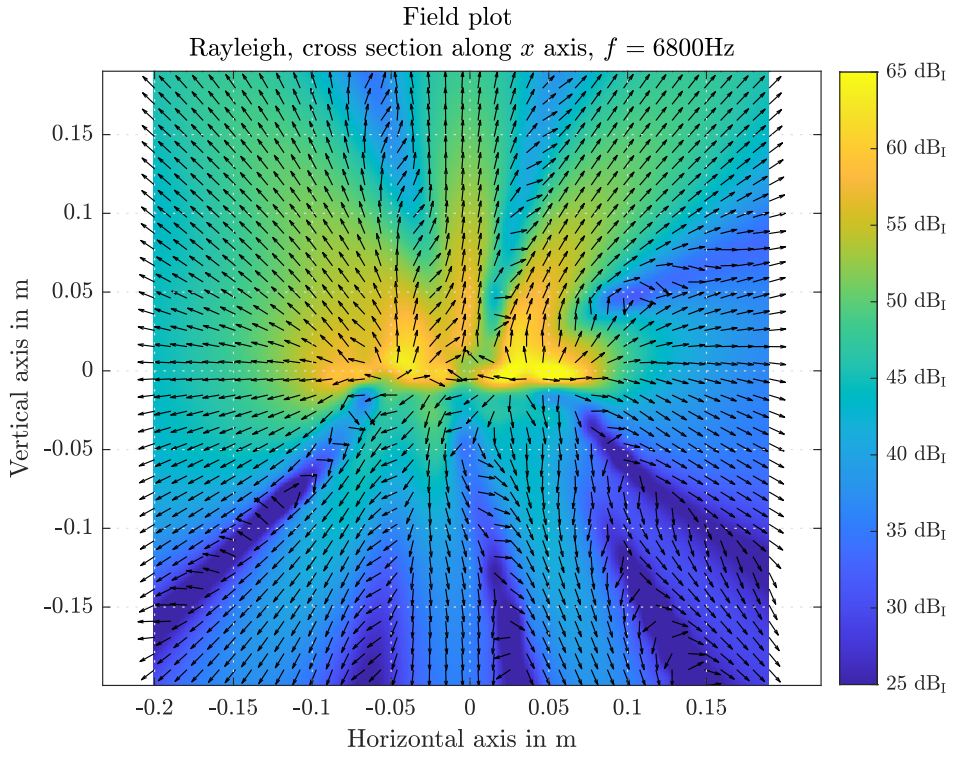


(a)

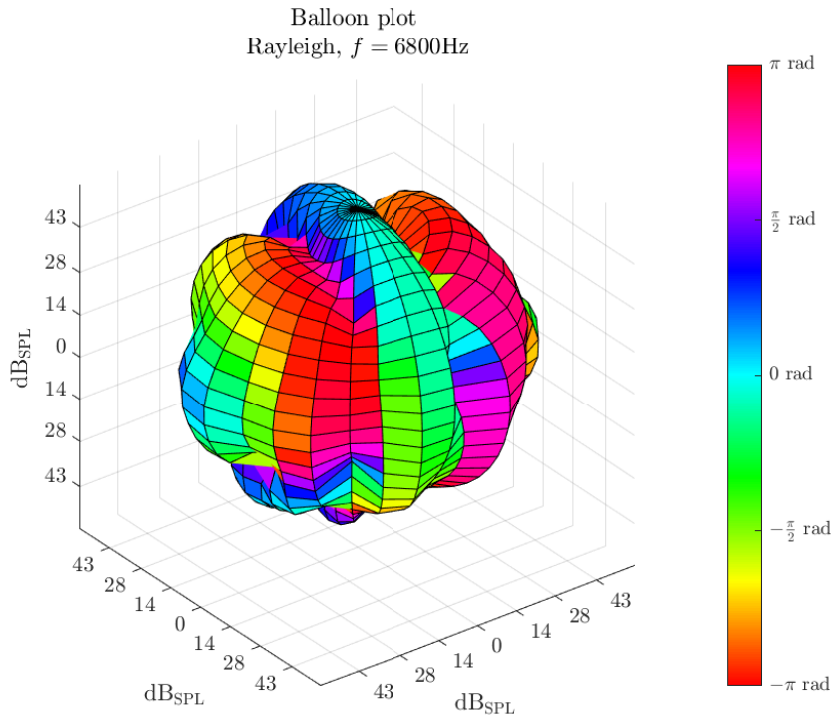


(b)

Figure D.19 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 6.8 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

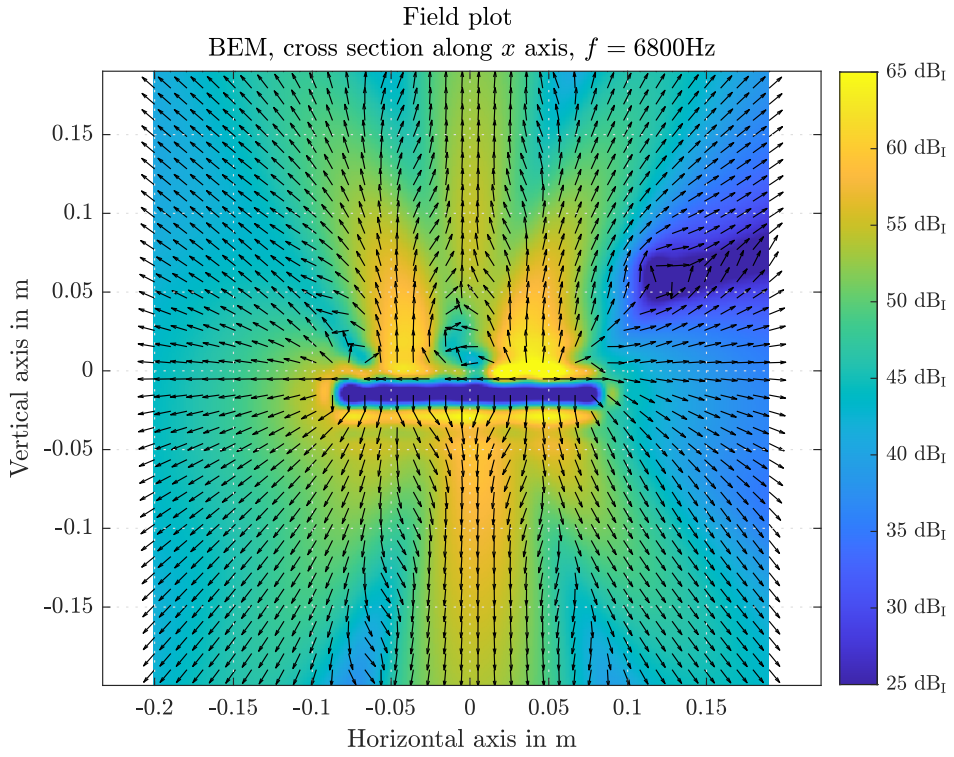


(a)

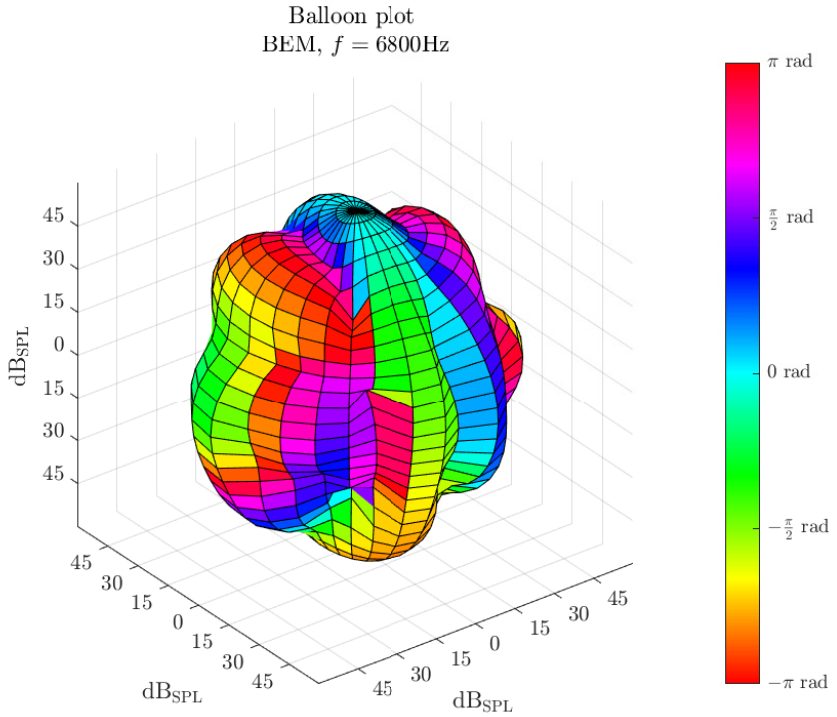


(b)

Figure D.20 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 6.8 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of* ± 20 cm. *(b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at* 30×30 *positions, at* $r = 0.1$ m *distance to the device's center. The axes of the plot are (left to right):* (z, y, x).

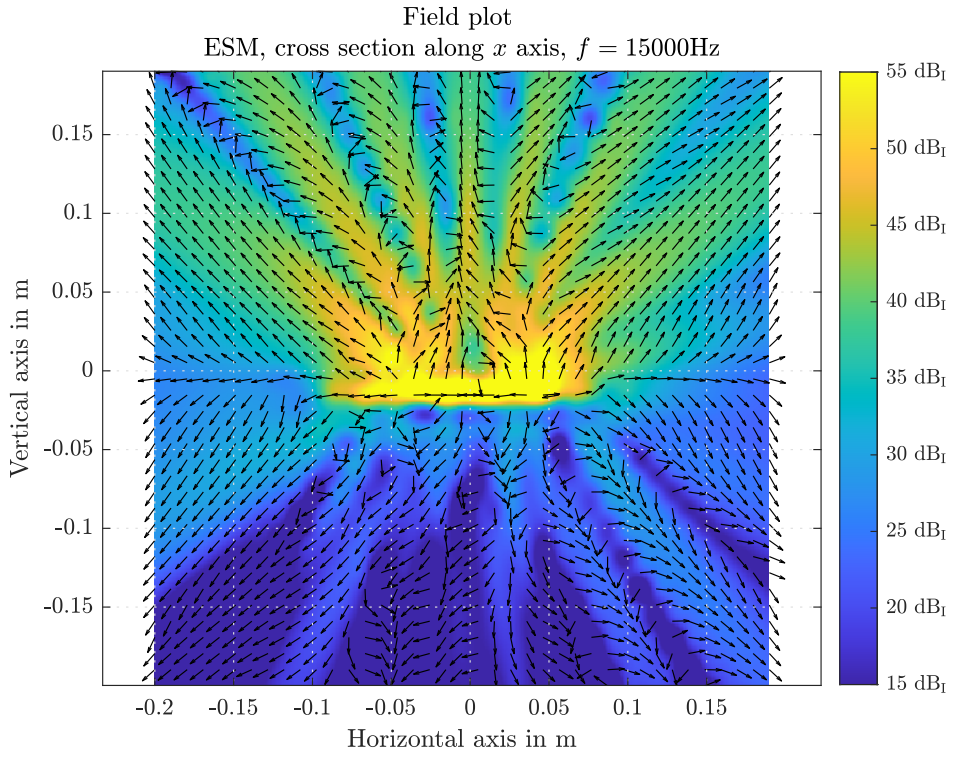


(a)

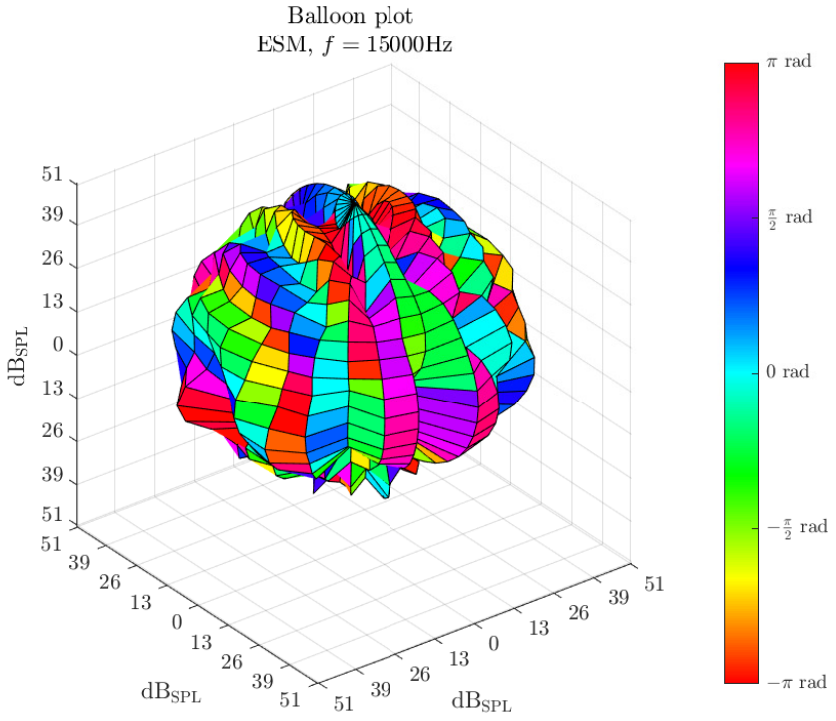


(b)

Figure D.21 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 6.8 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*

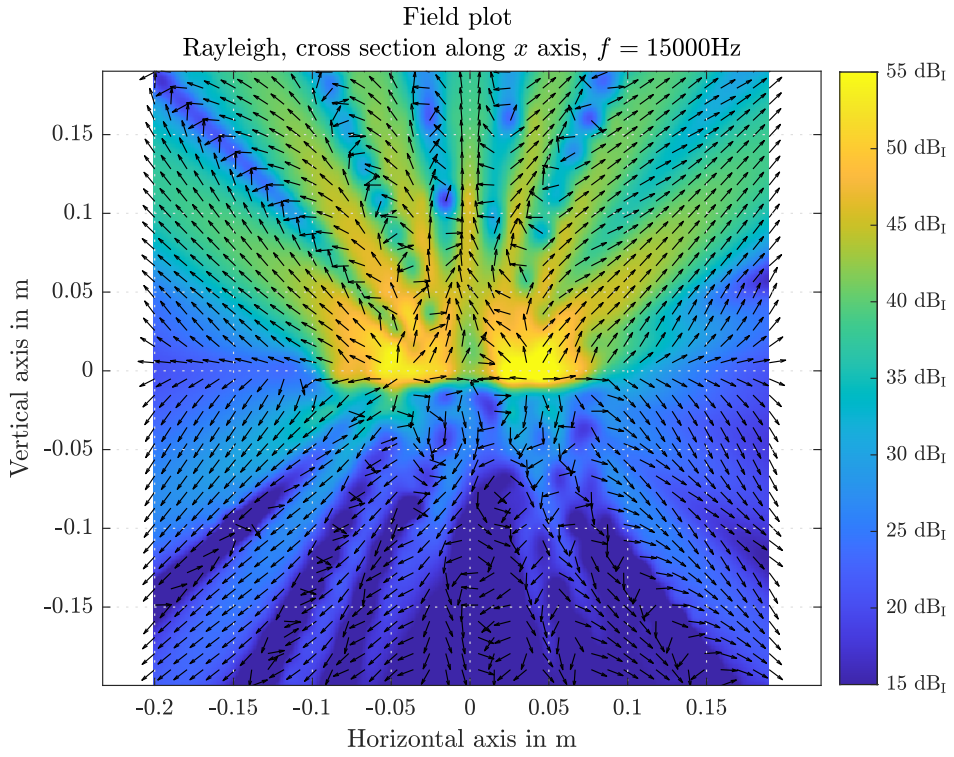


(a)

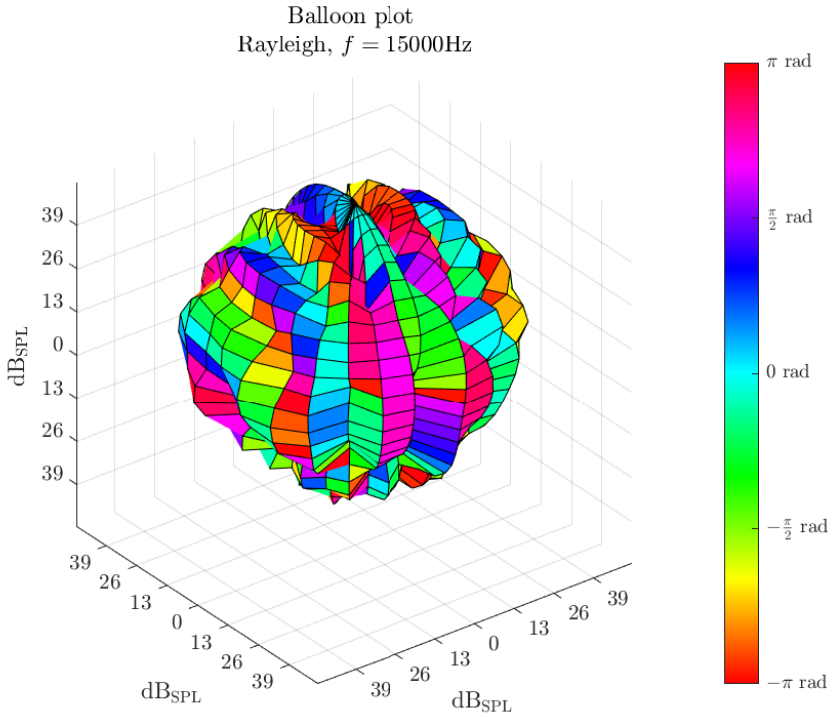


(b)

Figure D.22 *Spatial full-space sound field of the smartphone dummy, computed by the ESM, at* 15 kHz. *(a): acoustic intensity field of the cross-section along the* x *axis, at* $y = 0$. *The field is evaluated at a* (40×40) *grid in a range of* ± 20 cm. *(b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at* 30×30 *positions, at* $r = 0.1$ m *distance to the device's center. The axes of the plot are (left to right):* (z, y, x).

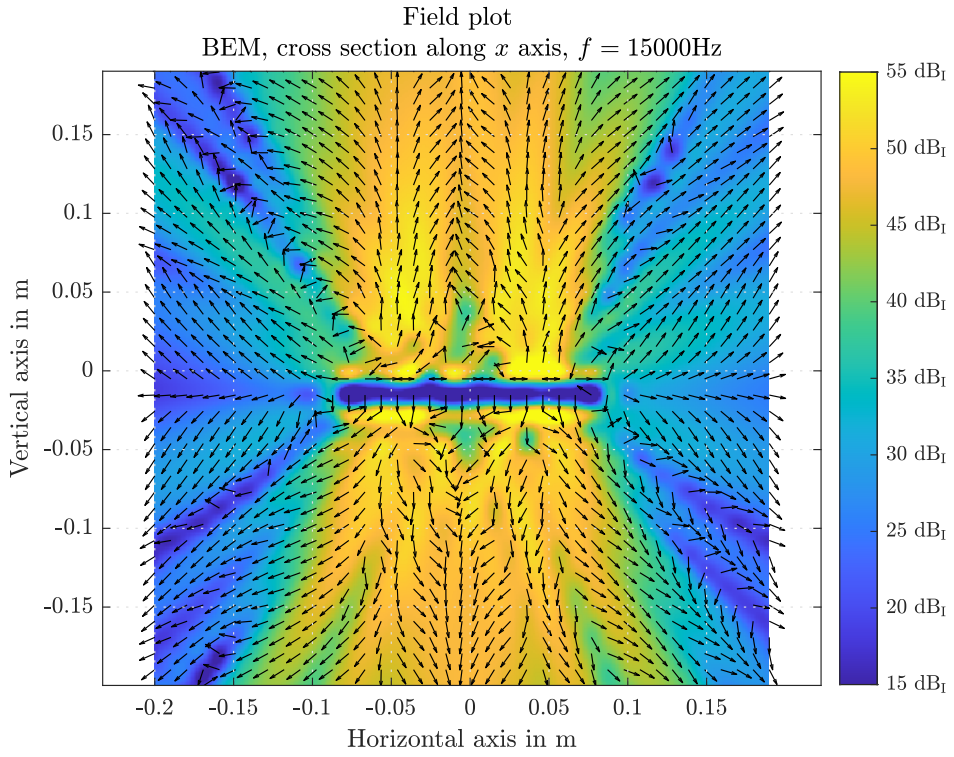


(a)

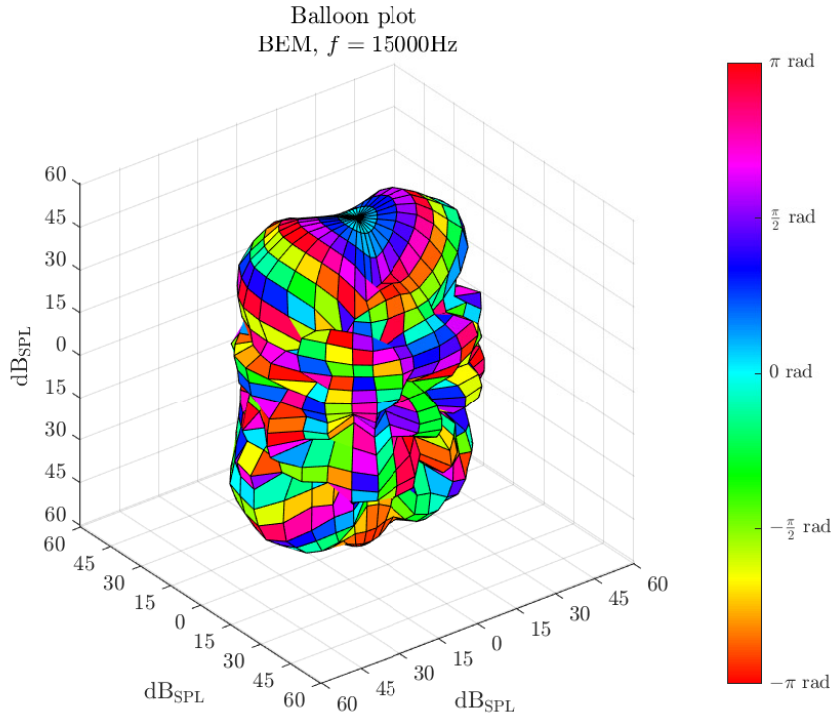


(b)

Figure D.23 *Spatial full-space sound field of the smartphone dummy, computed by the combined-Rayleigh formulation, at* 15 kHz. *(a): acoustic intensity field of the cross-section along the x axis, at $y = 0$. The field is evaluated at a (40×40) grid in a range of ± 20 cm. (b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at 30×30 positions, at $r = 0.1$ m distance to the device's center. The axes of the plot are (left to right): (z, y, x).*



(a)



(b)

Figure D.24 *Spatial full-space sound field of the smartphone dummy, computed by the BEM, at* 15 kHz. *(a): acoustic intensity field of the cross-section along the* x *axis, at* $y = 0$. *The field is evaluated at a* (40×40) *grid in a range of* ± 20 cm. *(b) balloon plot with angle-dependent coloration, plotted with balloonPlot(). The balloon plot is evaluated at* 30×30 *positions, at* $r = 0.1$ m *distance to the device's center. The axes of the plot are (left to right):* (z, y, x).

Bibliography

[1] J. Angus, "Distributed Mode Loudspeaker Polar Patterns", *Journal of the Audio Engineering Society*, Sep. 1999.

[2] V. P. Gontcharov and N. P. Hill, "Diffusivity Properties of Distributed Mode Loudspeakers", *Journal of the Audio Engineering Society*, Feb. 2000.

[3] M. J. Hawksford and N. Harris, "Spatial Bandwidth of Diffuse Radiation in Distributed-Mode Loudspeakers", *Journal of the Audio Engineering Society*, Nov. 2001.

[4] J. Panzer and N. Harris, "Distributed-Mode Loudspeaker Radiation Simulation", *Journal of the Audio Engineering Society*, Sep. 1998.

[5] M. Kuster, D. de Vries, D. de Beer, and S. Brix, "Structural and Acoustic Analysis of Multi-Actuator Panels", vol. 54, Nov. 2006.

[6] S. Lee, "Review: The Use of Equivalent Source Method in Computational Acoustics", *Journal of Computational Acoustics*, vol. 25, no. 01, p. 1 630 001, 2017.

[7] M. Munjal, F. Mechel, M. Vorländer, P. Költzsch, M. Ochmann, A. Cummings, W. Maysenhölder, W. Arnold, and O. Rudenko, *Formulas of Acoustics*, F. P. Mechel, Ed., ser. Formulas of Acoustics. Springer Berlin Heidelberg, 2004, ISBN: 978-3-540-42548-9.

[8] M. Ochmann and R. Piscoya, "The Source Simulation Technique with Complex Source Points for Computing Acoustic Radiation Problems", 2006.

[9] ——, *Theory and Application of Acoustic Sources Using Complex Analysis*. Springer Singapore, 2022, ISBN: 978-981-336-042-6.

[10] L. Song, G. H. Koopmann, and J. B. Fahnline, "Numerical errors associated with the method of superposition for computing acoustic fields", *The Journal of the Acoustical Society of America*, vol. 89, no. 6, pp. 2625–2633, 1991.

[11] R. Piscoya, H. Brick, M. Ochmann, and P. Költzsch, "Numerical aspects of the Equivalent Source Method applied to combustion noise", Jan. 2005.

[12] R. Jeans and I. C. Mathews, "The wave superposition method as a robust technique for computing acoustic fields", *The Journal of the Acoustical Society of America*, vol. 92, no. 2, pp. 1156–1166, 1992.

[13] L. Zhang, J. Wang, D. Yang, B. Hu, and D. Wu, "Optimization of the Equivalent Source Configuration for the Equivalent Source Method", *Journal of Marine Science and Engineering*, vol. 9, p. 807, Jul. 2021.

[14] Y. J. Gounot and R. E. Musafir, "Genetic algorithms: A global search tool to find optimal equivalent source sets", *Journal of Sound and Vibration*, vol. 322, pp. 282–298, Apr. 2009.

[15] M. R. Bai, C.-C. Chen, and J.-H. Lin, "On optimal retreat distance for the equivalent source method-based nearfield acoustical holography", *The Journal of the Acoustical Society of America*, vol. 129, no. 3, pp. 1407–1416, 2011.

[16] N. P. Valdivia, "Advanced equivalent source methodologies for near-field acoustic holography", *Journal of Sound and Vibration*, vol. 438, pp. 66–82, 2019, ISSN: 0022-460X.

[17] S. Kirkup. (). The Boundary Elements Method, [Online]. Available: http://www.boundary-element-method.com (visited on 05/10/2022).

[18] ——, "Fortran codes for computing the acoustic field surrounding a vibrating plate by the Rayleigh integral method", Oct. 2008.

[19] ——, (). The Rayleigh Integral Method in Matlab. Project, [Online]. Available: https://www.researchgate.net/project/The-Rayleigh-Integral-Method-in-Matlab (visited on 05/10/2022).

[20] M. Arunkumar, J. Pitchaimani, K. Gangadharan, and M. Leninbabu, "Effect of Core Topology on Vibro-acoustic Characteristics of Truss Core Sandwich Panels", *Procedia Engineering*, vol. 144, pp. 1397–1402, 2016, International Conference on Vibration Problems 2015, ISSN: 1877-7058.

[21] S. H. Hashemi, K. Khorshidi, and H. Rokni Damavandi Taher, "Exact acoustical analysis of vibrating rectangular plates with two opposite edges simply supported via Mindlin plate theory", *Journal of Sound and Vibration*, vol. 322, no. 4, pp. 883–900, 2009, ISSN: 0022-460X.

[22] S. Kirkup, "Computational solution of the acoustic field surrounding a baffled panel by the Rayleigh integral method", *Applied Mathematical Modelling*, vol. 18, pp. 403–407, Jul. 1994.

[23] D. Herrin, F. Martinus, T. Wu, and A. Seybert, "An assessment of the high frequency boundary element and Rayleigh integral approximations", *Applied Acoustics*, vol. 67, no. 8, pp. 819–833, 2006, ISSN: 0003-682X.

[24] YiminSun, EricVerschuur, and R. Borselen, "Acoustic propagation operators for pressure waves on an arbitrarily curved surface in a homogeneous medium", *Journal of Applied Geophysics*, vol. 150, Nov. 2017.

[25] S. Marburg, "Six boundary elements per wavelength: Is that enough?", *Journal of Computational Acoustics*, vol. 10, pp. 25–51, Mar. 2002.

[26] F. Zotter and S. Spors, "Is Sound Field Control Determined at All Frequencies? How Is It Related to Numerical Acoustics?", Sep. 2013.

[27] H. A. Schenck, "Improved Integral Formulation for Acoustic Radiation Problems", *The Journal of the Acoustical Society of America*, vol. 44, no. 1, pp. 41–58, 1968.

[28] A. F. Seybert and T. K. Rengarajan, "The use of CHIEF to obtain unique solutions for acoustic radiation using boundary integral equations", *The Journal of the Acoustical Society of America*, vol. 81, no. 5, pp. 1299–1306, 1987.

[29] S. Marburg, "Boundary Element Method for Time-Harmonic Acoustic Problems", in *Computational Acoustics*, M. Kaltenbacher, Ed. Cham: Springer International Publishing, 2018, pp. 69–158, ISBN: 978-3-319-59038-7.

[30] P. M. Juhl, "The Boundary Element Method for Sound Field Calculations", Thesis, The Acoustics Laboratory, TU Denmark, 1993. [Online]. Available: http://www.openbem.dk/docs/PMJ_Thesis.pdf (visited on 04/06/2022).

[31] P. M. Juhl, Ed. (2015). OpenBem, Open source Matlab codes for the Boundary Elements Method, [Online]. Available: http://www.openbem.dk (visited on 04/08/2022).

[32] (). Salome, The open source platform for numerical simulation, [Online]. Available: https://www.salome-platform.org (visited on 05/12/2022).

[33] M. Kaltenbacher, *Computational Acoustics*, ser. CISM International Centre for Mechanical Sciences. Springer, Cham, 2018, ISBN: 978-3-319-59038-7.

[34] L. Beranek and T. Mellow, *Acoustics: Sound Fields, Transducers and Vibration*, Second Edition. Academic Press, 2019, ISBN: 9780128152287.

[35] F. Fahy, "6 - Sources of Sound", in *Foundations of Engineering Acoustics*, F. Fahy, Ed., London: Academic Press, 2001, ISBN: 978-0-12-247665-5.

[36] A. D. Pierce, *Acoustics*, An Introduction to Its Physical Principles and Applications, Third Edition. 1994, ISBN: 0883186128.

[37] D. G. Duffy, *Green's Functions with Applications*, Second Edition, D. Zwillinger, Ed. CRC Press, 2015, ISBN: 978-1-4822-5103-6.

[38] G. B. Arfken, H. J. Weber, and F. E. Harris, *Mathematical Methods for Physicists*, A Comprehensive Guide, Seventh Edition. 2013.

[39] M. R. Bai, J.-G. Ih, and J. Benesty, "Appendix: Acoustic Boundary Element Method", in *Acoustic Array Systems: Theory, Implementation, and Application*. 2013, pp. 501–511. [Online]. Available: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470827253.app1 (visited on 04/08/2022).

[40] C. Moler, "Least Squares", in *Numerical Computing with Matlab*, ch. 5, pp. 139–163.

[41] F. Zotter and M. Frank, *Ambisonics, A Practical 3D Audio Theory for Recording, Studio Production, Sound Reinforcement, and Virtual Reality*. Springer Cham, 2019, ISBN: 978-3-030-17207-7. [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-17207-7 (visited on 05/24/2022).

[42] Mathworks, Ed., *MATLAB® Mathematics, user reference*, R2019b, 2019. [Online]. Available: https://de.mathworks.com/help/releases/R2019b/pdf_doc/matlab/math.pdf (visited on 04/15/2022).

[43] C. Moler, "Linear Equations", in *Numerical Computing with Matlab*, ch. 2, pp. 53–92.

[44] P. Schmidt, "Improvements in Localization of Planar Acoustic Holography", Master's Thesis, Institute of Electronic Music, Acoustics at University of Music, and Performing Arts, Graz, 2012. [Online]. Available: https://iem.kug.ac.at/fileadmin/media/iem/projects/2011/schmidt.pdf (visited on 03/14/2022).

[45] E. G. Williams, *Fourier Acoustics*, Sound Radiation and Nearfield Acoustical Holography, San Diego, Calif, 1999.

[46] O. Lezoray and L. Grady, *Image Processing and Analysis with Graphs*, Theory and Practice, First Edition, ser. Digital imaging and computer vision series. Baton Rouge: CRC Press, 2012, vol. 5, p. 562, ISBN: 9781138071766.

[47] M. Pagavino, Ed., *Alternative Diskretisierungen zur Auswertung des Rayleigh- Integrals basierend auf der Fourier-Methode*, 2019. [Online]. Available: https://phaidra.kug.ac.at/open/o:92197 (visited on 03/22/2022).

[48] J. H. McClellan, *DSP first*, Second Edition, ser. Matlab Curriculum Series. 2016, ISBN: 0132431718.

[49] G. Heinzel, A. Rüdiger, and R. Schilling, Eds., *Spectrum and spectral density estimation by the Discrete Fourier transform (DFT), including a comprehensive list of window functions and some new flat-top windows*. 2002. [Online]. Available: https://holometer.fnal.gov/GH_FFT.pdf (visited on 03/25/2022).

[50] A. W. Doerry, "Catalog of Window Taper Functions for Sidelobe Control", Sandia National Laboratories, SANDIA REPORT, 2017. [Online]. Available: https://www.researchgate.net/profile/Armin-Doerry/publication/316281181_Catalog_of_Window_Taper_Functions_for_Sidelobe_Control/links/58f92cb2a6fdccb121c9d54d/Catalog-of-Window-Taper-Functions-for-Sidelobe-Control.pdf (visited on 03/22/2022).

[51] L. Martens, H. Nijmeijer, I. Lopez Arteaga, and E. Moers, "Wave field extrapolation in spatial-domain", D & C Report, 2014. [Online]. Available: https://pure.tue.nl/ws/files/3791311/404695480215586.pdf (visited on 03/25/2022).

[52] I. Lopez Arteaga, R. Scholte, and H. Nijmeijer, "Improved source reconstruction in Fourier-based Near-field Acoustic Holography applied to small apertures", *Mechanical Systems and Signal Processing*, vol. 32, pp. 359–373, 2012, Uncertainties in Structural Dynamics.

[53] K. Saijyou and S. Yoshikawa, "Reduction methods of the reconstruction error for large-scale implementation of near-field acoustical holography", *The Journal of the Acoustical Society of America*, vol. 110, pp. 2007–23, Nov. 2001.

[54] R. Bremananth, "A Robust Extrapolation Method for Curtailed Aperture Reconstruction in Acoustic Imaging", *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 6, pp. 973–988, 2012.

[55] L. Gaul, M. Kögl, and M. Wagner, *Boundary Element Methods for Engineers and Scientists, An Introductory Course with Advanced Topics*. 2003.

[56] C. Sobtzick, "Optimierte Schallwand für akustische Messungen an Miniaturlautsprechern", Masterarbeit, Institut für Signalverarbeitung und Sprachkommunikation der Technischen Universität Graz, 2014.

[57] C. J. Struck and S. F. Temme, "Simulated Free Field Measurements", *Journal of the Audio Engineering Society*, vol. 42, no. 6, pp. 467–482, Jun. 1994.

[58] A. Savitzky and M. J. E. Golay, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures", *Analytical Chemistry*, 1964. [Online]. Available: http://xrm.phys.northwestern.edu/research/pdf_papers/1964/savitzky_analchem_1964.pdf (visited on 03/02/2022).

[59] (). Gmsh, A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, [Online]. Available: https://gmsh.info (visited on 05/17/2022).