MASTER THESIS IN PHYSICS/ACOUSTICS

# Finite element modeling and experimental characterization of piezoelectric ceramic disk in air

Espen Fosse

June 2022

**UNIVERSITY OF BERGEN**

Department of Physics and Technology

NORWAY

# Abstract

Understanding the piezoelectric disk characteristics and behaviors is essential to achieve a good system model and can increase accuracy and reduce measurement uncertainties. This work uses finite element modeling to compare simulations to experimentally obtained measurements of electrical and acoustic characteristics of the piezoelectric disk Pz27.

The electrical measurements are performed with an impedance analyzer, which measures the electrical properties of the piezoelectric disk conductance and susceptance. For the acoustic measurements of the directivity, on-axis pressure, and 2-D horizontal pressure field of the piezoelectric disk, a MatLab app has been developed to automate the measurements and control the acoustic measuring setup. The sensitivity of the directivity related to small dislocations of the transducer has been studied.

The finite element simulations compared to experimentally obtained results agree well, but somewhat low signal-to-noise ratio for some of the frequencies are observed. These results imply that the finite element simulation software used in this work proves to be a good tool for predicting measurements.

# Acknowledgment

I want to express my gratitude to my supervisor, Prof. Per Lunde, and co-supervisors Mathias Myrtveit Sæther and Assoc. Prof. Emer. Magne Vestrheim for their guidance, informative meetings, and remarks that have been helpful to me in completing this master's thesis.

I want to thank Sverre Kongsro Finstad for his introduction to the acoustic measurement setup. I want to give a big thanks to Eivind Nag Mosland for helping me answer questions on FEMP, sharing knowledge of previously used scripts in the acoustic measurement setup, and providing general information about instruments in the acoustic measurement setup. I also want to thank Assoc. Prof. Audun Oppedal Pedersen for always keeping his door open for questions and discussions that have been very useful. I thank Philip Trætteberg for sharing knowledge, fruitful discussions, and motivation.

Lastly, I want to thank my family and friends, and my biggest thanks to my partner for her patience and support and for taking extra good care of our son in times I have worked a lot.

E.F.

Bergen 2022

# Contents

# Chapter 1

# Introduction

## 1.1 Background and motivation

The use of ultrasound in industries, commercial products, and science is numerous. In these areas, ultrasound applications can be topography mapping the sea floor [58][34], detecting cracks in pipes both on and offshore [77][10][74], fiscal flow measuring [33][3], pregnancy check [75], or motion detections [59]. A piezoelectric material is often used in such applications due to the material's ability to transmit or receive ultrasound due to the piezoelectric effect and with given structural dimensions [73]. Therefore, there is a need to understand the behaviors and characteristics of these piezoelectric materials. A good understanding can increase accuracy and reduce uncertainties in measurements. Fiscal flow measurements for exporting and selling gas [3][33][69][43] are one field where accuracy and low uncertainties are crucial. Suppose ultrasonic flow meters are chosen over non-ultrasonic. In that case, it is often due to one or more reasons, such as reasonable purchase price, operation, maintenance, and installation costs, or the equipment is easy to use [45]. Ultrasonic gas meters use several measurement methods such as transit time, doppler, and correlation [30]. The more commonly used method is the transit time. This method measures the time difference of sound propagation between the transmitter and receiver caused by fluid velocity and produces high accuracy measurements [14].

To better understand an ultrasound measurement system, it is necessary to describe a complete system model beyond the transmitter and receiver [72]. A complete system model often consists of a computer, signal generator, transmitter electronics, transmitter transducer, propagation medium, receiver microphone/transducer, receiver electronics, and termination at oscilloscope [72]. Describing this system model as a whole or as individual parts can theoretically improve the understanding of the measurement system considerably. It also opens up to analyze each individual part of the system model and optimize parameters within each individual block, which further leads to improvements in the measurement system as a whole.

Understanding the piezoelectric element characteristics and behaviors is essential to achieving a good system model. In a transducer construction, piezoelectric elements are the main component to transmit and receive ultrasound. Since piezoelectric elements are often only a part of several components in a larger transducer structure, other components can be used, for example, to improve the impedance matching of the transducer structure to the medium [72]. A transducer construction could be matching layers, backing, piezoelectric material, and housing designed to increase the efficiency of transmitting and receiving ultrasound in a given medium [73]. Before designing a transducer structure, it is essential to have appropriate piezoelectric and material parameters to perform sufficiently good finite element simulation approximations, such as admittance, axis pressure, and directionality, that will largely match measurements.

Simulating piezoelectric elements requires knowledge and control over the parameters of the surrounding mediums, materials, and elements. Several years of research on the parameters of the element used in the present work have provided good approximations of the parameters. These good approximations come from comparing finite element modeling with actual measurements and fine-tuning the parameters. This method is well established within the research community when studying transducer elements.

## 1.2 Previous work

Knowledge and a good understanding of a transmit-receive ultrasonic measurement system are necessary to predict the measurement system's behaviors and electrical and acoustical characteristics. After years of research on transmitter receiving ultrasonic measurement systems, the developed modeling software FLOSIM was investigated for use in simulations of 1-D ultrasonic transit time systems. These investigations concluded that it would be an effective tool in developing future ultrasonic transit time flowmeters [44]. In later years, software such as FLOSIM (1-D), FEMP (2-D), and COMSOL (3-D) is used extensively to simulate and predict various parts of a transmit-receive ultrasonic measuring system. Examples of works that have used these different software are [71][44][49][22].

Work done by Benny et al. [11] investigates a method to predict and measure the acoustic radiation induced by ultrasonic transducers radiating in the air for frequencies less than 1 MHz. This work used a laser vibrometer to map the surface displacement. This surface displacement is used to predict the 2-D sound field, and the predicted sound field is compared with finite element modeling (FEM) simulations and measured pressure using an ultrasonic detector. These predictions and measurements were compared in the near-field range of $\pm$20mm in the radial direction and from 5 to 300 mm in the axial direction relative to the transducer surface. The results gave good agreement between the theory and experimental obtained results. With this method of predicting the sound field of the transducer, problems arising from standing waves and non-planar waves in near fields are avoided.

Sanabria et al. [62] use a closed reradiation method that combines the Rayleigh-Sommerfeld integral and time-reversal acoustics, which allows for calculating the entire near and far-field based on a single 2-D sound pressure field measurement. This sound pressure field is produced by an air-coupled ultrasound (ACU) probe, where measurements are done in a plane parallel to the probe's surface in the near field, and measurements are performed with a calibrated microphone. This method works for both 3-D (circular, square) and 2-D (rectangular) planar transducers in the frequency range from 50 to 230 kHz, and it is stated that this method outperforms baffled piston models.

The article by Øyerhamn et al. [49] developed a model describing a transit-receiver measurement system based on the transducer's radial mode operating in a homogeneous fluid medium. It uses axisymmetric FEM of two piezoelectric ceramic disks as transmit receiver pair and sound propagation in the medium air. The simulated model is compared with experimental measurements of the transmit-receive voltage-to-voltage transfer function in the frequency range over the two first radial modes of the two disks and at 1 atm. It is also compared simulation with the time domain electrical terminals voltages signals of the transmitting and receiving transducers. Compared to the measurements, this model simulation of the measurement system provides good agreement, but improvements are potentially identified and discussed.

An article by Chillara et al. [22] studies the generation of ultrasonic Bessel beams from radial modes to a piezoelectric disk transducer. Laser Doppler Vibrometry is used to measure the Bessel vibration pattern of the radial mode, which is found to agree well with the numerical simulations. The beam profiles from the four first radial modes of the disc are measured with a hydrophone in a water tank and compared to predicted results obtained from the analytical model, and they are in good agreement. These experimental measurements use similar methods of measuring directivity, on-axis pressure, and 2-D sound pressure field as the measurements conducted in this work.

There are few articles published regarding radial modes and characterization of the piezoelectric disk's radial modes. This leads to the majority of the articles covered in this section being written about complete transducer constructions and other transducers then used in this work. However, several of the measurement methods and models are similar to what has been done in this work.

### 1.2.1 Previous work at UiB

Over the years, much work has been conducted at the University of Bergen with the acoustical measurement setup, focusing on investigating the behaviors of piezoelectric disks and transducers in transmitting and receiving ultrasound in the air. The primary investigation by the work performed by Storheim [65] is diffraction correction of non-uniformly vibrating sources. These sources are baffled uniform piston, baffled piezoceramic disk, unbaffled piezoceramic disk, and side baffled piezoceramic disc. It is used finite element (FE) to simulate and investigate these sources and then compares them to numerical and analytical expressions and electrical and acoustic measurements. The diffraction correction behavior is defined by the sound pressure generated by the source simulations and compared to the widely used baffled piston diffraction correction. The FE simulation program used to perform the simulations was developed by Kocbach [38] in cooperation between the University of Bergen (UiB) and Christian Michelsen Research AS (CMR) and called Finite Element Modeling of Piezoelectric structures (FEMP).

The work of Mosland and Hauge [48][29] is partially co-written. Mosland's primary focus is developing and implementing a modified three-transducer reciprocity calibration method for use in the surrounding fluid air in a frequency range of 50-300 kHz. This calibration method includes correction due to absorption in air, diffraction effects, and transmitting and receiving electronics. The experiment's final results are compared to simulations and acoustical measurements with a calibrated condenser microphone. The produced transmitting voltage response result by Mosland is given in Fig. 1.1 [48] and is used in this work's discussion of signal-to-noise results. Hauge's primary focus is developing and implementing an FE-based linear system model that describes an air measurement system with arbitrary distances between transmitter and receiver and compares the system model with experiments performed in air.

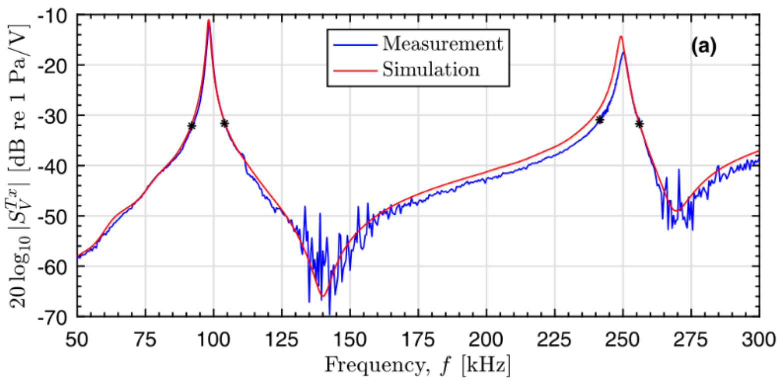



Figure 1.1: Comparison of the measurement and simulation of the transmitting voltage response $|S_V|$ for the transmitting piezoelectric disk Pz27 with d/t = 10 and plotted from 50 kHz to 300 kHz. The boundaries between "×" in the resonance peaks is $V_{0_{pp}}$ = 2 V and $V_{0_{pp}}$ = 20 V elsewhere. The result is taken from [48].

Andersen [7] used the three-transducer reciprocity calibration method to calibrate two piezoelectric disks for air in a frequency range of 50-300 kHz and compared the experiment's final results with FE simulations. Andersen [7] also implemented lasers to conduct high-accuracy separation measurements between the two disks.

The work of Søvik [64] is based upon the work of [48][29], where it is further developed the FE-based linear system model describing a measurement system for gas. This further development includes the slowly varying phase in the air at 1 atm, with room temperature and assumption of no airflow, and then compared to experimental measurements.

Hagen [27] is studying improvements to the work of [7] measurement system to simplify the measurements of the system model's transmission functions at short distances between transmitter and receiver. It is presented a method to reduce crosstalk, and the experimental result performed is compared to FE simulation by using both FEMP and COMSOL.

Work conducted by Grindheim [26] investigates and measures the transfer function of a transmit-receiver system and compares results with FE simulations. It is provided good similarities between calculated and simulated results, but misalignment causes uncertainty in the transfer function $H^{VV}_{15_{open}}$ amplitude, especially around the second radial mode.

The work executed by Finstad [24] measures the electrical characteristics with an impedance analyzer and acoustical characteristics with a condenser microphone in the air with a piezoelectrical disk Pz27 and compares results to FE simulations. The frequency of interest covers the two first radial modes of the disk, where the electrical measurement results show good comparison with FE simulations. For the acoustical measurements of directivity, it shows good similarities compared to FE simulations. However, for the acoustical measurement results of on-axis pressure and transmitter sensitivity, a deviation of around 8 dB is seen.

## 1.3 Objectives

The main objectives of this work are to experimentally measure the electrical and acoustical characteristics of a circular piezoelectric disk of type Pz27 with a diameter/thickness (d/t) ratio of 10, and measure in the frequency range covering the two first radial modes, then compare the measured results with the FE simulations performed in FEMP. Other primary objectives of this work are to improve and facilitate the positioning of the piezoelectric disk in the measurement setup and to automate and reduce the measuring time of the measurements.

The electrical characteristics measurements are the conductance and susceptance, used to calculate the admittance, and measurements are conducted in the frequency range of 1-300 kHz. The acoustical characteristics measurements are the on-axis pressure, directivity, and 2-D horizontal pressure field in the near and far field of the piezoelectric disk.

These acoustical measurements are performed with several different frequencies covering the disk's first radial mode, and directivity measurements also cover the second radial mode. All measurements are then compared with FE simulations of a modeled piezoelectric disk radiating in air. The FE simulation program used to perform the simulations is called FEMP and was developed by Kocbach [38] in cooperation between UiB and CMR. The current version of FEMP is 6.1.

The method for improving and facilitating the positioning of the piezoelectric disk is firstly done by building a MatLab app that controls all stages in the measurement setup. Secondly, creating a setup guide to find the origo on the piezoelectric disk's front center with the app significantly reduces the setup time. It allows for visual control over the position of the disk via the app by having control of the absolute position of the stages and by knowing the disk's position. The necessity of automating the measurements is due to work done by Finstad [24], which states that acoustical measurements are highly time-consuming, especially for the 2-D pressure field. Integrating communication with all instruments in the experimental setup through the app and controlling the disk's position makes it possible to automate the measurements.

Method for analyzing the measurements includes Fourier transforming the measured signal from time to frequency domain with Matlab's fast Fourier transform (FFT) function. This transformation opens up to extracting the peak-to-peak voltage amplitude of the transmitting frequency. This method of finding the voltage is used when calibrating the microphone sensitivity, where microphone sensitivity allows for converting the measured voltage into pressure. This method for calculating pressure is used when comparing simulated pressure with measurements.

## 1.4 Thesis outline

Chapter 2 presents the theoretical background and equations used in this work. Chapter 3 presents the instruments used, the experimental setup of electrical and acoustic measurements, and the method for analyzing measurement results. Chapter 4 presents the method of finding the correct position of the piezoelectric disk relative to the microphone and the developed MatLap app. Chapter 5 presents the finite element method to simulate the piezoelectric disk Pz27 vibrating in a vacuum and fully immersed in air. Chapter 6 presents the results obtained with the FE simulations method, the results obtained from the measurements of the electrical characteristics, the results obtained from the measurements of the acoustical characteristics, and a discussion of all results. Chapter 7 gives a conclusion and suggestions for further work.

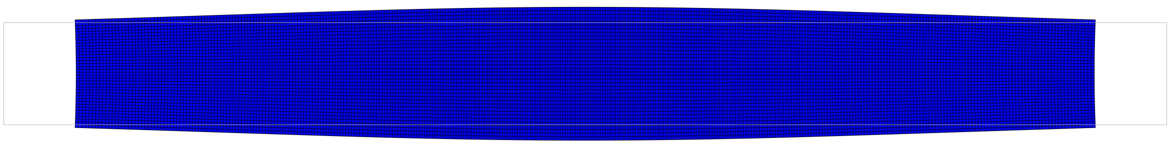
# Chapter 2

# Theory

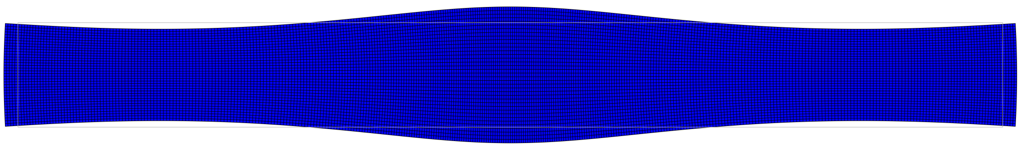
This chapter presents the theoretical background and equations used in this work. Sect. 2.1 goes through modes and resonances in the piezoelectric disk and the resonances central to this work. Sect. 2.2 describes the system model used in the present work. Sect. 2.3 presents the Fourier transformation theory and its use in this work. Sect. 2.4 presents how the speed of sound is calculated and used to estimate signal arrival time and length. Sect. 2.5 goes through the coordinate system used. Sect. 2.6 describes the theory of absorption in air. Sect. 2.7 presents the transfer functions of transmitter and receiver electronics to calculate electrical corrections. Sect. 2.8 represents the theory behind the calculations of microphone sensitivity. Sect. 2.9, the last section in this theory chapter, summarizes the theory behind finite element modeling.

## 2.1 Modes in the piezoelectric element

The vibration of a piezoelectric element shaped like a disk with electrodes on each end surface can be described as a weighted superposition of eigenmodes, either with an applied voltage or displacement [73]. For large d/t ratios, where d is the diameter of the disk and t is the thickness of the disk, studies show that the piezoelectric disks can oscillate in two kinds of radial modes (R-modes and A-modes) and four types of thickness modes (T-mode, TE-mode, TS-mode, and E-mode) [41][38]. The type of modes of interest in this work is the two first radial extension modes (R-modes), which correspond to standing waves in the radial direction, as illustrated in Fig. 2.1. These modes are the lowest kinds in piezoelectric disks [28], where R1 is the fundamental radial mode, and R2 is the second radial mode. These first two modes occur at 98.1 kHz and 249 kHz for the simulated Pz27, as shown as an example in Fig. 2.2, with a d/t ratio matching the element used in this work, see Table 5.3. The displacement will be at its maximum at these resonance frequencies and is known as the serial resonance $f_s$ [25].



(a)



(b)

Figure 2.1: Simulated displacement of the piezoelectric disk's Pz27 two first radial extension modes with a d/t ratio ≈ 10 mm **(a)** The fundamental radial mode R1, occurring at 98.1 kHz **(b)** the second radial mode R2, occurring at 249 kHz.

Because of the maximum displacement, the piezoelectric disk transmits a maximal amount of mechanical energy leading to an optimal transmission performance [25]. By performing measurements of the admittance $Y$ and over the frequency span from 1-300 kHz, which covers the two first radial modes of the disk used in this work, the series resonances of the piezoelectric disk occur when conductance $G$ is at its maximum [32]. The expression for the admittance is

$$Y(f) = G(f) + iB(f) = \frac{1}{Z(f)} = \frac{1}{R(f) + iX(f)} \tag{2.1}$$

where $B$ is the susceptance and the unit for admittance is Siemens [S], $Z$ is the impedance, $R$ is the resistance, $X$ is the reactance, and the unit for impedance is ohm [Ω], and $f$ is the frequency.

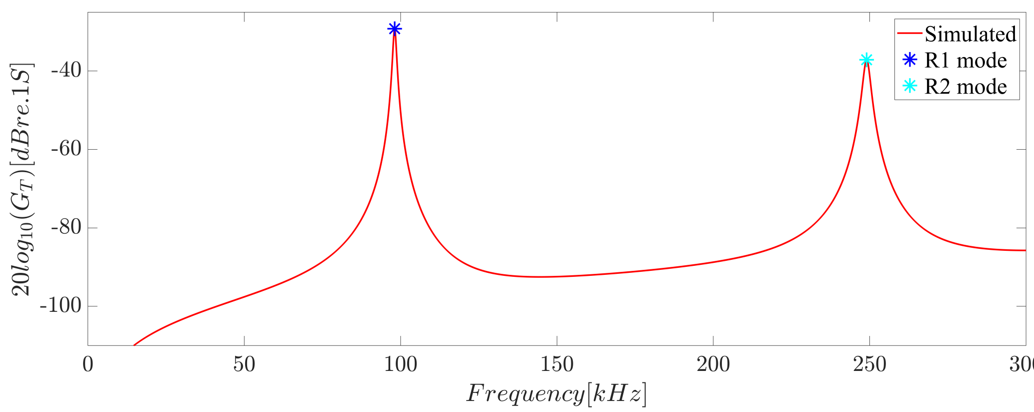



Figure 2.2: Simulated $G_T(f)$ of a piezoelectric disk with matching d/t ratio of disk used in present work and markings of the corresponding R1 and R2 modes of the max$(G(f)_T)$.

## 2.2 System Model

The system model is used to describe the electro-acoustic signal propagation chain via an ultrasonic measurement system, all the way from the signal generator to the measuring instrument consisting of an oscilloscope [49]. The system model in this work, shown in Fig. 2.3, consists of modules connected by nodes. These connections between the modules generate inputs and outputs. With the assumption that the system can be described as linear relations, the different nodes can be characterized by transfer functions [72]. An example of such a transfer function is

$$H_{12}^{vV} = \frac{v_2}{V_1}\,, \tag{2.2}$$

where $V_1$ is the voltage over the transmitter and $v_2$ is the particle velocity normal to the transmitter surface. Each node in Fig. 2.3 represents either the input or the output variables for the associated modules [72].

| **node** | |
|---|---|
| **0** | $V_0$ is the generated output voltage by the signal generator. |
| **0m** | $V_{0m}$ is the measured voltage at the oscilloscopes channel one or seen as the voltage into the transmitting electronics. |
| **1** | $V_1$ is the voltage over the electrodes of the transmitter or seen as the voltage out from transmitting electronics. |
| **2** | $v_2$ is the particle velocity normal to the transmitter surface. |
| **3** | $p_3$ is the free field, on axis pressure in the medium. |
| **4** | $p_4$ is the free field on axis pressure at the front surface of the receiver. |
| **5** | $V_5$ is the voltage out from the receiver or seen as the voltage into the receiver electronics. |
| **5m** | $V_{5m}$ is the measured voltage at the oscilloscopes channel two or seen as the voltage out of the receiver electronics. |

A piezoelectric ceramic disk Pz27 is used as the transmitter, and transmitter electronics couples the transmitter to the oscilloscope, which receives a generated signal from the signal generator. The receiver in this work is a pressure-field microphone, which measures the received signal, and induces a voltage. The receiver electronics, consisting of an amplifier and a filter, before being terminated in the oscilloscope.

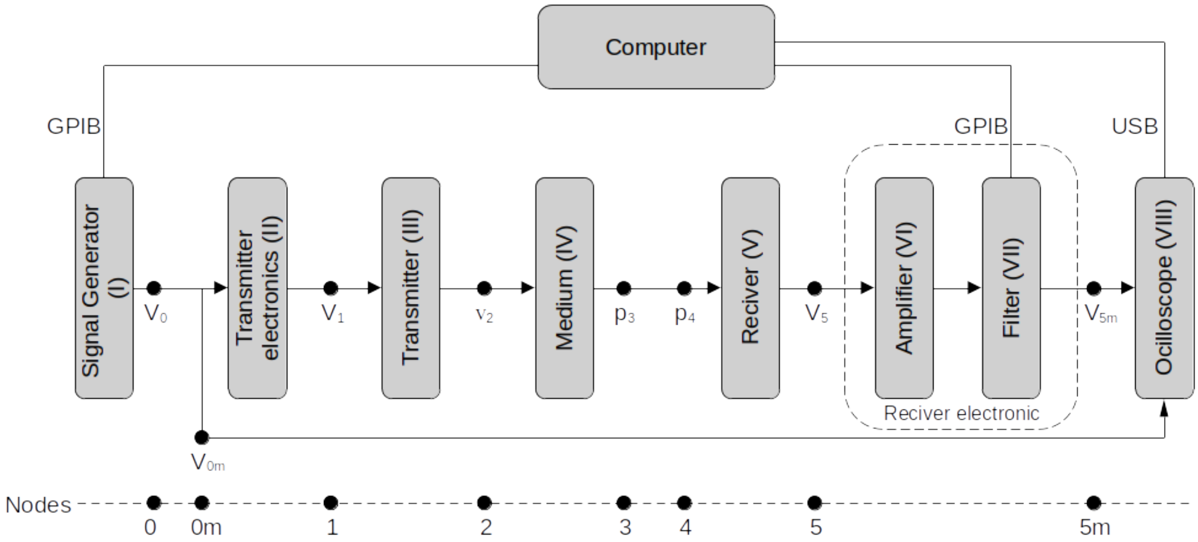



Figure 2.3: Representation of the system model used in this work in the form of a block diagram which is based on [72]

When analyzing the voltages $V_{0m}(t)$ and $V_{5m}(t)$ measured at the oscilloscope, the voltages are Fourier transformed from the time domain to the frequency domain. This transform enables voltage extraction $V(f)$ for its corresponding frequency spectrum and acts as an additional frequency filter.

## 2.3 Fourier transform

When measuring a signal on an oscilloscope, the amplitude is dependent on the time in the time domain. If the amplitude of interest lies in a specific frequency, such as the transmitting frequency, Fourier transform (FT) is used to extract the information on the amplitude for that given frequency in the frequency domain. This method converts the measured voltage signal from time domain $V(t)$ to frequency domain $V(f)$. There is also a method to reverse this process by taking the inverse Fourier transform (IFT), which takes the signal from the frequency domain $V(f)$ to the time domain $V(t)$. Both these transformations, FT and IFT, can mathematically be expressed [15] as

$$V(f) = FTV(t) = \int_{-\infty}^{\infty} V(t)e^{-i2\pi ft}dt \tag{2.3}$$

$$V(t) = IFTV(f) = \int_{-\infty}^{\infty} V(f)e^{i2\pi ft}df \tag{2.4}$$

In the present work, the FFT algorithm in MatLab is used [68], which is just an efficient and fast computation method of discrete Fourier transform (DFT), and returns both negative and positive frequencies. The negative frequency values correspond to the conjugates of the positive frequency values. The FFT algorithm also uses zero-padding, which increases the length of the signal time in the form of an array of zero amplitude. This array of zeroes increases the bins in the signal, increasing the frequency resolution of the solved FFT and finding the amplitude more accurately for the transmitted signal or other frequencies present in the signal.

## 2.4 Speed of sound in air

For a signal with a known frequency and number of cycles, it is of interest in this work to roughly estimate the signal length and the arrival time of the signal at the receiver. This is of interest because it is used to set the correct time window of the oscilloscope for a given distance between the transmitter and receiver, which efficiently discards unnecessary signal information. Since only a rough estimation is needed, the sound speed in the air $c_{air}$ used to calculate the signal length and arrival time does not need to be as accurate. Assuming the fluid is an adiabatic process, and the gas preserves the ideal gas laws, this will give a speed of sound that only depends on temperature as [36]

$$c_{air} = \sqrt{\frac{\gamma R T}{M}}. \tag{2.5}$$

Here the $\gamma$ is the adiabatic constant, $R$ is the gas constant, $T$ is the absolute temperature, and $M$ is the molar mass of air. Calculating the speed of sound for the temperature $T_K$ in kelvin equal to 273.15 K will give a constant equal to approximately 331 meters per second. Then, by measuring the temperature in kelvin, the sound speed becomes

$$c_{air} \approx 331\sqrt{\frac{T}{T_K}}, \tag{2.6}$$

which is the method used to roughly calculate the sound speed in the air in this work.

## 2.5 Coordinate system

Two coordinate systems are used in this work. The coordinate system X, Y, and the Z-axes correspond to the surface of the piezoelectric disk, see Fig. 2.4, and the coordinates of the X, Y, and Z-stages which is defined by the travel direction of the different positioning motor's in the experimental setup. X, Y, and Z-stages coordinates are found with induction sensors where the stages reference points are defined, see Table 4.1. There are two categories of coordinate systems based on the reference points: the machine coordinate system, defined by the reference points and travel limits of the positioning motor's; the user coordinate system, defined by an offset from the reference point. Ideally, the user coordinate system should overlap the coordinate system of the piezoelectric disk, such that when the X, Y, and Z-stages move, they are ideally moved in the coordinate system of the X, Y, and Z axis in Fig. 2.4. A rotating stage, R-stage, is also used in this work, where the axis out of the R-stage should ideally be parallel and overlap the Y-axis. It is studied in Sect. 4.5 what happens if the user coordinates do not overlap the coordinate system of the piezoelectric disk and what uncertainty this entails.

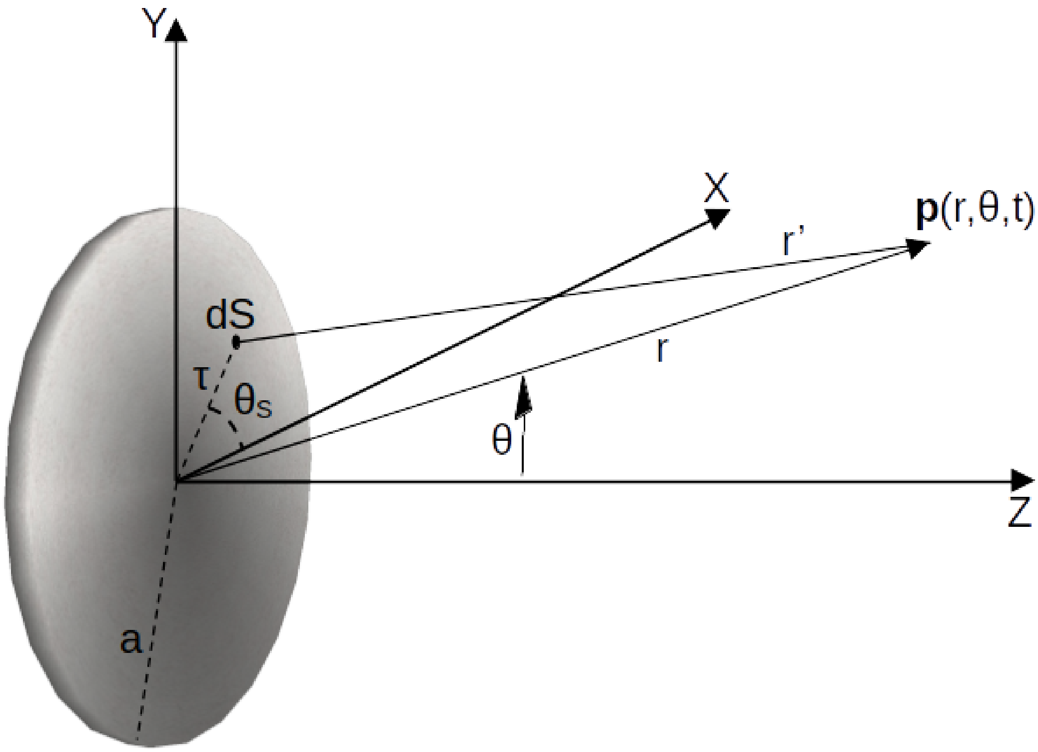



Figure 2.4: Illustration of the front surface of the piezoelectric disk with a radius $a$ and the corresponding X, Y, and Z-axis with the origo in the disk's center, an angle $\theta$ to, and arbitrary pressure point p(r,$\theta$,t) and with a $\theta_S$ representing the symmetrical angle around Z-axis.

## 2.6 Absorption in air

When using FEMP to simulate the pressure, it considers a lossless fluid. But in reality, when measuring sound pressure that has propagated through a fluid, there are different mechanisms in the fluid that absorb the sound pressure depending on the propagation distance $z$. To compare the experimentally measured pressure with FE simulated pressure in this work, then absorption must be known. This leads to the equation for the measured free field pressure [72]

$$p_4 = p_i e^{-\alpha_{Np/m} \cdot z}\,, \tag{2.7}$$

where $p_i$ is the initial free field pressure without losses and $\alpha$ is the absorption coefficient given in Neper per meter. Generally, the absorption coefficient is favorable to describe in dB per meter, which leads to the absorption coefficient [72]

$$\alpha_{Np/m} = \frac{\alpha_{dB/m}}{20log10(e)} \approx 0.1151\alpha_{dB/m}. \tag{2.8}$$

This atmospheric absorption coefficient combines several different attenuations, such as shear viscosity, thermal conductivity, molecular relaxation, and thermal diffusion [5]. This combination of attenuation sums up as [72]

$$\alpha_{Np/m} = \alpha_{cl} + \alpha_{rot} + \alpha_{vib,O} + \alpha_{vib,N}, \tag{2.9}$$

where $\alpha_{cl}$ stands for the classical absorption coefficient, $\alpha_{rot}$ stands for rotational relaxation, and $\alpha_{vib,O}$ and $\alpha_{vib,N}$ are the molecular vibrational relaxation absorption coefficient of oxygen and nitrogen in the air. The cause of $\alpha_{cl}$ is shear viscosity, heat conduction, and thermal diffusion [5]. In terms of dB per meter, the combined absorption coefficient becomes [5]

$$\begin{aligned}\alpha_{dB/m} =& 8.686 f^2 \left( \left[ 1.84 \cdot 10^{-11} \left( \frac{p_r}{p_a} \right) \left( \frac{T}{T_r} \right)^{1/2} \right] + \left( \frac{T_r}{T} \right)^{5/2} \right. \\ & \left. \left[ 0.01275 e^{\frac{-2239.1}{T}} \frac{f_{rO}}{f_{rO}^2 + f^2} + 0.1068 e^{\frac{-3352.0}{T}} \frac{f_{rN}}{f_{rN}^2 + f^2} \right] \right),\end{aligned} \tag{2.10}$$

where the relaxation frequency of oxygen is [5]

$$f_{rO} = \frac{p_a}{p_r} \left( 24 + \frac{(4.04 \cdot 10^4 h)(0.02 + h)}{0.391 + h} \right), \tag{2.11}$$

and the relaxation frequency of nitrogen is [5]

$$f_{rN} = \frac{p_a}{p_r} \left( \frac{T_r}{T} \right)^{1/2} \left( 9 + 280 h \cdot e^{-4.170 \left( \left( \frac{T_r}{T} \right)^{1/3} - 1 \right)} \right). \tag{2.12}$$

Here is $p_a$, the measured pressure in kPa, and $p_r$ the reference pressure, 101.325 kPa—further, $T$ is the measured air temperature, and $T_r$ the reference temperature, 293.15 K. Furthermore, $h$ is the percent molar concentration of water vapor, and $f$ is the frequency used [5]. In the given molar concentration of water vapor [5]

$$h = h_{rel} \left( \frac{p_{sat}}{p_r} \right) \left( \frac{p_r}{p_a} \right), \tag{2.13}$$

there is a need to know the relative humidity $h_{rel}$ and saturated pressure $p_{sat}$. Measurements could provide the relative humidity and give the necessary temperature $T$ value to calculate

[5]

$$
\begin{aligned}
V =&10.79586\left(1-\frac{T_{01}}{T}\right)-5.02808 log_{10}\left(\frac{T}{T_{01}}\right)+1.50474\cdot 10^{-4}\left(1-10^{-8.29692\left(\frac{T}{T_{01}}-1\right)}\right) \\
&+0.42873\cdot 10^{-3}\left(10^{4.76955\left(1-\frac{T_{01}}{T}\right)}-1\right)-2.2195983,
\end{aligned}
\tag{2.14}
$$

used to calculate the saturated pressure [5]

$$p_{sat} = p_r 10^V. \tag{2.15}$$

$T_{01}$ is the isothermal triple point temperature for water, 273.16 K.

The calculated absorption coefficient $\alpha$ in dB per meter and the measured pressure open up to calculating the initial pressure using Eq. 2.7 and 2.8, giving

$$p_i = \frac{p_4}{e^{-0.1151\alpha_{dB/m}\cdot z}}. \tag{2.16}$$

Using this Eq. 2.16, one can compare $p_i$ calculated from the measured pressure $p_4$ with the simulated pressure after the simulated pressure has been adjusted for the differences in the voltage across the piezoelectric disk and simulated voltage, see Sect. 5.4. It is worth noting that this absorption coefficient used in this work does not consider factors such as refraction, scattering by turbulence, and non-linear propagation effects [5], which are assumed to be negligible.

## 2.7 Electronics

Cables connect instruments to the piezoelectric disk and the microphone together within the transmitting or receiving electronics. In these systems, the ideal voltages across the piezoelectric disk $V_1$ and the microphone output $V_5$ can not be measured directly. This is due to the finite impedance of the measurement system as a whole, which affects the voltage signal flowing through the system. Therefore it is deduced transfer functions that transfer the input voltages of the transmitted voltage $V_{0m}$ and received voltage $V_{5m}$ to ideal voltage $V_1$ and $V_5$.

### 2.7.1 Cables

A well-known phenomenon is that signals flowing through cables are affected by the characteristics of the cable. This phenomenon is experimentally proven and, therefore, essential to consider when calculating the voltage $V_1$ and $V_5$, which lead to seeing the cable as an ideal uniform transmission line using distributed constants [61]. The coaxial cables terminated in a load impedance $Z_L$ can then be described as an equivalent circuit [73], as shown in Fig. 2.5.

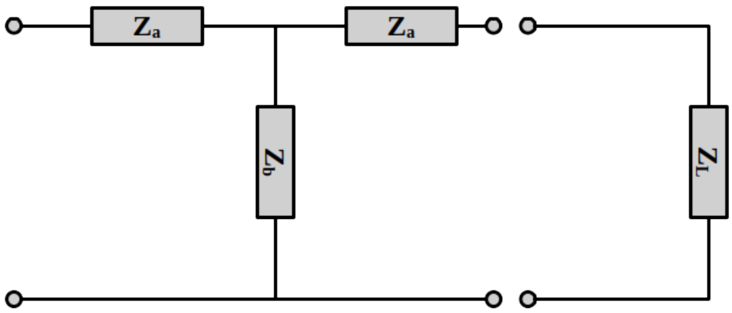



Figure 2.5: Equivalent circuit of a ideal lossless transmission line with coaxial cable terminated in a load impedance $Z_L$, where distributed constants are $Z_a$ and $Z_b$.

The distributed constants used in describing the coaxial cable could be defined as two different impedances $Z_a$ and $Z_b$, as seen in Fig. 2.5. These impedances are determined by the characteristic impedance of the cable $Z_0$, the electromagnetic wave number $k_{em}$, and the cable length $x$ in meters and given as [73]

$$Z_a = iZ_0 tan\left(\frac{k_{em}x}{2}\right) \tag{2.17}$$

and

$$Z_b = \frac{Z_0}{isin(k_{em}x)}. \tag{2.18}$$

The characteristic impedance and the electromagnetic wave number is [73]

$$Z_0 = \sqrt{\frac{L_x}{C_x}} \tag{2.19}$$

and

$$k_{em} = \omega\sqrt{L_x C_x}, \tag{2.20}$$

respectively, where $L_x$ and $C_x$ are the inductance and capacitance per meter, respectively, and $\omega$ is the angular frequency.

### 2.7.2 Transmitting electronics

In this work, transmitting electronics links the signal generator to the oscilloscope and the piezoelectric disk with coaxial cables. This part of the system model, seen in Fig. 2.3, can be described with a circuit diagram, as shown in Fig. 2.6. In this diagram, the voltage over the piezoelectric disk is $V_1$, and the voltage measured with the oscilloscope is $V_{0m}$. A transfer function can describe this relationship between these two voltages as

$$H_{0m1}^{VV} = \frac{V_1}{V_{0m}}. \tag{2.21}$$

From the circuit diagram, the voltage $V$ describes the Thevenin equivalent voltage of the signal generator, and $Z_{GEN}$ is the Thevenin equivalent impedance. The $Z_{OSC}$ is the oscilloscope's termination impedance, $Z_T$ is the piezoelectric disk's impedance, and $Z_a$ and $Z_b$ are the coaxial cable impedances. The impedance of the piezoelectric disk is measured with an impedance analyzer.

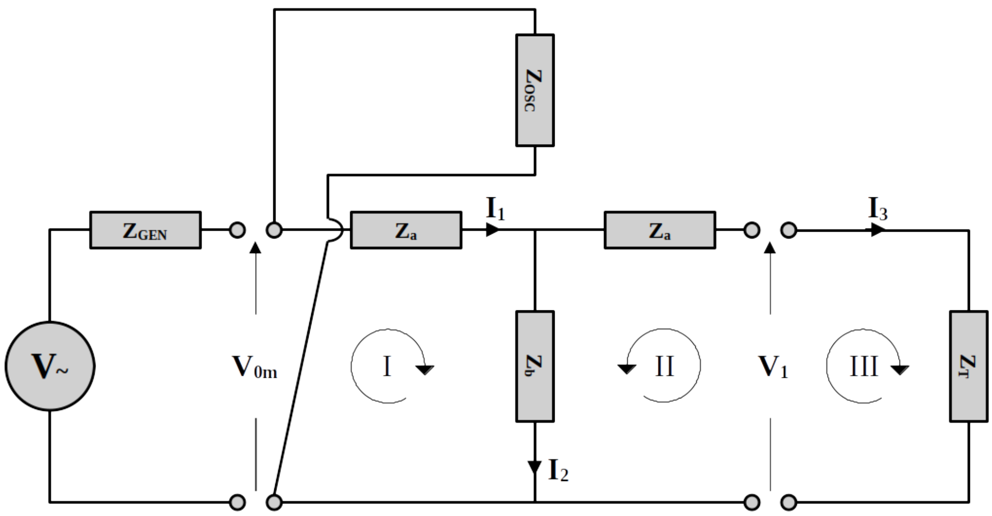



Figure 2.6: Circuit diagram of the transmitting electronics, connecting to a signal generator as a Thevenin equivalent circuit, the piezoelectric disk, and the oscilloscope.

Applying Kirchhoff's voltage law in the three directions indicated by arrows in Fig. 2.6, three voltage equations can be deduced,

$$V_{0m} = Z_a I_1 + Z_b I_2 \tag{2.22}$$

$$V_1 = Z_b I_2 - Z_a I_3 \tag{2.23}$$

$$V_1 = Z_T I_3 \tag{2.24}$$

and with Kirchhoff's current law, the relation between the three currents $I_1$, $I_2$, and $I_3$ shown in Fig. 2.6 is

$$I_1 = I_2 + I_3. \tag{2.25}$$

By setting Eqs. 2.23 and 2.24 equal to another and solving for the current $I_3$,

$$I_3 = \frac{Z_b I_2}{Z_T + Z_a}, \tag{2.26}$$

and using substitution and algebraic manipulation of Eqs. (2.22,2.24-2.26), then the transfer function in Eq. 2.21 is easily derived and gives

$$H_{0m1}^{VV} = \frac{Z_T Z_b}{Z_T(Z_a + Z_b) + (Z_a + Z_b)^2 - Z_b^2}\ , \tag{2.27}$$

which is a transfer function given in only known impedances and, therefore, used to solve the voltage $V_5$ with measured voltage $V_{0m}$.

### 2.7.3 Receiving electronics

In this work, receiver electronics link the amplifier and filter to the oscilloscope and the microphone with coaxial and microphone cables, respectively, as seen in Fig. 2.3. Describing the voltage signal that flows in via input on the filter or amplifier and out via output can quickly become unnecessarily complicated. This becomes complicated due to the electronics inside the filter or amplifier, which are unknown. Therefore, the electronics are described as a complex frequency-dependent factor $F(f)$ to simplify this part, which is given as the black dotted lines, illustrating unknown electronics in Fig. 2.7, separating the input and the output. In this circuit description of the filter (or amplifier), the input is seen as an open input with a voltage $V_{5'open}$ and an impedance $Z_{filt\_open}$. The output is seen as a Thevenin equivalent voltage generator $V_{5'}$ with a Thevenin equivalent impedance $Z_{filt\_out}$ which is the output impedance given by the manufacturer. The relation between the input and output of the filter becomes

$$V_{5'} = V_{5'open} F(f) \tag{2.28}$$

and applies to the amplifier and all filters in this work.

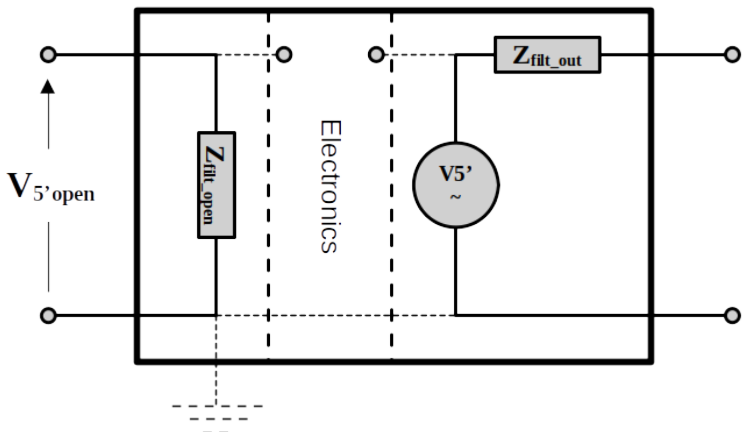



Figure 2.7: Illustration of input and output of the filter separated by unknown electronics. This illustration applies to the amplifier and all filters in this work.

With the theory of voltage flow from input to the output of the amplifier and the filters covered, it is necessary to describe what happens between the instruments. In this work, the microphone cable is neglected, and only the coaxial cables are considered. This leads to three circuit diagrams, shown in Fig. 2.8. These three circuit diagram transfer functions are almost identical to derive as for the transmitter electronics. The only difference between the transmitter and receiver electronics is that with the receiver electronics, one of Kirchoff's voltage law directions indicated changes, and the values of the receiver electronics are changed.

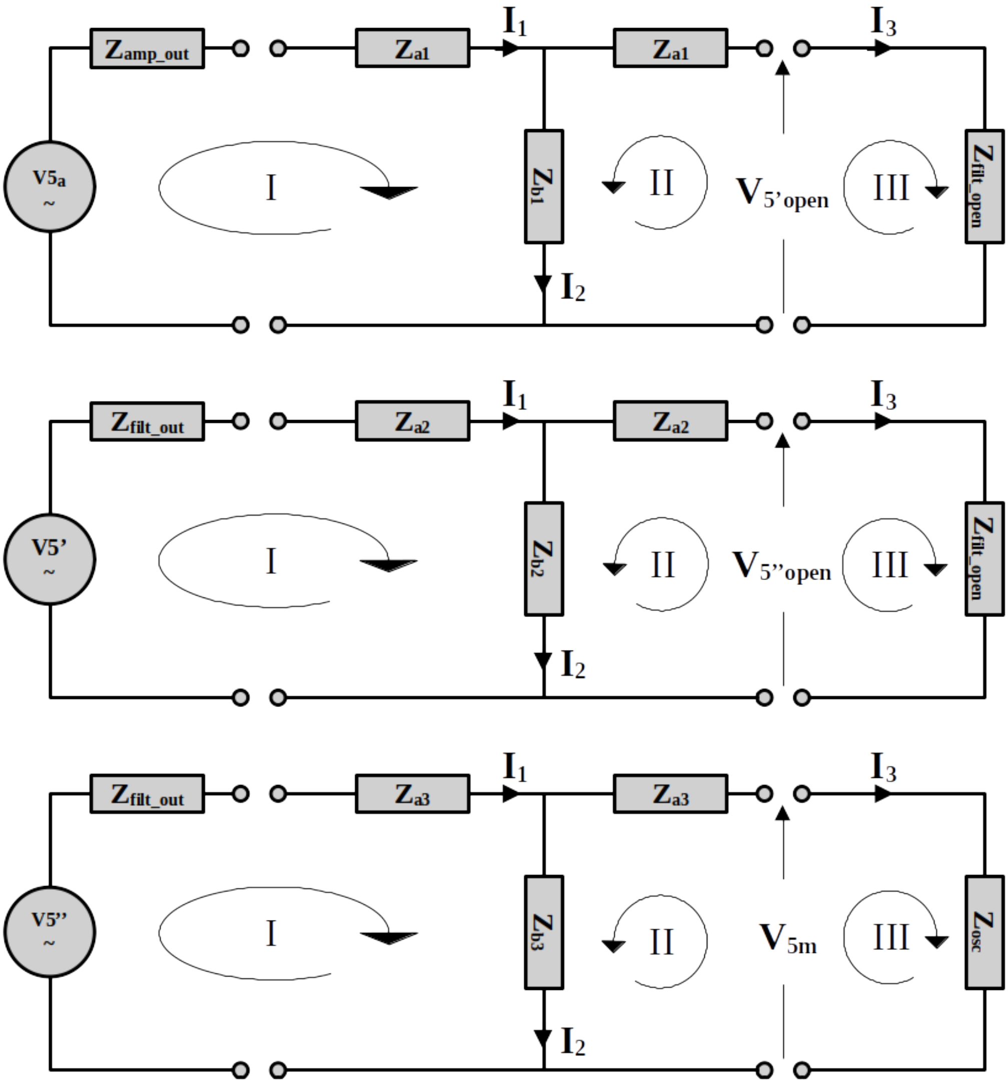



Figure 2.8: Three circuit diagrams: Top circuit diagram is the Thevenin equivalent generated output voltage from the amplifier connected to the filter's open input channel one via coaxial cable; the middle circuit diagram is the Thevenin equivalent generated output voltage from the filter's output channel one connected to the filter's open input channel two via coaxial cable; the bottom circuit diagram is the Thevenin equivalent generated output voltage from the filter's output channel two connected to the oscilloscope's channel two via coaxial cable.

The transfer function describing the relation of the output voltage of the microphone $V_5$ and the input voltage in channel two of the oscilloscope $V_{5m}$ is given as

$$H_{55m}^{VV} = H_{55_a}^{VV} \cdot H_{5_a5'_{open}}^{VV} \cdot H_{5'_{open}5'}^{VV} \cdot H_{5'5''_{open}}^{VV} \cdot H_{5''_{open}5''}^{VV} \cdot H_{5''5m}^{VV} = \frac{V_{5m}}{V_5}. \tag{2.29}$$

This transfer function is a combination of multiple transfer functions: the relation of the voltage $V_5$ and the output voltage of the amplifier $V_{5_a}$ as $H_{55_a}^{VV}$; the relation of the voltage $V_{5_a}$ and the input voltage of the filters channel one $V_{5'_{open}}$ as $H_{5_a5'_{open}}^{VV}$; the relation of the voltage $V_{5'_{open}}$ and the output voltage of the filters channel one $V_{5'}$ as $H_{5'_{open}5'}^{VV}$; the relation of the voltage $V_{5'}$ and the input voltage of the filters channel two $V_{5''_{open}}$ as $H_{5'5''_{open}}^{VV}$; the relation of the $V_{5''_{open}}$ and the output voltage of the filters channel two $V_{5''}$ as $H_{5''_{open}5''}^{VV}$; the relation of the voltage $V_{5''}$ and the $V_{5m}$ as $H_{5''5m}^{VV}$.

The complex frequency-dependent factor given in Eq. 2.28 equals the transfer functions over the amplifier and the filters. The factor is given as a dB gain over the amplifier. The factor is assumed lossless and equal to one over the filters. These three transfer functions become

$$H_{55_a}^{VV} = 10^{\frac{Gain}{20}}, \tag{2.30}$$

$$H_{5'_{open}5'}^{VV} = 1, \tag{2.31}$$

and

$$H_{5''_{open}5''}^{VV} = 1. \tag{2.32}$$

Using substitution and algebraic manipulation of the equation that is possible to derive from each circuit in Fig. 2.8, the transfer function from the output voltage of the amplifier to the input voltage of the filter's channel one becomes

$$H_{5_a5'_{open}}^{VV} = \frac{Z_{filt,open}}{\left[(Z_{amp,out+Z_{a_1}})\left(1+\frac{Z_{filt,open+Z_{a_1}}}{Z_{b_1}}\right)\right] + Z_{filt,open+Za_1}}, \tag{2.33}$$

the transfer function from the output voltage of the filter's channel one to the input voltage of the filter's channel two becomes

$$H_{5'5''_{open}}^{VV} = \frac{Z_{filt,open}}{\left[(Z_{filt,out+Z_{a_2}})\left(1+\frac{Z_{filt,open+Z_{a_2}}}{Z_{b_2}}\right)\right] + Z_{filt,open+Za_2}}, \tag{2.34}$$

and the transfer function from the output voltage of the filter's channel two to the input voltage of the oscilloscope channel two becomes

$$H_{5''5m}^{VV} = \frac{Z_{osc}}{\left[(Z_{filt,out+Z_{a_3}})\left(1+\frac{Z_{osc+Z_{a_3}}}{Z_{b_3}}\right)\right] + Z_{osc+Za_3}}. \tag{2.35}$$

## 2.8 Microphone sensitivity

Calibration of the Brüel & Kjær 4138 microphone [17] is performed with a Brüel & Kjær 4228 pistonphone [16], which operates at a known frequency of 250 Hz and produces a known sound pressure level, SPL [20]. The frequency response of the Brüel & Kjær 4138 microphones is given relative to the microphone reference sensitivity $|M_{ref}(250Hz)|$ when it is electrically unloaded [19]. It is possible to use the specified microphone sensitivity shown in the calibration papers, but it is recommended to calibrate the microphone by measuring and calculating microphone sensitivity using the pistonphone. This is due to changes in conditions such as air pressure, temperature, and any other conditions that may affect the sensitivity. Other conditions are not known, but aging effects or the receiver electronics may have additional effects that give a reason to calibrate. To calculate, in general, the free-field open circuit microphone sensitivity it is used the definition [73]

$$M_V(f) = \frac{V_5(f)}{p_4(f)} = |M_V(f)|e^{i\phi_{M_V}}, \tag{2.36}$$

where $V_5$ is the measured voltage at the electrically unloaded receiver, $p_4$ is the free field pressure at the microphone position when the microphone is not present, $|M_V(f)|$ is the magnitude of the free-field open circuit microphone sensitivity, and $\phi_{M_V}$ is the phase difference [73]. Since the open circuit pressure response relative to $|M_{ref}(250Hz)|$ for Brüel & Kjær 4138 microphone [19] does not specify the phase difference, it is disregarded, and only the microphone sensitivity is taken into account. The microphone sensitivity is then defined as [73]

$$|M_V(f)| = \frac{V_{eff}(f)}{p_{eff}(f)} , \tag{2.37}$$

where $V_{eff}$ is the effective voltage amplitude calculated from the measured voltage $V_{5m}$ as

$$V_{eff}(f) = \frac{V_{5_{pp}}(f)}{2\sqrt{2}} = \frac{1}{2\sqrt{2}} \frac{V_{5m_{pp}}(f)}{H^{VV}_{55m}(f)} , \tag{2.38}$$

and the pressure $p_{eff}$ is the effective pressure amplitude and can be calculated from the sound pressure level defined as

$$SPL = 20log_{10}\left(\frac{p_{eff}(f)}{p_{ref}}\right) . \tag{2.39}$$

where $p_{ref}$ is the reference pressure for air which is 20 $\mu$Pa, and SPL in this work is the known sound pressure level produced by the pistonphone at 250 Hz .

## 2.9 Finite element modeling

This work uses Finite Element Modeling of Piezoelectric structures (FEMP) software to simulate the electrical characteristic of $Y_T(f)$ (admittance in a vacuum and air) and acoustic characteristics of, $D(r, \theta, f)$ (directivity), $p_{ax}(z,f)$ (on-axis pressure), and $p(x,z,f)$ (2-D sound pressure field) of a piezoelectric disk. This section presents only a brief overview of the theory behind FEMP, where [38] provides a full description. In FEMP, the FE simulation is simplified by reducing the structure from being a 3-D structure to 2-D by assuming symmetry in the axisymmetric disk [38]. This assumption implies [38]

$$\frac{\partial}{\partial \theta_S} = 0 \, ,$$

where $\theta_S$ is the angle around the Z-axis in Fig. 2.4, and it further assumes no torsion modes [38]

$$u_{\theta_S} = 0 \, ,$$

where $u_{\theta_S}$ is displacement in the angle $\theta_S$ direction. Other benefits these assumptions provide are the reduction of the number of piezoelectric constants for the dielectric stiffness $[\epsilon^S]$, piezoelectric stiffness $[e]$, and elastic stiffness $[c^E]$ matrices used in FE calculations, which decreases computation time [38].

In the FE simulations, a structure is approximated using a finite number of volume elements [38]. Within each volume element, it is defined a number of nodes, see Fig 5.1. With an increasing number of volume elements, the accuracy of the simulation increases. Other factors that define the precision of the simulation are the number of nodes, and the interpolations function used to solve for quantitative values in an arbitrary position inside the structure's elements [38]. The starting equations to find the FE formulation for an unloaded piezoelectric disc with no surrounding fluids are as follows [32]

$$-\omega^2 \rho_p u_i = T_{ij,i} \, , \tag{2.40}$$

$$D_{i,i} = 0 \, , \tag{2.41}$$

$$T_{ij} = c^E_{ijkl} S_{kl} - e_{kij} E_k \, , \tag{2.42}$$

and

$$D_i = e_{ikl} S_{kl} + \epsilon^S_{ik} E_k \, , \tag{2.43}$$

where $T_{ij,i}$ is the motion for a piezoelectric medium, $D_{i,i}$ is the Maxwell equation, $T_{ij}$ and $D_i$ is the constitutive relations for the piezoelectric medium [38], and the rest of the variable's descriptions are given in Table 2.2.

Table 2.2: Description of variables used in Eqs.(2.42) and (2.43) [38].

| Variable | Description [38] | Unit |
|---|---|---|
| $T_{ij}$ | Mechanical stress tensor | $[N/m^2]$ |
| $S_{kl}$ | Mechanical strain tensor | $[-]$ |
| $D_i$ | Electric flux density | $[C/m^2]$ |
| $E_k$ | Electric field vector | $[V/m]$ |
| $c^E_{ijkl}$ | Elastic stiffness constant tensor with constant electric field | $[N/m^2]$ |
| $e_{ikl}$ | Piezoelectric constant tensor | $[C/m^2]$ |
| $\epsilon^S_{ik}$ | Dielectric constant tensor with constant strain | $[F/m]$ |
| $\omega$ | Angular frequency | $[rad/s]$ |
| $\rho_p$ | Piezoelectric medium density | $[kg/m^3]$ |
| $u$ | Displacement | $[m]$ |

The piezoelectric structure's surface, facing a vacuum, is imposed with boundary conditions [38]. With boundary conditions introduced, the Eqs. 2.40-2.43 is used when setting up weak formulations using test functions and solving each part of the weak formulation equations using Gauss Legendre quadrature [38]. With each part solved using Gauss, the final FE formulation for the unloaded case is given as [38]

$$-\omega^2 \begin{bmatrix} M_{uu} & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \hat{u} \\ \hat{\phi} \end{Bmatrix} + \begin{bmatrix} K_{uu} & K_{u\phi} \\ K_{\phi u} & K_{\phi\phi} \end{bmatrix} \begin{Bmatrix} \hat{u} \\ \hat{\phi} \end{Bmatrix} = \begin{Bmatrix} F \\ -Q \end{Bmatrix} , \tag{2.44}$$

and the variables of this Eq. 2.44 are given in Table 2.3. The FE formulation, Eq. 2.44 are transformed to H-form to simplify the calculation of the piezoelectric structure's resonance frequencies and response function [38]. With the potentials in the nodes of the elements condensed out from FE equations and $V$ and $I$ are introduced as voltage and current [39], Eq. 2.44 becomes [38]

$$-\omega^2 \begin{bmatrix} M_{uu} & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \hat{u} \\ V \end{Bmatrix} + \begin{bmatrix} H_{uu} & H_{u\phi} \\ H_{\phi u} & H_{\phi\phi} \end{bmatrix} \begin{Bmatrix} \hat{u} \\ V \end{Bmatrix} = \begin{Bmatrix} F \\ -I/i\omega \end{Bmatrix} . \tag{2.45}$$

Solving the piezoelectric disk with the direct harmonic analysis method in a vacuum where there is no outer traction (F = 0) [38], the first expression in Eq. (2.45), is used to calculate the particle displacement of a piezoelectric structure as [38]

$$\{\hat{u}\} = -[D]^{-1} \{H_{u\phi}\} V , \tag{2.46}$$

where Figs. 2.1a and 2.1b are examples of solved piezoelectric structure displacement $\{\hat{u}\}$, and the matrix $[D]$ is just a simplified notation and is given as [38]

$$[D] = [H_{uu}] - \omega^2 [M_{uu}] \ . \tag{2.47}$$

By inserting the global displacement vector into the second expression in Eq. 2.45, and if the admittance $Y$ is defined by the current $I$ divided by the voltage $V$, the admittance of the piezoelectric structure is given as [38]

$$Y(\omega) = \frac{I}{V} = i\omega \left( \{H_{u\phi}\}^T [D]^{-1} \{H_{u\phi}\} - H_{\phi\phi} \right) \ . \tag{2.48}$$

In the case of the piezoelectric structure is in a vacuum, the only volume studied is $\Omega_p$, which is the volume of the piezoelectric structure, see Figs. 5.2/5.3. Reviewing the disc submerged into an inviscid and irrotational fluid, a new region, $\Omega_f$, occurs which is the fluid volume [38], see Figs. 5.2/5.4. In the time-harmonic case, the relationship between the acoustic pressure $p$ and the velocity potential $\psi$ in the fluid is [38]

$$p = i\omega\rho_f\psi. \tag{2.49}$$

where $\rho_f$ is the density of the fluid. The Helmholtz equation governing the fluid velocity potential $\psi$ [36] is given as

$$\psi_{,ii} = -k^2\psi \ , \tag{2.50}$$

where $k$ is the wave number, which is $k = \omega / c_f$, where $c_f$ is the sound speed in the fluid. The piezoelectric structure's surface, facing a fluid, is imposed with new boundary conditions [38]. With boundary conditions introduced, the FE-formulations Eqs 2.40-2.43 and 2.50 is used when setting up weak formulations using test functions and solving each part of the weak formulation equations using Gauss Legendre quadrature [38]. With each part solved using Gauss, the final FE formulation for the fluid loading is written as [38]

$$-\omega^2 \begin{bmatrix} M_{uu} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -M_{\psi\psi} \end{bmatrix} \begin{Bmatrix} \hat{u} \\ \hat{\phi} \\ \hat{\psi} \end{Bmatrix} + i\omega \begin{bmatrix} 0 & 0 & C_{u\phi} \\ 0 & 0 & 0 \\ C_{\psi u} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \hat{u} \\ \hat{\phi} \\ \hat{\psi} \end{Bmatrix} + \begin{bmatrix} K_{uu} & K_{u\phi} & 0 \\ K_{\phi u} & K_{\phi\phi} & 0 \\ 0 & 0 & -K_{\psi\psi} \end{bmatrix} \begin{Bmatrix} \hat{u} \\ \hat{\phi} \\ \hat{\psi} \end{Bmatrix} = \begin{Bmatrix} 0 \\ -Q \\ 0 \end{Bmatrix} , \tag{2.51}$$

and the variables of this Eq. 2.51 are given in Table 2.3.

Table 2.3: Description of variables and matrices used in Eq.2.44 and 2.51 [38].

| Variable | Description [38] |
|---|---|
| $[M_{uu}]$ | Global mass matrix |
| $[M_{\psi\psi}]$ | Global fluid mass matrix |
| $[C_{u\psi}]$ | Global fluid/structure coupling matrix |
| $[C_{\psi u}]$ | Global fluid/structure coupling matrix |
| $[K_{uu}]$ | Global stiffness matrix |
| $[K_{u\phi}]$ | Global piezoelectric stiffness matrix |
| $[K_{\phi u}]$ | Global piezoelectric stiffness matrix |
| $[K_{\phi\phi}]$ | Global dielectric stiffness matrix |
| $[K_{\psi\psi}]$ | Global fluid stiffness matrix |
| $\{F\}$ | Global force vector |
| $\{Q\}$ | Global charge vector |
| $\{\hat{u}\}$ | Global displacement vector |
| $\hat{\phi}$ | Global electric potential |
| $\hat{\psi}$ | Global fluid velocity potential |
| $V$ | Voltage |
| $I$ | Current |

The FE formulation, Eq. 2.51 are transformed to H-form and given as [38]

$$-\omega^2 \begin{bmatrix} M_{uu} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -M_{\psi\psi} \end{bmatrix} \begin{Bmatrix} \hat{u} \\ V \\ \hat{\psi} \end{Bmatrix} + i\omega \begin{bmatrix} 0 & 0 & C_{u\phi} \\ 0 & 0 & 0 \\ C_{\psi u} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \hat{u} \\ V \\ \hat{\psi} \end{Bmatrix} + \begin{bmatrix} H_{uu} & H_{u\phi} & 0 \\ H_{\phi u} & H_{\phi\phi} & 0 \\ 0 & 0 & -K_{\psi\psi} \end{bmatrix} \begin{Bmatrix} \hat{u} \\ V \\ \hat{\psi} \end{Bmatrix} = \begin{Bmatrix} 0 \\ -I/i\omega \\ 0 \end{Bmatrix} . \tag{2.52}$$

From Equation (2.52), the global fluid velocity potential and global displacement vector can be written as [38]

$$\{\hat{\psi}\} = -i\omega \left(\omega^2 [M_{\psi\psi}] - [K_{\psi\psi}]\right)^{-1} [C_{\psi u}] \{\hat{u}\} \ , \tag{2.53}$$

and

$$\{\hat{u}\} = -[E]^{-1} \left\{H_{u\phi}\right\} V \ , \tag{2.54}$$

respectively, where the matrix $[E]$ is just a simplified notation and is given as [38]

$$[E] = \left\{[H_{uu}] - \omega^2 [M_{uu}] + \omega^2 [C_{u\psi}] \left(-[K_{\psi\psi}] + \omega^2 [H_{\psi\psi}]\right)^{-1} [C_{\psi u}]\right\}, \tag{2.55}$$

The admittance is obtained by inserting the global displacement into the second expression in Eq. 2.52 and written as [38]

$$Y(\omega) = \frac{I}{V} = i\omega \left\{ [H_{u\phi}]^T [G]^{-1} [H_{u\phi}] - [H_{\phi\phi}] \right\}. \tag{2.56}$$

The global fluid velocity potential is inserted into Eq. (2.49) for the acoustic pressure and then expressed as [38]

$$\{\hat{p}\} = -i\omega\rho_f \{\hat{\psi}\}. \tag{2.57}$$

# Chapter 3

# Experimental setup and measurement method

This chapter presents the instruments used, the experimental setup of electrical and acoustic measurements, and the method for analyzing measurement results. Sect. 3.1 shows a table of all the equipment that has been used. Sect. 3.2 goes through the method for electrical measurements. Sect. 3.3 goes through the method for acoustic measurements. Sect. 3.4 describes how the microphone sensitivity is calibrated. Sect. 3.5, motor setup and adjustment of the Y-stage travel length is described. Sect. 3.6 describes reflections that can affect the measured signal. Finally Sect. 3.7 describes the signal setup and processing of the signals.

## 3.1 Equipment list

Table 3.1 is the overview of the equipment connected to the movements of the experimental setup, and Table 3.2 shows the remaining equipment used in the experimental setup

Table 3.1: Equipment used in present work.

| Brand/Model | Equipment | Serial number | Documentation/Manual |
|---|---|---|---|
| SMC Hydra TT | Motion Controller | 1404-0153 | [56] |
| PI C-843.41 | Motion Controller | 0095103296 | [55] |
| PI C-852.12 | Signal processor/Encoder | 1460497 | [51] |
| PI M-531.DG | Linear stage (X-stage) | - | [53] |
| PI M-535.22 | Linear stage (Y-stage) | 1460497 | [52] |
| PI LS270 | Linear stage (Z-stage) | 414000926 | [57] |
| PI M-037.PD | Rotation stage (R-stage) | 109040312 | [54] |

Table 3.2: Equipment used in present work.

| Brand/Model | Equipment | Serial number | Documentation/Manual |
|---|---|---|---|
| Mitutoyo M310-25 | Micrometer | 102-301 | [47] |
| Cocraft HL10-S | Cross-line laser level | 18081157898 | [23] |
| HP 4192A | Impedance analyzer | 2150J01344 | [31] |
| Vaisala HMT313 | Humidity and temperature | F4850018 | [70] |
| ASL F250 MkII | Thermometer | 1365026993 | [8] |
| Paroscientific 740 | Barometer | 67325 | [50] |
| Agilent 33220A | Signal generator | MY44023589 | [4] |
| Tektronix DPO3012 | Oscilloscope | C010246 | [66] |
| Krohn-Hite 3940 | Filter | AM2626 | [40] |
| Brüel & Kjær 2636 | Measurement amplifier | 1815638 | [18] |
| Brüel & Kjær 4138 | 1/8-inch pressure-field microphone | 1832479 | [17] |
| Brüel & Kjær UA-160 | Adaptor - microphone to preamplifier | - | [17] |
| Brüel & Kjær 2633 | Preamplifier | - | [17] |
| Brüel & Kjær 4228 | Pistonphone | 1918465 | [16] |
| KEYENCE LK-G3001PV | Controller with display | 1741187 | [35] |
| KEYENCE LK-G32 | Laser sensor | 2041141/2041143 | [35] |
| Meggitt A/S Pz27 Ceramic disc | Piezoelectric element | - | [46] |

## 3.2 Electrical measurement setup

The impedance analyzer HP4192A [31] in Fig. 3.1 measures the electrical conductance and susceptance of the piezoelectric disk used in this work. The conductance, $G_T$, and susceptance, $B_T$, determine the piezoelectric element's electrical characteristics and are used to calculate $Y_T$. The admittance is given as

$$Y_T(f) = G_T(f) + iB_T(f) \ . \tag{3.1}$$

Before measurements are conducted of the piezoelectric disk, the impedance analyzer is turned on for at least 30 minutes to warm up the electronics and stabilize the instrument [31]. After the warm-up, it is necessary to do a zero calibration of the impedance analyzer to account for the electrical properties of external wires connected to the instrument. The calibration is performed in two steps: The first calibration is when the external wires connected to the analyzer are separated from each other, leading to an open circuit where the impedance is $Z = \infty\ \Omega$; The second calibration is when the external wires are kept together, making a shorted circuit where the impedance $Z \approx 0\ \Omega$. These calibrations are generally performed with the highest measuring frequency of the measurement series, which is 300 kHz in this work. The complete calibration description is in section 3-50 in the manual [31]. Then by connecting the wires to each of the electrodes or wires soldered to the electrodes, see Fig. 3.2, the element of interest is ready for measurement. The MatLab script **impanal.m** in Appendix

[A.1] sets all measurement parameters before performing measurements and connects to the analyzer through GPIB. The adjustable parameters are the frequency range, frequency resolution, time delay before measurement after a frequency change, and the root mean square voltage $V_{rms}$. The frequency range of interest in the present work is 1-300 kHz. This range covers the piezoelectric element's first and second radial modes relevant for air/gas measurements. After performing a rough measurement of the $G_T$ and $B_T$, the radial modes are identified. A new measurement is conducted from 1-300 kHz with a higher frequency resolution around the radial modes. The voltage range of the impedance analyzer is from 0.1-1.1 $V_{rms}$, where higher voltage can trigger non-linear effects but reduce the signal-to-noise ratio [13] or lower voltage increase uncertainties [31]. Previous work done by [48][29] has used 0.3 $V_{rms}$ to maximize the accuracy of the measurement of the $G_T$ and $B_T$, and therefore is this voltage used in the present work.

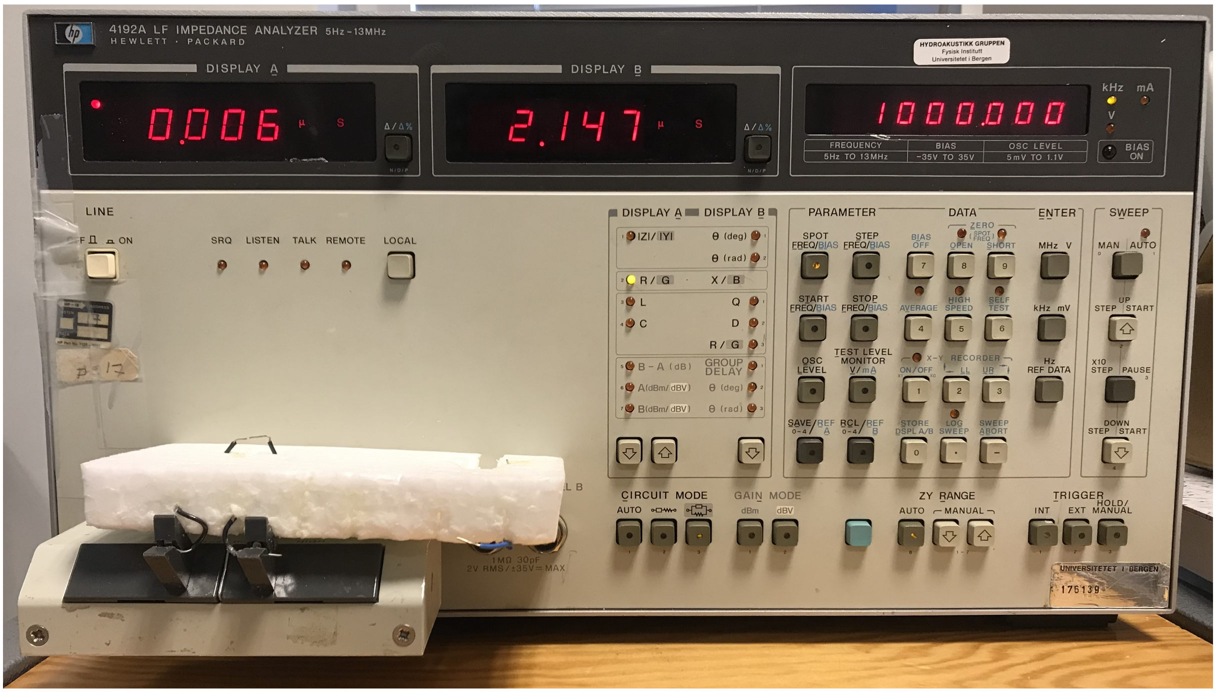
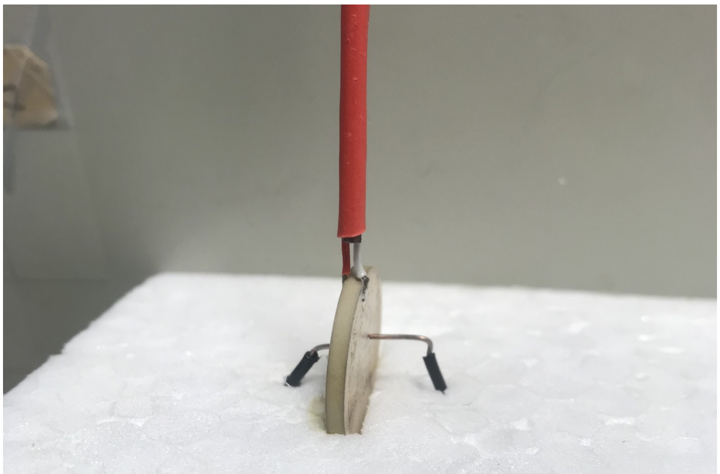
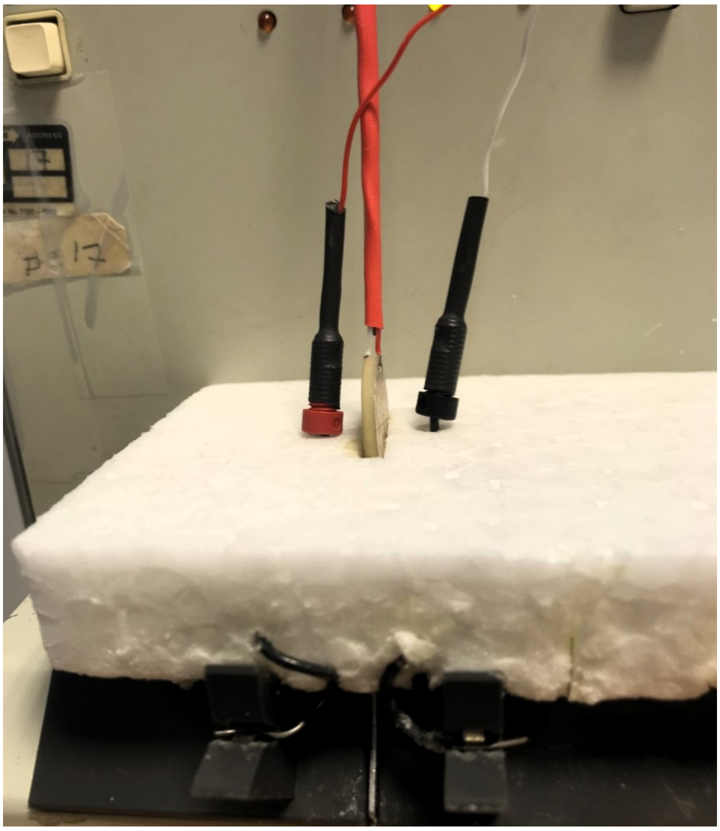



Figure 3.1: Impedance Analyzer HP 4192A [31], with a styrofoam block on the left side and the external wires used to connect the analyzer to the electrodes on the piezoelectric disk, pass through the styrofoam block.

When conducting measurements of the piezoelectric element, it is desirable to minimize the mechanical load on the disk. With a styrofoam block, the disk placed in a groove in the styrofoam, see Fig. 3.2, and wires connected to the electrodes with a slight inward force, this setup intends to minimalize this load. This minimization helps the element to vibrate more freely and is essential when compared to FE simulations of the admittance of the piezoelectric disk in vacuum and fluid. It also helps to give more accurate measurements. However, in previous work, repeatability problems have been observed around the resonances of measurements performed by [48] [29]. It has been suggested that repeatability problems arise due to variations in the disk location in the styrofoam groove, the location of the wires on the electrodes, or different spring-like forces exerted by the wires placed on the electrodes [48] [29]. It is also performed measurements directly on the wires plugs, where the wires are soldered to the electrodes, see Fig. 3.2b. These measurements are used to compare to the measurements conducted directly on the electrodes.



(a)



(b)

Figure 3.2: A styrofoam block with a groove holds the piezoelectric element when measuring the admittance. (a) Measurement is performed with wires from the analyzer placed directly on the electrodes and (b) to the wires soldered to the electrodes.

## 3.3 Acoustical measurement setup

Previous works [65][48][29] started developing the current acoustical measurement setup and have been continued to be developed by [6][7][64][26][24]. The block diagram in Fig. 3.3 is a detailed description of the current acoustical measurement setup. Figs. 3.4 and 3.5 gives the setup's visual overview, and the remaining instruments are shown in Fig. 3.6. There are two ways to go forward with acoustical measurements, set the settings directly in MatLab app for single measurements or predefining settings in the **MeasurementParameters.m** in Appendix A.3 script for a measurement series, see Sect. 4.4 for changeable settings.

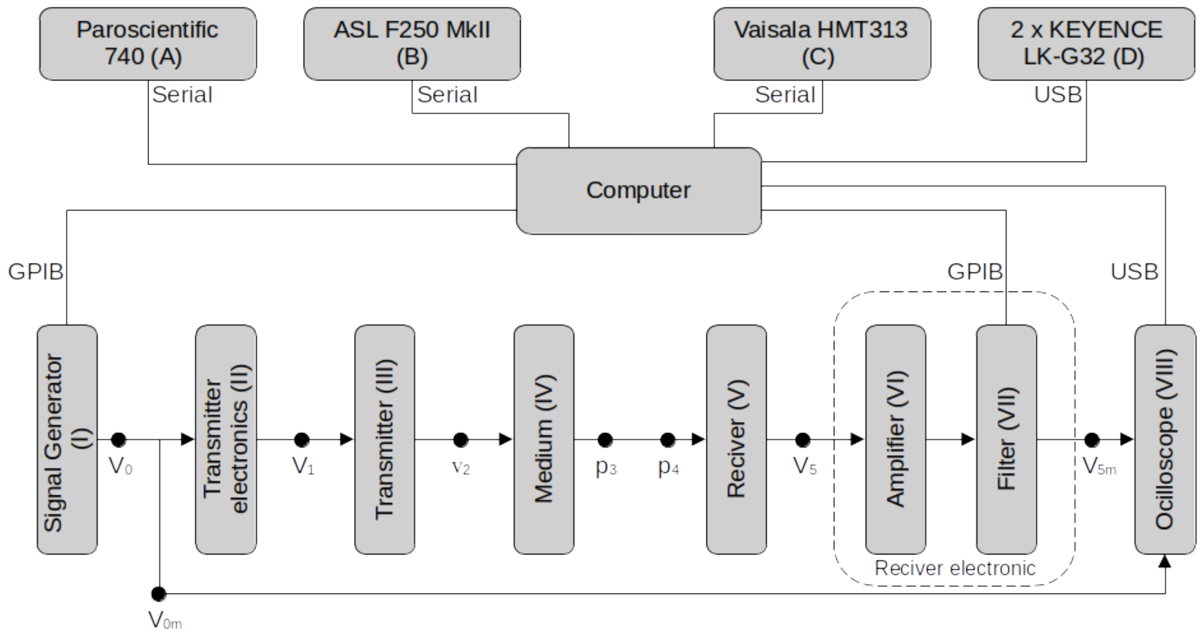



Figure 3.3: Block diagram of the acoustical setup and signal path. The blocks represent different equipment and are given in Table 3.2.

From the measurement system shown in Fig. 3.3, the measurements start with settings being sent from the computer to the signal generator (I), oscilloscope (VIII), and filter (VII). The signal generator creates the desired signal and sends it out through the output port. The oscilloscope reads the signal in channel one, which is connected to the signal generator and transmitter electronics (II) via BNC T-connector. The signal continues through the transmitter electronics, connected to the transmitter (III) via coaxial cable. Further, the transmitter sends the signal through the air (IV) before reaching the receiver (V). The received signal continues further to receiver electronics which amplifies (VI) the signal and filters the signal with a band-pass filter. The termination of the signal happens at the oscilloscope channel two after being amplified and filtered.


Figure 3.4: A overview of some of the equipment in the acoustical setup, where (A) is the barometer, (I) is the signal generator, (VI) is the amplifier, (VII) is the filter, and (VIII) is the oscilloscope (Table 3.2).

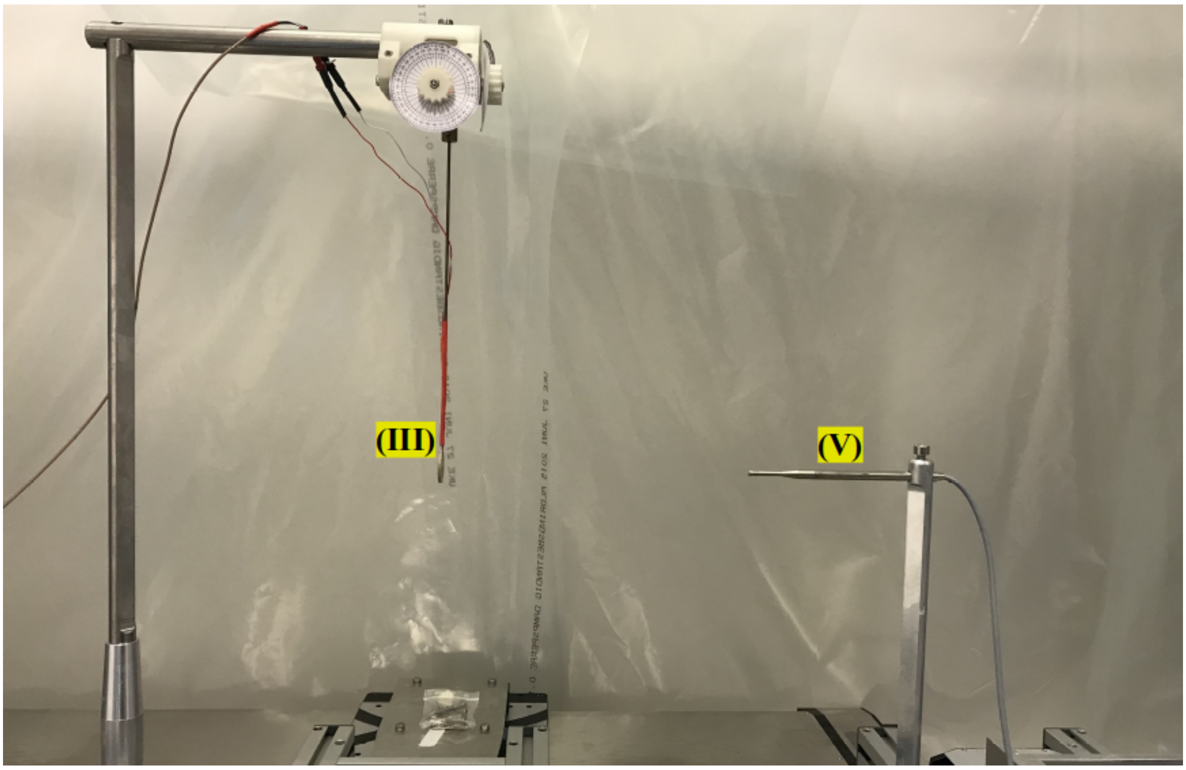



Figure 3.5: A overview of the chamber in the acoustical setup, where (III) is the piezoelectric disk/transmitter and (V) is the microphone/receiver (Table 3.2).

The setup frame is built up of aluminum profiles and placed inside a plastic sheet chamber. The chamber intends to lower the airflow to increase the accuracy of measurements. The measurements of the environmental parameters that take place inside the chamber are temperature (B)(C) and humidity (C), see Fig. 3.6. Inside the chamber is the placement of the transmitter, receiver, lasers stage (D), and all the moving stages as well. The only environmental parameter measured outside the chamber is the pressure (A).

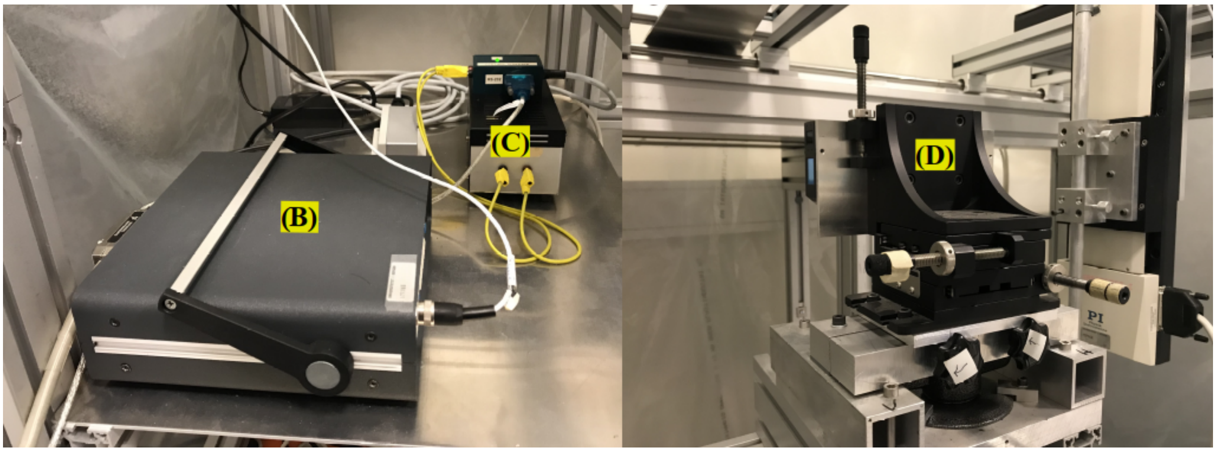



Figure 3.6: A overview of some of the equipment in the acoustical setup, where (B) is the ASLF250 measuring the temperature, (C) the Vaisala measuring temperature and humidity, and (D) is the laser stage.

### 3.3.1 Signal generator (I)

The signal generator used in the present work is the Agilent 33220A [4], which communicates with the computer through GPIB. By predefining the settings, the instrument generates a sinusoidal burst and sends it out through the output port as $V_0$. Changeable settings are the number of cycles, frequency, burst rate, and signal voltage amplitude. This work uses a 60-cycle sine burst and several different frequencies, a burst rate of 25Hz, and a peak-to-peak voltage signal $V_{0_{pp}}$ of 1 volt to avoid non-linearities of the piezoelectric disk at resonance frequencies. The low burst rate frequency used is because of letting reverberations of the piezoelectric disk die out and not letting any reflection interfere with the next burst. The output impedance of the signal generator is 50 Ω, and at low output impedance, the output voltage $V_0$ usually is twice the programmed voltage. This doubling of the voltage is because, with high frequency and broadband signals, coaxial cables have a characteristic impedance of 50 Ω for practical reasons and will halve the voltage $V_0$ to 1 volt peak-to-peak. This halving of the voltage could easily be calculated by putting a load impedance in series to a Thevenin circuit, and the output voltage over the load impedance becomes

$$V_{0m} = V_0 \left( \frac{50\Omega}{50\Omega + 50\Omega_{sig}} \right) = V_0 \frac{1}{2} \,. \tag{3.2}$$

However, since the termination impedance of the oscilloscope is 1 MΩ, the read voltage will be approximately 2 volts peak-to-peak due to

$$V_{0m} = V_0 \left( \frac{1E6\Omega}{1E6\Omega + 50\Omega_{sig}} \right) \approx V_0 \, . \tag{3.3}$$

With the piezoelectric disk considered in this Thevenin circuit and parallel to the load impedance of the oscilloscope, this voltage $V_{0m}$ measured changes dependent on the frequencies. Outside resonance of the disk, the perceived impedance is much higher than at the resonances. This change in impedance leads to the voltage $V_{0m}$ measured being closer to 2 volts outside resonances and closer to 1 volt at resonance.

### 3.3.2 Transmitter electronics (II)

Transmitting electronics in this work is only a cable linking the piezoelectric disk to the BNC-T connector at the oscilloscope. It is assumed that this cable has the same characteristics as an RG58 coaxial cable, see Sect. 3.3.10. This transmitting electronics transfer the voltage $V_{0m}$ measured at channel one at the oscilloscope to the piezoelectric disk electrodes. The voltage $V_1$ over the disk is calculated using the transfer function Eq. 2.21.

### 3.3.3 Transmitter (III)

The transmitter used in the present work is the piezoelectric ceramic disk Pz27. This ceramic disk, made by Meggit, states low aging rates and stable performance [46]. This ceramic disk type is a soft lead zirconate titanate (PZT) [46]. It has characteristics such as; high Curie temperature, low-temperature coefficients, and low mechanical quality factors [46], which allows the disk to be used in a variety of applications, such as flow meters, and therefore of interest in this work. The stated dimension of the disk is 2 mm in the thickness direction and 20 mm in diameter. However, the more accurate dimensions are measured and given in Table 5.3 in Sect. 5.3. The disk used in this work has previously been used by [26][7][64], and named #7. This disk has two wires soldered to the electrodes and hung up in a steel rod attached to the axis of rotation. A voltage $V_1$ applied to the electrodes is generated at the signal generator and sent through the transmitting electronics. The voltage sets the disk into motion and starts oscillating with a given frequency. This oscillation excites the piezoelectric disk with a velocity $v_2$ normal to the electrode front and is transmitted into longitudinal free-field pressure waves $p_3$ out in the medium.

### 3.3.4 Medium (IV)

The medium in this work that surrounds the transmitter and receiver is air. By monitoring the environmental parameters by measuring the temperature T in °C, relative humidity RH in percentage %, and pressure P in hPa, the air absorption is calculated by Eq. 2.10. The absorption is essential when correcting for losses in pressure amplitude $p_4$, which makes the measurement pressure comparable to the FE simulation pressure.

### 3.3.5 Receiver (V)

The receiver in this work is a 1/8-inch pressure-field microphone of the brand Brüel & Kjær and is the type 4138 [17]. It measures the free-field pressure $p_4$ and converts it into a voltage $V_5$. More about the microphone in Sect. 3.4, later in this chapter. This section also covers the calibration of the microphone sensitivity.

### 3.3.6 Amplifier (VI)

The amplifier used in the present work is the Brüel & Kjær 2636 Measuring Amplifier and is adjusted manually [18]. This amplifier is one of two receiver instruments in this acoustical setup. In this work, the amplifier gain setting is 60 dB and split between the input and output ports by 40 dB and 20 dB, respectively. Brüel & Kjær state the amplifier's frequency response to be flat, 0 dB in the range from 1 Hz to 200 kHz, see Fig 3.7 and can provide up to 100 dB gain with a step of 10 ± 0.05 dB [18]. However, previous work done by Mosland and Hauge [48][29] did investigate the amplifier's frequency response and found a more accurate frequency response, see Fig 3.8. This new frequency response shows that in the frequency range from approx. 170 kHz to 220 kHz, an additional gain of 0.1 dB must be taken into account.

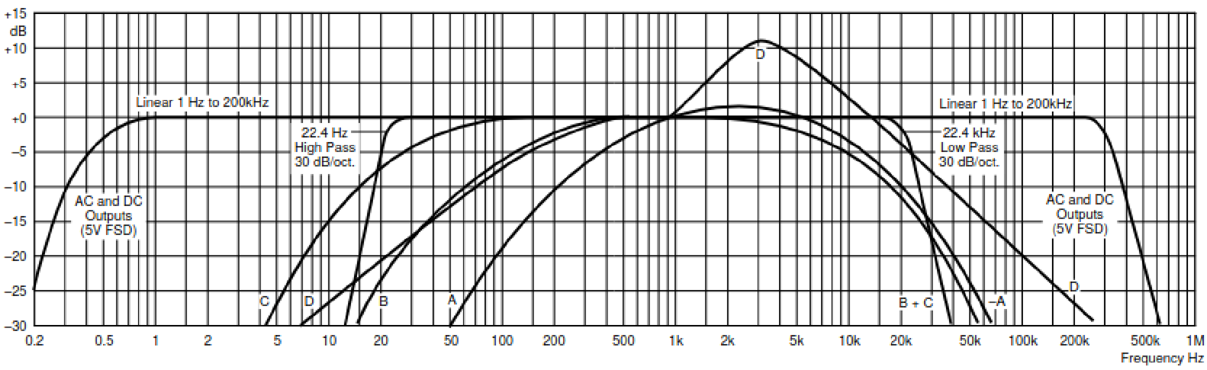



Figure 3.7: Typical overall frequency response of the Brüel & Kjær 2636 Measuring Amplifier [18].

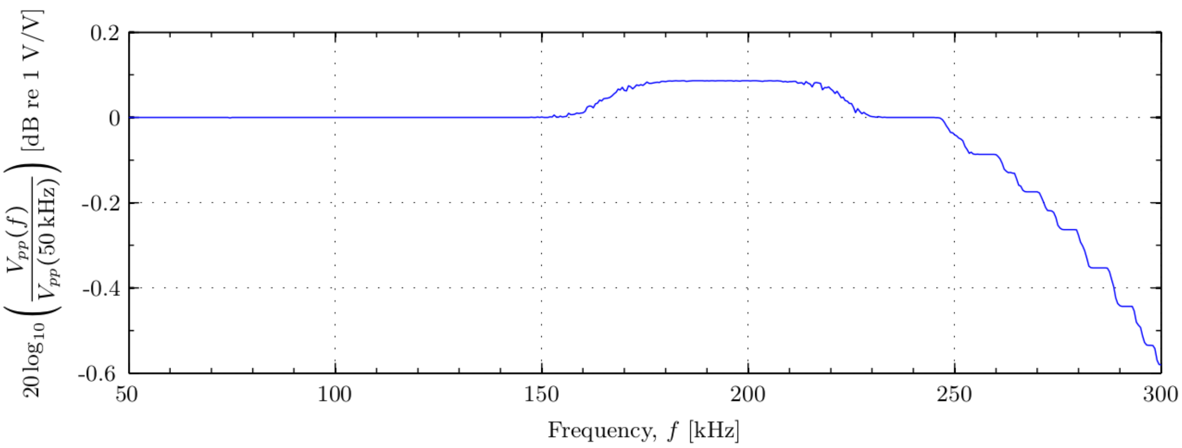



Figure 3.8: The frequency response of the Brüel & Kjær 2636 Measuring Amplifier created by [48] by conducting peak-to-peak voltage measurement in the frequency range from 50-300 kHz and normalizing measurements to the peak-to-peak voltage at 50 kHz. This frequency response is taken from [48].

### 3.3.7 Filter (VII)

The filter used in the present work is the Krohn-Hite 3940 filter [40], which communicates with the computer through GPIB. It has two channels, the first for a high-pass filter and the second for a low-pass filter. The cutoff frequency for the high-pass filter is set to half the measurement frequency and twice the measurement frequency for the low-pass filter. The attenuation outside the band-pass for this filter is 24 dB per octave [40]. It is possible to adjust the filter setting with the **MeasurmentParemeters.m** script in Appendix A.3.

### 3.3.8 Oscilloscope (VIII)

The oscilloscope used in the present work is the Tektronix DPO3012 [66], which communicates with the computer through USB. The oscilloscope has two input channels with termination impedance of 1 MΩ and 11 pF. Channel one read voltage $V_0$ from the signal generator connected to the oscilloscope by coaxial cable and measured as $V_{0m}$. Channel two read the output voltage $V_5$ from the microphone treated by the amplifier and band-pass filter connected by coaxial cables, terminated at the oscilloscope, and measured as $V_{5m}$. The oscilloscope's changeable settings are averaging, sample count, time per division (t/div), voltage per division (V/div), and bits. The voltage signal measured is averaged by 128 bursts. The t/div and V/div are automatically updated with scripts in the MatLab app, where t/div and V/div depend on the currently displayed signal length and voltage amplitude. The sample count is set to 10 000 samples, the t/div is usually 40 $\mu$s or 100 $\mu$s, and it is ten divisions on the oscilloscope, leading to a 0.4 ms to 1 ms time window. This time window leads to a sampling frequency of 25 MS/s or 10 MS/s and is more than sufficient for frequencies between 50 kHz to 250 kHz. The bits resolution of the oscilloscope is set to 16-bit.

### 3.3.9 Environmental parameters

The monitored environmental parameters are pressure, temperature, and relative humidity. The barometer Paroscientific 740 (A) measures the pressure in hPa with a DIGIQUARTZ® pressure transducer. This pressure transducer gives high accuracy, resolution, and long-term stability resulting in an uncertainty of ±0.01 % [50]. Both the ASL F250 MkII (B) and the Vaisala HMT313 (C) measure the temperature inside the plastic sheet chamber. They use a PT100 sensor with an uncertainty of ±0.01 °C and ±0.1 °C, respectively [8][70]. Vaisala HMT313 also measures relative humidity with a HUMICAP® with an uncertainty of ±0.6 % when relative humidity is below 40 % and ±1.0 % for above 40 % [70].

### 3.3.10 Cables

In the acoustical air setup, are two types of cables used. The first cable connects the piezoelectric disk to the BNC-T connector at the oscilloscope and is assumed to have the same characteristics as an RG58 coaxial cable. The second type is RG58 coaxial cables connecting the instruments. The coaxial cables are of different lengths, see Table 3.3, and the characteristic impedance of type RG58 coaxial cable is 50 Ω. There are also typical specifications for the RG58 coaxial cable listed in Table 3.4 [73].

Table 3.3: Measured or given length of the cables in the measurement setup.

| Cable connection | | Cable length |
|---|---|---|
| Input1 | Input2 | |
| Signal generator | BNC-T connector | 0.25 m |
| BNC-T connector | Piezoelectric disk | 3.00 m |
| Amplifier | Filter (Ch.1) | 0.50 m |
| Filter (Ch.1) | Filter (Ch.2) | 0.80 m |
| Filter (Ch.2) | Oscilloscope | 1.50 m |

Table 3.4: Typical specifications for RG58 coaxial cables [73]

| Description | Value | Unit |
|---|---|---|
| Inductance (L) per meter | 250 | [nH/m] |
| Capacitance (C) per meter | 100 | [pF/m] |

## 3.4 Brüel & Kjær 4138 microphone

The receiver in the acoustical setup is a microphone of the brand Brüel & Kjær type 4138 [17], which is a 1/8-inch microphone. This microphone is used together with a 1/4-inch Brüel & Kjær preamplifier type 2633 [17]. The preamplifier is necessary since the microphone needs a polarization voltage of 200 V. Brüel & Kjær states that the preamplifier has a flat frequency response [17]; therefore, only the microphone's frequency response $|M(f)|$ is considered in calibration. The given frequency response from the calibration chart of the microphone [19] goes from 20 Hz to 200 kHz, and is calibrated with an electrostatic actuator and plotted relative to reference microphone sensitivity $|M_{ref}(250Hz)|$ at 250 Hz. The calibration chart is digitized by importing a scanned image of the frequency response into an online software [60] and recreating the chart by manually inserting data points. The final result of the data points is plotted, see Fig. 3.9. This method of recreating a plot will give some errors. Factors that give errors are the thickness of the line, manually placing the data points on the line, and limitation on pixel movement of the points. Previous work by [24] estimated this error to be ±0.1 dB.

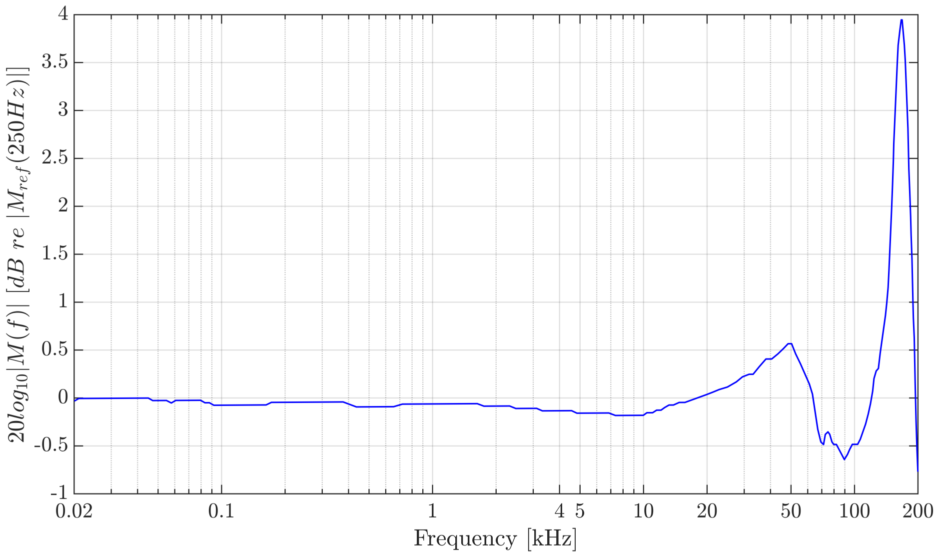



Figure 3.9: Open-circuit pressure response relative to 250 Hz for Brüel & Kjær microphone type 4138 with the serial number: 1832479.

In addition to the open-circuit pressure response, it is necessary to include the free field correction for a given incident angle of sound pressure $\theta_i$. The free-field correction describes the increase of sound pressure caused by diffractions [17]. This increase happens at high frequencies when the wavelength becomes comparable to the diameter of the microphone. By assuming the incident angle of sound pressure between transmitter and receiver is $\theta_i$ equals 0 degrees, which is the case in this work, the Brüel & Kjær free-field correction chart is digitized for this angle and added to the open-circuit pressure response. The chart used is for the 4138 microphone with a protection grid over the diaphragm [17] and digitized with the same method earlier, and the result is given in Fig. 3.10. This chart goes from 4-200 kHz. The open-circuit pressure response added to the free-field correction gives the microphone system's free-field open-circuit pressure response from 4-200 kHz, see Fig. 3.11.

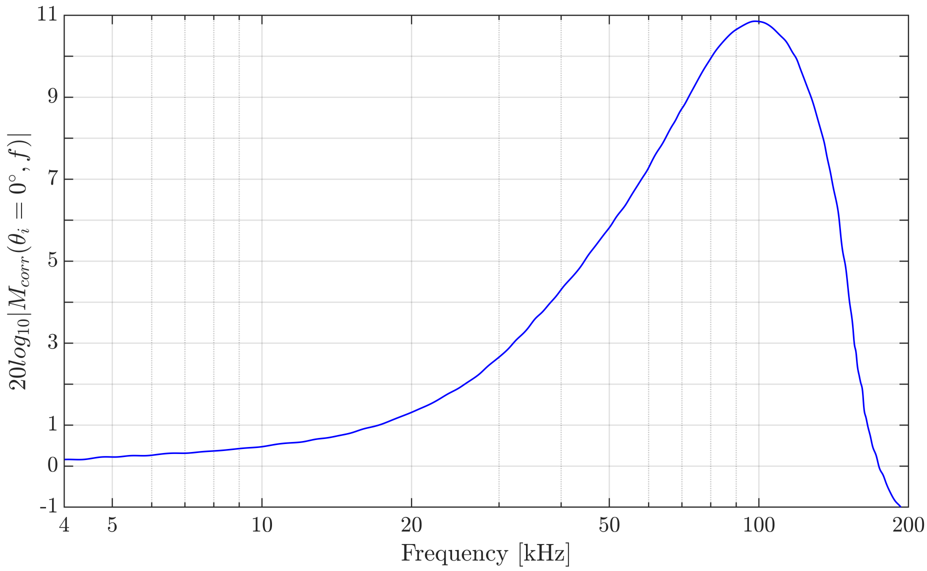



Figure 3.10: Free-field correction relative to 250 Hz for Brüel & Kjær microphone type 4138 with protection grid [17].

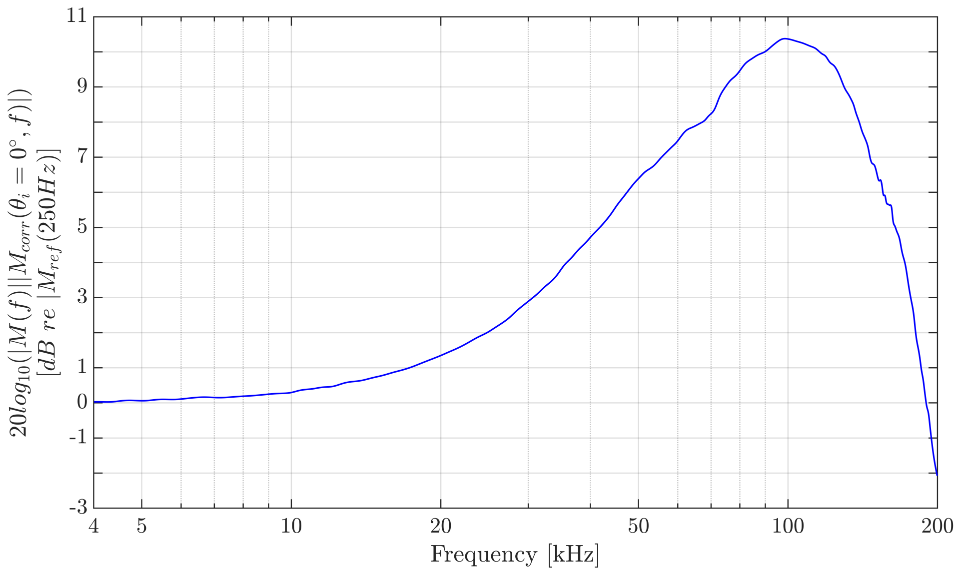



Figure 3.11: Open-circuit pressure response, including free-field correction, relative to 250 Hz for Brüel & Kjær microphone type 4138 with the serial number: 1832479.

### 3.4.1 Microphone sensitivity calibration using a pistonphone

This work uses a Brüel & Kjær 4228 pistonphone to find the microphone's system sensitivity [16]. This pistonphone produces a nominal sound pressure level $_{Stated}\ SPL$ of 124.11 ± 0.09 dB relative to 20 $\mu$Pa [20]. It also has a nominal frequency of 250 Hz, or the exact frequency is $10^{2.4}$ Hz (ISO266) ±0.1 % [20]. If the calibration is under conditions other than the reference condition, the sound pressure level is given as [20]

$$_{Actual}\ SPL = {}_{Stated}\ SPL + \Delta L_V + \Delta L_p\ , \tag{3.4}$$

where $\Delta L_V$ is the load volume correction, $\Delta L_p$ is a correction if ambient pressure deviates from 1013 mbar, and $_{Actual}\ SPL$ is the corrected sound pressure level. The load volume correction $\Delta L_V$ for the given microphone 4138 is 0 dB when using the pistonphone adaptor DP 0774, see Table 3.5. The ambient pressure correction $\Delta L_p$ can be read directly off barometer UZ0004 supplied with the pistonphone calibration set [16], see Fig. 3.12. Alternatively, calculated using an interpolation function between different points read of the barometer UZ0004, see Table 3.6, and then using the ambient pressure measured with the Paroscientific 740, during the calibration.

Table 3.5: The load volume correction $\Delta L_V$ given by [20].

| Size | Piston-phone Adaptor | Micro-phone Type | Load Volume Correction $\Delta L_V$ (dB) | | | |
|---|---|---|---|---|---|---|
| | | | With Protection Grid | Without Protection Grid | With Adaptor Ring DB 0111 | With Adaptor Ring UA 0825 |
| 1″ | None | 4131/32 | −0.05 | — | +0.25 | — |
| | | 4144/45 | +0.05 | — | +0.25 | — |
| | | 4160 | +0.43 | +0.28 | — | — |
| | | 4179 | −0.05 | — | — | — |
| 1/2″ | DP 0776 | 4129/30 | −0.02 | — | — | — |
| | | 4133/34 | 0.00 | — | — | +0.08 |
| | | 4147 | 0.00 | — | — | +0.08 |
| | | 4148 | −0.04 | — | — | — |
| | | 4149 | 0.00 | — | — | — |
| | | 4155 | −0.03 | — | — | — |
| | | 4165/66 | −0.03 | — | — | — |
| | | 4176 | −0.02 | — | — | — |
| | | 4180 | — | +0.08 | — | — |
| | | 4181 | 0.00 | — | — | — |
| | | 4183 | −0.02 | — | — | — |
| 1/4″ | DP 0775 | 4135/36 | 0.00 | — | — | — |
| 1/8″ | DP 0774 | 4138 | 0.00 | — | — | — |

Table 3.6: Different points read of the supplied barometer UZ0004 in the pistonphone calibration set [16], see Fig. 3.12, used to calculate the ambient pressure correction $\Delta L_p$.

| Pressure | 685 mbar | 800 mbar | 940 mbar | 990 mbar | 1013 mbar | 1060 mbar |
|---|---|---|---|---|---|---|
| Correction | -3.4 dB | -2.05 dB | -0.65 dB | -0.20 dB | 0.00 dB | 0.39 dB |


Figure 3.12: Supplied barometer UZ0004 in the pistonphone calibration set [16].

Using the definition of the microphone sensitivity given in Eq. 2.37 and using the pistonphone, the microphone sensitivity $|M_V|$ is calculated as

$$M_V(250\ Hz) = \frac{V_{eff}(250\ Hz)}{p_{eff}(250\ Hz)}, \tag{3.5}$$

where $V_{eff}$ is the calculated open-circuit effective voltage with Eq. 2.38, and the pressure $p_{eff}$ is the free-field effective pressure at the front of the microphone with the assumptions of normal incidence and plane waves. The oscilloscope records the microphone's voltage signal $V_{5m}(t)$ produced by the pistonphone, see Fig. 3.13, and fast Fourier transformed to the voltage $V_{5m}(f)$, see Sect. 3.7.2, where $V_{5m_{pp}}(250\ Hz)$ is then calculated as

$$V_{5m_{pp}}(250\ Hz) = 4 * V_{5m}(250\ Hz) = 402.55\ mV\ , \tag{3.6}$$

where $V_{5m_{pp}}(250\ Hz)$ is the calculated peak-to-peak voltage from the fast Fourier transformed voltage $V_{5m}(250\ Hz)$. Then, with the calculated $V_{5m_{pp}}(250\ Hz)$, the effective voltage $V_{eff}$ is then deduced by using the Eq. 2.38, which gives

$$V_{eff}(250\ Hz) = \frac{V_{5_{pp}}(250\ Hz)}{2\sqrt{2}} = \frac{1}{2\sqrt{2}}\frac{V_{5m_{pp}}(250\ Hz)}{H^{VV}_{55m}(250\ Hz)} = \frac{1}{2\sqrt{2}}\frac{402.55}{9.998}\ mV = 14.235\ mV\ , \tag{3.7}$$

where $H^{VV}_{55m}$ is the transfer function in Eq. 2.29 and $V_{5_{pp}}$ is the peak-to-peak voltage out of the microphone. In this calibration, the amplifier gain is set to 20 dB.

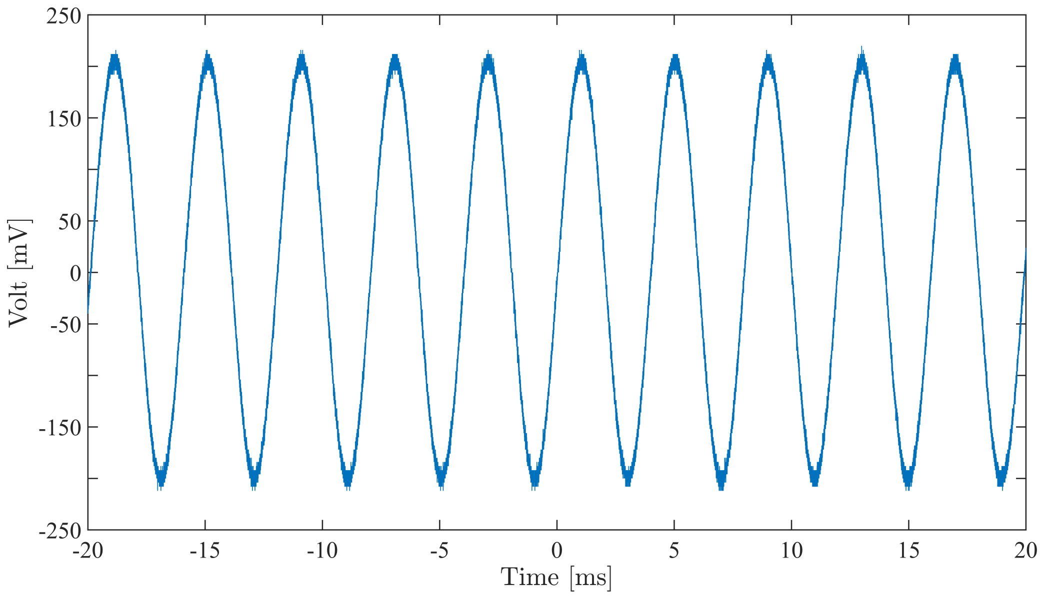



Figure 3.13: $V_{5m}(t)$ measurement generated by the pistonphone with an amplifier gain of 20 dB.

By calculating the $_{Actual}\ SPL$ with Eq 3.4, which gives

$$_{Actual}\ SPL = 124.11\ dB - 0.251\ dB + 0\ dB = 123.859\ , \tag{3.8}$$

where $\Delta L_V$ equals 0 dB and $\Delta L_p$ equals -0.251 dB, the free-field effective pressure can be calculated using the Eq. 2.39, which gives

$$p_{eff}(250\ Hz) = p_{ref} \cdot 10^{\frac{_{Actual}SPL}{20}} = 20\mu Pa \cdot 10^{\frac{123.859}{20}} = 31.1875\ Pa \tag{3.9}$$

where $p_{ref}$ is the reference pressure 20 $\mu$Pa. Then with Equation 3.5, the microphone sensitivity is calculated as

$$M_V(250\ Hz) = \frac{14.235\ mV}{31.1875\ Pa} = 0.456433\ mV/Pa. \tag{3.10}$$

This calibrated microphone sensitivity $|M_V(250\ Hz|$ at 250 Hz has changed largely in previous works and calculated to 0.3157 mV/Pa [1], 0.535 mV/Pa [6], 0.493 mV/Pa [48][29], and now 0.456433 mV/Pa. None of the previous studies answers the significant changes in calibration results. An observation made in this work was that the higher the gain, the higher the signal distortion of the measured signal of the pistonphone. This distortion is the reason why 20 dB gain is used in this work, while it has previously been used as 30 dB by [1] and [6] and 20 dB by [48][29]. The calibration and calculated microphone sensitivity at 250 Hz are then added to the open-circuit pressure response, including a free-field correction in Fig.

3.11, to deduce the free-field open-circuit receiver sensitivity, which is given as

$$20log_{10}|M_{cal}(f)| = 20log_{10}\left(\frac{|M(f)|\,|M_{corr}(\theta_i = 0, f)|\,|M_V(250Hz)|}{|M_{ref}(250Hz)|}\right)\ [dB\ re.\ 1V/Pa] \quad (3.11)$$

where $|M_{cal}(f)|$ is the calibrated microphone sensitivity and plotted in Fig. 3.14. The Fig. 3.15 is the equivalent free-field open-circuit receiver sensitivity in mV/Pa.

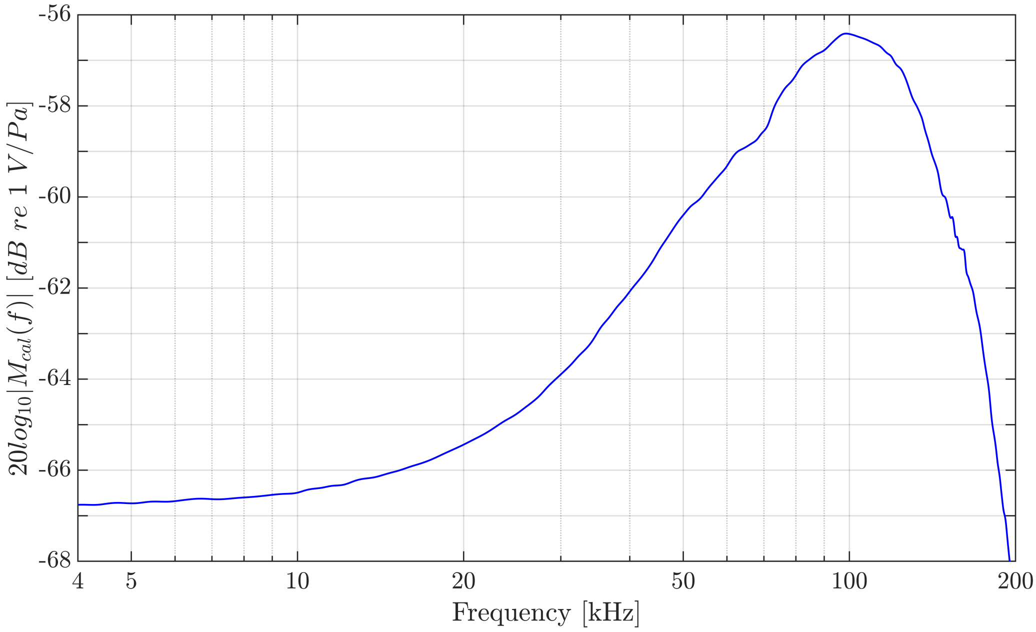



Figure 3.14: The free-field open-circuit microphones sensitivity from 4-200 kHz in dB.

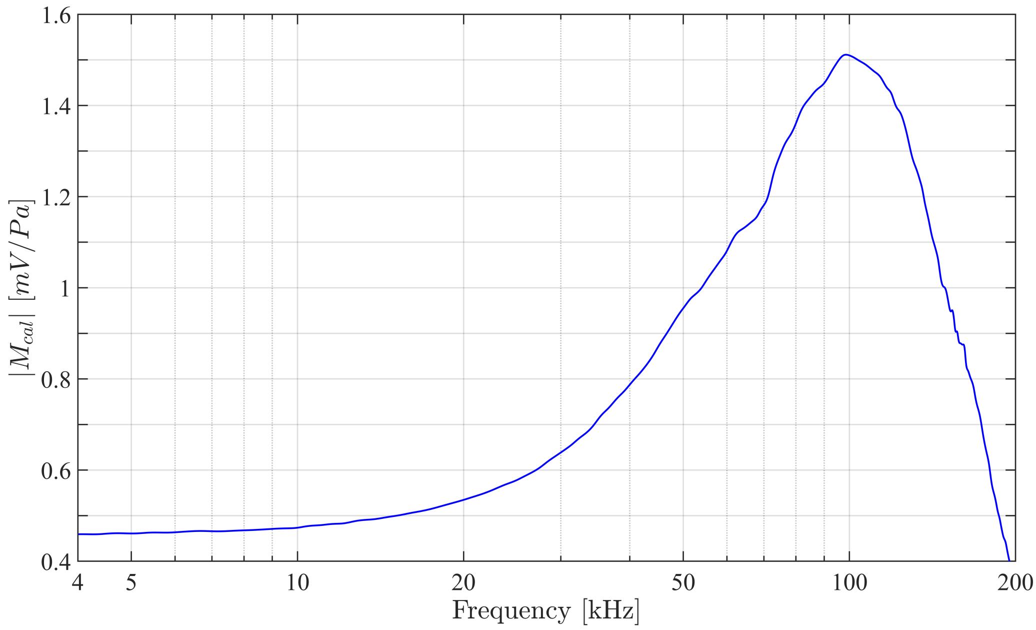



Figure 3.15: The free-field open-circuit microphones sensitivity from 4-200 kHz in mV/Pa.

## 3.5 Motor's setup

The present work's motor setup contains two motion controllers, one encoder for Y-stage, three linear stages (X, Y, and Z-stage), and one rotation stage (R-stage). For an illustration of the motor setup, see the block diagram of the motor setup, Fig. 3.16. All linear stages use reference coordinates found by an induction sensor at the end of each linear stage. These reference coordinates are repetitive and therefore used to find the origin position of the transducer. More details about using reference coordinates to find the origin position of the transducer are in the Chapter 4, and the importance of finding the origin position of the transducer is in the Sect. 4.5.

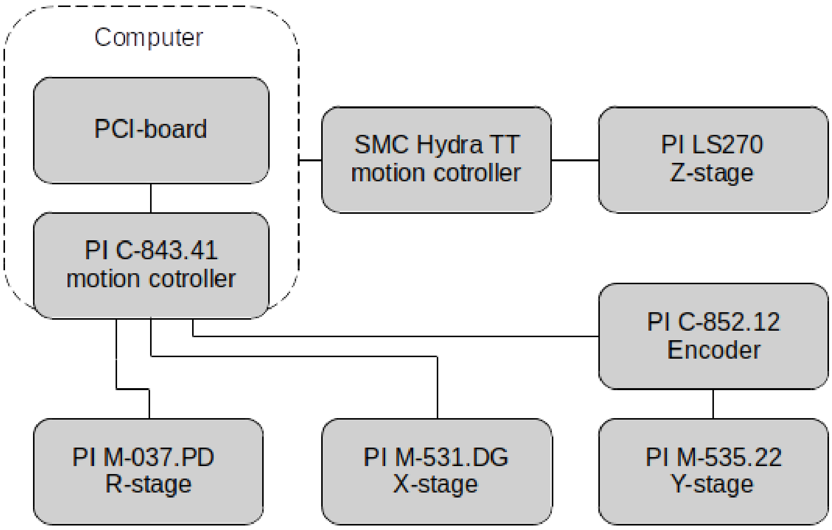



Figure 3.16: Block diagram of the acoustical motor setup and signal path. The blocks represent different motors and motor controllers and are given in Table 3.1.

The X and Y-stage set the microphone's position, and the Z-stage set the distance between the microphone and transducer. The R-stage rotate the transducer and is used to set the angle $\theta_R$ of the transducer relative to the microphone. The travel length of the X and Y-stage is 300 mm, and the Z-stage is 1016 mm, but it is limited to about 900 mm because the X and Y-stage are a part of the Z-stage linear rail system. The R-stage has no limits and can rotate an infinite number of degrees, but it is limited to roughly $\pm 90$ degrees in this work. All linear stages move in steps, where one step is equivalent to one mm, or ten step is equivalent to ten mm, for the rotation stage, one step is equivalent to one degree. In this work, it is needed to adjust the Y-stage after observing wrong travel distance in mm relative to the given input step.

### 3.5.1 Travel distance adjustments of Y-stage

The internal settings of the Y-stage are given by [52] and are:

- the ball screw pitch, $b$, equal to 2 mm per revolution
- the encoder resolution, $e_{res}$, equal to 2000 counts per revolution
- the backlash-free gear head, $g$, which is 29.64197531:1
- the stage resolution, $y_{res}$, (calculated from $b/e_{res}/g$) equal to 0.34 $\mu$m per count

Suppose any of the internal settings deviate from [52]. In that case, it will lead to the encoder interpreting an inaccurate $y_{res}$ and reading the incorrect number of counts leading to a wrong travel distance. It is observed that the Y-stage does not move the desired travel length but moves a shorter distance. The error is proportional to the input value by a common factor, e.g., an input value of one step results in a travel distance of half an mm, or two step results in a travel distance of one mm. The repetitive error in the travel distance indicates that the error comes from the internal settings of the Y-stage and not because of damaged parts. Since changing the internal settings has not been possible, a common alternative way of adjusting the travel length has been used. This alternative way is to look at the ratio of the output travel distance and the input value and define this ratio as travel resolution, $d_{res}$ as

$$d_{res} = \frac{d}{s} = 1 \,, \tag{3.12}$$

where $s$ is the step and $d$ is the travel distance in mm. If the input value is not equal to the travel distance, which is the case for the Y-stage, then $d_{res}$ deviates from one and must be adjusted with a factor $f$ as

$$f \cdot d_{res} = f \cdot \frac{d}{s} = 1 \,. \tag{3.13}$$

The purpose of this is to alter the number of counts to be counted by the encoder without physically changing the internal settings but still achieving the correct travel distance. The travel distance $d$ is measured with a dial gauge with an uncertainty of ±0.01 mm. This dial gauge is made by the brand TESA Technology [67]. The positioning of the dial gauge, shown in Fig. 3.17, is as parallel as possible to the Y-stage such that the dial gauge plunger moves parallel with the stage. The travel distance is measured ten times using $s$ set to 25, and the average of $d$ is calculated as

$$\hat{d} = \frac{\sum_{i=1}^{N} |d_i|}{N} \,, \tag{3.14}$$

and the uncertainty of the travel distance is

$$\sigma_d = \sqrt{\sigma_{std}^2 + \sigma_{dial}^2} \,, \tag{3.15}$$

where $\sigma_{dial}$ is the uncertainty of the dial gauge and $\sigma_{std}$ is the estimated standard deviation calculated as

$$\sigma_{std} = \sqrt{\frac{\sum_{i=1}^{N}(d_i - \hat{d})^2}{N-1}} \,. \tag{3.16}$$


Figure 3.17: Dial gauge lined up parallel to the Y-stage, ready to measure the travel distance of the stage.

After calculating the factor with Eq. 3.13, $f$ is tested to find the accuracy of $d_{res}$ by taking the input value, $s$, multiplying it with $f$, and repeating ten distance measurements. If the distance resolution still deviates from one, a new factor is calculated as

$$f_{new} \cdot d_{res} = f_{new} \cdot \frac{d}{s \cdot f_{old}} = 1 \,, \tag{3.17}$$

where $f_{new}$ is the new factor and $f_{old}$ is the old factor. Then to test $f_{new}$ to find the accuracy of $d_{res}$, $f_{new}$ is replaced with $f_{old}$, and the ten distance measurements are repeated.

This process is repeated until $d_{res}$ reaches appropriate values, see Fig 3.18. To calculate the uncertainty of the $d_{res}$ from Eq. 3.17 the uncertainty $\sigma_{d_{res}}$ becomes

$$\sigma_{d_{res}} = \sqrt{\left(\frac{\partial d_{res}}{\partial d} \cdot \sigma_d\right)^2 + \left(\frac{\partial d_{res}}{\partial s} \cdot \sigma_s\right)^2 + \left(\frac{\partial d_{res}}{\partial f_{old}} \cdot \sigma_{f_{old}}\right)^2} = \sqrt{\left(\frac{1}{s \cdot f_{old}} \cdot \sigma_d\right)^2}, \tag{3.18}$$

where $\sigma_{f_{old}}$ and $\sigma_s$ equals zero because the factor and step are exact quantities with zero uncertainty. The final adjustment factor, $f$ used, equals 1.69028, and the calculated distance resolution, $d_{res}$, equals $1.0000 \pm 0.00024$.

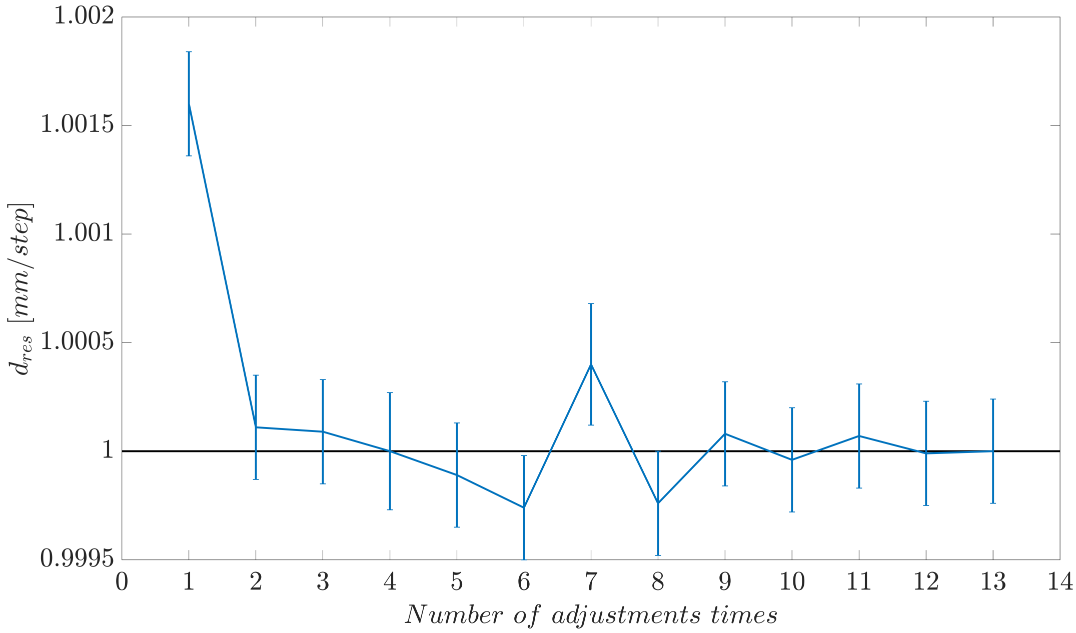



Figure 3.18: The change in distance resolution, $d_{res}$, is adjusted by the factor $f$ found using the input step, $s$, equal to 25 steps and the calculated distance $\hat{d}$ of ten distance measurements.

## 3.6 Reflections

This acoustic measurement setup contains many surfaces that can reflect a transmitted sound wave. Such as flat aluminum profiles, steel rods and plates, walls, floor and roof, plastic sheets, and wood frames. Most of these reflections are non-destructive on the received signal because of the late arrival time compared to the transmitted signal. The one destructive reflection that needs to be considered when measuring is the reflection from the vertical rod behind the transducer. The vertical steel rod has a wedge shape to reduce the reflections, see Figs. 3.5 and 3.19. This reflection from the vertical rod becomes apparent after the rotation angle $\theta_R$ deviates from 0 degrees, and the reflection starts to become a problem for the acoustic measurement signal, as illustrated in Fig. 3.19. Calculating the arrival time $t_{refl}$ of this reflection for every angle of $\theta_R$ is possible with the given equation

$$t_{refl}(\theta_R) = \frac{\sqrt{(d_{rod}cos(\theta_R) + z_0)^2 + (d_{rod}sin(\theta_R))^2} + d_{rod}}{c_{air}} = \frac{d_h(\theta_R) + d_{rod}}{c_{air}} , \tag{3.19}$$

where $d_{rod}$ is the distance between the rod and piezoelectric disk, $z_0$ is the direct length between disk and microphone, and $d_h$ is the distance from the vertical rod to the microphone, which is dependent on the R-stage rotation angle $\theta_R$.

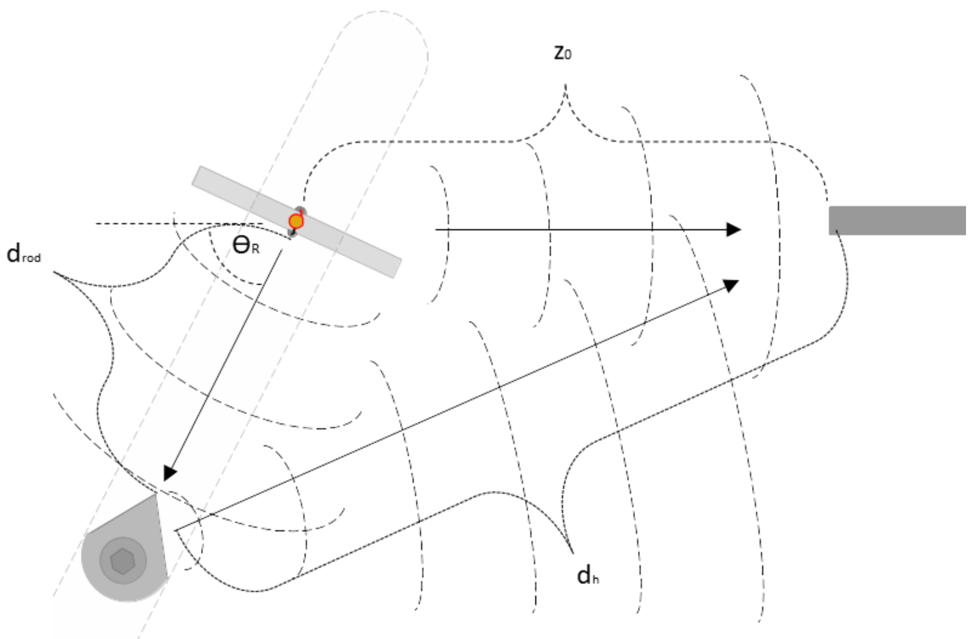



Figure 3.19: Illustration of direct travel path $z_0$, and reflection path through $d_{rod}$ and $d_h$ from piezoelectric disk to microphone.

By knowing the arrival time of the reflection $t_{refl}$ at every angle $\theta_R$, the interval used to calculate the voltage $V_{5m}(f)$ can be set from the start of the steady state of the acoustic measurement signal to the reflection's arrival time. This interval prevents the acoustic measurement signal from being interfered with by reflections and intends to get the best results possible by the acoustic measurement signal. However, this work uses a sinus burst with a calculated standard number of cycles, which is used throughout this work. In order to calculate the number of cycles to be used in the measurements in this work, it is first calculated the time difference between the acoustic measurement signal coming directly from the piezoelectric disk and the arrival time of the reflection from the vertical rod $\Delta t$ as

$$\Delta t(\theta_R, z_0) = t_{refl}(\theta_R) - \frac{z_0}{c_{air}} = t_{refl}(\theta_R) - t_{dir} , \tag{3.20}$$

where $t_{dir}$ is the direct signal between the disk and microphone. In this work, final measurements are not performed further out than distance $z_0$ equal to 0.3 m. To calculate the shortest $\Delta t$ for this distance ($z_0$ equal to 0.3), which is when $\theta_R$ is equal to 90 degrees, then $\Delta t$ can be used to calculate the number of cycles without reflection interfering with the direct signal of interest. The number of cycles is then calculated for the frequency 98860 Hz as

$$cycles = f \cdot \Delta t(\theta_R = 90\,^\circ, z_0 = 0.3\ m) = 98860\ Hz \cdot 0.737\ ms \approx 73\,. \tag{3.21}$$

With a calculated number of cycles to be 73, the standard number of cycles is set to 60-cycles. With a measurement signal combined as a short duration signal $t_{dur}$, which can be calculated as

$$t_{dur} = \frac{cycles}{f}\,, \tag{3.22}$$

and with a short burst period i.e. 25 Hz, it is accounted for that all reflections, and any reverberations in the piezoelectric disk die out before the next sine burst is transmitted.

By studying the received acoustic measurement signal, it is possible to confirm that the one destructive reflections come from the vertical steel rod. It can be confirmed by blocking the interference and seeing the reflection die out from the live signal at the oscilloscope. Alternatively, measuring and calculate the arrival time of the reflecting signal from different angles and compare it to the actual measurements, see Figs. 3.20-3.23. The present work uses both these methods. By analyzing the signals in Figs. 3.20-3.23, the estimated arrival time of the direct signal and reflection signal fits well with the several different angles' predictions.

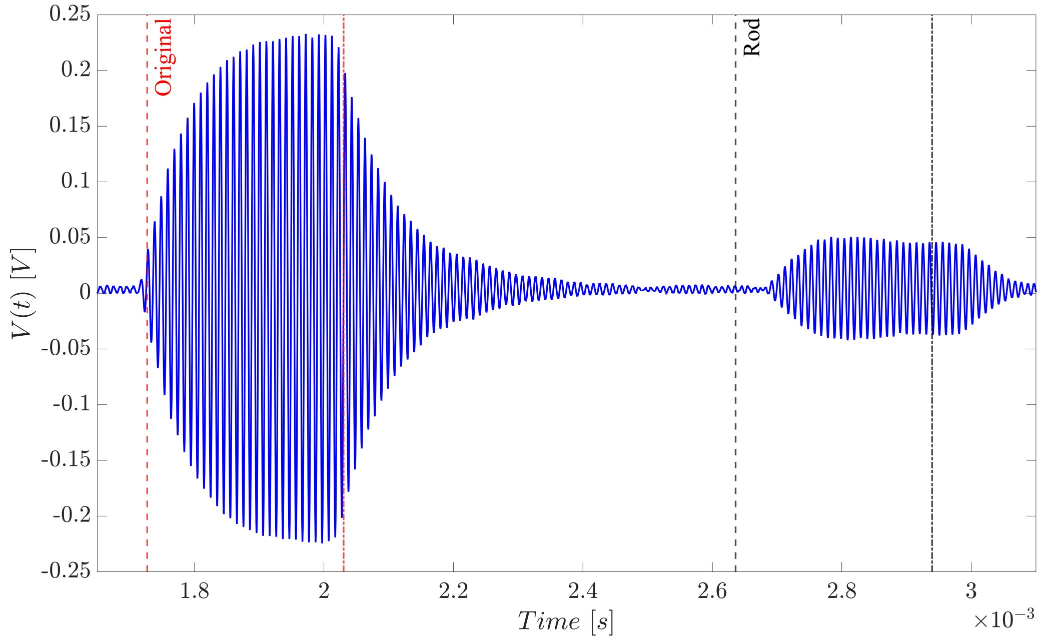



Figure 3.20: Reflection at $\theta_R$ equal to 55 degrees, a distance from the microphone to piezoelectric disk $z_0$ equals 600 mm, and a 30-cycle sine burst with frequency 98860 Hz.

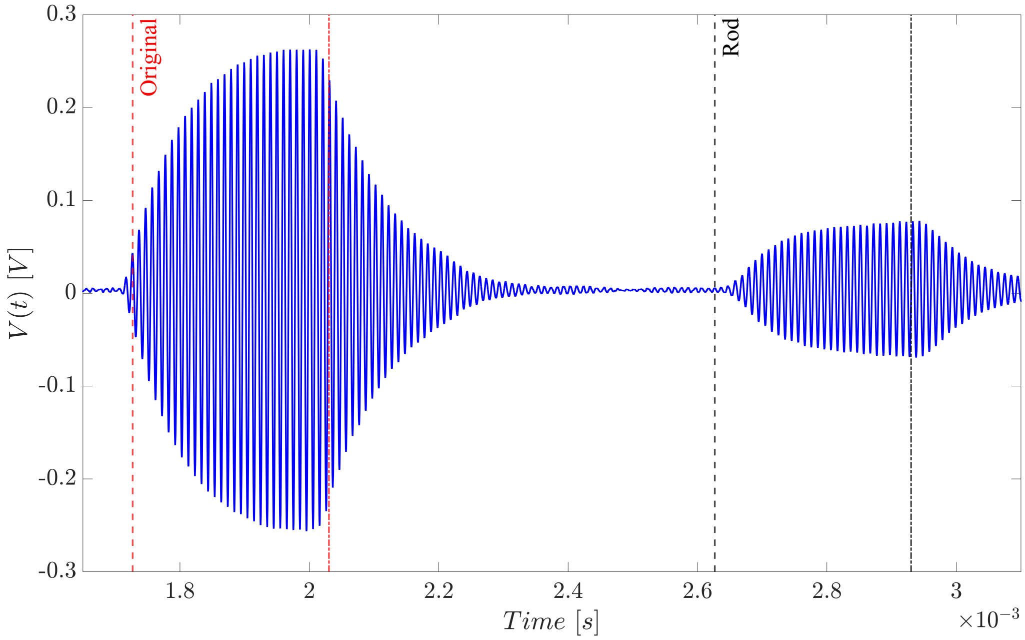



Figure 3.21: Reflection at $\theta_R$ equal to 60 degrees, a distance from the microphone to piezoelectric disk $z_0$ equals 600 mm, and a 30-cycle sine burst with frequency 98860 Hz.

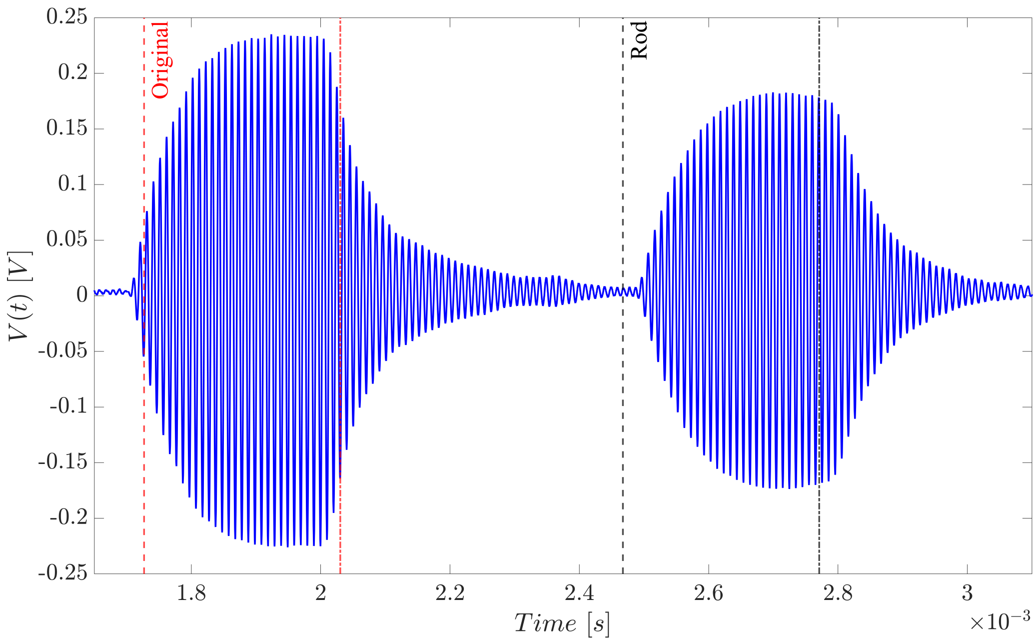



Figure 3.22: Reflection at $\theta_R$ equal to 80 degrees, a distance from the microphone to piezoelectric disk $z_0$ equals 600 mm, and a 30-cycle sine burst with frequency 98860 Hz.

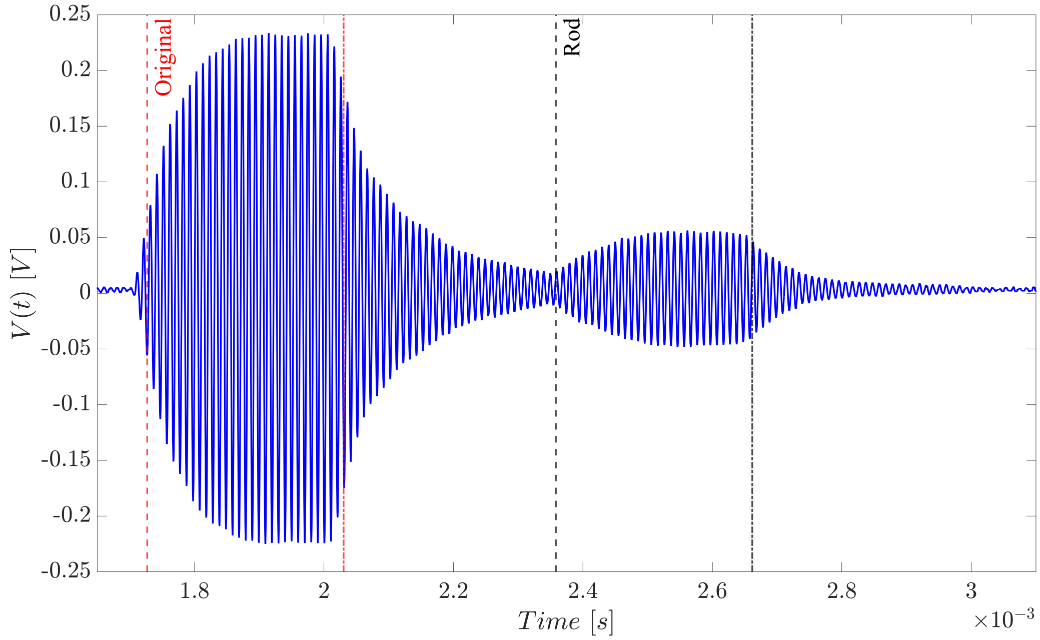



Figure 3.23: Reflection at $\theta_R$ equal to 90 degrees, a distance from the microphone to piezoelectric disk $z_0$ equals 600 mm, and a 30-cycle sine burst with frequency 98860 Hz.

## 3.7 Signal setup and processing

For all measurements performed in this work, the number of pulse cycles is set to 60. This selection is based on the arrival time of the first reflection from the vertical steel bar behind the transducer found in Sect. 3.6. To avoid reflection from other places in the measuring cage and give reflections from the cage time to die out and have no destructive effect on the measuring signal, a burst rate of 25 Hz is selected. This burst rate also gives the piezoelectric disk reverberations time to stop before the next cycle is sent out of the disk. The voltage out of the signal generator $V_{0pp}$ is 1 V peak-to-peak and is selected to avoid non-linearities in the piezoelectric disk that can occur at high input voltages. Because analyzing a vast amount of signals in this work, there is a need to find a standard interval range for the measured acoustic signal $V_{5m}(t)$ to be Fourier-transformed, which gives good and repetitive results. After performing measurements and analyzing the acoustic voltage signal $V_{5m}(t)$ over a more extended period, it is observed that the time for the signal to reach a steady state is approximately 30 pulses. The end time of the signal interval to Fourier transform is set to 5 pulses before the estimated end of the signal to ensure the interval is within the transmitted signal and no transient part is included. All measurements performed in this work are averaged 128 times.

### 3.7.1 Transmitted signal

The transfer function $H^{VV}_{0m1}$ in Eq. 2.21 is used to calculate the voltage $V_1$ across the electrodes of the piezoelectric disk by multiplying the transfer function with the Fourier-transformed measured voltage signal $V_{0m}(t)$. This measured voltage $V_{0m}(t)$ example is shown in Fig. 3.24, and to Fourier transform the signal, a steady state range is selected. This steady state range can change depending on the measuring frequency. The Fourier transform of the steady state range gives the voltage $V_{0m}(f)$.

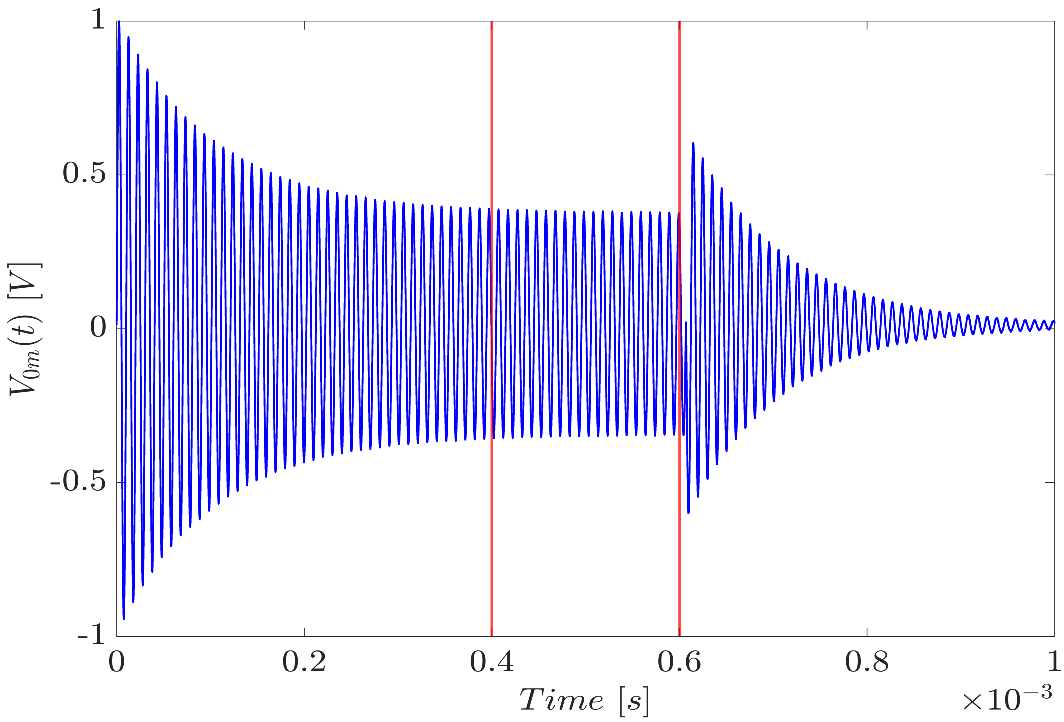



Figure 3.24: The measurement voltage $V_{0m}(t)$ measured at the input channel one on the oscilloscope with signal generator frequency and voltage $V_0(t)$ set to 98860 Hz and 1 Vp-p. The two red vertical lines represent the time interval's start and end, which is the interval that is FFT.

### 3.7.2 Received signal

The transfer function $H^{VV}_{55m}$ in Eq. 2.29 is used to calculate the output voltage $V_5$ from the microphone by taking the Fourier transformed measured acoustic voltage signal $V_{5m}(t)$ and dividing it by the transfer function. This measured voltage $V_{5m}(t)$ example is shown in Fig. 3.25, and to Fourier transform the signal, a steady state range is selected based on observation of vast measurements. These observations led to the interval start of 30 pulses from the estimated arrival time and 5 pulses before the estimated end time. This interval is used for all angles and all distances from the microphone. The Fourier transform of the steady state range gives the voltage signal $V_{5m}(f)$

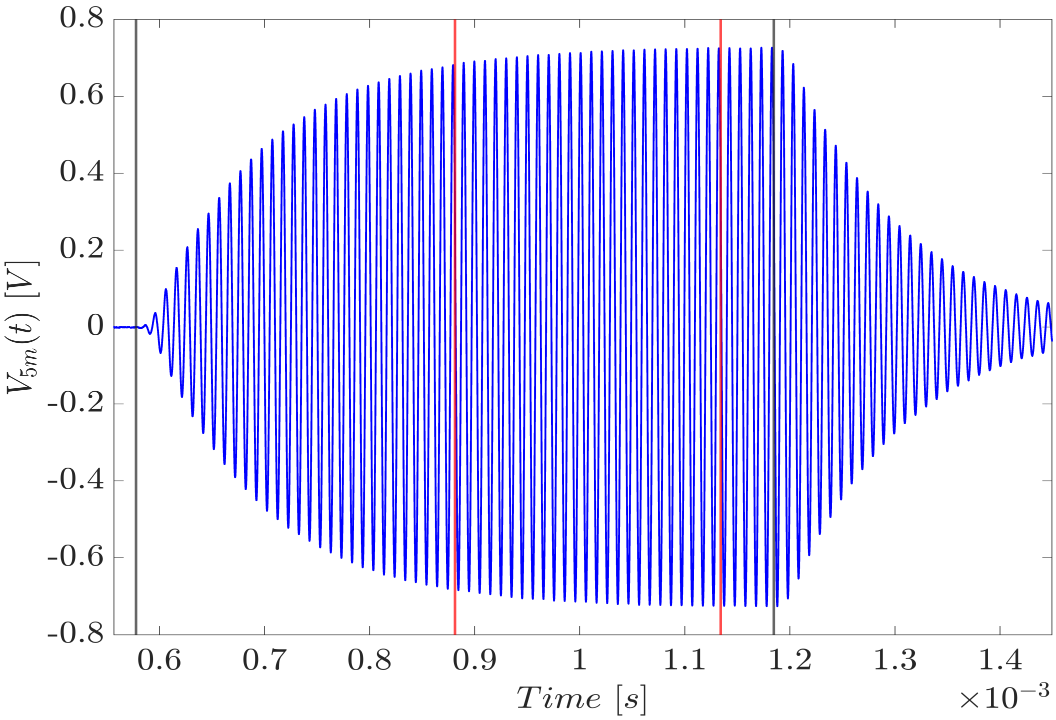



Figure 3.25: The measurement voltage $V_{5m}(t)$ measured at the input channel two on the oscilloscope with signal generator frequency and voltage $V_0(t)$ set to 98860 Hz and 1 Vp-p. The distance between the microphone and the piezoelectric disk Pz27 $z_0$ is 200 mm, and the rotation angle $\theta_R$ is 0 degrees. The first black vertical line represents the estimated arrival time of the signal at the microphone, and the second black vertical line estimates the end of the signal. The two red vertical lines represent the time interval's start and end, which is the interval that is FFT.

### 3.7.3 Signal filtering

The vertical resolution of the measured $V_{0m}(t)$ and $V_{5m}(t)$ depends on the vertical scaling selected and the oscilloscope's bit resolution. With 16-bit resolution, the signal is relatively good, but the waveform data can be somewhat choppy and uneven. To even out the vertical resolution of the measurement signal, a Savitzky-Golay filter [63] is used, which is a built-in function in MatLab's signal processing toolbox. A Savitzky-Golay filter tries to fit a polynomial of a selected degree to the dataset using a frame length to the date input vector, time, t, of a selected size. In this work, a fifth-degree polynomial is used for the measured signals and a frame length of 21. In Fig. 3.26, it is shown the effect of the Savitzky-Golay filter on a measured signal.

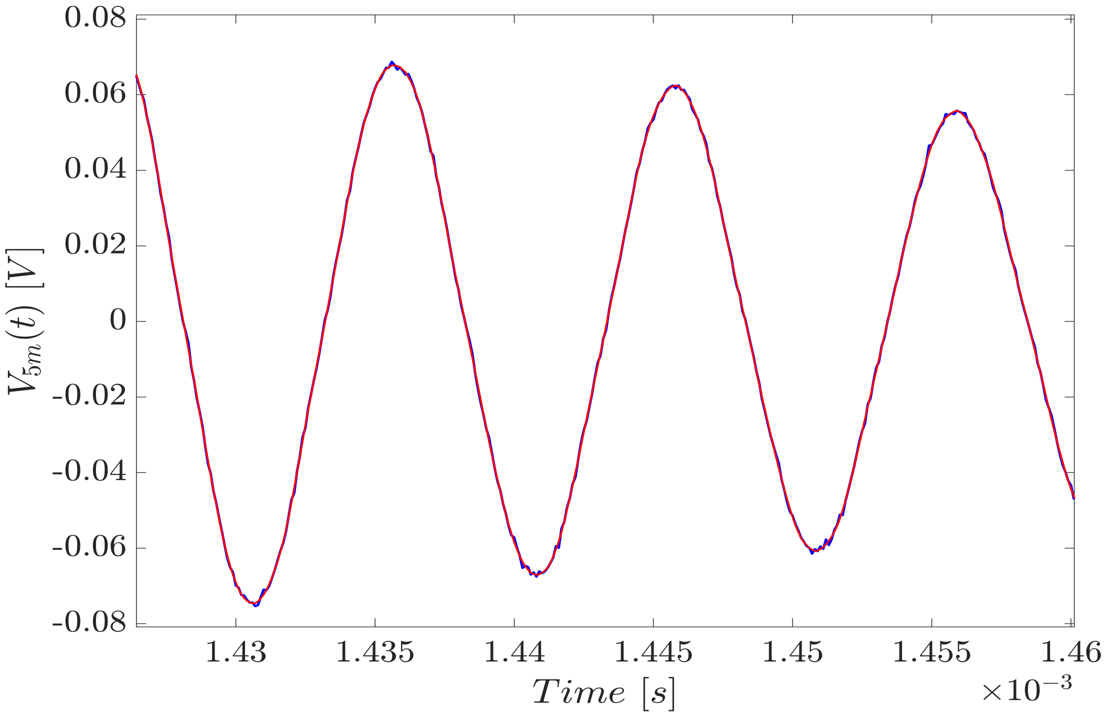



Figure 3.26: An example of measured receiver signal on input channel 2 of the oscilloscope filtered with Savitzky-Golay filter.

### 3.7.4 Method for calculating peak-to-peak voltage by using the fast Fourier transform

In this work, the Fourier transform method is used to convert a measurement signal that is filtered, DC-compensated, and range limited from being time-dependent to frequency-dependent,

$$V(t) \xrightarrow{FFT} V(f)\,. \tag{3.23}$$

The DC component is calculated by taking the mean value of the entire measured signal, which is given as

$$DC = \frac{1}{N}\sum_{i=1}^{N} V_{5m,i_{DC}}\,, \tag{3.24}$$

where N is the number of samples in $V_{5m_{DC}}(t)$, and $V_{5m,i_{DC}}$ is the discrete measurement sample amplitude. Then the DC component is subtracted from the signal as

$$V_{5m}(t) = V_{5m_{DC}}(t) - DC, \tag{3.25}$$

where $V_{5m_{DC}}(t)$ is the signal containing a DC component. The range limited area is further adjusted, such as the start and end of the interval falling in a zero point, such that it possesses an integer number of periods. The measured and range-limited signal transformed from time-dependent to frequency-dependent is done with the built-in MatLab function FFT and by using zero padding five times the signal length. Zero padding increases the Fourier-transformed signal $V(f)$ vector's frequency resolution, leading to increased voltage accuracy for the exact applied transmitting frequency, as seen in Fig. 3.27. To find the peak-to-peak voltage of the FFT signal, the FFT signal $V(f)$ vector must be multiplied by four and divided by the number of bins of the FFT signal. It must be multiplied by four because the $V(f)$ vector is split between negative and positive frequencies, which halves the voltage amplitude, and a voltage amplitude is half the peak-to-peak value. This leads to the equation,

$$V_{5m_{pp}}(f) = \frac{V(f) \cdot 2 \cdot 2}{N_{bins}} = 4V_{5m}(f) \,, \tag{3.26}$$

where $V_{5m}(f)$ is the vector output of the FFT, $N_{bins}$ is the number of samples in $V(f)$, and the final result plotted in Fig. 3.27.

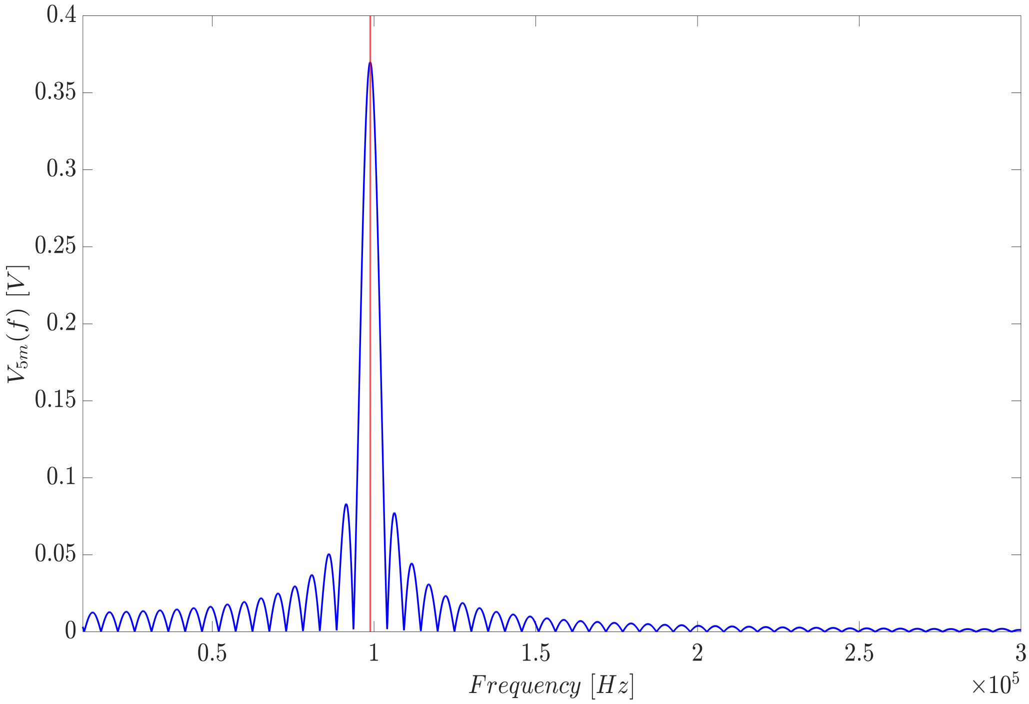



Figure 3.27: FFT of measurement receiver signal in Fig. 3.25 on input channel two of the oscilloscope filtered with Savitzky-Golay filter and with signal generator frequency and voltage $V_0(t)$ set to 98860 Hz and 1 Vp-p. The red vertical line marks the transmitting frequency 98860 Hz. The distance between the microphone and the piezoelectric disk Pz27 $z_0$ is 200 mm, and the rotation angle $\theta_R$ is 0 degrees.

### 3.7.5 Method of calculating pressure

In order to be able to calculate the peak-to-peak free-field pressure $p_{4_{pp}}(f)$, the voltage $V_5(f)$ and the microphone sensitivity $|M_{cal}(f)|$, see Fig 3.15, for the transmitter frequency $f$ must be calculated. Then, the free-field pressure is calculated using measurement voltage and calibrated microphone sensitivity, which is

$$p_{4_{pp}}(f) = \frac{V_{5_{pp}}(f)}{M_{cal}(f)}\ , \tag{3.27}$$

where $p_{4_{pp}}(f)$ is the calculated peak-to-peak free field pressure at the microphone.

### 3.7.6 Signal to noise ratio

This work uses the same method to calculate background noise as in previous works [29] [48] [24]. This method takes a time window with enough noise samples, which is about a 1 ms time window containing about 10 000 samples just before the transmission signal's estimated arrival. With the time window selected, the background noise is calculated by the root-mean-square-voltage as

$$V_{rms}^{noise} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(V_{5m,i} - \hat{V}_{5m}\right)^2}\ , \tag{3.28}$$

where $N$ is the number of samples of the measurement voltage signal interval $V_{5m}(t)$, $V_{5m,i}$ is the voltage value of a single sample, and $\hat{V}_{5m}$ is the mean value of the voltage signal. Furthermore, the acoustic measurement signal $V_{5_{pp}}(f)$ is calculated for the frequency of transmitted signal, which is further used to calculate the root mean squared voltage of the measured signal as

$$V_{5_{rms}} = \frac{V_{5_{pp}}(f)}{2\sqrt{2}}\ , \tag{3.29}$$

where $V_{5_{rms}}$ is the root mean squared voltage of the received acoustic signal. With the root mean squared voltage of the acoustic signal and background noise, the signal to noise [12] can be calculated as

$$SNR = 20log_{10}\left(\frac{V_{5_{rms}}}{V_{rms}^{noise}}\right)\ . \tag{3.30}$$

# Chapter 4

# Positioning setup and measurements with the MatLab app

At the beginning of this work, it was not possible to control the air setup's X, Y, Z, and R-stages through MatLab and has instead been controlled through the motion controller software PIMikroMove. In previous works, this has led to all measurements performed over different positions, such as directivity, on-axis pressure, and 2-D horizontal pressure field, being measured manually, which are highly time-consuming. If the measurement had only been over a few different positions, it would not have been a problem to manually move the position of the stages through PIMikroMove. However, automation is a must due to a single 2-D horizontal pressure field measurement consisting of nearly 10 000 different position measurements, where one measurement manually takes at least 20 seconds. There have previously been able to control some of the stages through MatLab. However, after the years have passed, the company Physik Instrumente no longer supports updates for some of the stages, which has led to communication problems for MatLab with the stages. This issue occurred because MatLab could not read a C ++ script, and this issue is resolved in this work.

This chapter goes through the Matlab app designed to easily control the air setup's X, Y, Z, and R-stages in Sects. 4.1 and 4.2, and find the correct position of the piezoelectric disk relative to the microphone with a setup wizard in Sect. 4.3. The Matlab app is also designed to take individual measurements directly through the app or load a set of measurement parameters and start a measurement series, see Sect. 4.4. Starting a measurement series is the automation part, where the app takes in parameters through **MeasurmentParameters.m** shown in Appendix A.3 and performs the measurements based on the given parameters. The app then takes the measurement results and saves them into a specified folder in an organized manner. The app also has built-in codes for safety to stop the motions of the machine, preventing undesirable crashes and personal injury or material damage. The app's script is included in Appendix B. Sect. 4.5 describes the importance of positioning the piezoelectric disk and what uncertainty the positioning entails.

## 4.1 MatLab app screen

The first showing when the air controller is opened is the app's screen, see Fig. 4.1. This screen contains buttons, labels, LEDs, input fields, and a text window.

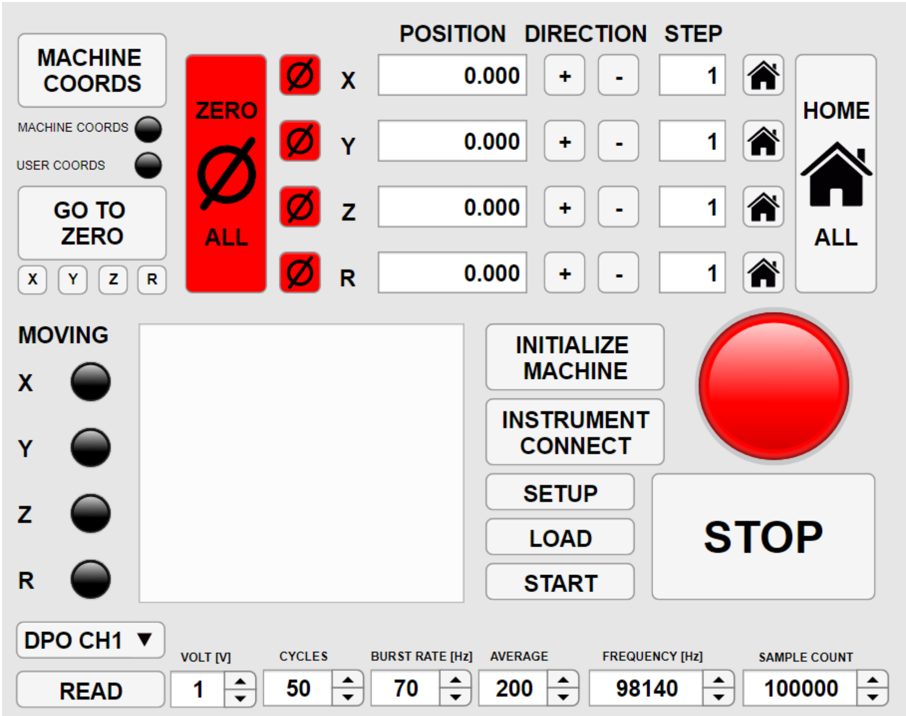



Figure 4.1: The MatLab Air Controller app's screen.

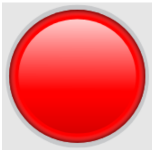
Buttons are an essential element of the app. Pressing one of the buttons executes an action, such as the stop button, see Fig. 4.2a, which stops the motion of the stages and halts all actions until the restart button is pushed, see Fig. 4.2b. All buttons use functions inside the Matlab app.


(a)


(b)

Figure 4.2: (a) Stop button and (b) Restart button. If the stop button is pushed, all motions of the stages stop and halt all actions, and the stop button turns into a restart button. The control over the app is regained when the restart button is pushed.

Labels serve as informative text. They can be constant, like all labels, not on a button. Labels on buttons can either be constant or change depending on the state of a button. An example of such a state change is the stop/restart button in Fig. 4.2. LEDs show a current state or action being active or not by either being on (green) or off (red/black), see Fig. 4.3, and the app automatically updates the LED's state.



(a) Off.

(b) On.

Figure 4.3: Indicates if the stop button is active or not. (a) The stop button is pushed if it is red and prevents stages movements. (b) If the LED is green, the system is up and running.

Input fields are fields with values that can be manually changed, and an example of a changeable input value is a step value, see Fig. 4.4. A step is a value related to travel distance, where one step is the same as one millimeter.


Figure 4.4: One out of the four input fields for step.

The text window is to display actions in the form of text. The text window is the white field in Fig. 4.1 or shown with information as in Fig. 4.5. It helps keep track of all actions, such as informing about the estimated time of completion of a measurement series. A log is kept for all messages displayed in the text window. If anything happens, such as the PC turning off, instruments freezing, or MatLab crashing, the app log can contain the last information displayed in the text window. In case of a failure happens in a middle of a large measurement series, the last known position can be extracted from the log, and the measurement series can quickly be resumed at the last known position.

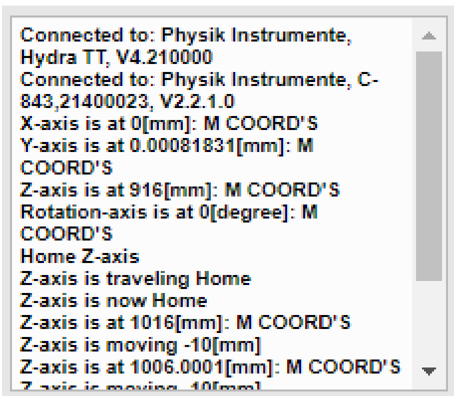



Figure 4.5: Text window with an illustration of live information from the action performed in the app.

## 4.2 Startup of the MatLab app

When starting the air controller app and the screen in Fig. 4.1 is displayed, certain steps must be taken to ready the measurement setup for measurements. The first action is initializing the machine with the button in Fig. 4.6. The initialize machine button is a function (Appendix B.2) that sets up a connection between the Hydra TT motion controller and PC and between the C-843 motion controller and PC, as illustrated in Fig. 3.16. The initialize machine button also assigns the correct parameter values, such as velocity and acceleration, to the stages. Parameters and other values assigned to the stages are within the script in Appendix B.2.

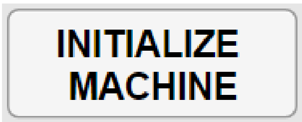


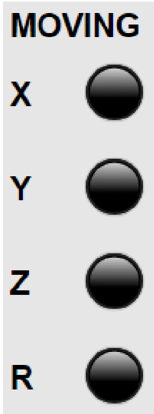

Figure 4.6: The initialize machine button.

After pressing the initialize machine button, all stages need to be "homed" with the home all button, Fig 4.8a, or the individual home button, Fig 4.8b. The home all button is a function (Appendix B.3) that starts a sequence in the following order Z, R, X, and Y-stage to search for their respective reference point. The search for a reference point is when the stage moves as far as possible in a specific direction until induction sensors detect that the stage is at its farthest limit range it can travel. At this limit, the position of the machine coordinates for each stage is defined as the reference point and given in Table 4.1. In the same Table 4.1, the travel limits of the stages are also given.

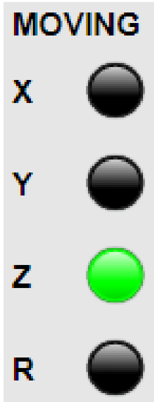
Table 4.1: Individual stages reference point after a homing function is run, and the individual stages travel range given as lower and upper limit.

| | Reference point | Lower limit | Upper limit | Unit |
|---|---|---|---|---|
| X-stage | 0 | 0 | 300 | [mm] |
| Y-stage | 0 | 0 | 300 | [mm] |
| Z-stage | 1016 | 0 | 1016 | [mm] |
| R-stage | 0 | -Inf | Inf | [degrees] |

Whenever the stages search for a reference point or move in general, the LEDs in Fig. 4.7 indicate which stage is moving.




(a) No stages in motion.




(b) Z-stage is moving.

Figure 4.7: Indicates the movement of the stages. If the stage is in motion, the LED is green. No light, no motion.

The individual home button executes the same function as the home all button, but "homes" only the stage associated with the button. The great thing about the reference points is that they are repeatable. This means if the app is closed with Z-stage at the machine coordinates 500 mm, and the app is opened again, the machine coordinates 500 mm of the Z-stage can be found by first using homing function and then traveling from reference points to the machine coordinates 500 mm.

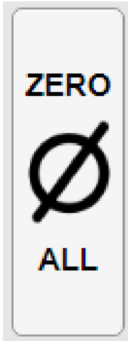

(a) Home all.

(b) Home.

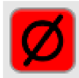
Figure 4.8: The home all button (a) and the individual home button (b).

With all stages "homed", the user coordinates need to be found. User coordinates are an offset value from reference points, where the user coordinates are set to 0 mm at this offset. Setting the coordinate to 0 mm means that if the machine coordinates for Z-stage are at 500 mm, and the zero button for that individual stage is used, Figs. 4.9c/4.9d, the user coordinates will display this coordinate position as 0 mm in the app screen position field (Fig. 4.1). If this stage is moved 10 steps with writing 10 in the input field shown in Fig. 4.4 and moved in the positive direction with the (+) button seen in the app screen in Fig. 4.1, the user coordinates would display 10 mm, and the machine coordinates would display 510 mm for that individual stage.


(a) Not all stages is zeroed.




(b) All stages is zeroed.



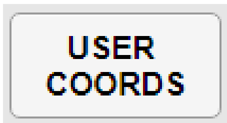
(c) Zero position not found for a individual stage.

(d) Zero position found for a individual stage.

Figure 4.9: The zero all button (a)/(b) sets all current positions of stages to 0 mm in the user coordinates. If it is white (b), it indicates that all stages are zeroed, and (b) indicates that some or all stages are not using offsets from reference points. The zero button (c)/(d) sets the current position of the individual stage to 0 mm in the user coordinate. If it is white (d), it indicates that the stage is zeroed, and (c) indicates that it is not using offsets from reference points.

The button in Figs. 4.10a/4.10b can be used to switch between machine and user coordinates, and the label on the button and the LEDs in Fig. 4.11 will change depending on which coordinate system is active.


(a)




(b)

Figure 4.10: (a) indicates that the machine coordinates are active. If pushed, it turns into (b) and indicates that the user coordinates are active.

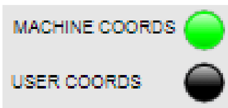



Figure 4.11: Indicates the active coordinate system. In this figure, it is the machine coordinates system that is active.

## 4.3 Positioning of the piezoelectric disk

User coordinates are used to find zero coordinates for X, Y, and Z-stages that are as close as possible to the origin of the X, Y, and Z axis defined in Fig. 2.4 and to avoid the effects that can occur, which is described in Sect. 4.5. The user coordinates for the R-stage $theta_R$ is zeroed when the piezoelectric disk's Z-axis is parallel with Z-stage. To find these coordinates, the setup wizard is used by pushing the setup button in Fig. 4.12.

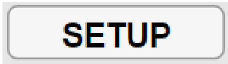



Figure 4.12: The setup button

This setup wizard provides guidance, and a choice of five setups, as seen in Fig. 4.13, where: the setup one is used to find $\theta_T$, see Figs. 4.19 and 4.32; the setup two is used to find $\sigma$, see Figs. 4.23 and 4.37; the setup three is used to set $\theta_R$ equals 0 degrees; the setup four is used to find the distance $z_0$; the setup five is used to find $\theta_T$ and $z_0$ and calculate constants of a slope such that the position of the microphone is adjusted respect to $\theta_T$ and the distance $z_0$ such that angle $\theta_T$ equals $\theta_M$, see Fig. 4.33, for all distances between the microphone and piezoelectric disk. However, this work never used setup five and linear compensation, and compensation was always turned off.

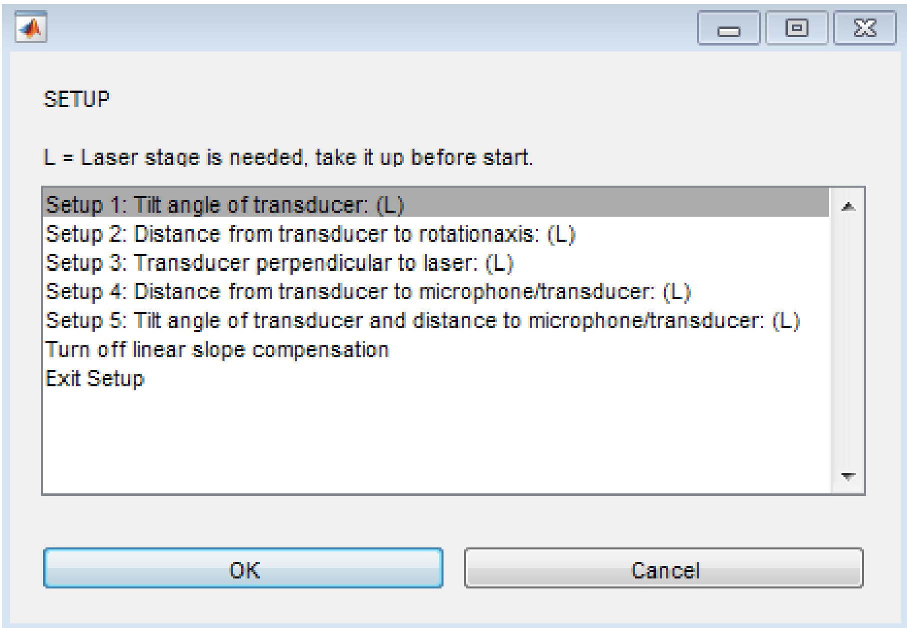



Figure 4.13: Setup wizard that makes the user able to choose different setups.

In Fig. 4.13, it says L equals laser stage (see Fig. 3.6 (D)), which is the laser stage set up by [7], and the laser sensors are of the type LK-G32 from Keyence used with a controller LK-G3001PV and listed in Table 3.2. For all the setup choices, the laser stage needs to be elevated all the way up before starting to measure. The distance $d_x$ denoted in Fig. 4.14 is the distance between the two fronts of the lasers and calibrated by [7] and given in Table 4.2. In Table 4.2, there are also listed key features of the lasers. The distance denoted $d_{ref}$ equals 30 mm and is the reference distance. From $d_{ref}$, the measuring range of the laser is ±5 mm.

When measuring with lasers 1 and 2, they measure a distance $d_1$ and $d_2$, which are negative values for distances from the laser front to the object that are greater than $d_{ref}$ or positive if the distance is less than $d_{ref}$.

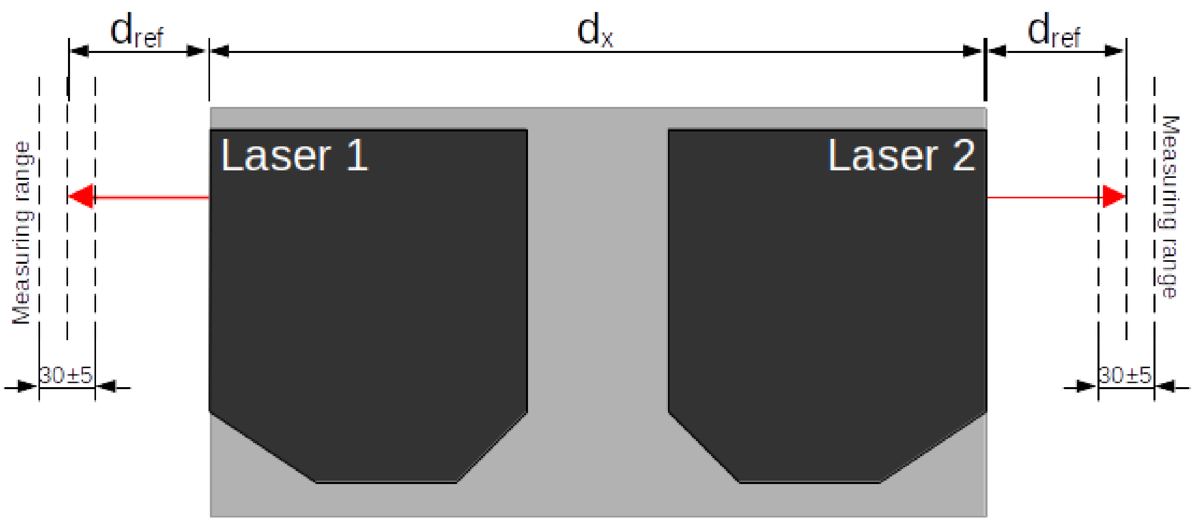



Figure 4.14: Schematic of the two lasers LK-G32 and illustrating the measuring rages. The distance $d_{ref}$ is the reference distance of 30 mm, and $d_x$ is the distance of 182.5692 mm between the two laser fronts. [35]

Table 4.2: Different key features of the two lasers LK-G32, given in [35], and the distance $d_x$ is the calibrated distance by [7].

| LK-G32 laser sensor | | Unit |
|---|---|---|
| $d_x$ | 182.5692 | [mm] |
| $d_{ref}$ | 30 | [mm] |
| Range | ±5 | [mm] |
| Spot diameter | Approx. ø30 | [$\mu$m] |
| Linearity | ±0.05% (of full scale = ±5 mm) | [-] |
| Repeatability | 0.05 | [$\mu$m] |
| Light source | (viable light) 655 | [nm] |

### 4.3.1 Setup 1

For all setup choices in Fig. 4.13, the Z-stage moves in front of the laser after clicking the measure button in Fig 4.15.

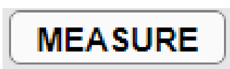



Figure 4.15: The measure button.

When the Z-stage has arrived in front of the laser and the piezoelectric disk is within the laser's measuring range, as illustrated in Fig. 4.16, the laser's software opens.

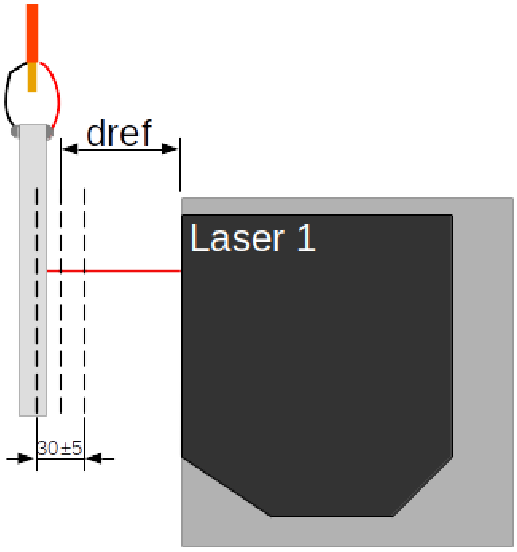



Figure 4.16: Side view of the piezoelectric disk being within the laser 1 measuring range.

The laser software that came with the purchase of the lasers is called LK-navigators. This LK-navigators screen is seen in Fig. 4.17.

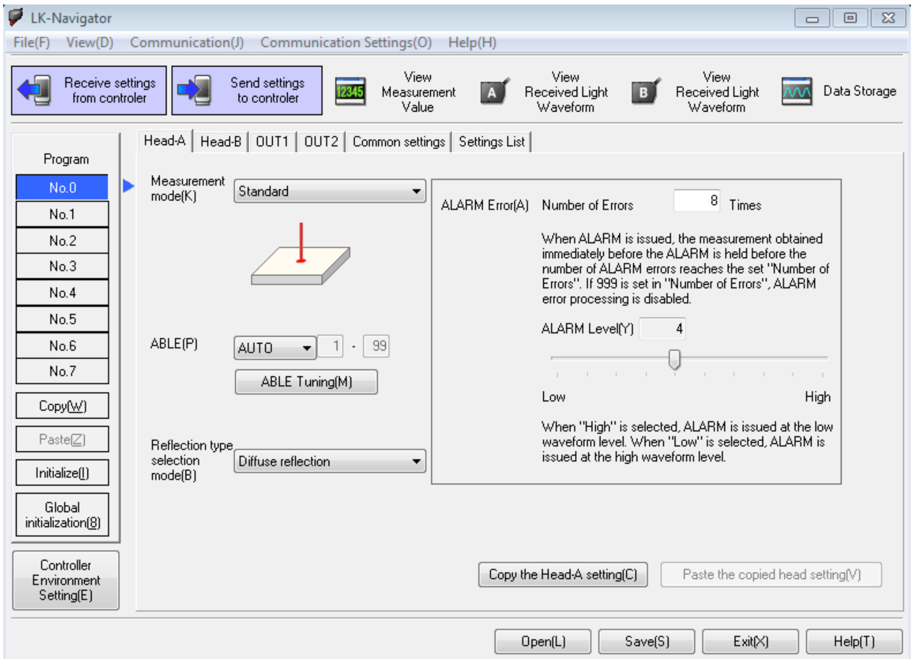



Figure 4.17: LK-navigator screen from Keyence.

Adjustment screws on the laser stage are used to move the laser point to the highest point of the piezoelectric disk front surface. The disk must be slightly rotated, clock, or counterclockwise, so the solder lump is not at the highest point of the disk's front surface. By clicking the view measurement value from the LK-navigator screen and clicking the measurement value acquisition start, see Fig. 4.18, it starts averaging the distance $d_{11}$ value from the reference distance $d_{ref}$. The measured value is then written into an open GUI, and measurements are repeated at the bottom of the disk's front surface, measuring $d_{12}$, see Fig 4.19

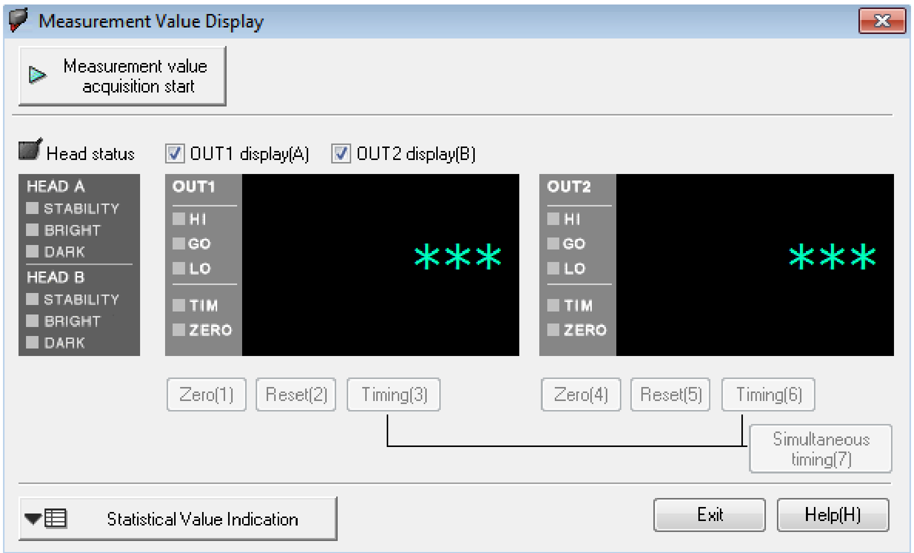



Figure 4.18: LK-navigator measurement value display.

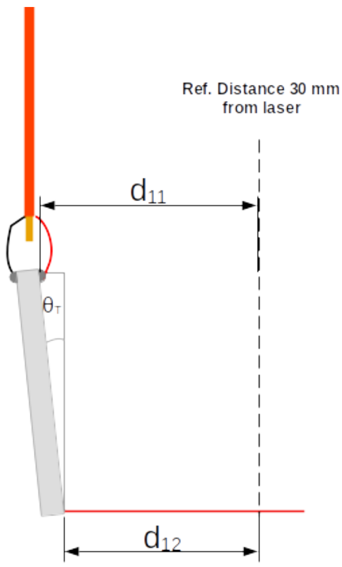



Figure 4.19: Side view of the tilt angle $\theta_T$ of the piezoelectric disk relative to the laser one beam direction and distances $d_{11}$ (highest point) and $d_{14}$ (lowest point).

After measuring the top and bottom distances and inserting the diameter of the piezoelectric disk in the open GUI (Graphical User Interface), the app calculates the tilting angle $\theta_T$ with the equation

$$\theta_T = sin^{-1}\left(\frac{d_{12} - d_{11}}{d}\right), \tag{4.1}$$

where d is the diameter of the disk. With known angle $\theta_T$, and if the angle of the piezoelectric disk is more significant than 0.5 degrees, the tilt is adjusted, and the setup one is repeated. This tilt is manually adjusted, and the goal is to get the angle as small as possible. This adjustment reduces the effect of $\theta_T$ modeled in Sect. 4.5 is not significant in the measuring frequency range up to 300 kHz.

### 4.3.2 Setup 2

Before starting this setup, it is essential to get the middle of the vertical rod and horizontal rod, which is attached to the R-stage, and the pointy bolt on the R-stage, which is aligned with the R-stage rotation axis, to construct a vertical plane in between them. The reason for creating this plane is to create a parallel plane with the YZ-stage plane. This opens up to move the front center of the piezoelectric disk with the 3-D-printed part that now moves in the XZ-stage plane, see Fig. 4.20. Then the Y-axis, which passes through the front center of the disk, is defined in Sect. 4.5 can be moved as close as possible to the R-stage axis of rotation after finding $\sigma$.

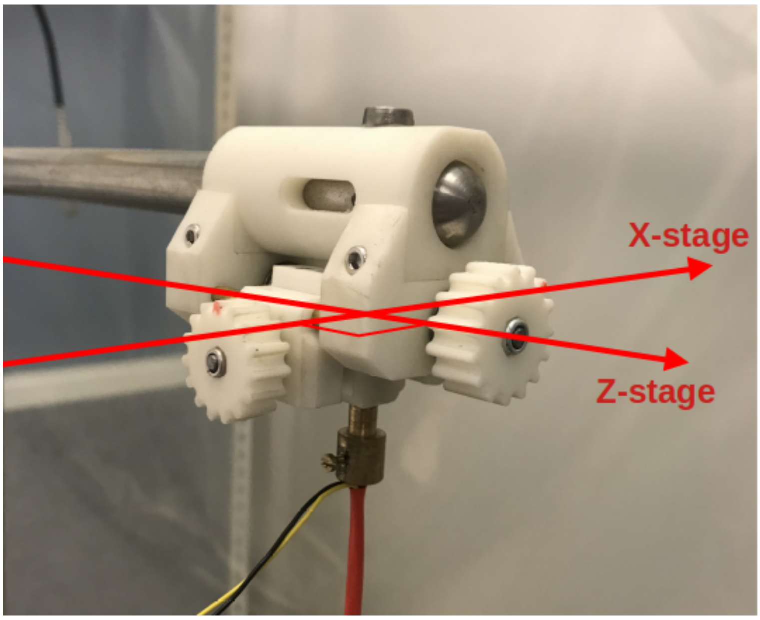
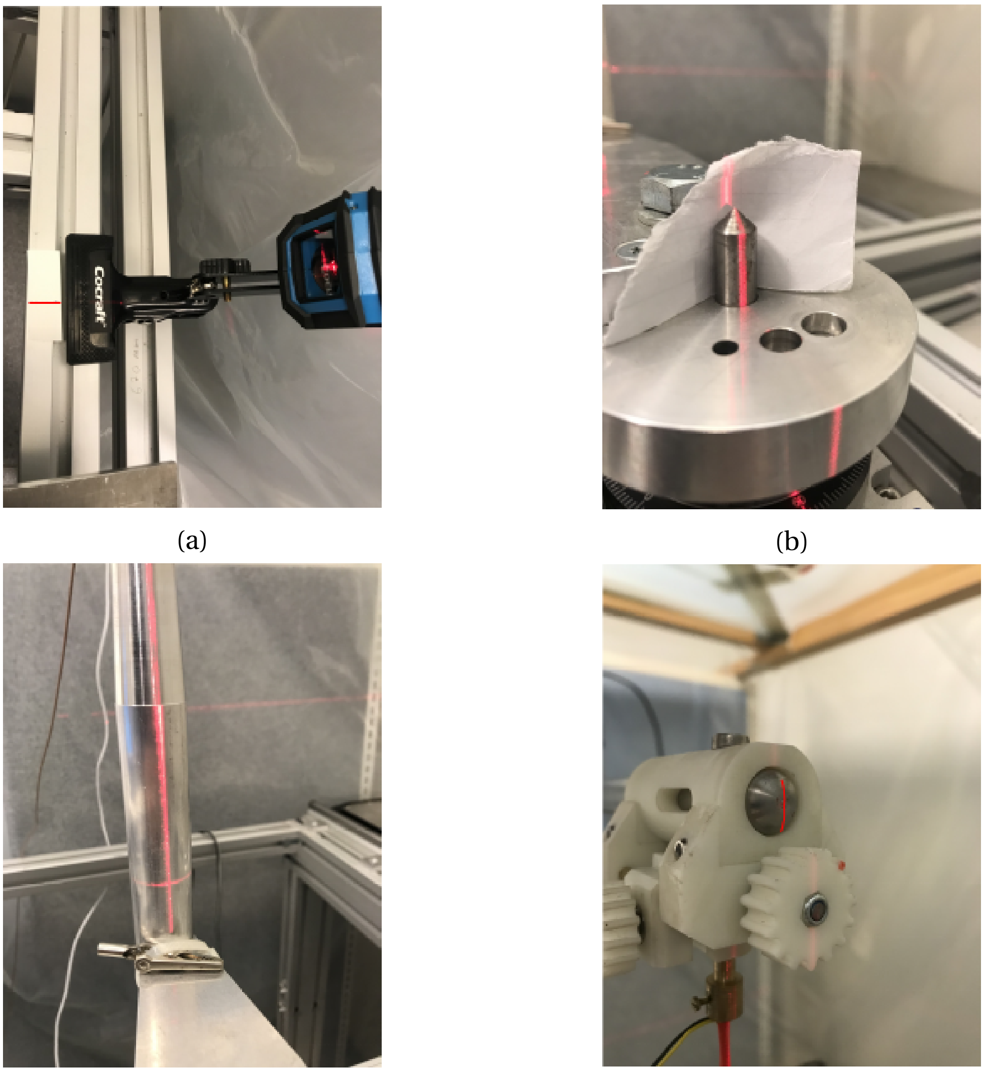



Figure 4.20: Movement of the piezoelectric disk front center in the XZ-stage plane with 3-D printed part adjustment screws.

A vertical laser is used to find this vertical plane passing through the rods and the bolt, see Fig. 4.21. First, in Fig. 4.21a, the vertical laser overlaps the black line on a white tape in the back of the measuring cage that has the same distance to the Z-stage as the pointy bolt. Secondly, in Fig. 4.21b, a piece of paper behind the pointy bolt is used to see the casted shadow of the bolt tip being in the center of the laser line. And lastly, in Figs. 4.21c and 4.21d, The laser line is in the middle of the vertical and horizontal rod.




(a) (b)

(c) (d)

Figure 4.21: Illustrating the vertical laser creating the vertical plane parallel with the YZ-stage plane by laser (a) overlapping the black line on a white tape in the back of the measuring cage, (b) casted shadow of the bolt tip being in the center of the laser line, (c) and (d) the laser line is in the middle of the vertical and horizontal rod.

When selecting setup two, the procedure is the same as the setup one. Instead of measuring the piezoelectric disk front surface's highest point $d_{11}$ and lowest point $d_{12}$ distances, it is measured on the front surface farthest right side of the disk $d_{13}$ and left side $d_{14}$, see Fig. 4.22. The measured values and diameter $d$ is written into an opened GUI. The app calculates the $\theta_r$ as

$$\theta_r = sin^{-1}\left(\frac{d_{13} - d_{14}}{d}\right). \tag{4.2}$$

Furthermore, $\theta_r$ is used in the subsequent calculations to determine the distance $\sigma$ to the R-stage rotation axis.

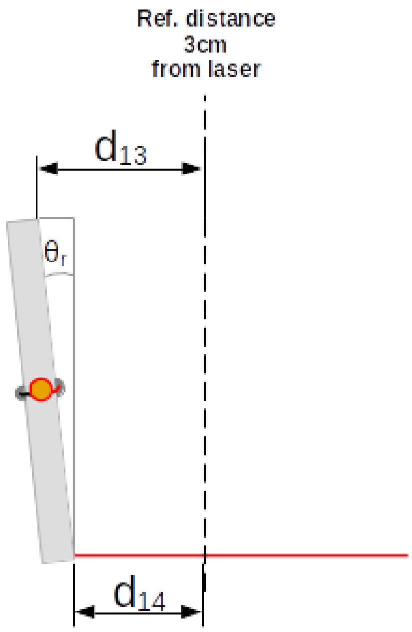



Figure 4.22: Top view of the rotation angle $\theta_r$ of the piezoelectric disk relative to the laser one beam direction and distances $d_{13}$ (farthest right point) and $d_{14}$ (farthest left point).

Then following three distances in Fig 4.23 are measured. First distance, $d_{15}$ on the piezoelectric disk front center surface, and the result is written into the open GUI. The R-stage then rotates with an angle $\theta_R$ equals $\gamma$, then the second distance $d_{16}$ is measured without adjusting the laser's position. It is worth noting that gamma is a negative angle because R-stage rotates in a clockwise direction which is a negative direction. Then the last distance, $d_{17}$, is measured after adjusting the laser stage position by moving the laser point back to the disk's center, and the results are written into the open GUI. Fig 4.23 illustrates these three measurements.

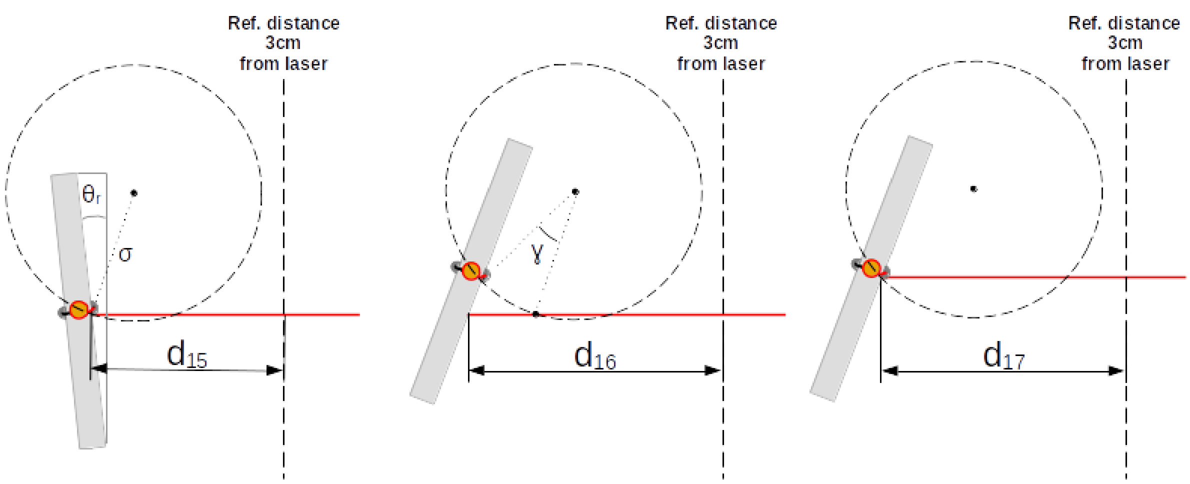



Figure 4.23: Illustration of the three measured distances $d_{15}$, $d_{16}$, and $d_{17}$, where $\sigma$ is the distance between the center of the piezoelectric disk and the horizontal distance to the R-stage rotation axis and $\gamma$ is the $\theta_R$ rotation angle.

With these three distances measured, $d_{15}$, $d_{16}$, and $d_{17}$ the app can calculate the distances $X_{mov}$ and $Z_{mov}$, which are the distances to move the piezoelectric disk front center with adjustment screws of the 3-D printed part seen in Fig. 4.20. This movement of the front surface will reduce the distance $\sigma$ and its effect analyzed in Sect. 4.5. By illustrating this trigonometric problem first, as in Fig. 4.24, and calculate the distance $a$ as

$$a = \frac{d_{16} - d_{17}}{tan(\theta_r + \gamma)} \tag{4.3}$$

where $d_{17} - d_{16}$ equals the opposite distance of the triangle with angle $\theta_r + \gamma$ in Fig 4.24, $\theta_r$ is the calculated value from Eq. 4.2 and $\gamma$ is the rotated angle by R-stage.

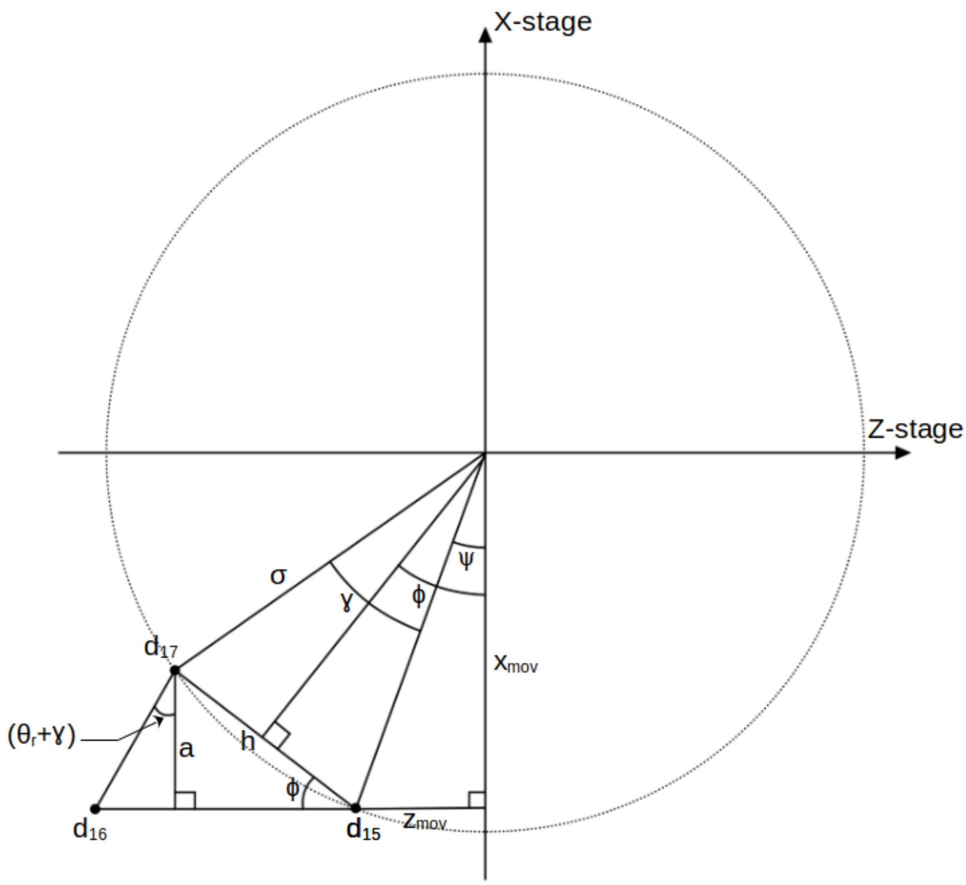



Figure 4.24: Trigonometric problem used in calculating the movement distances $X_{mov}$ and $Z_{mov}$, which is used to reduce the distance $\sigma$. The distances $d_{15}$, $d_{16}$, and $d_{17}$ are the measured distances and are marked as three points. The angle $\theta_r$ is a calculated angle from measurements, and $\gamma$ is a (negative) rotation angle of the R-stage. The angle $\phi$ and $\psi$ are calculated angles.

Then the angle $\phi$ in Fig. 4.24 is calculated by using the calculated distance $a$ and the distance $d_{15}$ and $d_{17}$ as

$$\phi = tan^{-1}\left(\frac{a}{d_{15} - d_{17}}\right), \tag{4.4}$$

where $d_{15} - d_{17}$ is the adjacent distance of the smaller triangle with angle $\phi$ in Fig. 4.24. Furthermore, the hypotenuse distance $h$ for this same triangle is calculated as

$$h = \frac{d_{15} - d_{17}}{cos(\phi)} \ . \tag{4.5}$$

Now, the angle $\psi$ and radius $\sigma$ in Fig. 4.24 can be calculated by using the distances $a$, and $h$, and the angle, $\phi$, which gives

$$\psi = \phi - \frac{|\gamma|}{2} \ , \tag{4.6}$$

and

$$\sigma = \frac{h/2}{sin(\gamma/2)} \ . \tag{4.7}$$

Then it is easy to calculate the distances $X_{mov}$ and $Z_{mov}$ by using the calculated radius $\sigma$ and the angle $\psi$, which gives

$$X_{mov} = cos(\psi)\sigma \ , \tag{4.8}$$

and

$$Z_{mov} = sin(\psi)\sigma \ . \tag{4.9}$$

After finding the $X_{mov}$ and $Z_{mov}$ distances with Eqs. 4.8 and 4.9, the app calculates the rotation of the 3-D printed part in Fig 4.20 adjustment screws to move the front center of the piezoelectric disk to the R-stage rotation axis and reduce the effects of $\sigma$. By using the model in Sect. 4.5, and analyzing the results for different values of $\sigma$ and the angle $\alpha$, it is found that for $\sigma$ equal to 2 mm, the effects are insignificant in the measuring frequency range up to 300 kHz.

### 4.3.3 Setup 3

In this setup, the same method is used for finding $\theta_r$ as in setup two. The difference is that $\theta_r$ is now used to straighten up the R-stage of the measured angle, so the piezoelectric disk surface is now perpendicular relative to the laser one's beam. When the piezoelectric disk is perpendicular to the laser beam, the R-stage is zeroed such that $\theta_R$ is equal to 0 degrees in the user coordinates. This angle for $\theta_R$ is now saved into a zero.mat file such that the angle is stored safely in case of computer failure. In case of failure, these coordinates are loaded into the app with the load button in Fig. 4.25.


Figure 4.25: The load button.

### 4.3.4 Setup 4

The last setup used is setup four, which is used to find the distance between the piezoelectric disk front surface and the microphone. This measurement is illustrated in Fig. 4.26. Before selecting this setup, the laser position is adjusted such that the laser point is in the center of the piezoelectric disk. After selecting setup four, the app asks to move the X and Y-stage to position the microphone's center in the laser.

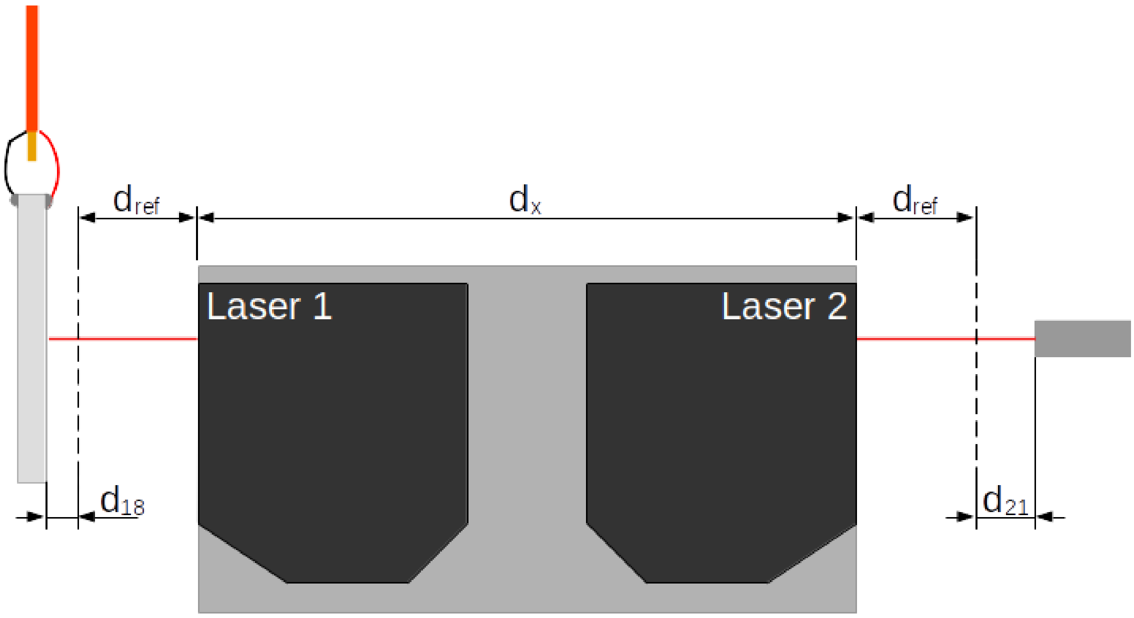



Figure 4.26: Illustration of laser stage measuring the distance between the piezoelectric disk (left side) and the microphone (right side).

With the laser in the center of the disk and microphone, the X and Y-stage are zeroed, setting the user coordinates equal to 0 mm. Then by clicking the measure button in Fig. 4.15, the LK-navigator software opens and is used as described earlier, and then the distances $d_{18}$ and $d_{21}$ in Fig. 4.26 are measured, and the results are written into the open GUI. The app calculates the total distance between the transducer and the microphone as

$$d_{tot} = (d_{ref} - d_{18}) + d_x + (d_{ref} - d_{21}) \tag{4.10}$$

where $d_{ref}$ and $d_x$ distances are given in Table 4.2. The app then uses $d_{tot}$ to take the current machine coordinate of the Z-stage and subtract that position whit $d_{tot}$, and that result is set as Z-stage user coordinates to equal 0 mm. When all the setup wizards are completed, the exit setup in Fig. 4.13 is chosen, the Z-stage backs away from the laser stage, and the laser stage is manually lowered.

## 4.4 Single or series of measurements of electrical and acoustical signals

When the instrument connect button is pushed, see Fig. 4.27; all instruments in the experimental setup, see Fig. 3.3, are connected to the app.

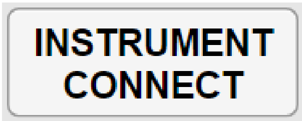



Figure 4.27: Instrument connect button.

When calibrating the microphone, it is used the DPO CH2 read-only in the drop-down list, accessed by the down arrow in Fig. 4.28, which only reads the oscilloscopes screen. When identifying and measuring the reflection, a single measurement is used by using DPO CH2 accessed by the down arrow in Fig. 4.28, and the settings are set in Fig. 4.29. When measuring the reflection, the setting used is 1 V which is the signal generator voltage $V_{0_{pp}}$, a sinus burst of 30 cycles, and a 25 Hz burst rate. It is also used in the reflection measurement frequency of 98860 Hz and a sample count of 10 000 samples, and the signal is averaged 128 times.

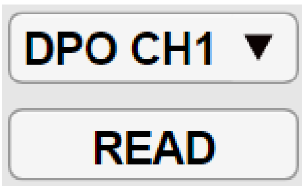



Figure 4.28: Measurement read button.

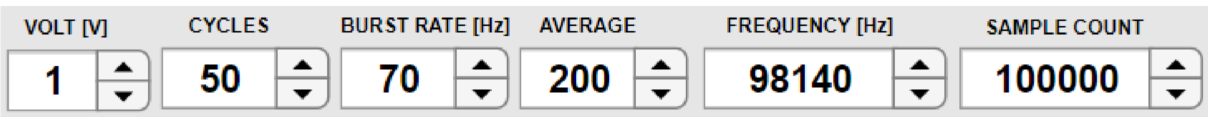



Figure 4.29: Single measurement adjustable parameters.

When it is performed, large measurements, such as directivity, on-axis pressure, and 2-D sound pressure field, the **MeasurmentParameters.m** script in Appendix A.3 is used. This script is used to preset the type of measurement being conducted, the measurement frequency, the measuring range in angle and distance direction, the peak-to-peak voltage of the signal generator, the signal averaging, the number of cycles of the measurement signal, the burst rate, the filter setting, the number of sample count, and which channel that measures the electrical or acoustical signal at the oscilloscope. When the correct parameters are set in the script for the type of measurement being conducted, the parameters are then loaded into the app with the load button in Fig. 4.25, and the start button in Fig. 4.30 is pushed.


Figure 4.30: Start button.

This starts the measurement series by the app automatically driving the stages to the first measuring position. When the stages arrive at the first measuring position, the app sends all settings to the different instruments, and the electrical voltage $V_{0m}(t)$ is first measured. Then the time window of the oscilloscope is adjusted automatically to fit the acoustical signal $V_{5m}(t)$, the acoustical voltage is measured, and environmental parameters are measured. After the first measurement is complete, the results are automatically saved into a specific folder, and the measurement series continues to its next position and repeats the measurements cycle until it is finished with the measurement series.

## 4.5 Transmitter and receiver mounting and positioning sensitivity analysis

It is vital to go through the MatLab Air Controller app's setup, which is designed to adjust and find the correct position of the piezoelectric disk relative to the microphone. The setup wizard helps to set the user coordinates to be as near as possible to the origin of the X, Y, and Z axis given in Fig. 4.31. Sects. 4.3.1-4.3.4 goes through the actual setup wizard in more detail and explains the user coordinates. The coordinates of the X, Y, and Z axis origin are defined in Figs. 2.4 and 4.31 as the piezoelectric disk front center, where Y-axis is parallel to the Y-stage and overlaps the R-stage rotation axis. The XZ-axis-plane is the horizontal plane. Further, the XY-axis-plane is vertical and matches the piezoelectric disk surface plane under perfect conditions of the orientation of the piezoelectric disk.

However, the world is rarely under perfect conditions, leading to different orientations that can affect the measured pressure. These influences lead to changes in the directivity, on-axis pressure, and 2-D sound pressure measurements from their true values relative to perfect orientations. In this work, three different orientations are analyzed that can affect the measured pressure and modeled how it affects the FE simulation directivity. These modeled deviations can be analyzed and indicate how large a deviation can be allowed from perfect orientation without affecting the measurement results.

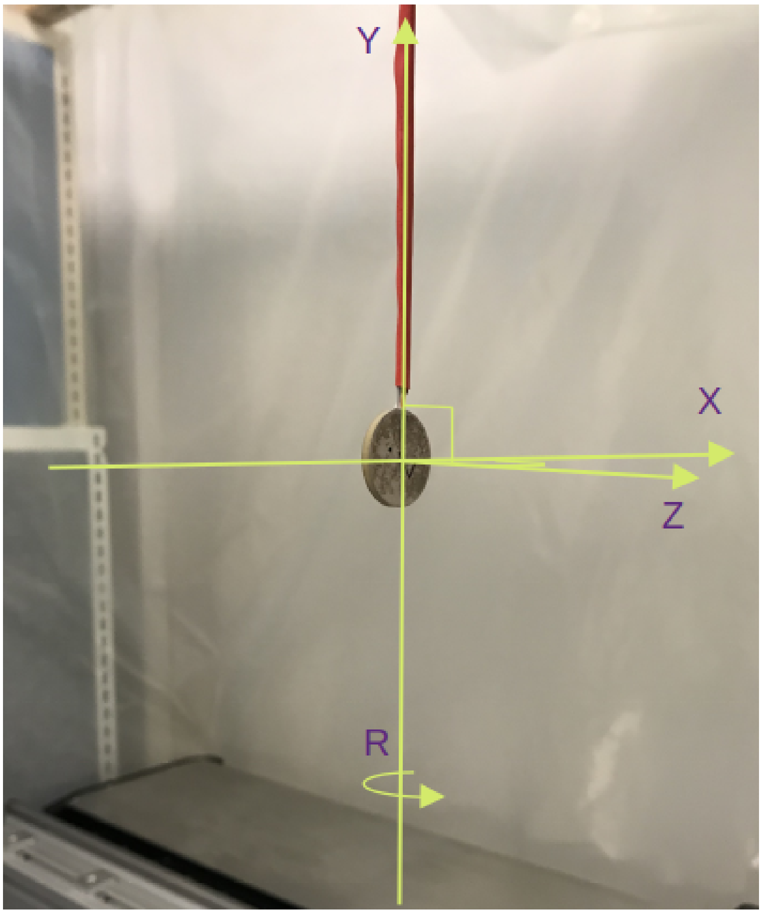



Figure 4.31: X, Y, and Z coordinate with origin in the piezoelectric disk's front center, where the XZ-axis-plane is the horizontal plane, and the XY-axis-plane is the vertical plane overlapping the piezoelectric disk's front surface. Y-axis is parallel with Y-stage and lies on the R-stage rotation axis.

The three analyzed orientations that can affect the measurements come from tilt angles or offset distances. The first orientation is the piezoelectric disk being tilted at an angle $\theta_T$ forward or backwards, see Fig. 4.32. The second orientation is an offset value $\beta$ from the XZ-axis-plane to the microphone's front center resulting in an angle $\theta_M$ relative to the Z-axis, see Fig. 4.33. The difference between the first and second orientations is that the tilt angle $\theta_T$ of the piezoelectric disk is constant for all distances $z_0$ between the piezoelectric disk and microphone centers, and the angle $\theta_M$ depends on $\beta$ and the distance $z_0$ between the piezoelectric disk and microphone centers. Since $\theta_M$ depends on $\beta$ and the distance $z_0$ between the microphone and the piezoelectric disk centers, $\theta_M$ can have a significant impact in the near field for small $\beta$. However, into the far field, the impact of $\theta_M$ would decrease rapidly. The third and last orientation is a horizontal offset $\sigma$ from the piezoelectric disk's front center or seen as an offset from the X, Y, and Z-axis origo to the R-stage rotation axis, as illustrated later in Fig. 4.37. This orientation leads to the center of the microphone's front no longer orbits with a constant distance of $z_0$ around the piezoelectric disk front center. This orbit changes the distance between the microphone and the piezoelectric disk depending on the

R-stage rotation angle $\theta_R$.

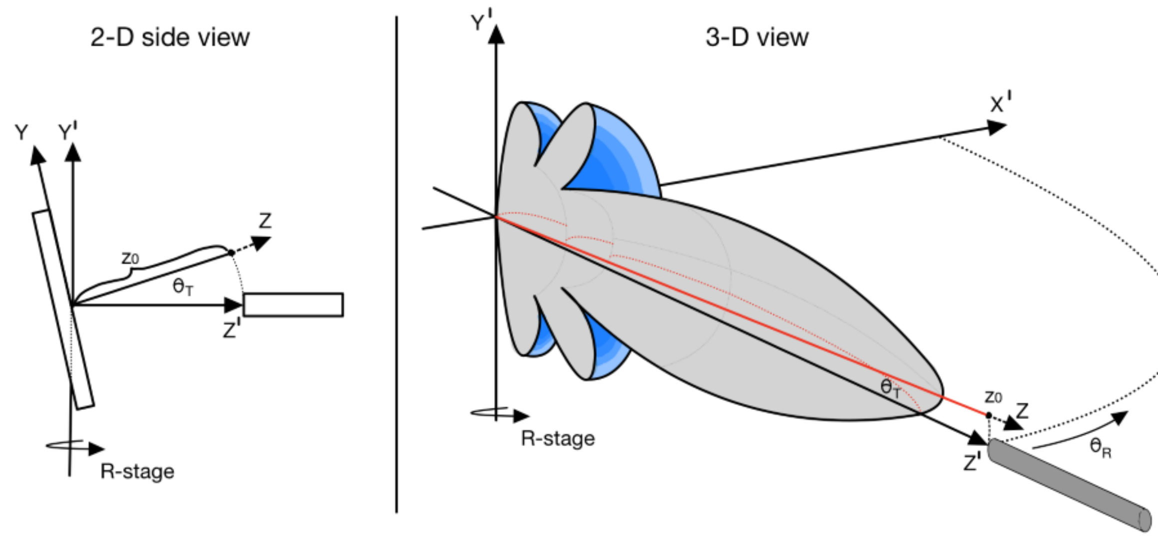



Figure 4.32: Left illustrates the 2-D view of the YZ-plane where the XY-plane crosses the piezoelectric disk's front center. The disk is tilted with an angle $\theta_T$ and distanced a distance of $z_0$ from origo to the microphone's front center. The right illustrates the case in a 3-D view.

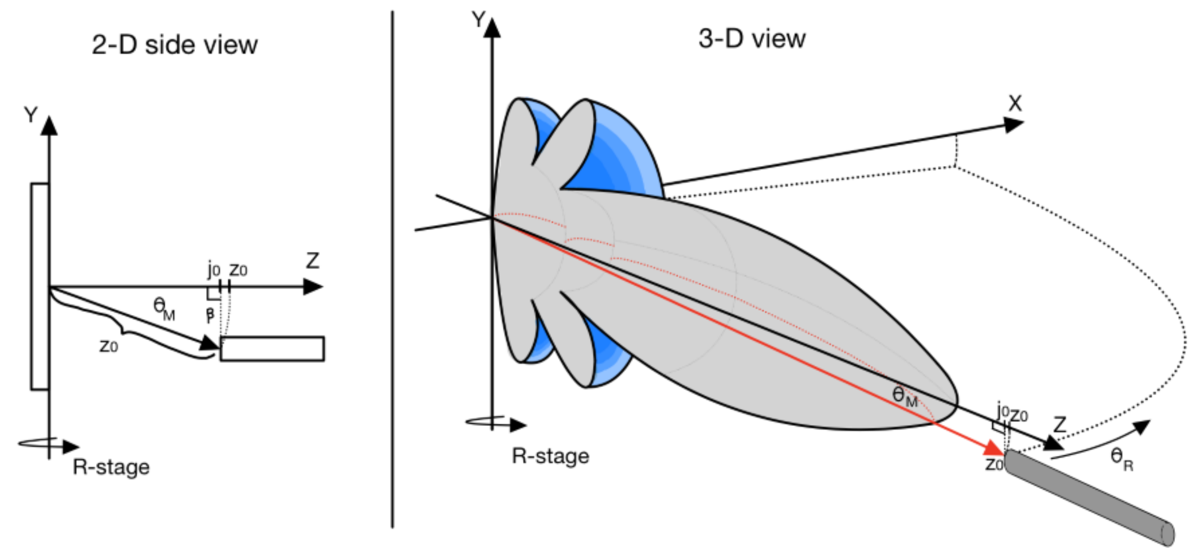



Figure 4.33: Left illustrates the 2-D view of the YZ-plane where the XY-plane lies on the piezoelectric disk's surface. The microphone's front center is offset a distance $\beta$ from the XZ-plane and distanced at a distance of $z_0$ from origo. The right illustrates the case in a 3-D view.

The first and second orientations can be described as two points A and B on a sphere with a radius of $z_0$ and an angular distance $\theta$ between the points, see Fig. 4.34. The distance from the X, Y, and Z-axis origin to the directivity $D(\theta = 0, z_0, f)$ is described as the OA vector in Fig. 4.34, where $\theta$ equal to zero is the on-axis perpendicular to the piezoelectric disk surface, $z_0$ as the simulated distance and $f$ as simulated frequency. The distance from the X, Y, and

Z-axis origin to the front center of the microphone is described as OB in Fig. 4.34. The $\theta_T$ is the tilt of the piezoelectric disk, $\theta_M$ is the angle to the center of the microphone's front dependent on $\beta$ from X'Z'-axis-plane in Fig. 4.34 and $z_0$, $\theta$ is the angle between points A and B on the sphere, and $\theta_R$ is the R-stage rotation in Fig. 4.34.

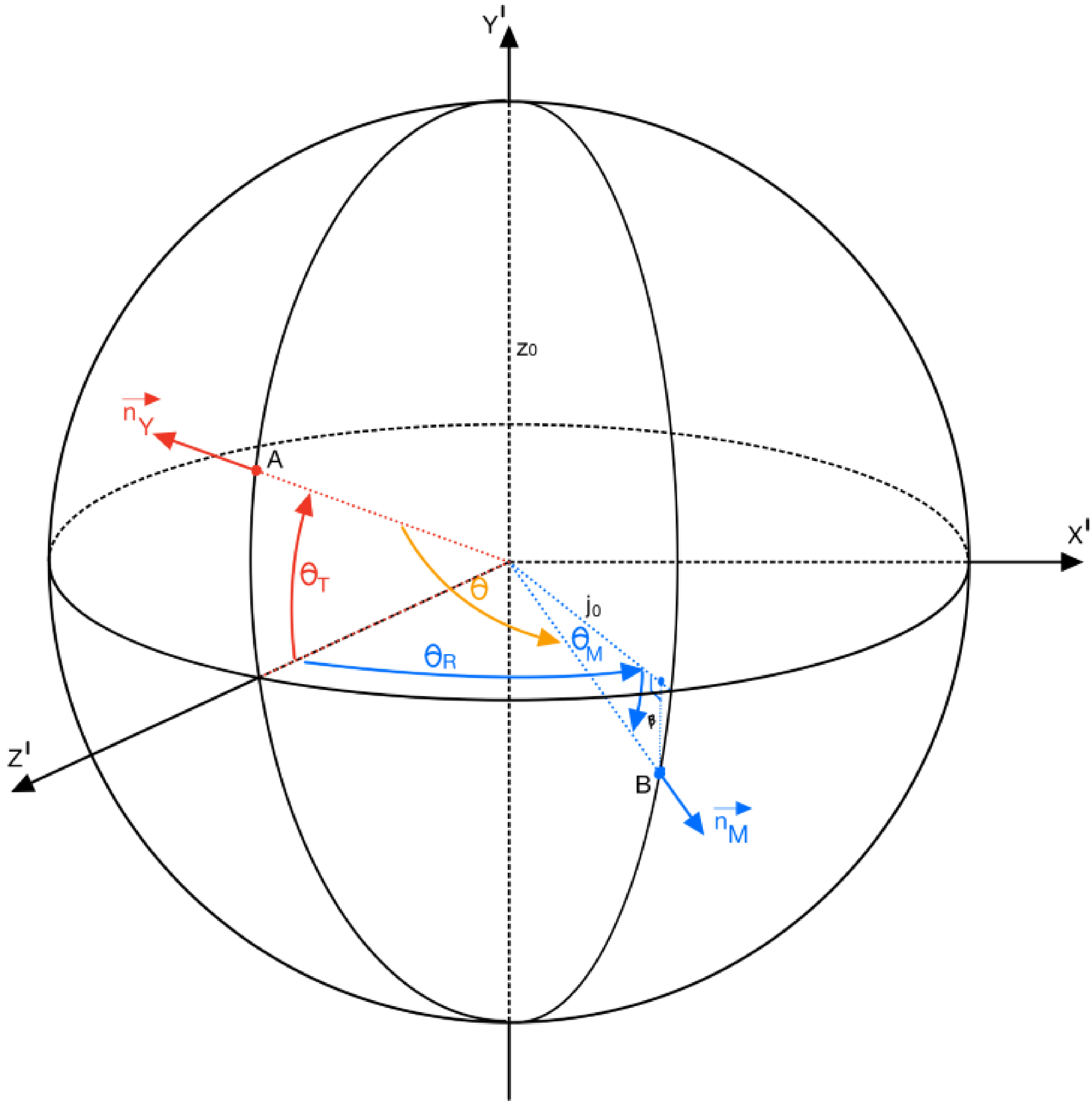



Figure 4.34: Illustration of the first (OA) and second (OB) orientations and the different values of $\theta_T$ disk tilt, $\beta$ microphone offset, $z_0$ sphere radius, $\overrightarrow{n_Y}$ normal unit vector of point A, $\overrightarrow{n_M}$ normal unit vector of point B, $\theta$ angle distance between A and B, and $\theta_R$ R-stage rotation [76].

The angle $\theta_R$ ranges from 0 to $2\pi$, and $\theta_T$ and $\theta_M$ range from $-\pi/2$ to $\pi/2$. To derive the angle distance $\theta$, one first looks at the dot product of the vectors OA and OB as

$$\overrightarrow{OA} \cdot \overrightarrow{OB} = z_0^2 cos\theta \ , \tag{4.11}$$

which is equivalent to the dot product between the normal unit vectors of OA and OB, which equals to

$$\overrightarrow{n_Y} \cdot \overrightarrow{n_M} = cos\theta \ . \tag{4.12}$$

In the X', Y', and Z'-axis frame in Fig. 4.34, the normal unit vectors can be written as

$$\overrightarrow{n_Y} = \begin{pmatrix} 0 \\ sin(\theta_T) \\ cos(\theta_T) \end{pmatrix} , \tag{4.13}$$

and

$$\overrightarrow{n_M} = \begin{pmatrix} cos(\theta_M)sin(\theta_R) \\ sin(\theta_M) \\ cos(\theta_M)cos(\theta_R) \end{pmatrix} , \tag{4.14}$$

and solving Eq. 4.12 for $\theta$ by inserting the normal unit vectors, the angle is written as

$$\theta = cos^{-1}\left(sin(\theta_M)sin(\theta_T) + cos(\theta_M)cos(\theta_T)cos(\theta_R)\right) \ . \tag{4.15}$$

With an equation for $\theta$, it can now be analyzed the effect of $\theta_T$ and $\theta_M$ of the R-stage rotation range $\theta_R$ from -90 to 90 degrees and plot the logarithmic result of D($\theta$,$z_0$,$f$) against the measured angle $\theta_R$. Plotting against $\theta_R$ is due to it being the perceived angle, while $\theta$ describes the actual angle between the microphone's front center and D($\theta = 0$,$z_0$,$f$), see Figs. 4.35 and 4.36 for the results of the modeling. The script used to model this is included in Appendix A.2.

By analyzing Fig. 4.35, it can be seen that simulated directivity with the frequency 98100 Hz is affected by the tilt of two degrees. The modeled tilts affect the sidelobes by increasing dB strength, where the first side lobes increase by approximately 0.64 dB, and the maximum of the first sidelobes shifts slightly inwards. By analyzing Fig. 4.36, it can be seen that simulated directivity with the frequency 249050 Hz is affected by the tilt of two degrees in a higher degree than for 98100Hz. The modeled tilts affect the sidelobes by increasing dB strength, and the main lobe is highly affected by having a decrease of 3.835 dB. Such an effect in the main lobe should be easily recognized in measurements.

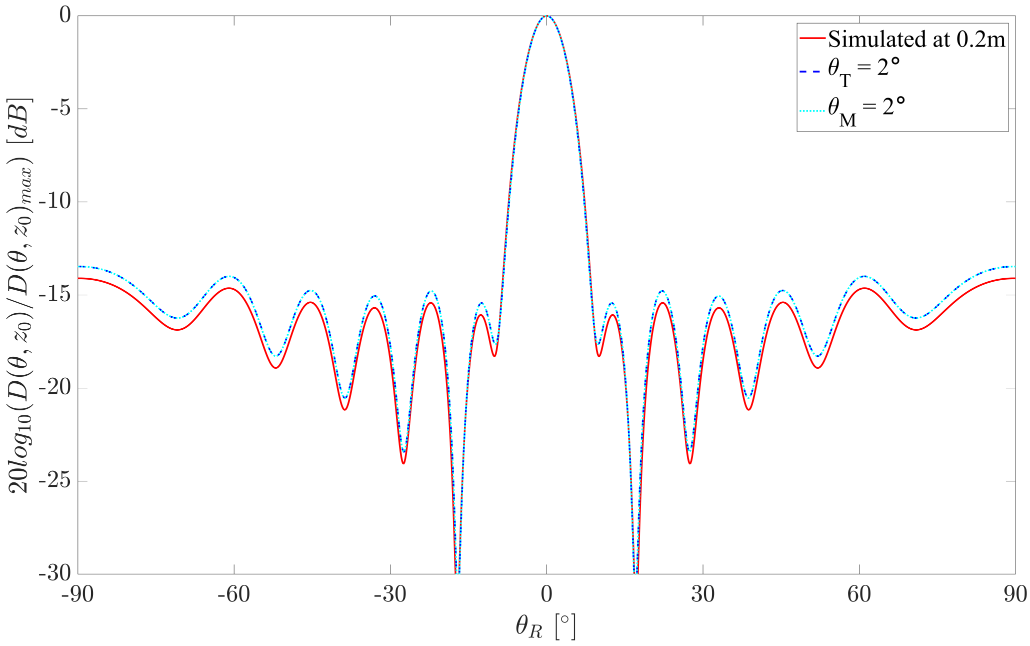



Figure 4.35: Simulated directivity at 0.2 m and frequency of 98100 Hz and analyzing the effect of the $\theta_T$ and $\theta_M$ equals two.

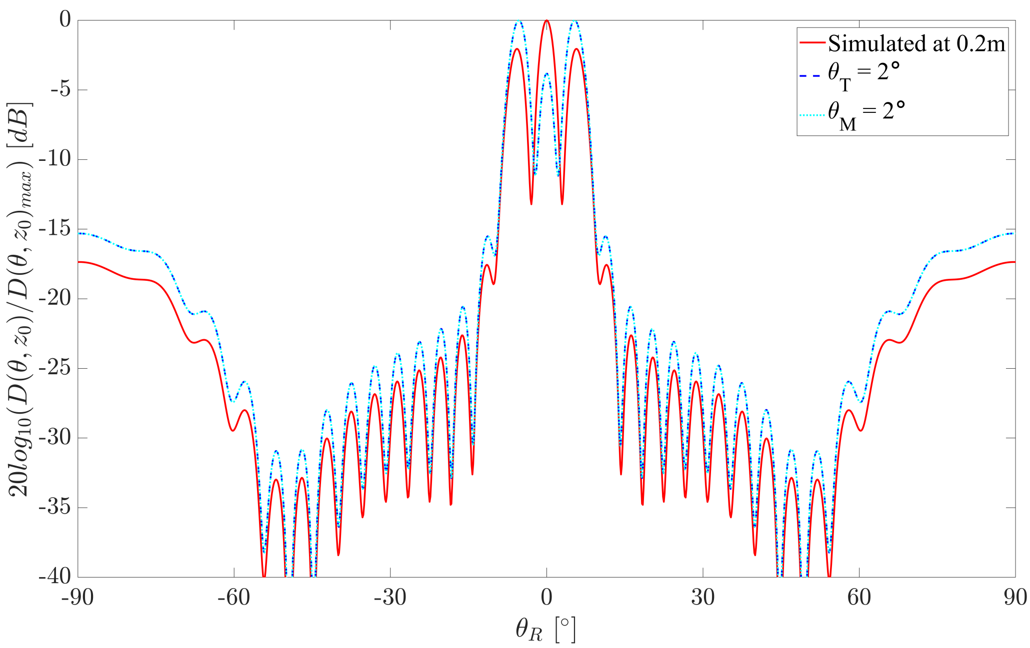



Figure 4.36: Simulated directivity at 0.2 m and frequency of 249050 Hz and analyzing the effect of the $\theta_T$ and $\theta_M$ equals two.

The third orientation is a horizontal offset $\sigma$ from the piezoelectric disk's front center to the R-stage rotation axis, as illustrated later in Fig. 4.37. The distance from the R-stage rotation axis to the microphone front center is $R$ and is constant. The center of the microphone's front no longer orbits with a constant distance of $z_0$ around the piezoelectric disk front center, instead orbits with an changing distance $z(\theta_R)$ dependent on the rotation angle. The angle $\alpha$ is the angle from the line passing through the R-stage rotation axis and microphone front center when $\theta_R$ equals zero and to the piezoelectric disk's front center. The distance $z_0$ is the horizontal distance from the piezoelectric disk to the microphone when $\theta_R$ equals zero.

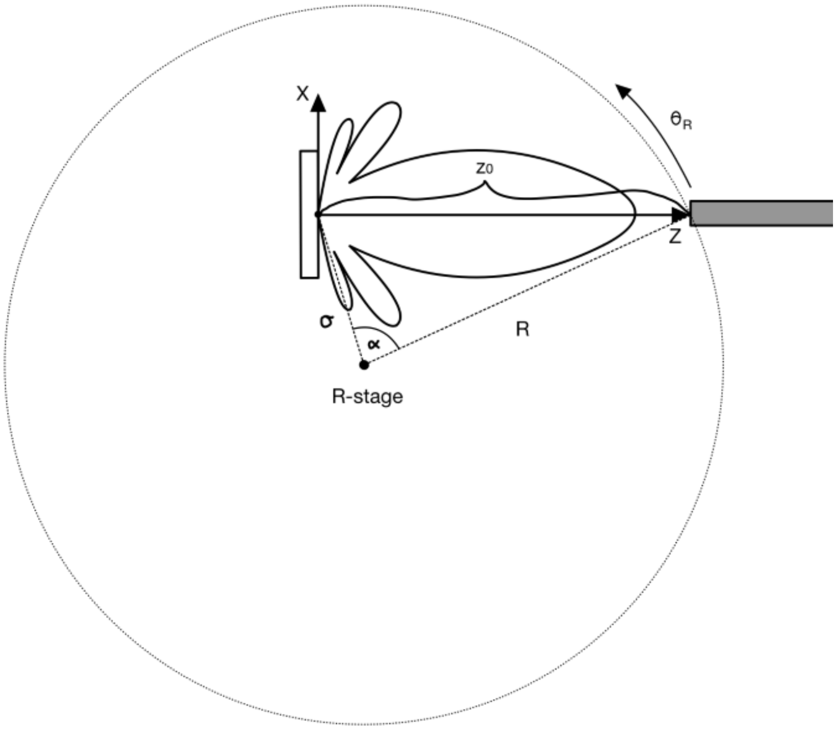



Figure 4.37: Illustration of the top view XZ-plane of the third orientation with a distance sigma from R-stage to disk's front center. The orbital radius of the microphone is the distance R. The angle alpha is the angle from the R line to the disk's front center.

The third, including first and second orientations, can be described as two points on a sphere with a radius of $z_0$ and an angular distance $\theta$ between the points. The distance from the X, Y, and Z-axis origin to the directivity D($\theta = 0$, $z_0$, $f$) is described as the OA vector in Fig 4.38, where $\theta$ equal to zero is the on-axis perpendicular to the piezoelectric disk surface, $z_0$ as the simulated distance and $f$ as simulated frequency. The distance from the X, Y, and Z-axis origin to a point on the sphere where the line passes through the front center of the microphone is described as OB in Fig 4.38. The $\theta_T$ is the tilt of the piezoelectric disk, $\theta_m$ is the angle to the center of the microphone's front dependent on $\beta$ from X'Z'-axis-plane and $j$, $\theta$ is the angle between points A and B on the sphere, and the $\theta_r$ is the horizontal rotation in Fig 4.38.

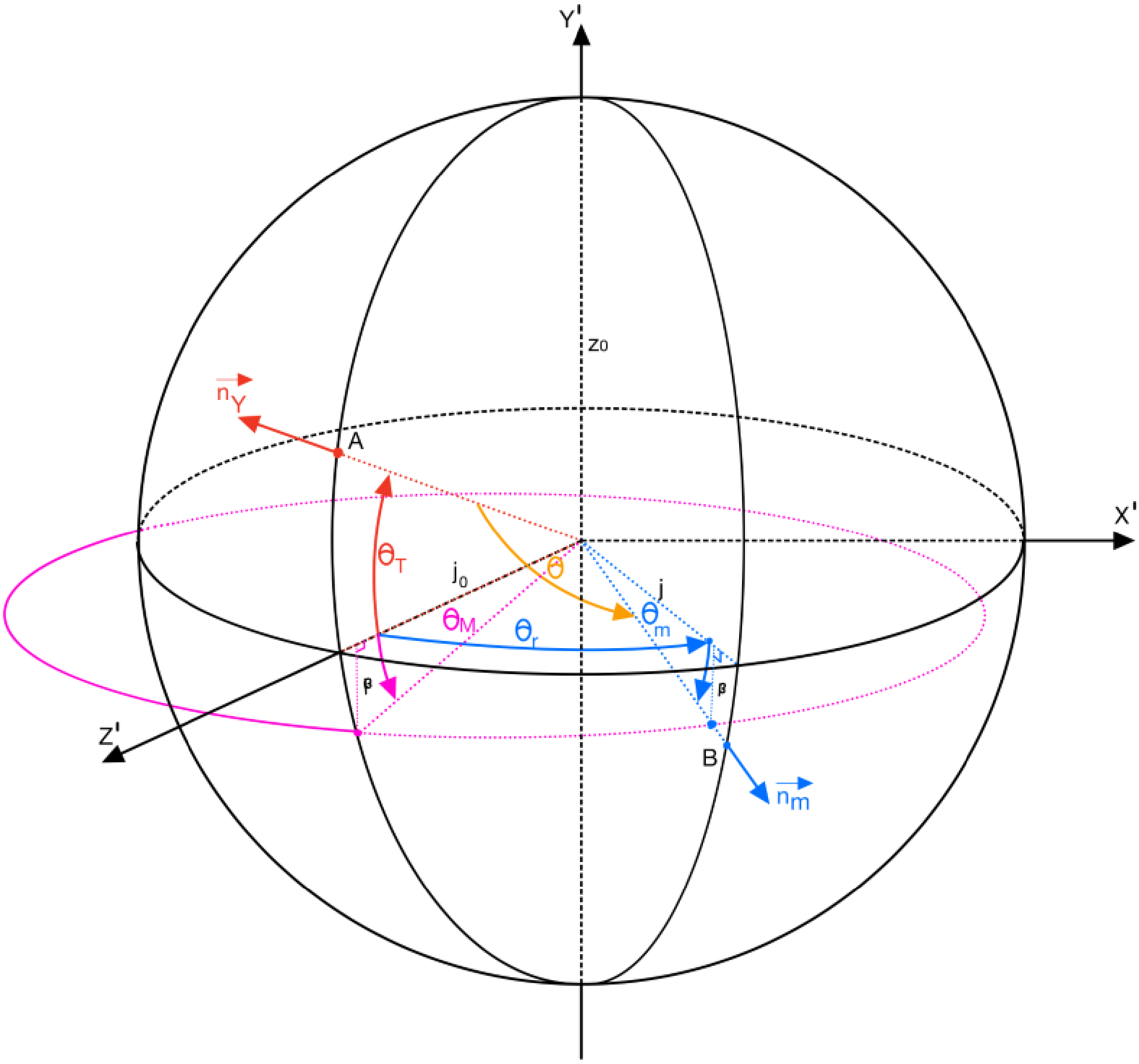



Figure 4.38: Illustration of the third, including first and second orientations, OA and OB, and the different values of $\theta_T$ disk tilt, $\beta$ microphone offset, $z_0$ sphere radius, $\overrightarrow{n_Y}$ normal unit vector of point A, $\overrightarrow{n_m}$ normal unit vector of point B, $\theta$ angle distance between A and B, and $\theta_r$ horizontal rotation.

The dot product between the normal unit vectors of OA and OB equals to

$$\overrightarrow{n_Y} \cdot \overrightarrow{n_m} = cos\theta \ , \tag{4.16}$$

where in the X, Y, and Z-axis frame, the normal unit vector can be written as

$$\overrightarrow{n_m} = \begin{pmatrix} cos(\theta_m) sin(\theta_r) \\ sin(\theta_m) \\ cos(\theta_m) cos(\theta_r) \end{pmatrix} . \tag{4.17}$$

The offset beta can be set to a constant or calculated with $\theta_M$ and the distance $z_0$ and equals

$$\beta = z_0 sin(\theta_M) , \tag{4.18}$$

and the distance $j_0$, which is the horizontal distance from the piezoelectric disk to the microphone when $\theta_R$ equals zero, is

$$j_0 = z_0 cos(\theta_M) . \tag{4.19}$$

The radius of the microphones orbit $R$ is derived in Appendix D.1 and equals

$$R = \sigma cos(\alpha) + \sqrt{j_0^2 + \sigma^2(cos^2(\alpha) - 1)} , \tag{4.20}$$

and the distance $j$ from the piezoelectric disk front center to any point on the microphones orbit is derived in Appendix D.2 and is

$$j = \sqrt{R^2 - 2R\sigma cos(\alpha - \theta_R) + \sigma^2} . \tag{4.21}$$

With the equation for the distance $j$ and offset $\beta$, the angle $\theta_m$ can be calculated as

$$\theta_m = tan^{-1}\left(\frac{\beta}{j}\right) , \tag{4.22}$$

and with the distances $j$, $j_0$, and $R$, the $\cos(\theta_r)$ is derived in Appendix D.3 and equals to

$$cos(\theta_r) = \frac{j^2 + j_0^2 - 4R^2 sin^2(\theta_R/2)}{2 j j_0} . \tag{4.23}$$

With Eqs 4.22 and 4.23, it is possible to calculate $\theta$ with Eq. 4.16 by inserting the normal unit vectors and $\theta$ equals to

$$\theta = cos^{-1}\left(sin(\theta_m)sin(\theta_T) + cos(\theta_m)cos(\theta_T)cos(\theta_r)\right) . \tag{4.24}$$

Now with $\theta$ deduced for all three orientations, the effects of the distance $\sigma$ and starting angle $\alpha$ can be analyzed in the R-stage rotation range $\theta_R$ from -90 to 90 degrees. The results are then plotted by taking the logarithmic result of $\mathrm{D}(\theta, z(\theta_R), f)/\max(\mathrm{D}(\theta, z(\theta_R) f))$ and plotting it against the measured angle $\theta_R$. The distance between the microphone's front center to the piezoelectric disk's front center $z$ is now dependent on the change in $\theta_m$'s angle. This change in the distance $z$ is solved by using interpolation between several distance simulations. In Fig. 4.39, the simulated directivity at 98100 Hz and the effect of distance $\sigma$ equals 5 mm, and the angle $\alpha$ equals 0, 90, 180, and 270 degrees are analyzed. From Fig. 4.39, it is seen that the modeled effects are largely dependent on the starting angle $\alpha$. At $\alpha$ equals 0 degrees, the entire signal is compressed closer to $\theta_R$ equals 0 degrees, and for $\alpha$ equals 180 degrees, the entire signal is stretched out relative to $\theta_R$ equals 0 degrees. At $\alpha$ equal 90 and 270 degrees, the signal is either stretched out on one side of $\theta_R$ equals 0 degrees and compressed closer to $\theta_R$ equals 0 degrees on the other side. In Fig. 4.40, the simulated directivity at 249050 Hz and the effect of distance $\sigma$ equals 5 mm, and the angle $\alpha$ equals 0, 90, 180, and 270 degrees are analyzed. From Fig. 4.40, it is seen that the modeled effects of $\sigma$ and $\alpha$ behave in the same matter as in Fig 4.39.

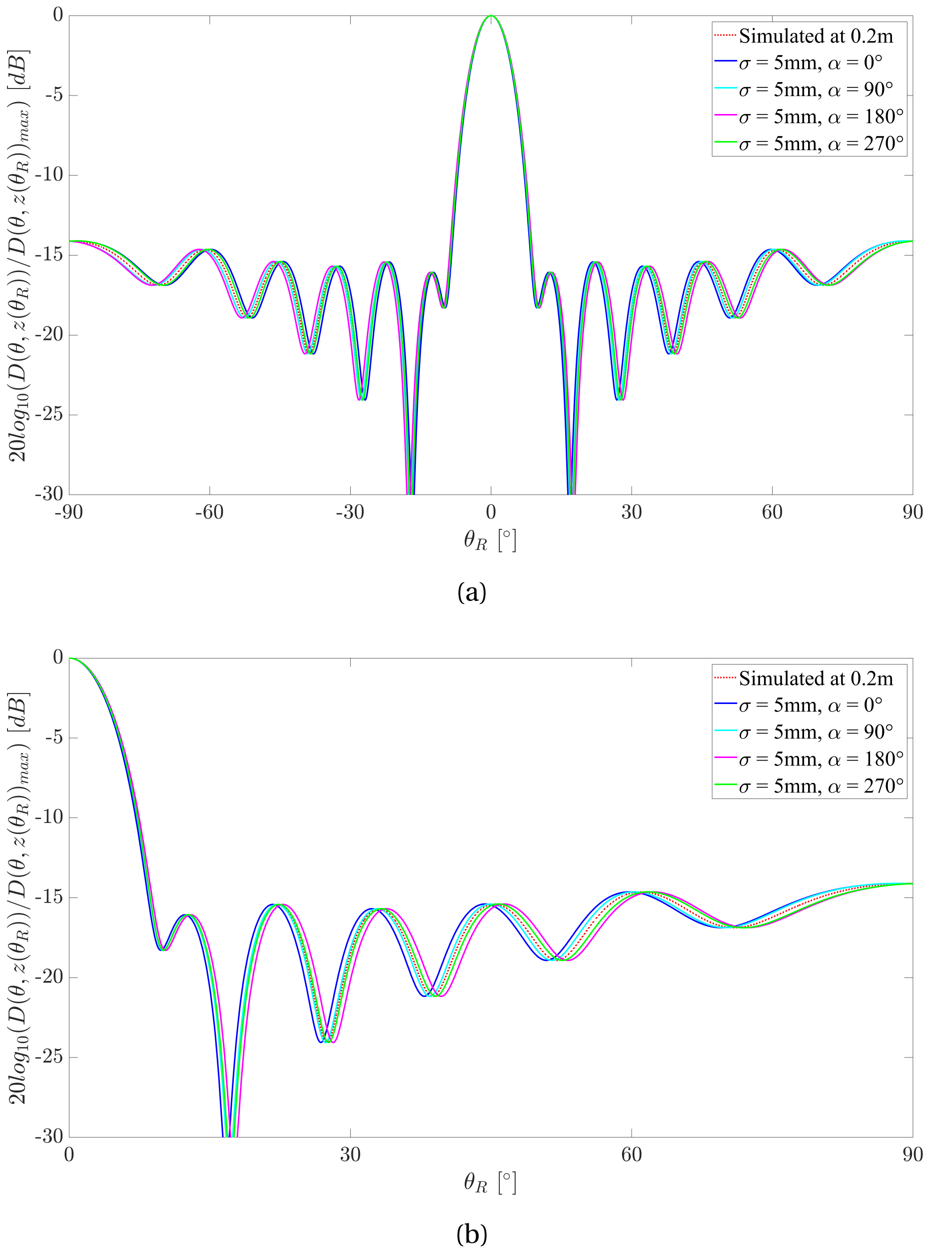
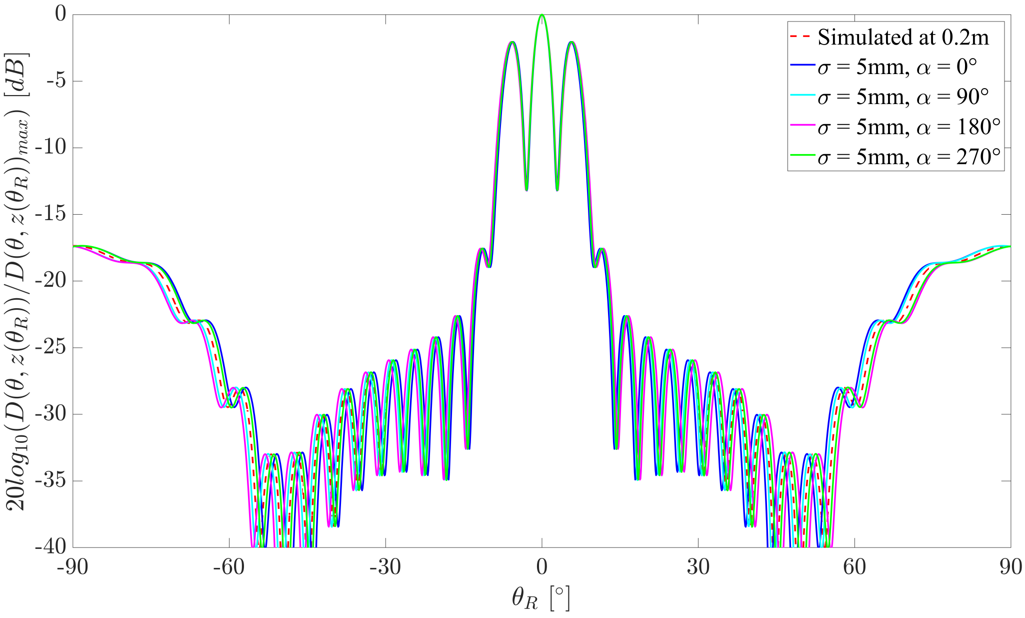
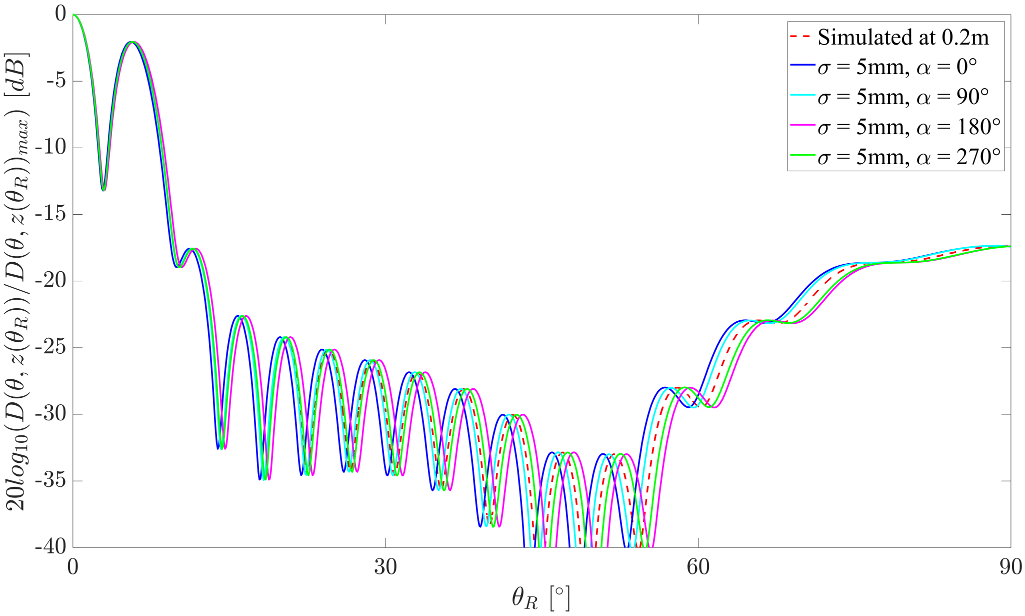



Figure 4.39: Simulated directivity at 0.2 m and frequency of 98100 Hz and analyzing the effect of the $\sigma$ equals 5 mm and $\alpha$ equals 0, 90, 180, 270 degrees.




(a)




(b)

Figure 4.40: Simulated directivity at 0.2 m and frequency of 249050 Hz and analyzing the effect of the $\sigma$ equals 5 mm and $\alpha$ equals 0, 90, 180, 270 degrees. (a) is the simulated and modeled effect from -90 to 90 degrees (b) is the simulated and modeled effect from 0 to 90 degrees.

# Chapter 5

# Finite element setup

This work uses the finite element method to simulate the piezoelectric disk Pz27 vibrating in a vacuum and fully immersed in a air. Sect. 5.1 gives information on FEMP and the current version used in this work, the different files FEMP uses, and a brief description of FEMP. Sect. 5.2 gives info on the material parameters, the material parameters of the piezoelectric element, and the parameters for the fluid. Sect. 5.3 gives information on the simulation parameters used in this work and their associated values. Sect. 5.4 describes the conversion of simulated pressure to comparable pressure with the measured pressure. Sect. 5.5 gives a visualization of the simulation structures.

## 5.1 FEMP 6.1

This work performs finite element simulations with the simulation tool FEMP developed by Kocbach [38] in a cooperation between the University of Bergen (UiB) and Christian Michelsen Research AS (CMR). Students and researchers at UiB and CMR/NORCE have continued developing FEMP ever since, and the current version used is FEMP 6.1. The programming language MATLAB [68] allows for fast implementation and visualization [38], making it the choice for the programming language of FEMP.

Different files are needed to alter variables, parameters, areas, points, boundary conditions, and materials to be able to run simulations. The file **matertial_.dat** in Appendix C.5 defines the transducer material and fluid parameters. The file **_.inn** in Appendix C.4 defines the set parameters; dimension of the structure, elements in the fluid or structure, material or fluid number, radius from origo to infinite elements region. The rest of the file defines simulation variables and parameters; material file, elements per wavelength frequency, order of the finite elements, order of the infinite elements, the frequency range, and type of simulations like direct harmonic analysis used to calculate, e.g., admittance. The scripts **read_inn_project.m** and **init_const_project.m** in Appendix C.1, C.2, and C.3 defines structure, areas, points and boundary conditions. It also uses the set parameters from the **_.inn** file.

FE approximates structures with a given number of volumes/elements, where computational time relies on the size and complexity of the structures. To reduce the complexity and, thereby, the computational time, FEMP implements axis symmetry which reduces volumes to rectangles. A problem modeled in FEMP may consist of up to two domains, finite elements and infinite elements domains. The finite elements domain uses 8 node isoparametric elements to solve the parameters inside the element, and the global to local coordinates transition with interpolation functions [21], see Fig. 5.1. In this domain, finite elements consist of the transducer region $\Omega_p$ and can consist of the inner fluid region $\Omega_{f1}$ [38], depending on the type of simulation. The infinite elements domain is not isoparametric but described by conjugates Astley-Leis infinite elements [9] and is solved using 10th order infinite elements. In this domain, infinite elements consist of the outer fluid region $\Omega_{f2}$ [38], see Fig. 5.2.

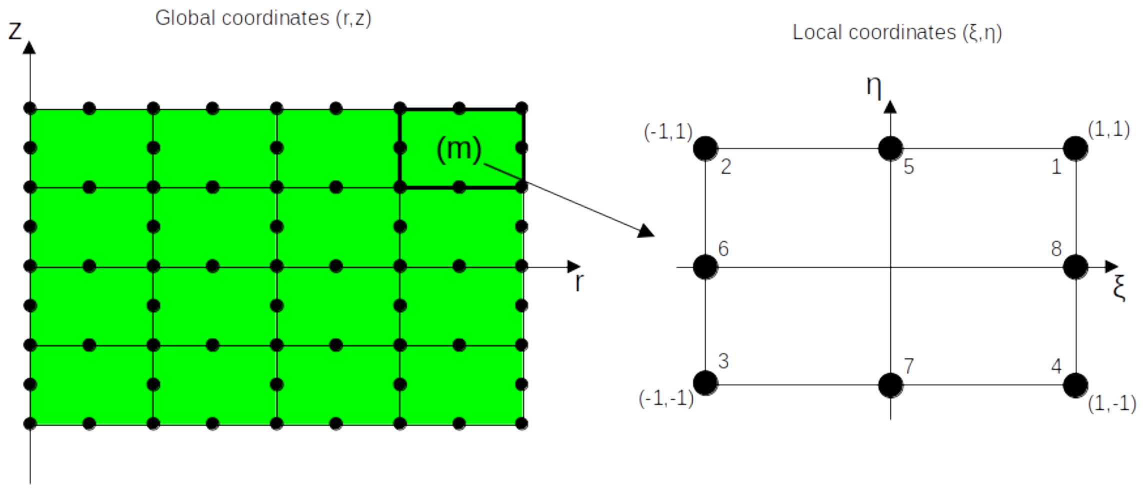



Figure 5.1: The colored area is the piezoelectric discs volume $\Omega_p$ split into a number of elements in the global coordinates and shows the connection from one element, m, from the global coordinates to the local coordinates for the 8 node isoparametric element [38].

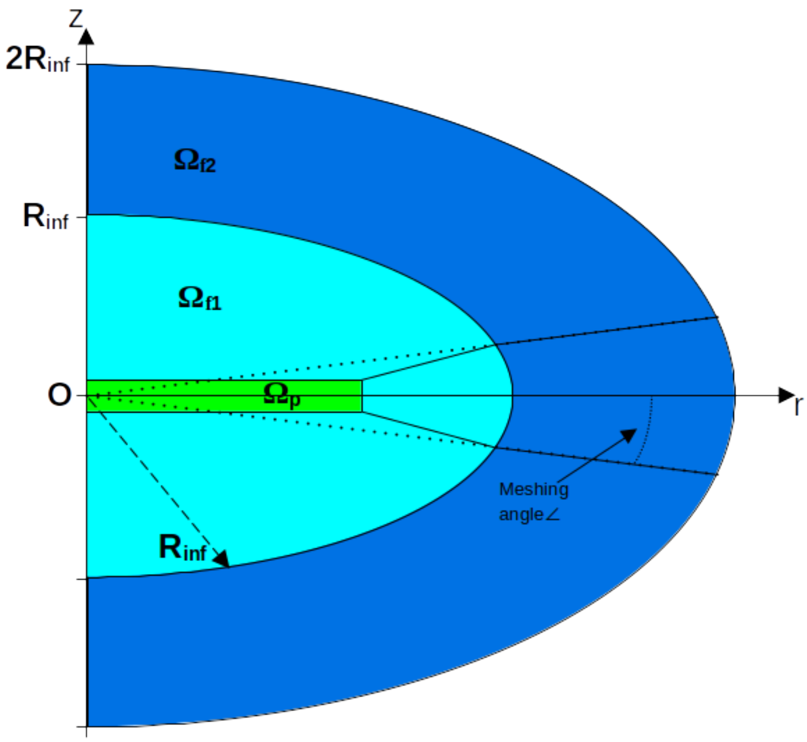



Figure 5.2: Illustration of the piezoelectric disc volume $\Omega_p$, the surrounding finite fluid volume $\Omega_{f1}$ and the infinite volume $\Omega_{f2}$, the distance $R_{inf}$ described in Sect. 5.3, and the meshing angle $\angle$ splitting the surrounding fluid volume $\Omega_{f1}$ and $\Omega_{f2}$ into three regions [38].

## 5.2 Material parameters

Piezoelectrical material data provided by manufacturers can often be incomplete or inaccurate. This incompleteness and inaccuracy can arise from variations in the composition in different batches, uncertainties, or lack of methods to provide better results of the material parameters. In simulations, it is essential to have as accurate material parameters as possible such that simulations match measurements. This matching is a demanding task to accomplish, and this work uses material parameters found in [37] and presented in Table 5.1. The material parameters of the piezoelectric element that FEMP uses are elastic stiffness matrix with a constant electric field $[c^E]$, piezoelectric stiffness matrix $[e]$, and dielectric stiffness matrix with constant strain $[\epsilon^S]$ [38]. The fluid parameters that FEMP uses are bulk modulus $K$ and density $\rho$. With complex values of the piezoelectric constants, it is possible to describe mechanical and electrical losses in the piezoelectric element, but fluid losses are not considered [38]. The material numbers are set in the **_.inn** file to select the materials or fluid for simulations. These numbers correspond to the material found in the **material_.dat** file.

### 5.2.1 Piezoelectric element, Pz27

The manufacturer of the piezoelectric element Pz27 used in this work is Meggitt, earlier known as Ferroperm. They state that their element has a variation of ±5% for all parameters and the lowest variation from batch to batch [46]. Previous work has shown that the parameters provided by Meggitt are not in the best agreement when used in simulations and compared with measurements [42][37][1]. This inaccuracy led to an adjusted dataset of Pz27 by Lohne and Knappskog [42][37], which has improved the agreement in the first and second radial mode. This dataset is shown in Tabel 5.1 and used to compare the simulations and measurements in this work.

Table 5.1: Because of inaccuracy in the original dataset from Meggit and others manufacturers, there is used in present work the adjusted dataset set for piezoelectric element Pz27 made by Lohne and further improved by Knappskog [42][37].

| Parameters | Unit | Lohne/Knappskog [42][37] |
|---|---|---|
| $c_{11}^E$ | $[10^{10} Pa]$ | $11.8750\left(1+i\frac{1}{95.7500}\right)$ |
| $c_{33}^E$ | $[10^{10} Pa]$ | $11.2050\left(1+i\frac{1}{177.990}\right)$ |
| $c_{44}^E$ | $[10^{10} Pa]$ | $2.11000\left(1+i\frac{1}{75.0000}\right)$ |
| $c_{12}^E$ | $[10^{10} Pa]$ | $7.43000\left(1+i\frac{1}{71.2400}\right)$ |
| $c_{13}^E$ | $[10^{10} Pa]$ | $7.42500\left(1+i\frac{1}{120.190}\right)$ |
| $e_{15}$ | $[C/m^2]$ | $11.2000\left(1-i\frac{1}{200.000}\right)$ |
| $e_{31}$ | $[C/m^2]$ | $-5.40000\left(1-i\frac{1}{166.000}\right)$ |
| $e_{33}$ | $[C/m^2]$ | $16.0389\left(1-i\frac{1}{323.770}\right)$ |
| $\epsilon_{11}^S$ | $[10^{-9} F/m]$ | $8.11043\left(1+i\frac{1}{50.0000}\right)$ |
| $\epsilon_{33}^S$ | $[10^{-9} F/m]$ | $8.14585\left(1+i\frac{1}{86.2800}\right)$ |
| $\rho$ | $[kg/m^3]$ | 7700 |

### 5.2.2 Air

When doing finite element simulations in a fluid, the dataset needs the parameters density $\rho$ and bulk modulus $K$ for the given fluid. Since the given fluid in this work is air, and density and bulk modulus are dependent on environmental conditions, setting the temperature and pressure is necessary. At $20°C$ and $1atm$, Appendix A10 in Kinsler & Frey [36] gives the density and sound speed. Using the definition of the thermodynamic speed of sound given as [36]

$$c = \sqrt{\frac{K}{\rho}}\,, \tag{5.1}$$

the bulk modulus can be calculated by rewriting this definition as

$$K = c^2 \rho \ . \tag{5.2}$$

The values found, calculated, and used in the material dataset are given in Table 5.2.

Table 5.2: The material dataset for air used in this work, for 1 $atm$ and 20 °C. [36].

| Name | Parameter | Unit | Value |
|---|---|---|---|
| Sound speed | c | $[m/s]$ | 343 |
| Density | $\rho$ | $[kg/m^3]$ | 1.21 |
| Bulk modelus | $K$ | $[10^5 Pa]$ | 1.42355 |

## 5.3 Simulation parameters

Simulation in FEMP requires a set of parameters defined in the **_.inn** file, where the number of parameters depends on the type of simulation or structural complexity. The parameters used in this work are only those accounted for here.

Radius and thickness are the dimensions that define the piezoelectric element in this work. They are measured with a Mitutoyo M310-25 micrometer [47] and given in Table 5.3. These dimensions are then used in FEMP when simulating. The element simulation's accuracy and computational time depend on elements per wavelength and frequency [38]. In the case of vacuum, the elements per wavelength used are 98 in radius and thickness direction. For the non-vacuum case, the element per wavelength is 7 in the fluid, 25 in radius, and 98 in thickness. When calculating the wavelength, the frequency is the maximum simulated frequency [38]. Fluids use the wavelength for longitudinal waves, while materials use the wavelength for shear waves [38]. The reason for using shear waves, the shortest wavelength, over longitudinal weaves is that the number of elements per wavelength needed to obtain a specific accuracy is almost equal in radial and thickness directions [38]. This investigation of comparing shear weaves against longitudinal waves is done by [38].

Table 5.3: Measured dimensions of the piezoelectric element Pz27 with calculated uncertainties. The dimensions measured with Mitutoyo M310-25 micrometer [47] and are mean values of ten measurements.

| Element number in batch | Diameter | Thickness | Unit |
|---|---|---|---|
| #7 | 20.25±0.01 | 2.048±0.011 | [mm] |

Studies of [38] make it possible to calculate the optimal distance from origo to the boundary between finite and infinite elements. Using 10th order infinite elements in simulation, [38] suggest the normalized distance

$$R_{inf}/(a^2/\lambda) = S \tag{5.3}$$

should be larger or equal to the critical distance $S = 0.32$. This normalized value corresponds to $R_{inf}$, approximately 28 mm at the maximum simulated frequency of 300 kHz and sound speed in the air. Therefore $R_{inf}$ is set to 30 mm in this work. When it comes to the order of finite elements, 8 nodes are used, corresponding to 2nd order isoparametric elements [2]. Typically the meshing angle $\angle$ for fluid regions is set to 1.3 rad [2], see Fig. 5.2.

FEMP uses direct harmonic analysis to solve the FE problem over a set range of frequencies [38]. For simulation of the element fully immersed in the fluid, a wide frequency range can take a lot of computing power and computation time. Therefore, all simulations performed in this work only simulate the frequency range of interest from 1-300 kHz with a step of 50 Hz. These simulations also account for complex losses in the element. The main computed quantities in this work are admittance $Y_T(f)$, directivity $D(r,\theta,f)$, 2-D near-field and far-field pressure $p(x,z,f)$, and on-axis pressure $p_{ax}(z,f)$.

## 5.4 Simulated pressure

The voltage $V_1$ over the piezoelectric disk, described in Sect. 2.7.2, will change according to the measuring frequency. At around resonance, this voltage decreases rapidly relative to the $V_0$ voltage generated by the signal generator. The voltage over the piezoelectric disk can be calculated by measuring $V_{0m}$ with the oscilloscope and using the transfer function $H_{0m1}^{VV}$, see Eq. 2.21. Due to the simulated pressure amplitude being linear dependent on the input voltage amplitude

$$\frac{\{p_{sim}\}}{V_{sim}} = \frac{\{p_{pp}\}}{V_{1_{pp}}}\ , \tag{5.4}$$

where $p_{sim}$ is the simulation pressure, $V_{sim}$ is the simulation voltage applied to the piezoelectric disk, $V_{1_{pp}}$ is the peak-to-peak voltage calculated over the piezoelectric disk, and $p_{pp}$ is the adjusted peak-to-peak simulation pressure, which is used to compare to peak-to-peak measurement pressure $p_{4_{pp}}$. The simulated pressure $p_{sim}$ can be scaled by the measured peak-to-peak voltage $V_{1_{pp}}$ as

$$\{p_{pp}\} = \frac{\{p_{sim}\}}{V_{sim}} V_{1_{pp}}\ , \tag{5.5}$$

where $V_{sim}$ is equal to 1 V in these simulations.

## 5.5 Structure setup

This work uses two different **read_inn_project.m** scripts given in Appendix C.1 C.2, one for the element fully immersed in the fluid and one for vacuum simulations. Plotting the mesh view, Fig. 5.3 and 5.4, visualize the result of the two scripts and the FE problem to solve.

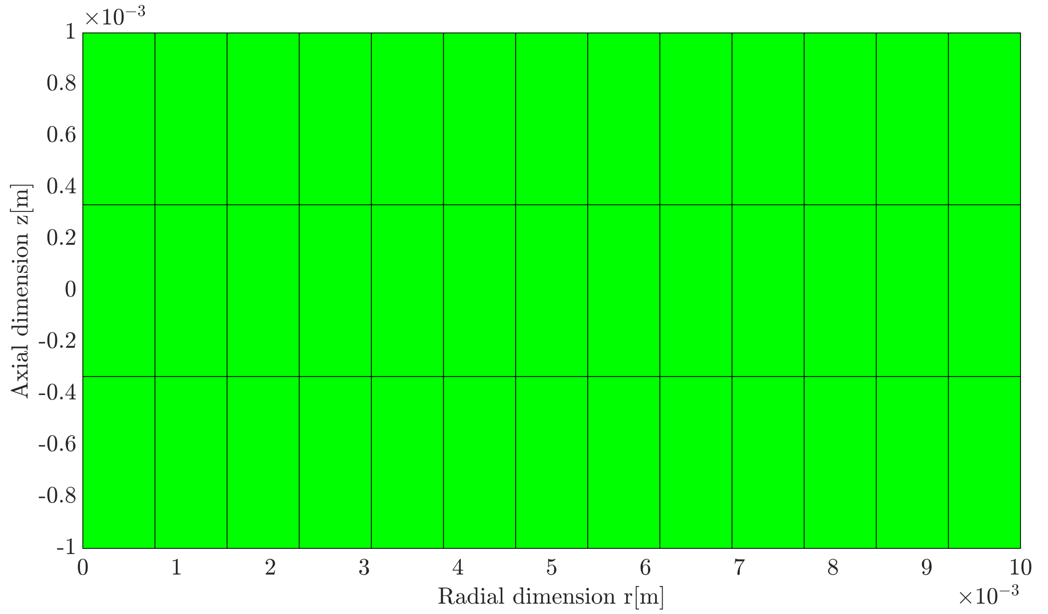



Figure 5.3: For illustration purposes is the mesh view of vacuum simulation, plotted with 7 elements per wavelength in radial and thickness direction at a frequency of 300 kHz.

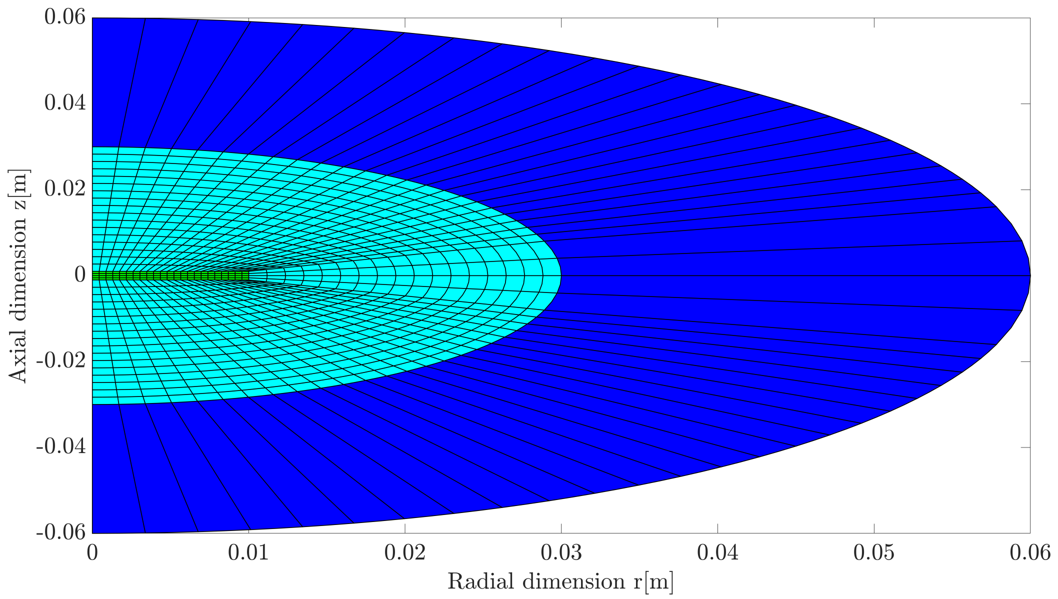



Figure 5.4: For illustration purposes is the mesh view of piezoelectric disc fully immersed in fluid simulation, plotted with 2 element per wavelength in fluid at a frequency of 100 kHz. Elements per wavelength in radial and thickness direction are set to 7 and 28.

# Chapter 6

# Results and discussion

This chapter presents the results obtained with the FE simulations method described in Chapter 5. The results obtained by performing measurements of the piezoelectric disk's electrical properties with an impedance analyzer (Table 3.2) described in Sect. 3.2 are also presented. And finally, acoustic measurements with the piezoelectric disk transmitting ultrasound and measured with a condenser microphone by Brüel & Kjær are presented. Sect. 6.1 presents all the electrical properties from FE simulations and the measurements results. Sect. 6.2 presents all the acoustically measurements results compared to FE simulations.

## 6.1 Electrical properties of the piezoelectric disk

This section studies the FE simulated electrical properties of a piezoelectric disk with the same dimensions as piezoelectric disk in Table 5.3. The electrical properties of piezoelectric disk are studied by measuring the properties and compared with the FE simulated electrical properties performed in FEMP.

### 6.1.1 Comparison of electrical properties between FE simulations in a vacuum and air

The piezoelectric disk's electrical admittance FE simulations are done in a vacuum and air from 1-300 kHz. This frequency range covers the two first radial modes, given the datasets in Tables 5.1 and 5.2, and the dimension of the piezoelectric disk in Table 5.3 is used. The reason for doing the simulations in both vacuum and air is to see how the air impacts the electrical properties of the piezoelectric disk. Based on the plots in Fig. 6.1, there is considerable overlap between simulated admittance for vacuum and air. One has to zoom in significantly to see that they do not overlap perfectly in the plot. The reason for this is that the acoustic impedance in the air is small, which can cause the piezoelectric disk to be considered to oscillate in a vacuum.

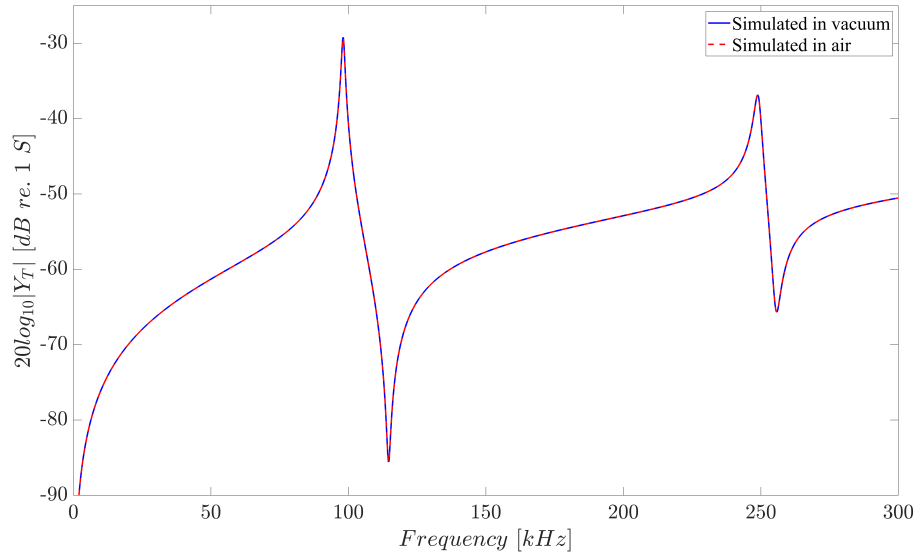



Figure 6.1: FE simulation of the piezoelectric's admittance plotted in dB relative to one siemens over the frequency range of 1-300 kHz. FE simulation of the disk in a vacuum is the blue line, and the FE simulation of the disk in the air is the stippled red line.

The radial modes are found by determining the maximum conductance defined as the serial resonance frequency [32](Sect. 2.1). By studying Fig. 6.2, the maximum conductance is found at 98.1 kHz for the first radial mode R1 and 249.05 kHz for the second radial mode R2. For completeness of the comparison of the FE simulated piezoelectric disk are, the susceptance shown in Fig. 6.3.

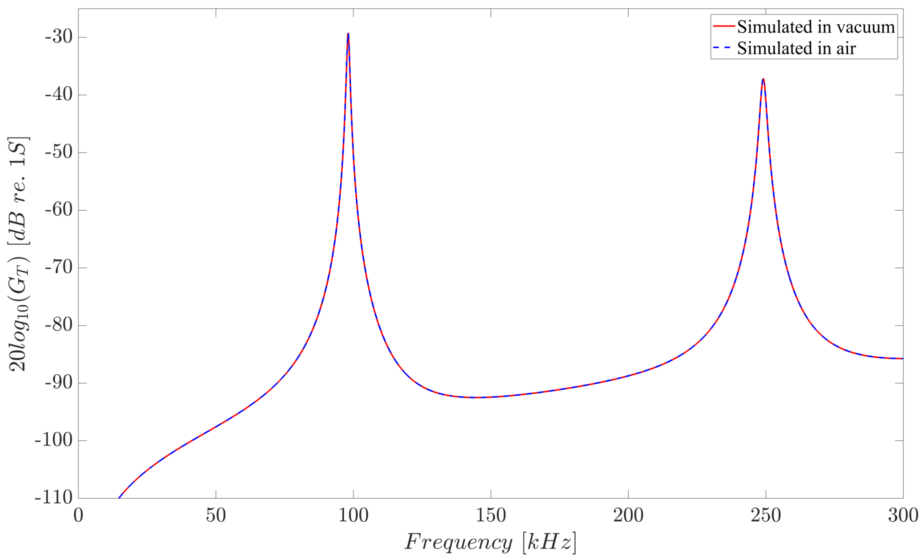



Figure 6.2: FE simulation of the piezoelectric's conductance plotted in dB relative to one siemens over the frequency range of 1-300 kHz. FE simulation of the disk in a vacuum is the blue line, and the FE simulation of the disk in the air is the stippled red line.

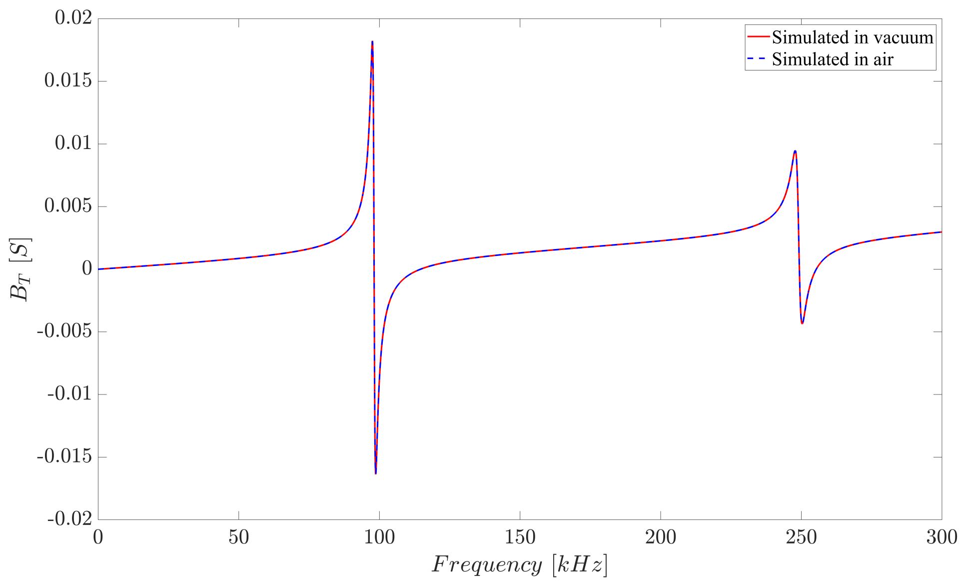



Figure 6.3: FE simulation of the piezoelectric's susceptance over the frequency range of 1-300 kHz. FE simulation of the disk in a vacuum is the blue line, and the FE simulation of the disk in the air is the stippled red line.

### 6.1.2 Comparison of electrical properties between measurements and FE simulations

The piezoelectric disk's electrical conductance and susceptance are measured with the impedance analyzer as described in Sect. 3.2. It is measured directly on the electrodes and measured on the wires that are soldered onto the electrodes. This is done to see if the short, thin wires soldered to the electrodes affect the measurement results. The results from the impedance analyzer are then compared to the FE simulation in air. The frequency range of the measurements of the electrical properties on the piezoelectric disk matches the FE simulations frequency range, 1-300 kHz. Fig. 6.4 compares the conductance of the measurement results directly on the electrodes, the measurements on the wires, and the FE simulations. In Fig. 6.4, there is a large overlap between the electrode and wire measurements. However, in the frequency range from approx. 50 kHz to 75 kHz in this figure, there are small irregular spikes in the conductance of the electrode measurement. There is a much more consistent measurement result on the wires than the electrodes' measurement. This smoother result may be due to the wires soldered to the electrodes providing better current flow than measuring directly on the electrodes. However, this statement does not explain why irregularities only appear in the specific measuring range from 50 kHz to 75 kHz and are only positive in order of magnitude. Furthermore, it also does not explain why it is only shown in the conductance and not for the susceptance and admittance of the same measurement, see Figs. 6.5 and 6.6. Since the irregularity is frequency-dependent, it is possible that the wires touching the electrodes may have a spring load effect on the electrodes. This "spring" then vibrates close to a resonant frequency and contributes to the irregularities in this frequency range. One thought is that the spikes may occur due to a certain vibration of the disk, but this is weakened because they are not seen in the wire measurement. The last thought is that these irregularities happen due to noise. Regarding the large difference between measured and simulated conductance in the frequency range from approx. 110 kHz to 165 kHz. It is thought that this arises due to the piezoelectric constants used in the FE simulations not perfectly matching the piezoelectric constant of the piezoelectric disk or the fact that these measurements are performed with solder lump, which is not considered in the FE simulations.

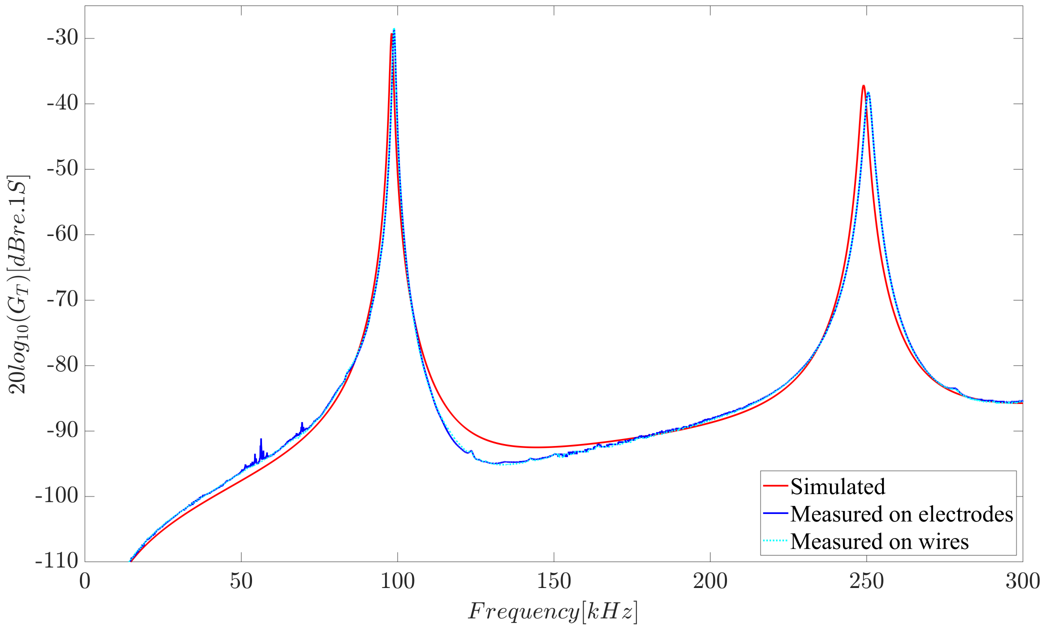



Figure 6.4: FE simulation and measurement of the piezoelectric's conductance plotted in dB relative to one siemens over the frequency range of 1-300 kHz. FE simulation of the disk in a vacuum is the red line, the measured on the disk electrodes is the blue line, and the measured on the wired soldered to the disk electrode is the stippled turquoise line.

By comparing the measurement and FE simulated conductances, it is seen that the simulated conductance's first radial extension mode, R1, frequency is shifted slightly down, and the magnitude is not as large as the measured values. The same shift is seen in comparing simulated and measured radial mode, R2, but to a greater extent. The frequencies of the maximum conductances are given in Table 6.1, and the frequency differences and magnitude differences of the maximum conductances of FE simulated relative to the measured are shown in Table 6.2.

Table 6.1: Comparison of the frequencies of the maximum conductances found in Fig. 6.4 of the FE simulation in air and measurements made with the impedance analyzer. R1 is the first radial extension mode, and R2 is the second radial extension mode.

| Mode | Simulated | Measured on wires | Measured on electrodes | Unit |
|---|---|---|---|---|
| R1 | 98100 | 98860 | 98870 | [Hz] |
| R2 | 249050 | 250500 | 250550 | [Hz] |

Table 6.2: Comparison of the magnitude difference ΔMagnitude and frequency difference Δf of the maximum conductances found in Fig. 6.4 of the FE simulation in air relative to the measurements made with the impedance analyzer. R1 is the first radial extension mode, and R2 is the second radial extension mode.

| Simulated vs Measured on wires | | |
|---|---|---|
| Mode | ΔMagnitude | $\Delta f$ |
| R1 | 0.86 dB | 760 Hz |
| R2 | -1.01 dB | 1450 Hz |
| **Simulated vs Measured on electrodes** | | |
| Mode | ΔMagnitude | $\Delta f$ |
| R1 | 0.61 dB | 770 Hz |
| R2 | -1.1 dB | 1500 Hz |

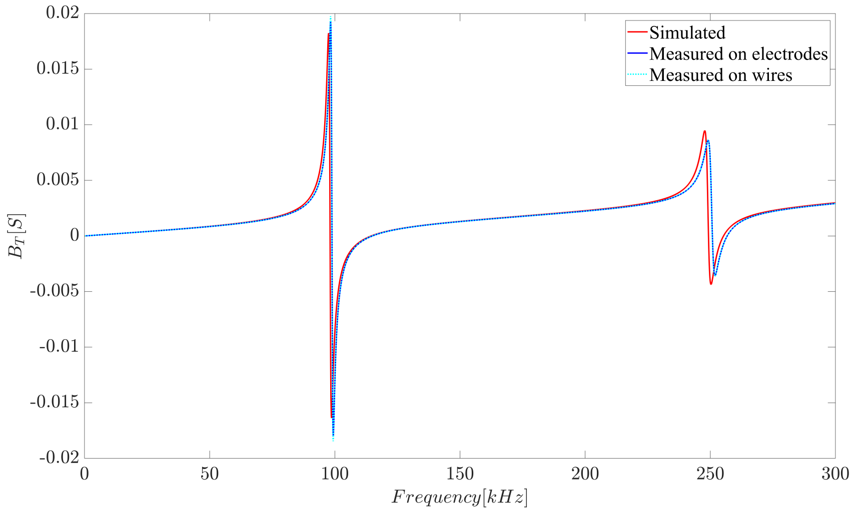



Figure 6.5: FE simulation and measurement of the piezoelectric's susceptance over the frequency range of 1-300 kHz. FE simulation of the disk in a vacuum is the red line, the measured on the disk electrodes is the blue line, and the measured on the wired soldered to the disk electrode is the stippled turquoise.

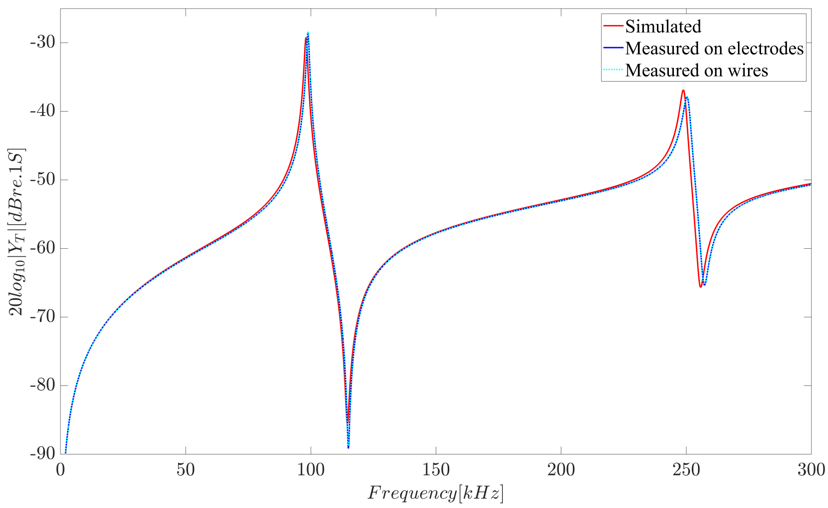



Figure 6.6: FE simulation and measurement of the piezoelectric's conductance plotted in dB relative to one siemens over the frequency range of 1-300 kHz. FE simulation of the disk in a vacuum is the red line, the measured on the disk electrodes is the blue line, and the measured on the wired soldered to the disk electrode is the stippled turquoise line.

## 6.2 Acoustic characteristics of the piezoelectric disk

In this section, acoustic measurements made with the piezoelectric disk are studied and compared with FE simulations. First, the frequencies used in the acoustic measurements and which simulation frequencies to compare with the measurements are determined. Then directivity measurements are presented, then on-axis pressure measurements, and finally, 2-D sound pressure field of the near and far field.

### 6.2.1 Frequency selection for measurements and FE simulations

For the on-axis pressure and 2-D pressure field, the frequencies used correspond to 1:2, 2:3, 1:1, 3:2, and 2:1 of the first radial modes given in Table 6.1. For the directivity, the frequencies used correspond to the same frequencies as for the on-axis pressure and 2-D pressure field and, in addition, the second radial mode given in Table 6.1. This method of scaling the frequency relative to the first radial modes is due to the FE simulation, and the measurement results of the radial modes in Tabel 6.1 are not equal. In order to be able to compare the acoustic characteristics of simulations and measurements with each other, it is then chosen to use the first radial modes of the FE simulation and measurement as the reference frequency when comparing results outside the resonance. The final frequencies used for comparing measurements to FE simulations are shown in Table 6.3.

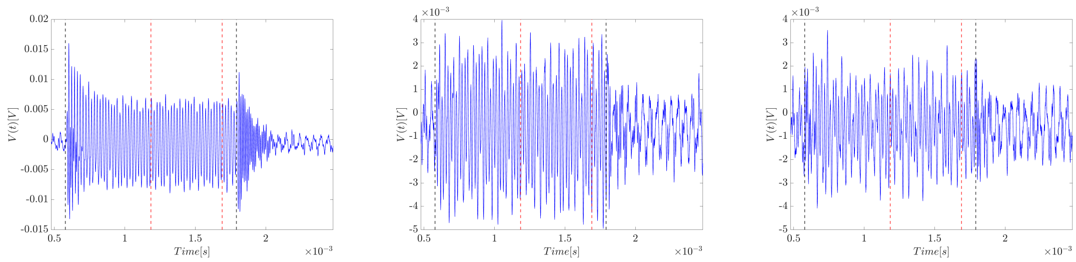
A reason for only measuring directivity at the second radial mode is due to directivity being a normalized quantity, making it feasible to compare FE simulations with directivity measurements. Another reason is that the second radial mode only is used to measure directivity is that microphone sensitivity is not valid for such high frequencies, see Fig 3.14. Therefore, only qualitative measurements can be made for such frequencies, and only chosen to do so for directivity due to lack of time.

Table 6.3: The scaling factor of the modes in parentheses and the calculated frequency result, used to compare acoustic characteristics between simulation and measurement. Delta f is the frequency difference between the simulations and the measurements comparisons.

| Scaling of mode | | Simulation frequency | Measurement frequency | $\Delta f$ | Unit |
|---|---|---|---|---|---|
| 1:2 | (R1) | 49050 | 49430 | 380 | [Hz] |
| 2:3 | (R1) | 65400 | 65907 | 507 | [Hz] |
| 1:1 | (R1) | 98100 | 98860 | 760 | [Hz] |
| 3:2 | (R1) | 147150 | 148290 | 1140 | [Hz] |
| 2:1 | (R1) | 196200 | 197720 | 1520 | [Hz] |
| 1:1 | (R2) | 249050 | 250500 | 1470 | [Hz] |

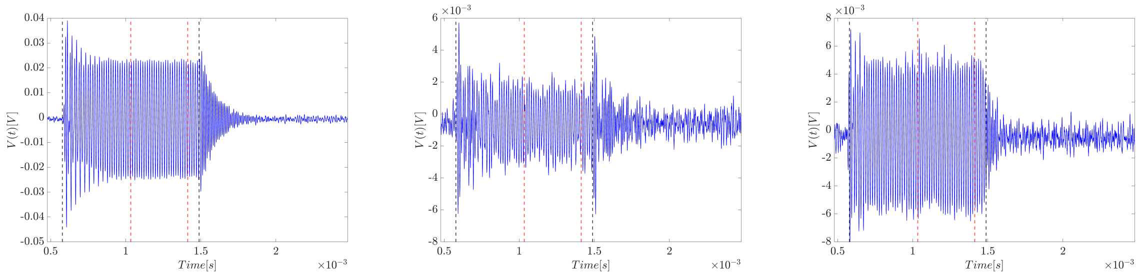
### 6.2.2 Acoustic signals examples over three different angles and for all used frequencies

This section gives examples of the measurement results of the acoustic signal $V_{5m}(t)$, measured at channel two on the oscilloscope at three different angles and for all frequencies in Table 6.3. These examples demonstrate that the acoustic signal quality changes with changing measuring angles. This change can be due to measurements performed in the nodes in the directivity pattern or with too low voltage amplitude $V_{0_{pp}}$ used outside resonance frequencies, which can lead to low signal-to-noise ratio (SNR) and low-pressure amplitudes.



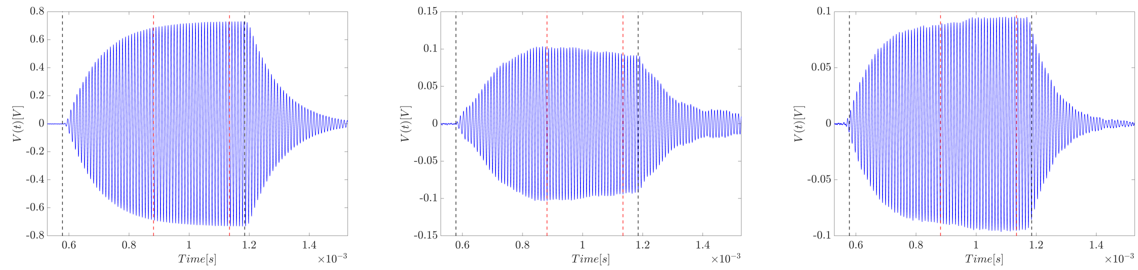

(a) (b) (c)

Figure 6.7: Comparison of the acoustic measurement signal for the frequency 49430 Hz, and for the angles (a) 0 degrees, (b) 25 degrees, and (c) 50 degrees. The plot is given in voltage vs. time. The red dotted lines represent the time domain used in FFT, and the black dotted represents the start and end of the signal. The distance between the piezoelectric disk and microphone is 0.2 m. (T = 25.2 °C, RH = 32.6 %, P = 990 hPa).



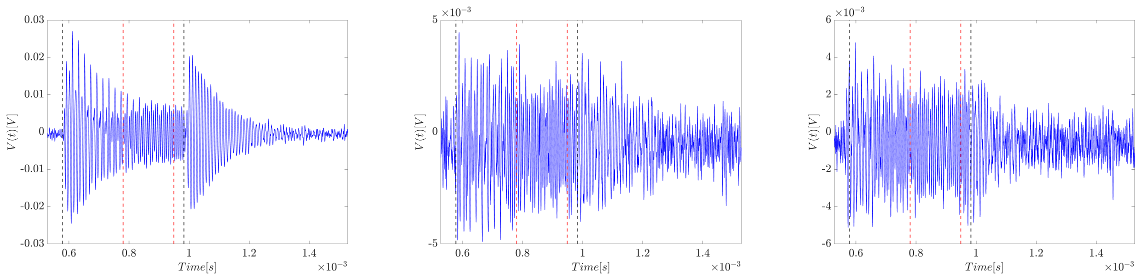

(a) (b) (c)

Figure 6.8: Comparison of the acoustic measurement signal for the frequency 65906 Hz, and for the angles (a) 0 degrees, (b) 25 degrees, and (c) 50 degrees. The plot is given in voltage vs. time. The red dotted lines represent the time domain used in FFT, and the black dotted represents the start and end of the signal. The distance between the piezoelectric disk and microphone is 0.2 m. (T = 25.3 °C, RH = 35.3 %, P = 990 hPa).



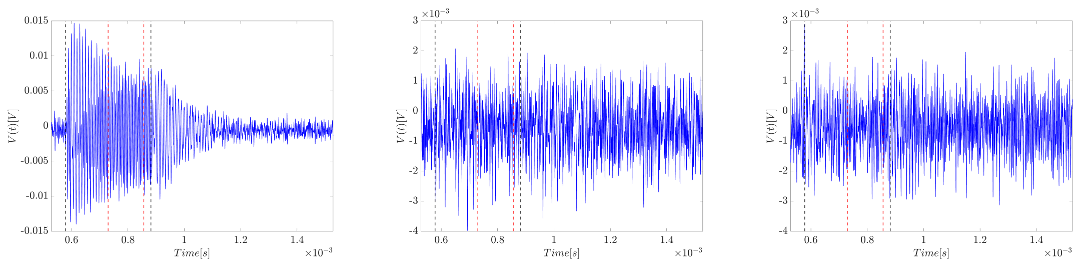

(a) (b) (c)

Figure 6.9: Comparison of the acoustic measurement signal for the frequency 98860 Hz, and for the angles (a) 0 degrees, (b) 25 degrees, and (c) 50 degrees. The plot is given in voltage vs. time. The red dotted lines represent the time domain used in FFT, and the black dotted represents the start and end of the signal. The distance between the piezoelectric disk and microphone is 0.2 m. (T = 25.1 °C, RH = 33.4 %, P = 991 hPa).



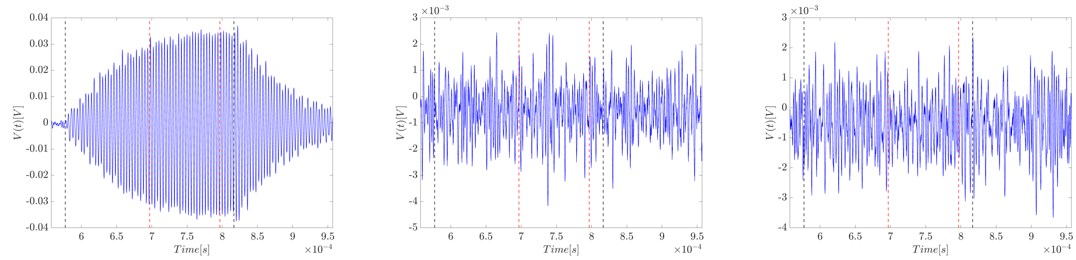

(a) (b) (c)

Figure 6.10: Comparison of the acoustic measurement signal for the frequency 148290 Hz, and for the angles (a) 0 degrees, (b) 25 degrees, and (c) 50 degrees. The plot is given in voltage vs. time. The red dotted lines represent the time domain used in FFT, and the black dotted represents the start and end of the signal. The distance between the piezoelectric disk and microphone is 0.2 m. (T = 25.3 °C, RH = 36.3 %, P = 990 hPa).



(a) (b) (c)

Figure 6.11: Comparison of the acoustic measurement signal for the frequency 197720 Hz, and for the angles (a) 0 degrees, (b) 25 degrees, and (c) 50 degrees. The plot is given in voltage vs. time. The red dotted lines represent the time domain used in FFT, and the black dotted represents the start and end of the signal. The distance between the piezoelectric disk and microphone is 0.2 m. (T = 25.4 °C, RH = 37.8 %, P = 991 hPa).



(a) (b) (c)

Figure 6.12: Comparison of the acoustic measurement signal for the frequency 250500 Hz, and for the angles (a) 0 degrees, (b) 25 degrees, and (c) 50 degrees. The plot is given in voltage vs. time. The red dotted lines represent the time domain used in FFT, and the black dotted represents the start and end of the signal. The distance between the piezoelectric disk and microphone is 0.2 m. (T = 26.3 °C, RH = 40.2 %, P = 1007 hPa).

### 6.2.3 Comparison of directivity between FE simulations and measurements

The directivity measurements of the piezoelectric disk are performed at the frequencies given in Table 6.3 and over the angles -90 to 90 degrees with one-degree resolution. These measurement results are compared to the corresponding FE simulations of the directivity with the frequencies in Table 6.3. All directivity measurements and simulations are conducted at a distance of 0.2 m between the disk and microphone. The voltage from the signal generator was 1 $V_{0_{pp}}$, and the received signal was amplified by 60 dB. The measurement results are normalized relative to the maximum voltage and plotted linearly and logarithmically. At the measuring distance of 0.2 m, all measurements are performed except for one frequency in the far field, see Table 6.4, where the Rayleigh distance defines the far field as

$$z_R = \frac{\pi a^2}{\lambda}\,, \tag{6.1}$$

where $\lambda$ is the wavelength defined by sound speed divided by the frequency. The measurements are desirable to carry out in the far field but not too far out in the far field because it is also desirable to obtain a good SNR.

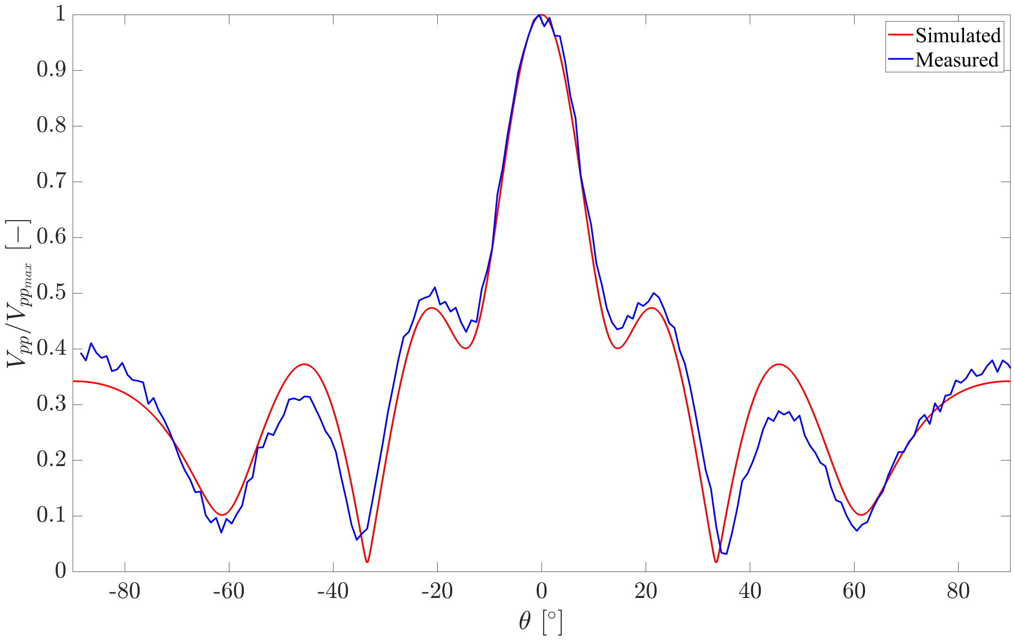
For Fig. 6.13, the measurement shows good agreement in the main and first side lobe relative to the simulation. However, from the first side lob and with the angle continuing to increase, there appears to be noise present in the measurement. In the first side lobe, effects due to tilting $\theta_T$ or angle $\theta_M$ from the model in Sect. 4.5 can be seen as being present in the measurement, and a $\sigma$ effect at $\alpha$ approx. 0 degrees can also seem to be present. However, these effects are not consistent for all degrees, which leads to the probability that there are other reasons why these effects occur. In the second side lobe, the magnitude of the measurements is lower compared to the simulation and can be due to SNR. At this second side lobe, which is at about 50 degrees, Fig. 6.7c does show a significant amount of noise is present relative to the measurement signal.

Table 6.4: Rayleigh distance for the different measurement frequencies where sound speed c = 343 m/s and radius of the disk a = 10 mm are used when calculating the distance.

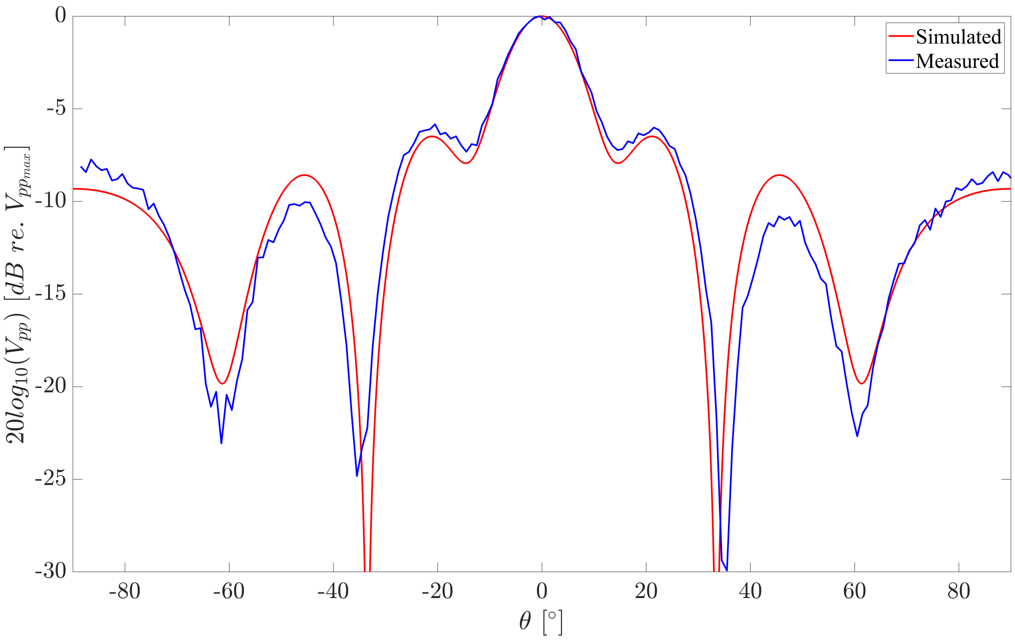
| Rayleigh distance | Frequency |
|---|---|
| 46.3 mm | 49430 Hz |
| 60.4 mm | 65907 Hz |
| 90.6 mm | 98860 Hz |
| 135.8 mm | 148290 Hz |
| 181.1 mm | 197720 Hz |
| 229.4 mm | 250500 Hz |

In Fig. 6.14, the measurement shows good agreement in the main lobe relative to the simulations. Regarding the first side lobe, there is a magnitude difference between the negative and positive degrees where the negative is greater than the positive degree. Furthermore, this magnitude difference is shifted for the second side lobe, where the positive is greater than the negative degree. This directivity behavior is one of the reasons for the method in Sect. 4.5, were developed. This was to study if this behavior was due to the disk's positioning relative to the microphone when the R-stage rotates. But this behavior is not absent in the developed method in Sect. 4.5, which leads to the thought that this occurs due to the disk not being perfectly axisymmetric. This discrepancy may be due to the wires soldered to the electrodes and causes anti-axisymmetric effects, damage in the polarization after soldering on the electrodes, or some aging or structural defects.

In Fig 6.15, the measurement shows similar behaviors as Fig. 6.14, but both first side lobes are larger in magnitude relative to the simulation. When it comes to the second and third side lobes, they seem to melt into each other, especially seen on the negative angle side.

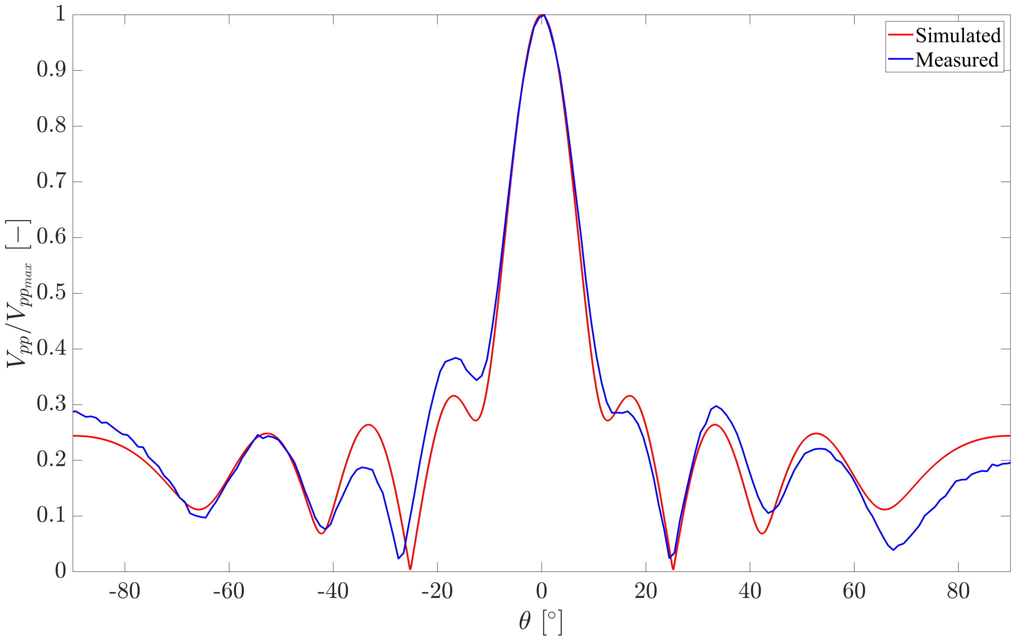



(a)

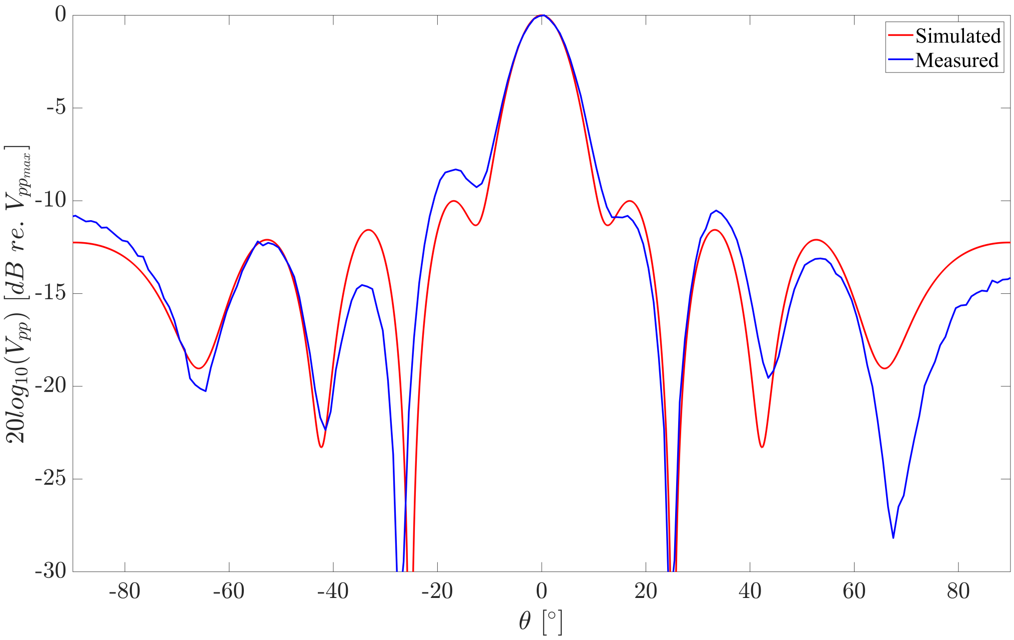



(b)

Figure 6.13: The directivity is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity frequency is 49430 Hz for measurement and 49050 Hz for simulation, both conducted at a distance of 0.2 m. (T = 25.2 °C, RH = 32.6 %, P = 990 hPa).

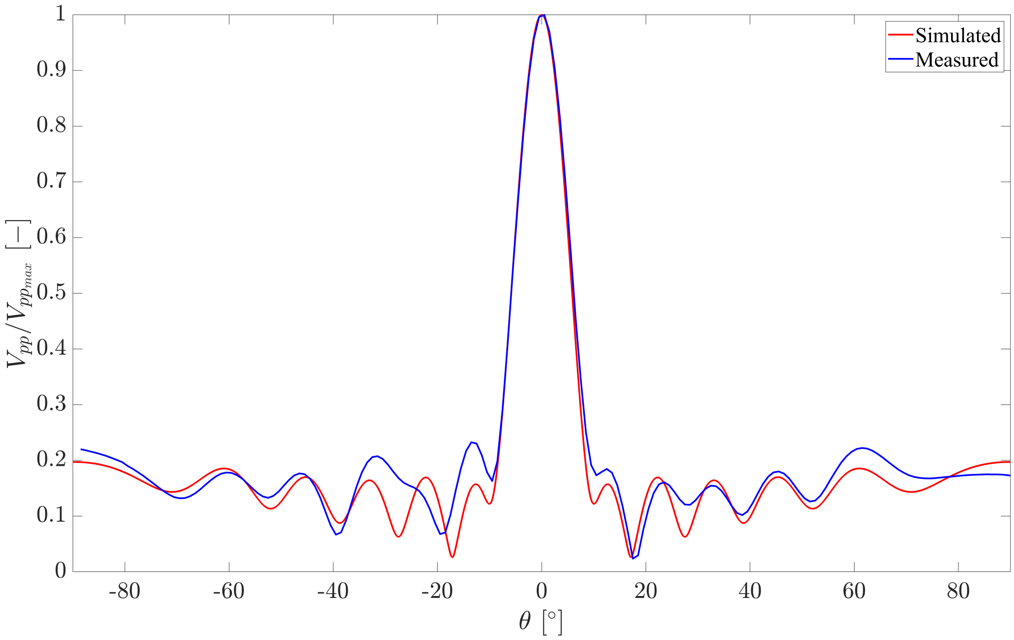



(a)

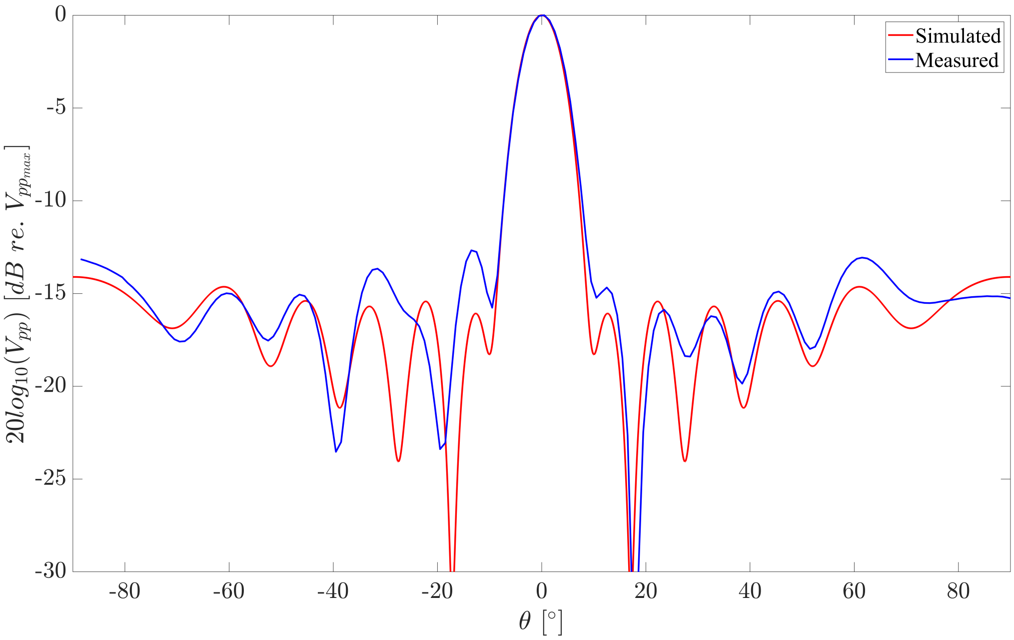



(b)

Figure 6.14: The directivity is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity frequency is 65907 Hz for measurement and 65400 Hz for simulation, both conducted at a distance of 0.2 m. (T = 25.3 °C, RH = 35.3 %, P = 990 hPa).

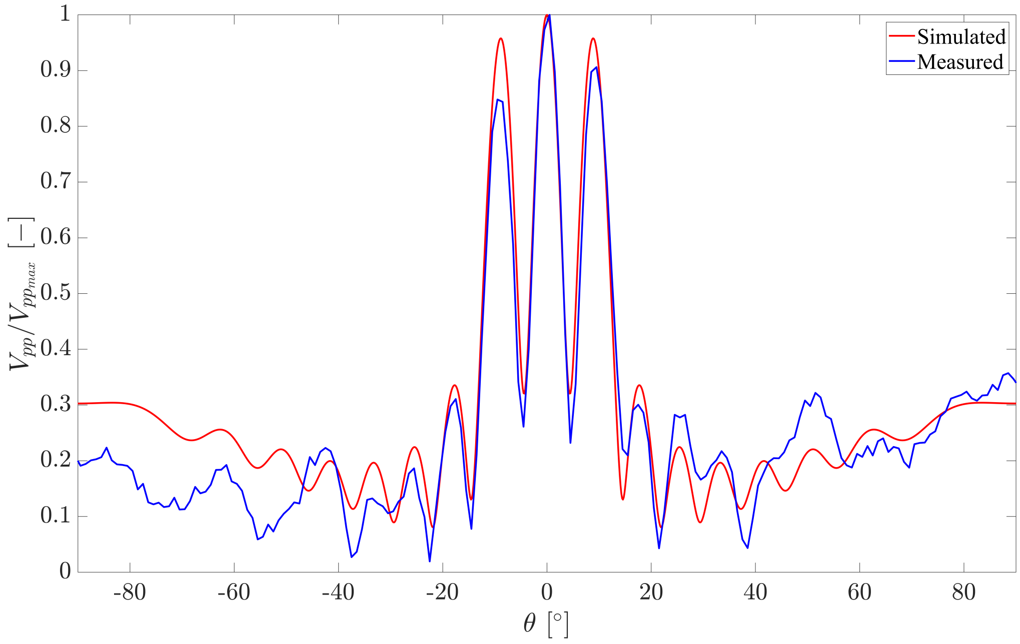



(a)

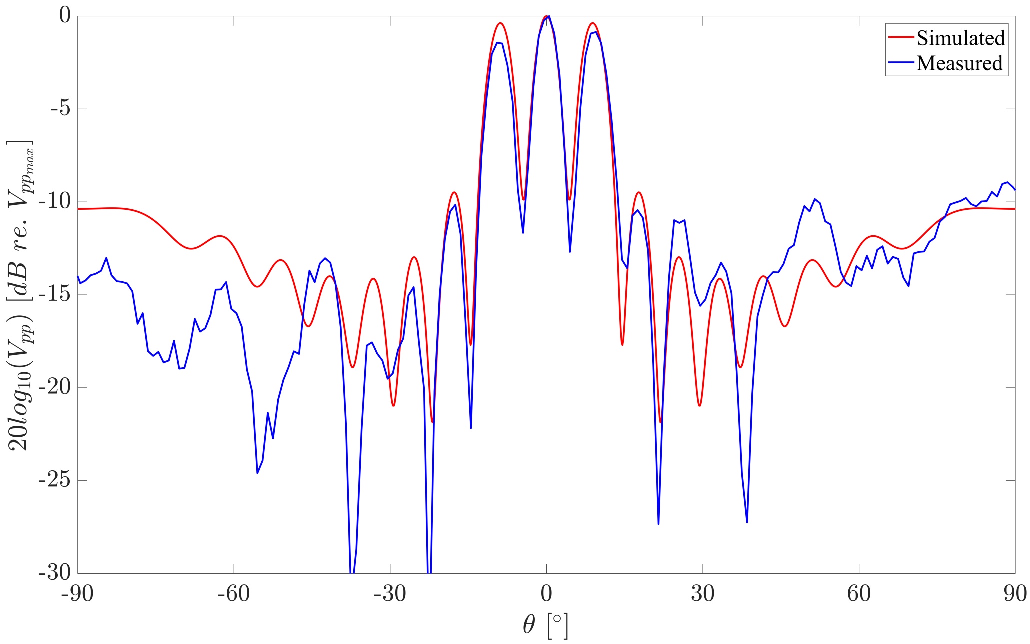



(b)

Figure 6.15: The directivity is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity frequency is 98860 Hz for measurement and 98100 Hz for simulation, both conducted at a distance of 0.2 m. (T = 25.1 °C, RH = 33.4 %, P = 991 hPa).

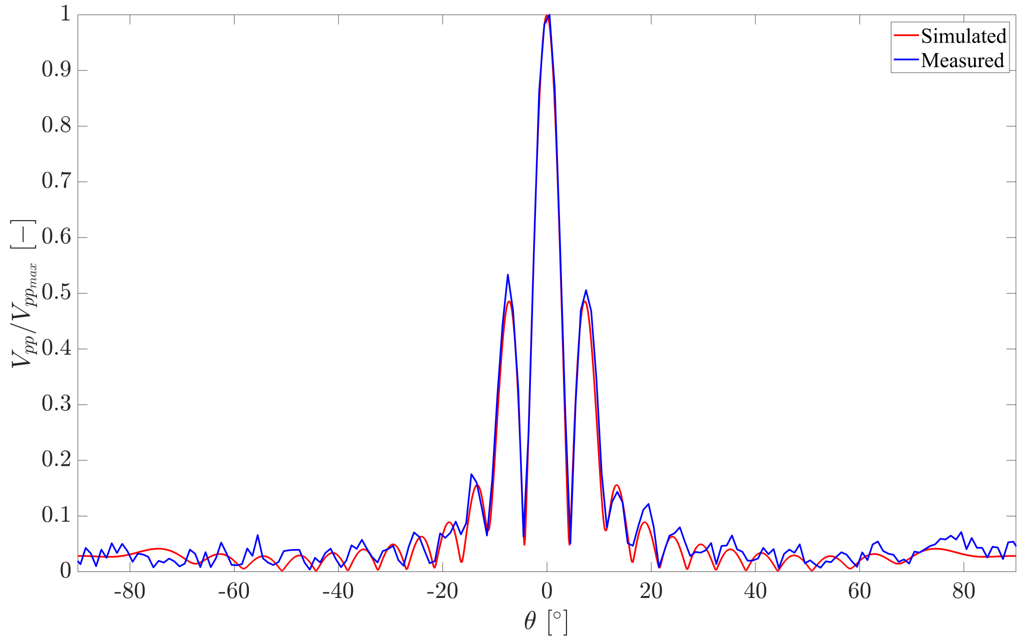



(a)

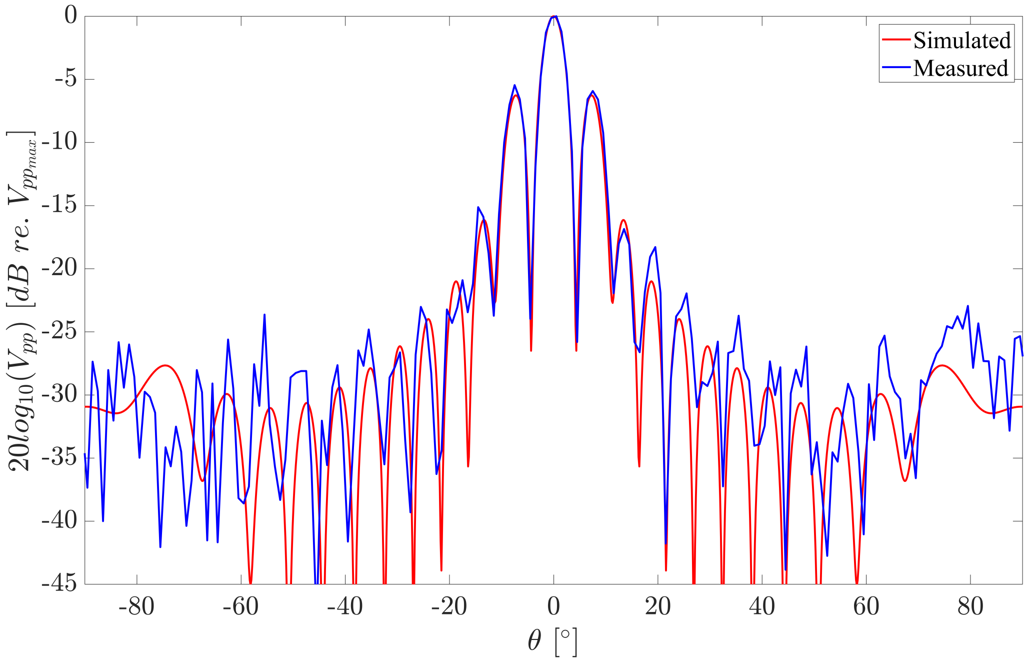



(b)

Figure 6.16: The directivity is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity frequency is 148290 Hz for measurement and 147150 Hz for simulation, both conducted at a distance of 0.2 m. (T = 25.3 °C, RH = 36.3 %, P = 990 hPa).

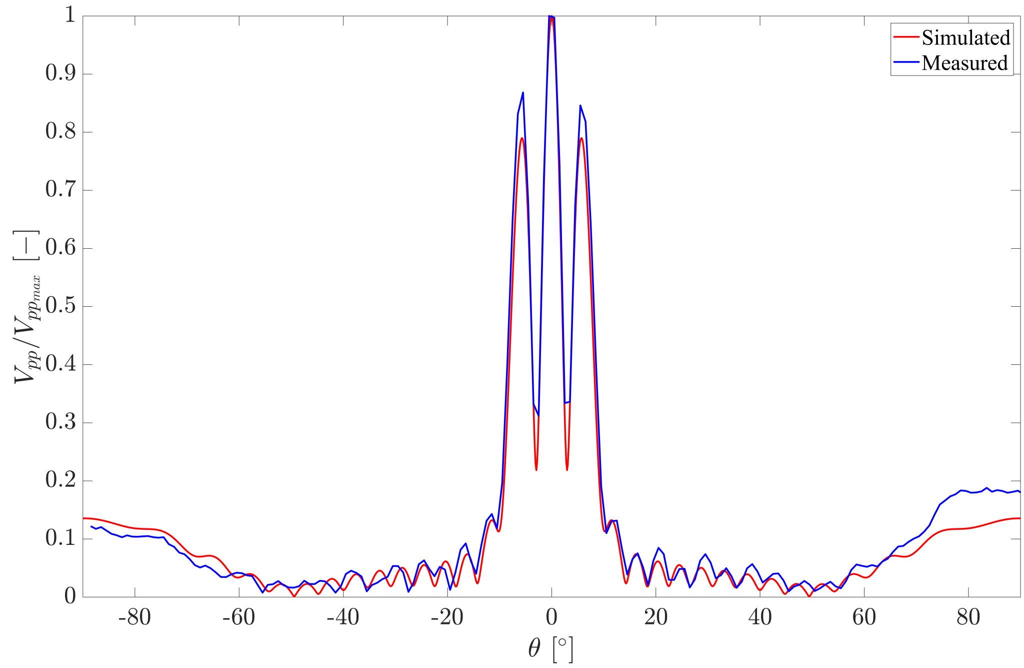



(a)

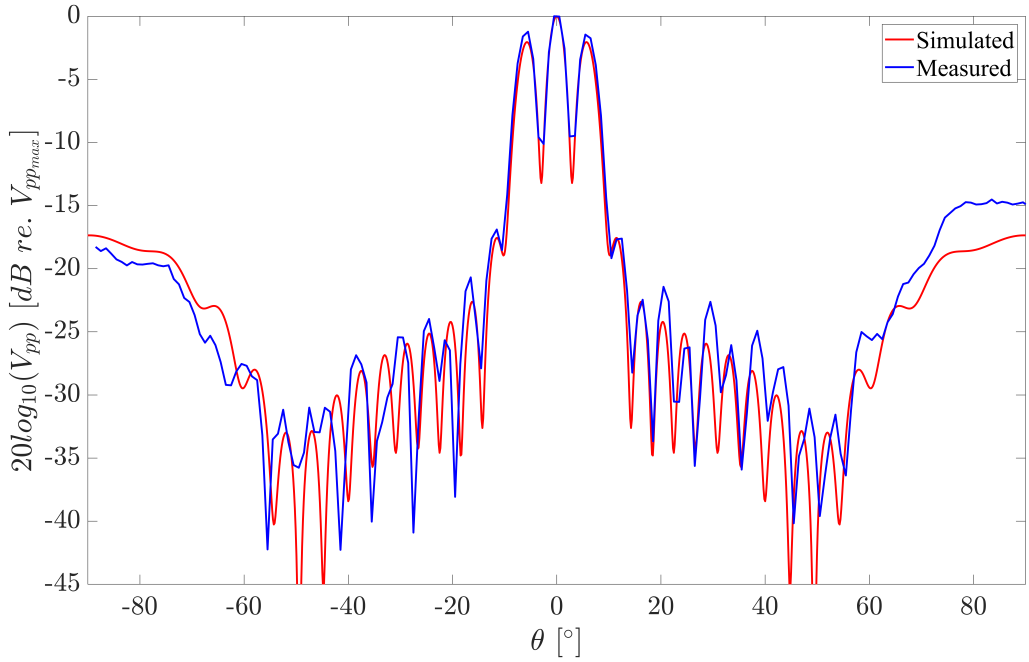



(b)

Figure 6.17: The directivity is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity frequency is 197720 Hz for measurement and 196200 Hz for simulation, both conducted at a distance of 0.2 m. (T = 25.4 °C, RH = 37.8 %, P = 991 hPa).




(a)




(b)

Figure 6.18: The directivity is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity frequency is 250500 Hz for measurement and 249050 Hz for simulation, both conducted at a distance of 0.2 m. (T = 26.3 °C, RH = 40.2 %, P = 1007 hPa).

In Fig 6.16, the measurement shows good agreement in the main lobe relative to the simulations and, to some degree, good agreement in the first and second side lobes. Beyond the second side lobes, there seems to be relatively much noise present in measurement, which seems to be true for the measurement signal as well, see Figs. 6.10b and 6.10c. It is worth noting that it is used a relatively low signal generator output voltage, $V_{0_{pp}}$, which is 1 V. With this low voltage and compared to Fig. 1.1, measured by [48], it can be seen that the transmitting voltage response $S_V$ of this type of piezoelectric disk for this particular frequency range is relatively low. This low $S_V$ impacts the transmitting efficiency, which most likely is the cause of much noise being present in the measurement due to bad SNR. Since this is not at a resonance frequency, the measuring voltage should and can be much higher and still not cause a non-linear effect in the piezoelectric disk.

In Fig 6.17, the measurement shows good agreement in the main, first and second side lobes. Beyond the second side lobes, there seems to be relatively much noise present in measurement, which is true for the measurement signal as well, see Figs. 6.11b and 6.11c. In the first side lobe, effects due to tilting $\theta_T$ or angle $\theta_M$ from the model in Sect. 4.5 can be seen as being present in the measurement. These effects can be due to $\theta_T$ or $\theta_M$ exceeding 0.5 degrees.

For the last directivity measurement seen in Fig. 6.18, the measurement results are similar to the result in Fig. 6.17, but with less noise.

### 6.2.4 SNR of directivity measurements

In Fig 6.19, the SNR of directivity measurements is presented for all measurement frequencies in Tabel 6.3 and at the measurement distance of 0.2 m between the piezoelectric disk and microphone. Generally, a good SNR magnitude is 20 dB when the measurement voltage $V_{5_{rms}}$ is 10 times greater than the rms noise voltage $V_{rms}^{noise}$. The highest SNR magnitude is 63 dB and is related to the frequency 98860 Hz (the first radial mode). The lowest SNR magnitude is -25.4 dB and is related to the frequency 197720 Hz. With an SNR less than 0 dB, the rms noise voltage $V_{rms}^{noise}$ is larger than the measurement voltage $V_{5_{rms}}$.

For the directivity measurement with a frequency of 49430 Hz, the SNR magnitude of the main lobe stays above 20 dB. The SNR magnitude starts to drop below 20 dB when the angle passes 30 degrees. When the SNR drops below 20 dB, noise starts appearing in the measurements, as seen in Fig. 6.13.

For the directivity measurement with a frequency of 65907 Hz, the SNR magnitude is generally above 20 dB, except for node points of the directivity beam where the SNR magnitude goes below 20 dB. This measurement is generally stable, as seen in Fig. 6.14.

For the directivity measurement with a frequency of 98860 Hz, the SNR magnitude stays well above 40dB, except for the node points of the directivity beam at approx. 20 degrees where the SNR magnitude goes below 40 dB. This measurement is stable, as seen in Fig. 6.15.

For the directivity measurement with a frequency of 148290 Hz, the SNR magnitude is below 20 dB. At some point, the $V_{rms}^{noise}$ is calculated to be larger than the measurement voltage $V_{5_{rms}}$. This measurement is generally unstable from approx 20 degrees, as seen in Fig. 6.16.

For the directivity measurement with a frequency of 197720 Hz, the SNR magnitude is generally below 20 dB, and for all angles larger than 20 degrees, the $V_{rms}^{noise}$ is calculated to be larger than the measurement voltage $V_{5_{rms}}$. This measurement is generally unstable from approx 15 degrees, but appear to follow the simulation even if there is noise visibly present, as seen in Fig. 6.17.

For the last directivity measurement with a frequency of 250500 Hz, the main, first, and second side lobes stay above the SNR magnitude of 20 dB. The angles between approximately 15 to 60 degrees remain between the SNR magnitude of 0 dB and 20 dB. For the angles increasing past 60 degrees, the SNR magnitude is larger than 20 dB. Between 0 dB and 20 dB, the measurements may appear to follow simulation to some degree in Fig. 6.18, but noise is visibly present.

Measurements of the directivity beam pattern for different frequencies show relatively good agreements with the simulations. However, the SNR is quite low and negative for some of the frequencies used, especially for the measurement frequencies 148290 Hz and 197720 Hz. This can be due to the low signal generator peak-to-peak voltage equal to 1 $V_{0_{pp}}$. If the transmitting voltage response $S_V$ information in Fig. 1.1 had been considered when conducting measurements, the signal generator voltage would have been increased for frequencies outside the first and second radial modes. This is due to work done by Mosland [48], which shows that high signal generator voltages at the first and second radial modes give non-linear effects at these frequencies. Therefore the work conducted by [48] did measure $S_V$ with two different signal generator voltages, which are $V_{0_{pp}}$ = 20 V outside resonance and $V_{0_{pp}}$ = 2 V at resonance to avoid non-linear effects. If these measurements conducted in this work had used the same voltages as [48], it would most likely increase the SNR ratio giving better measurement results.

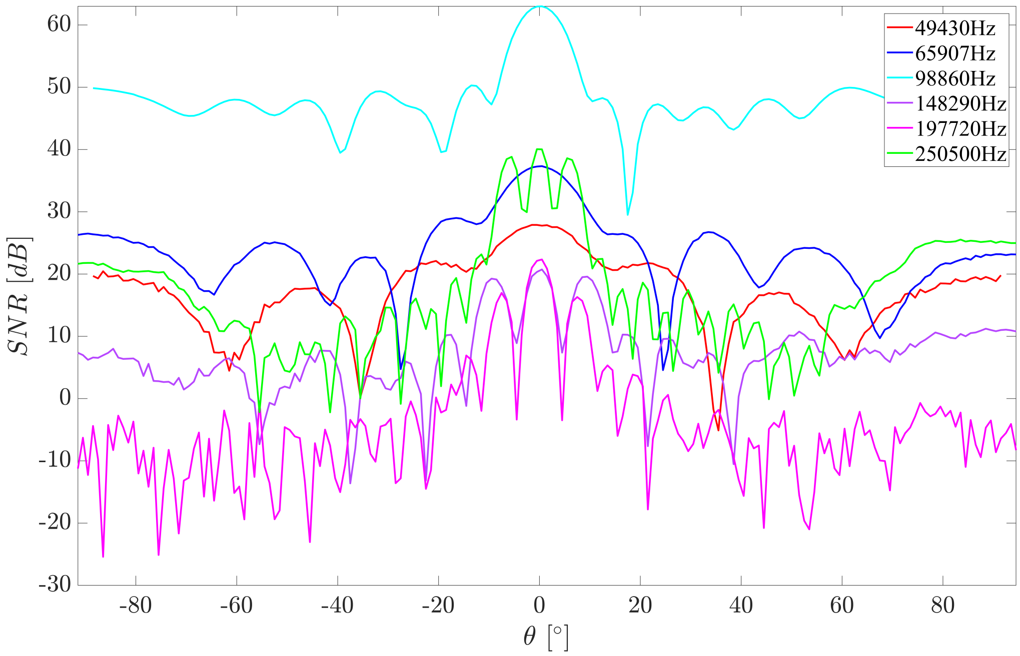
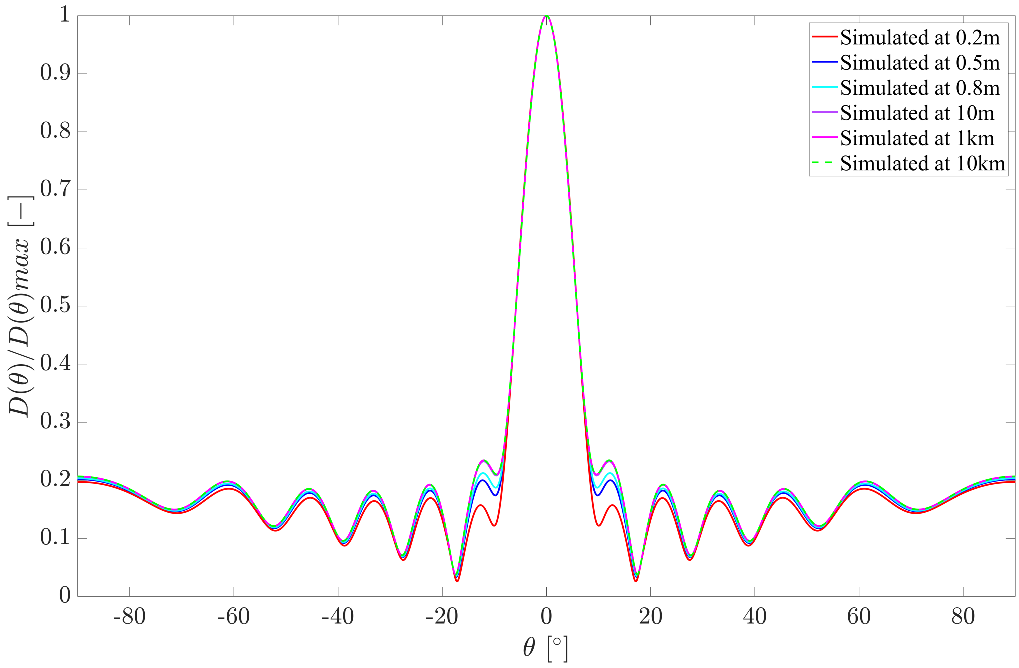



Figure 6.19: Signal-to-noise ratio of all directivity measurements at the distance of 0.2 m.

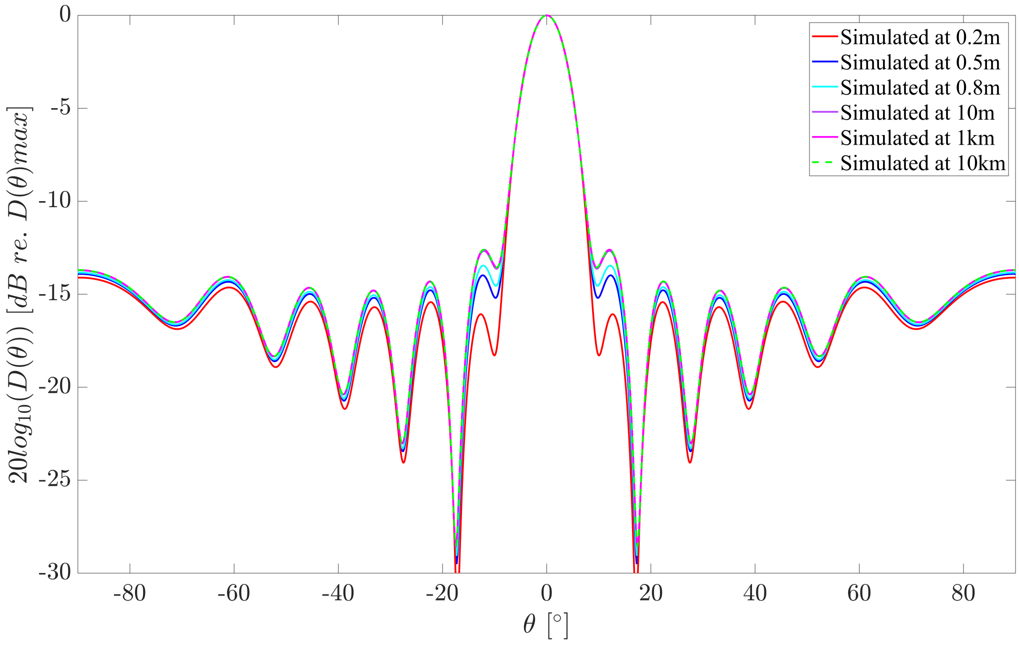
### 6.2.5 Comparison of directivity at different z distances of FE simulations and measurements

In Fig. 6.20 it is studied the change in the FE directivity simulations as a function of increasing distance z. This is to investigate how increasing the distance affects the directivity and investigate how large the distance must be before the directivity stabilizes. The same study is performed for directivity measurements, but not to the same extent since it is not measured further out than to z = 0.8 m, see Fig 6.21.

For the FE directivity simulations, shown in Fig. 6.20, it can be seen that the amplitude of the side lobes of the directivity changes with the distance. The amplitude difference is most noticeable for simulations that have not been performed far enough into the far field, where the far field distance for this frequency is 90.6 mm, see Tabel 6.4. The further into the distant far field that simulations are performed, the more stable the directivity becomes. After 10 m into the far field, there is minimal change up to 10 km and can be seen as they overlap in Fig. 6.20. The directivity measurements at the distances 0.2 m, 0.5 m, and 0.8 m, shown in Fig. 6.21, and the FE directivity simulations at 0.2 m, 0.5 m, and 0.8 m, shown in Fig. 6.20, do not share the same effects. For the directivity measurements in Fig. 6.21, the largest difference in the first side lobe is between 0.2 m and 0.8 m on the positive angle side and is 1.38 dB. Compare this magnitude to the FE directivity simulations first side lobe between 0.2 m and 0.8 m, which is 2.61 dB. The directivity measurement side lobes in Fig. 6.21 vary for which distance is greatest in magnitude.

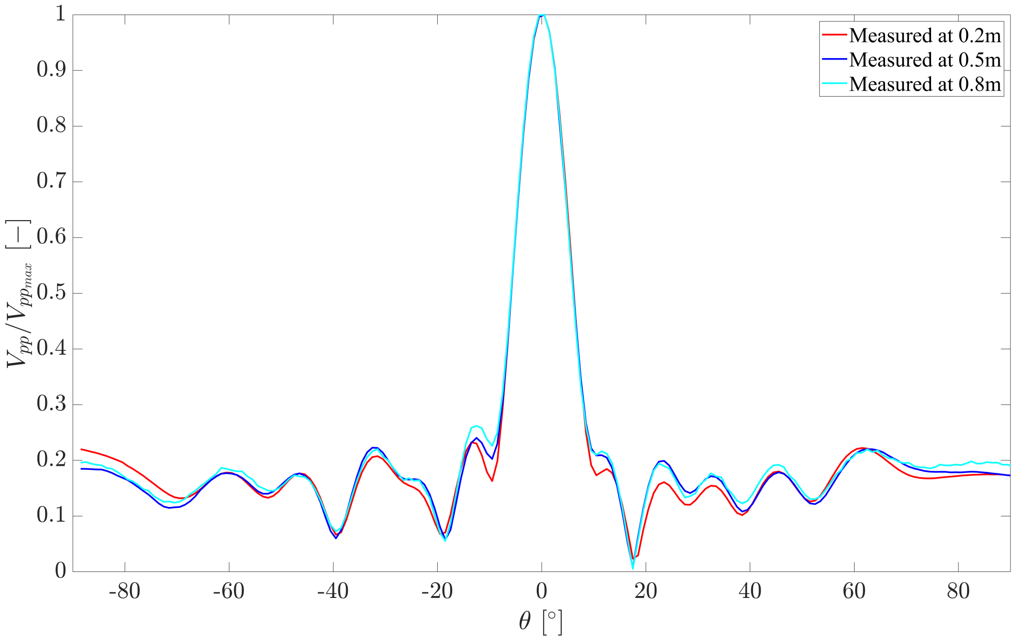



(a)

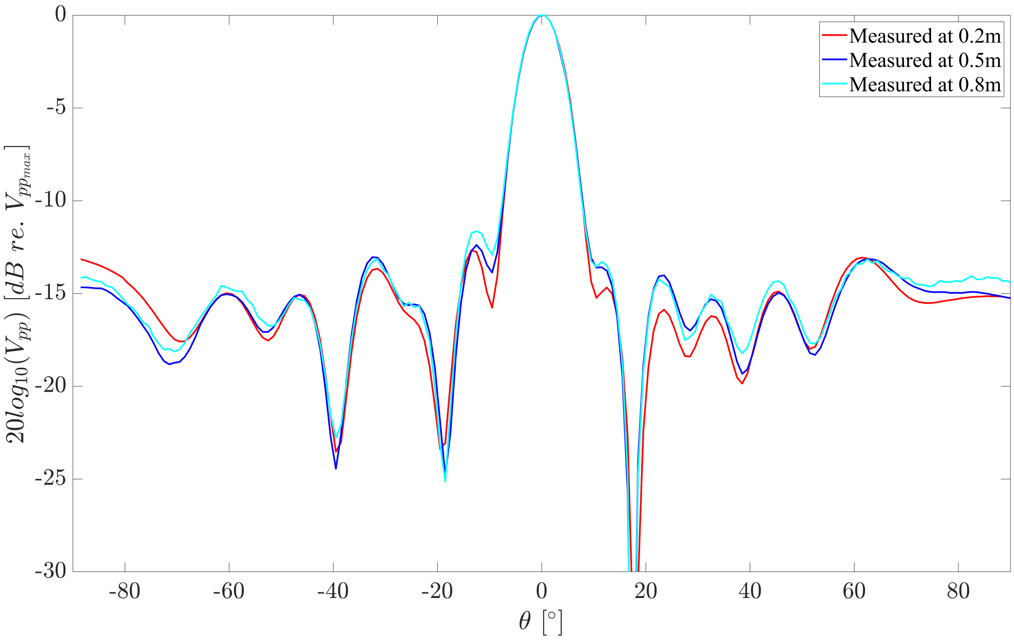



(b)

Figure 6.20: The FE directivity simulations is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity simulation frequency is 98100 Hz and conducted at a distance of 0.2 m, 0.5 m, 0.8 m, 10 m, 1 km, and 10 km.




(a)




(b)

Figure 6.21: The directivity measurements is given linearly (a) and logarithmically (b), and at the angles from -90 to 90 degrees with one-degree resolution. Directivity measurement frequency is 98860 Hz and conducted at a distance of 0.2 m, 0.5 m, and 0.8 m. (T = 25.2 °C, RH = 32.6 %, P = 990 hPa).

### 6.2.6 Comparison of on-axis pressure between FE simulations and measurements

The on-axis pressure of the piezoelectric disk is measured with frequencies given in Table 6.3, excluding 250500 Hz, and at the $z$ distance from 0.1 mm to 0.3 m. The spatial resolutions of the $z$ distance vary with increasing distance and are given in Table 6.5. With increasing $z$ distance, the spatial resolution decreases when it approaches far field, and the on-axis pressure approaches $1/z$ dependency. The resolution is more significant in the near field due to several pressure nodes and maximums that need to be documented. If the spatial resolution is too low in the near field, it is easy to miss relevant pressure changes. All on-axis pressure measurements and simulations are plotted as pressure amplitude vs. distance $z$ and sound pressure level (SPL) vs. distance $z$.

Table 6.5: Spatial resolution of on-axis pressure measurements from 0.1-300 mm.

| Start of Interval | Spatial resolution | Stop of Interval | Unit |
|---:|---:|---:|---|
| 0.1 | 0.1 | 30 | [mm] |
| 31 | 1 | 100 | [mm] |
| 105 | 5 | 300 | [mm] |

In Fig. 6.22, the on-axis pressure measurement shows good agreement with the FE simulation. It is seen some ripples in the measurements with the increasing $z$ distance. The distance between each maximum of the ripples is approximately 3.2-3.6 mm. The ripples slowly die out and can't be seen anymore from about 0.1 m. For the pressure node, the simulation pressure is higher than for measurement. The $1/z$ dependency shows good agreement with measurement until at around 0.15 m.

In Fig 6.23, the on-axis pressure measurement shows good agreement with the FE simulation. As for earlier on-axis pressure measurement, this also contains ripples with the increasing $z$ distance. The distance between each maximum of the ripples is approximately 2.2-2.7 mm. The ripples slowly die out and can't be seen anymore from about 0.1 m. The simulation pressure is higher for the first pressure maximum and node than for measurement. At the last pressure maximum, the measurement pressure is higher than for simulation. The $1/z$ dependency shows excellent agreement with measurement.

In Fig 6.24, the on-axis pressure measurement shows good agreement with the FE simulation but starts to deviate when $z$ becomes less than 0.0145 m. As for all earlier on-axis pressure measurements, this also contains ripples with the increasing $z$ distance. The distance between each maximum of the ripples is approximately 1.6-1.8 mm. The ripples slowly die out and can't be seen anymore from about 0.075 m. At the last pressure maximum, the simulation pressure is higher than for the measurement and continues to be higher with increasing distance $z$. The $1/z$ dependency shows good agreement with measurement and starts to deviate in the near field at approximately 0.075 m.

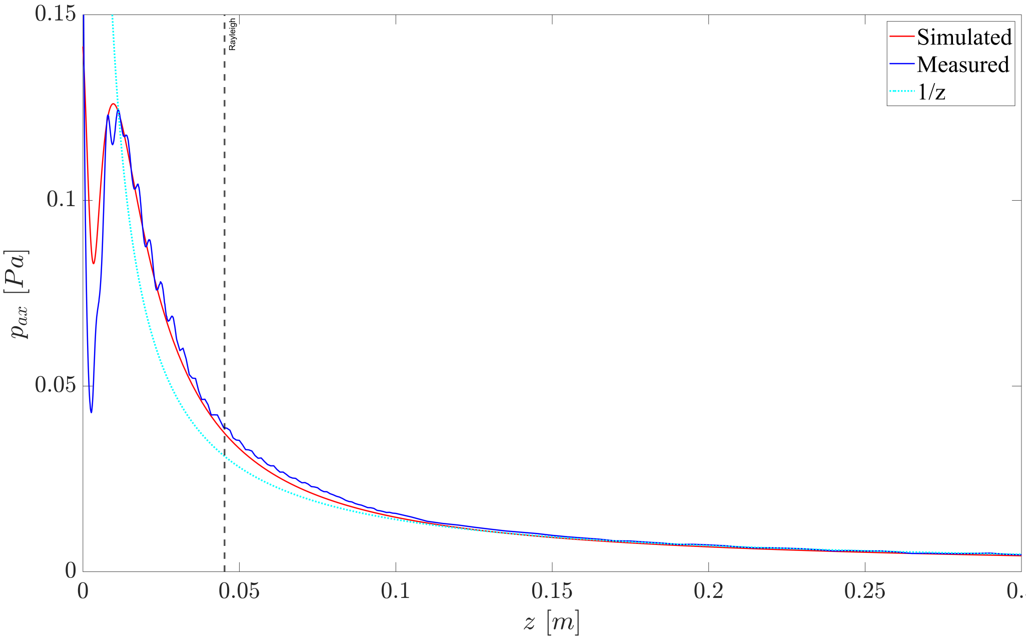
In Fig. 6.25, the on-axis pressure measurement shows some agreement with the FE simulation, but a significant deviation can be seen from the last pressure maximum. As for all earlier on-axis pressure measurements, this also contains ripples, but the ripples are more significant for this frequency than all earlier measurements. The distance between each maximum of the ripples is approximately 1.1-1.2 mm, and the ripples die abruptly out at about 0.03 m. The $1/z$ dependency shows some agreement with measurement in the far field. For the pressure nodes and maximums, it can be seen that they are shifted a little to the left.

In Fig. 6.26, the on-axis pressure measurement shows good agreement with the FE simulation, but a significant deviation can be seen. For the last maximum pressure, the measurements deviate by about 30 % less than for simulated pressure. As for all earlier on-axis pressure measurements, this also contains ripples, and they abruptly die out at about 0.029 m. The distance between each maximum of the ripples is approximately 0.8-0.9 mm. The pressure nodes and maximums show good agreement with the FE simulation but with lesser pressure amplitude.

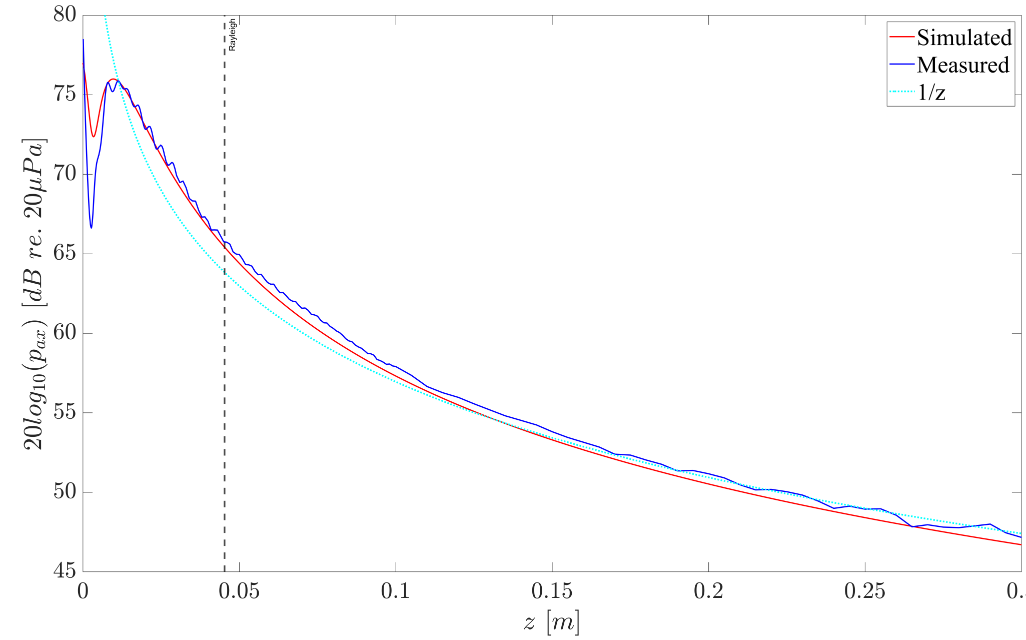
By comparing all distances between the ripple maximums with the wavelength in Table 6.6, it can be seen that all wavelengths divided by two lies within the distance ranges of the ripple maximums with the corresponding frequency. This implies that these ripples are probably caused by standing waves between the microphone and the piezoelectric disk. The pressure shape for all measurements seems to agree well with simulations when one disregards the magnitude. All on-axis pressure plots also have a $1/z$ dependency, which usually happens in the far field at around Rayleigh distance, where pressure amplitude decreases with the inverse of the increasing $z$ distance. For Fig 6.22, 6.25, and 6.26, there also seems to be some noise present at around 0.15 m and continues with increasing distance $z$.

Table 6.6: Signal length and wavelength $\lambda$ for the different frequencies used in on-axis measurements. It is used 60-cycle sine burst, and the sound speed c = 343 m/s.

| Frequency | Signal Length | $\lambda$ |
|---|---|---|
| 49430 Hz | 416.3 mm | 6.939 mm |
| 65907 Hz | 312.3 mm | 5.204 mm |
| 98860 Hz | 208.2 mm | 3.470 mm |
| 148290 Hz | 138.8 mm | 2.313 mm |
| 197720 Hz | 104.1 mm | 1.735 mm |

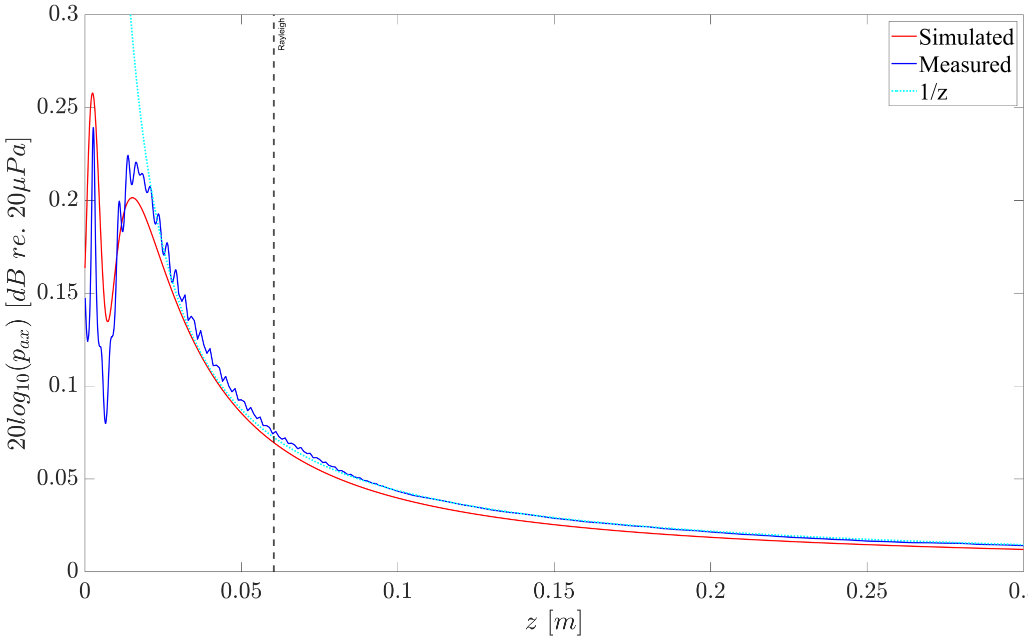



(a)

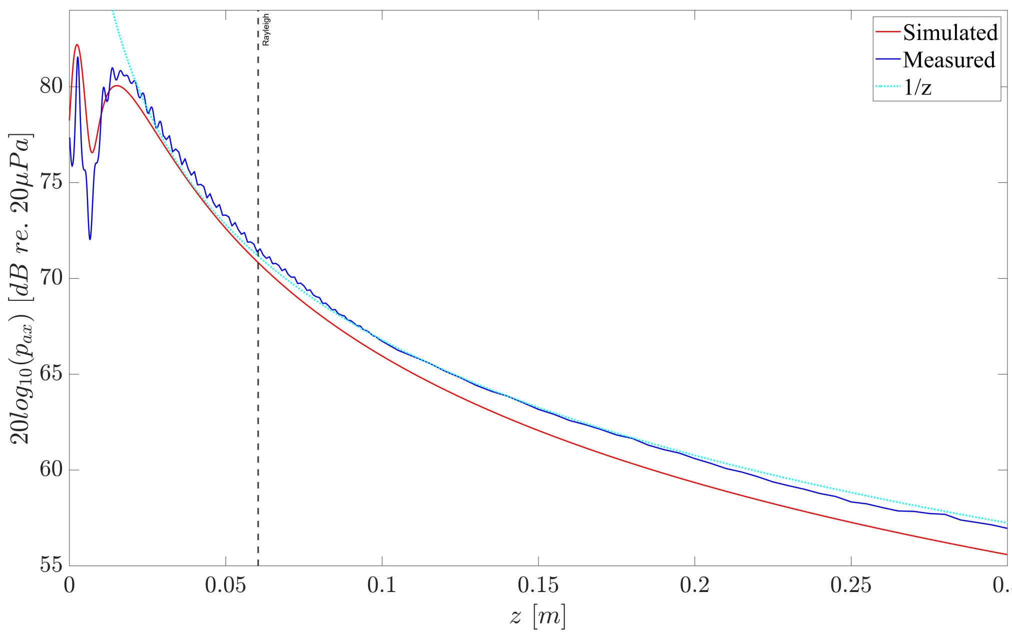



(b)

Figure 6.22: On-axis pressure given in pressure amplitude (a) and SPL (b), from 0.1 mm to 0.3 m with decreasing spatial resolution with increasing distance $z$. On-axis pressure frequency is 49430 Hz for measurement and 49050 Hz for simulation. The Black stippled line represents Rayleigh distance, and turquoise represents $1/z$ dependency. (T = 26.2 °C, RH = 31.7 %, P = 1010 hPa).

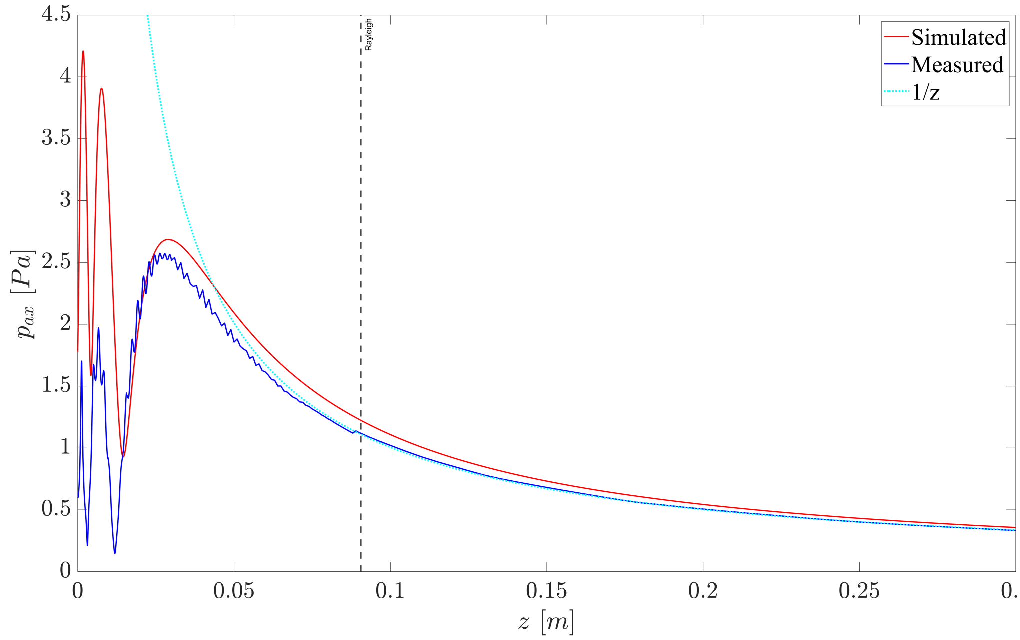



(a)

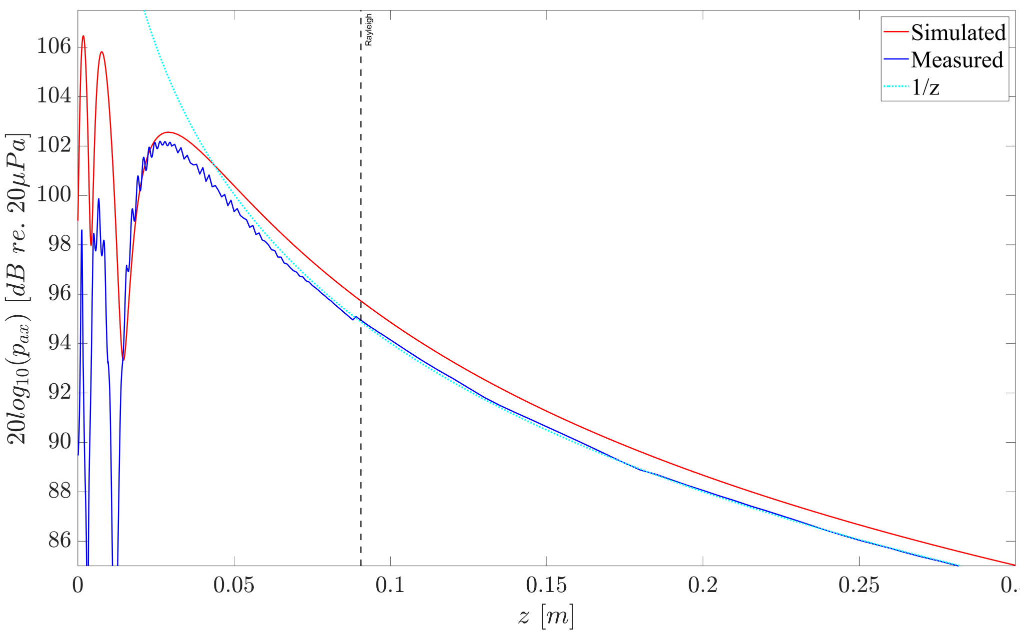



(b)

Figure 6.23: On-axis pressure given in pressure amplitude (a) and SPL (b), from 0.1 mm to 0.3 m with decreasing spatial resolution with increasing distance $z$. On-axis pressure frequency is 65907 Hz for measurement and 65400 Hz for simulation. The Black stippled line represents Rayleigh distance, and turquoise represents $1/z$ dependency. (T = 25.1 °C, RH = 42.1 %, P = 999 hPa).

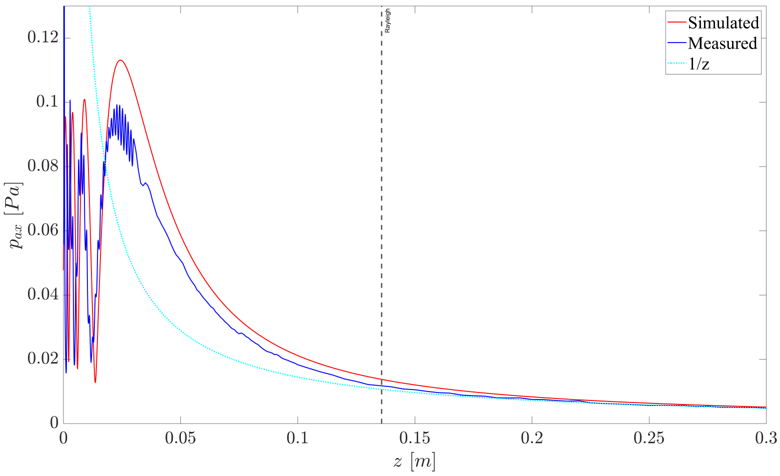



(a)

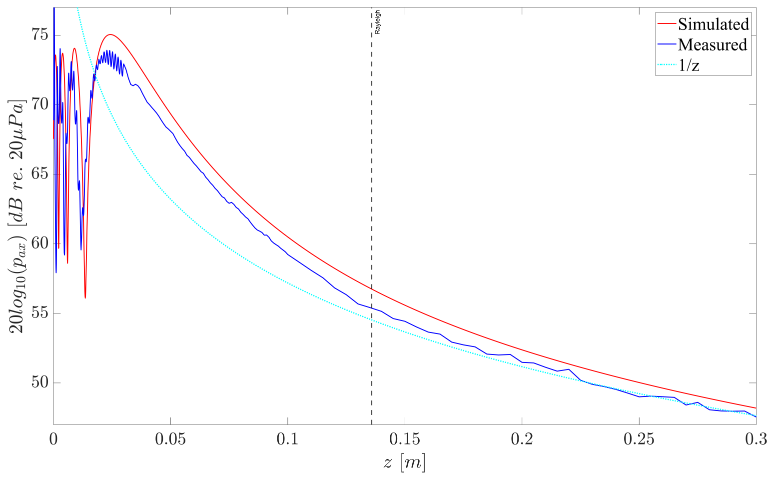



(b)

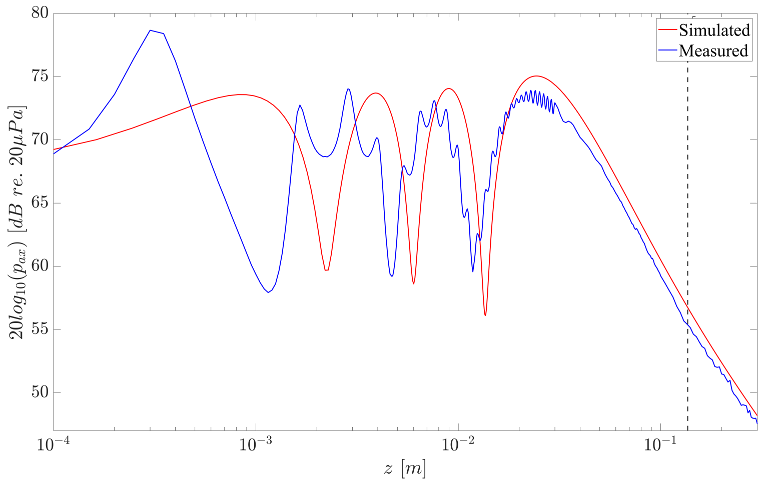
Figure 6.24: On-axis pressure given in pressure amplitude (a) and SPL (b), from 0.1 mm to 0.3 m with decreasing spatial resolution with increasing distance $z$. On-axis pressure frequency is 98860 Hz for measurement and 98100 Hz for simulation. The Black stippled line represents Rayleigh distance, and turquoise represents $1/z$ dependency. (T = 26.0 °C, RH = 35.0 %, P = 1012 hPa).




(a)

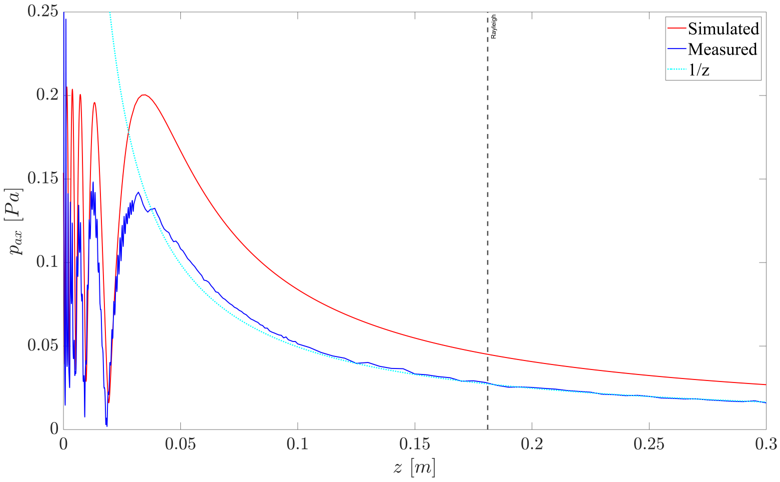



(b)

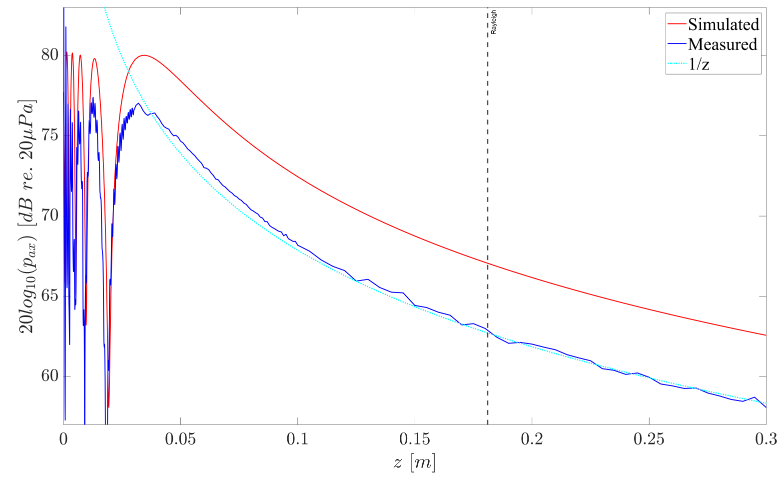



(c)

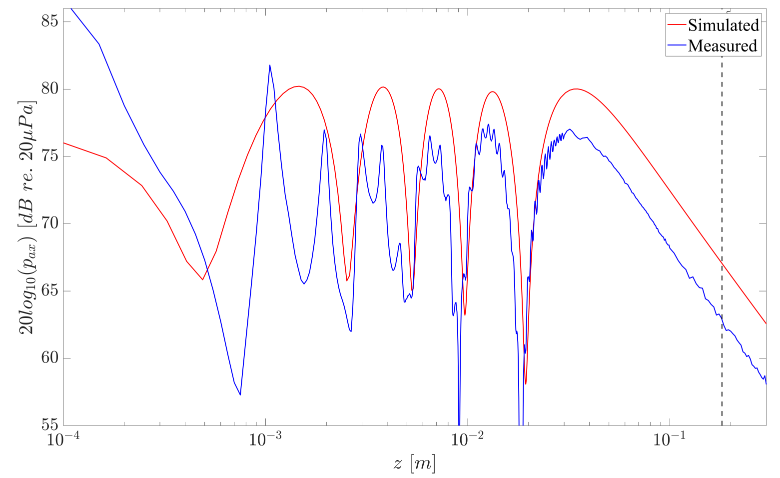
Figure 6.25: On-axis pressure given in pressure amplitude (a), SPL (b), and $z$ distance log scaled (c), from 0.1 mm to 0.3 m with decreasing spatial resolution with increasing distance $z$. On-axis pressure frequency is 148290 Hz for measurement and 147150 Hz for simulation. The Black stippled line represents Rayleigh distance, and turquoise represents $1/z$ dependency. (T = 25.2 °C, RH = 42.4 %, P = 998 hPa).




(a)




(b)




(c)

Figure 6.26: On-axis pressure given in pressure amplitude (a), SPL (b), and $z$ distance log scaled (c), from 0.1 mm to 0.3 m with decreasing spatial resolution with increasing distance $z$. On-axis pressure frequency is 197720 Hz for measurement and 196200 Hz for simulation. The Black stippled line represents Rayleigh distance, and turquoise represents $1/z$ dependency. (T = 24.5 °C, RH = 40.1 %, P = 1007 hPa).

### 6.2.7 SNR of on-axis pressure measurements

In Fig 6.27, the SNR of on-axis pressure measurements is presented for all measurement frequencies in Tabel 6.3, excluding 250500 Hz, and at the $z$ distance from 0.1 mm to 0.3 m. The on-axis pressure SNR is generally good for all frequencies and stays well above 20 dB. For the frequency 148290 Hz, the SNR measurement drops below 20 dB at about a z distance equal to 0.2 m. For the frequency 197720 Hz, the SNR measurement drops below 20 dB at about a z distance equal to 0.25 m.

These results in this work have not presented examples of the acoustic signal with increasing distance, such as it was presented for different angles in Sect. 6.2.2, due to lack of time. However, from the on-axis SNR in Fig 6.27, it can be assumed that with increasing $z$ distance, the SNR for different angles will be worsened relative to the SNR given in Fig 6.19.

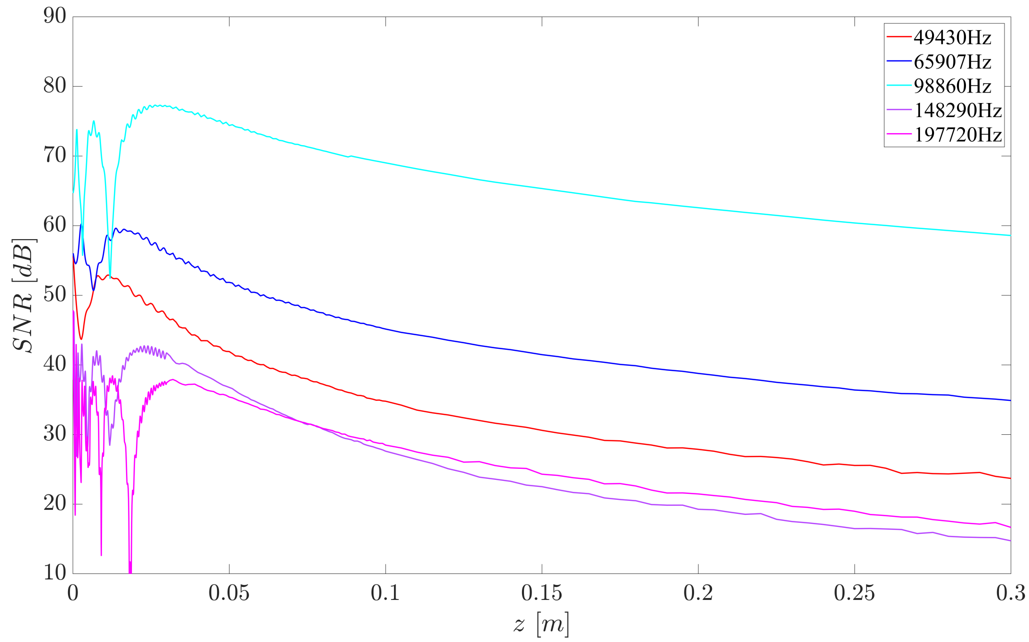



Figure 6.27: Signal to noise ratio of all measured on-axis pressure from 0.1 mm to 0.3 m. (T, RH, and P see individual measurements, Figs 6.22-6.26).

### 6.2.8 Comparison of 2-D sound pressure field between FE simulations and measurements

In this section, the 2-D sound pressure field for near and far fields is studied, and FE simulations are compared with measurements. Because the diameter of the piezoelectric disk is 10 mm in radius, the sound pressure field of the disk starts at a $z$ distance of 15 mm to avoid crashing with the microphone and extends as far as the $z$ distance of 300 mm. The measurements for each 2-D pressure field are composed of 9350 individual measurements for different positions and angles and performed with the frequencies given in Table 6.3, excluding 250500 Hz. The resolution of the angle stays constant throughout each measurement series and is set to one degree. The measurement is performed from -93 to 93 degrees for each measured 2-D pressure field with variating spatial resolution $z$ given in Table 6.7. The spatial resolution closest to the receiver provides the best spatial resolution, and with increasing distance $z$, the spatial resolution decreases.

Table 6.7: The $z$ distance spatial resolution of 2-D sound pressure field measurements from 15-300mm.

| Start Interval | Spatial resolution | Stop Interval | Unit |
|---:|---:|---:|---|
| 15 | 1 | 30 | [mm] |
| 35 | 5 | 100 | [mm] |
| 110 | 10 | 300 | [mm] |

The FE-simulated 2-D pressure field has no distance limitations in the near field, such as measurements performed with the piezoelectric disk. This means that simulations calculate the pressure from the structure's surface and out as far as 300 mm.

In Fig. 6.28, the measurement of the 2-D pressure field compared to the FE simulated pressure field shows good agreement. All lobes present in the measurement are present in the FE simulation.

In Fig. 6.29, the measurement of the 2-D pressure field compared to the FE simulated pressure field shows good agreement. All lobes present in the measurement are present in the FE simulation.

In Fig 6.30, the measurement of the 2-D pressure field compared to the FE simulated pressure field shows relatively good agreement. However, all lobes seen in the FE simulation are not as easy to identify in the measurement. The same effect in the 2-D pressure field measurement can be seen as for the directivity in Fig. 6.15, where some of the lobes discussed seem to melt into each other. This overlap between lobes mainly applies to the second and third side lobes for the negative angle side.

In Fig 6.31, the measurement of the 2-D pressure field compared to the FE simulated pressure field shows good agreement. However, it becomes more difficult to see the side lobes with the increasing angle, and some side lobes overlap. The difficulties occur probably due to low SNR shown in Fig. 6.19 and deviations seen in the directivity beam pattern from approximately 20 degrees in Fig. 6.16.

In Fig. 6.32, the measurement of the 2-D pressure field compared to the FE simulated pressure field shows reasonable agreement. However, it becomes more difficult to see the side lobes with the increasing angle, probably due to the low SNR shown in Fig. 6.19. Even though the SNR is low, it is possible to see that the side lobes are present. It is worth noting the one pressure node at zero degrees and from about 15-20 mm, see Fig 6.26, is documented in the measurement and agrees well with the FE simulation.

For all 2-D pressure field measurements seen in Fig. 6.28-6.32, the main lobe and first side lobes can be identified and correspond well to the FE simulations.

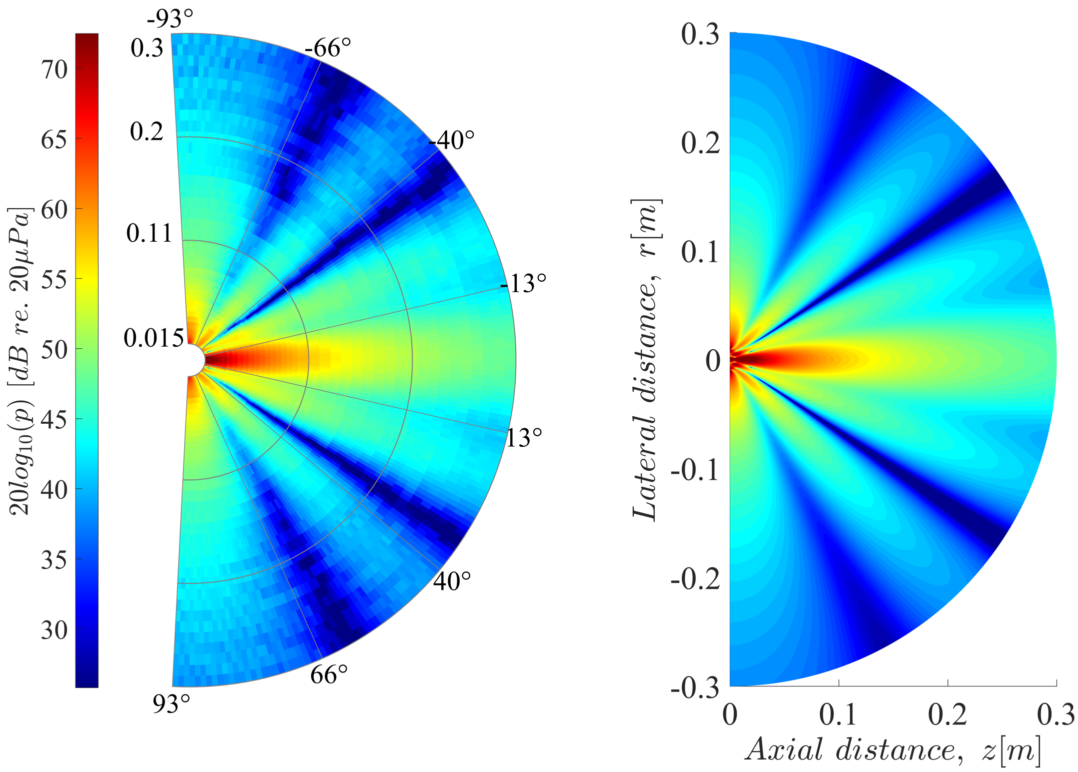



Figure 6.28: Measured (**left**) 2-D pressure field from 0.015-0.3 m with frequency 49430 Hz and simulated (**right**) from 0-0.3 m with frequency 49050 Hz with matching dB scale.

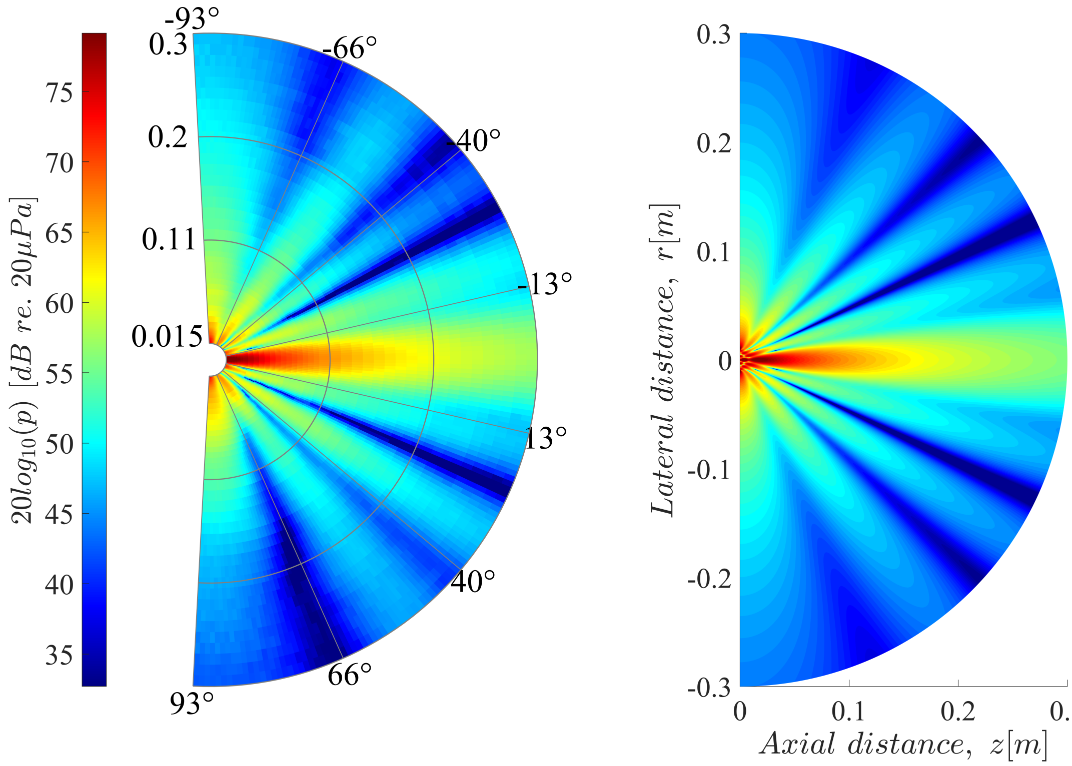



Figure 6.29: Measured (**left**) 2-D pressure field from 0.015-0.3 m with frequency 65907 Hz and simulated (**right**) from 0-0.3 m with frequency 65400 Hz with matching dB scale.

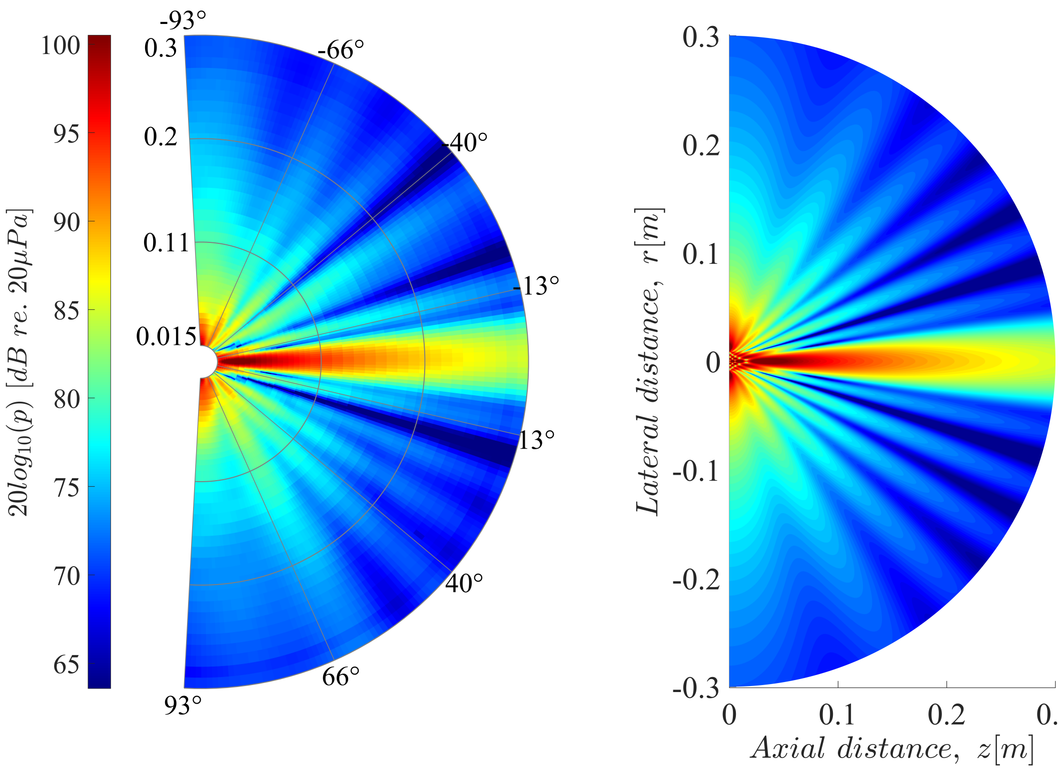



Figure 6.30: Measured (**left**) 2-D pressure field from 0.015-0.3 m with frequency 98860 Hz and simulated (**right**) from 0-0.3 m with frequency 98100 Hz with matching dB scale.

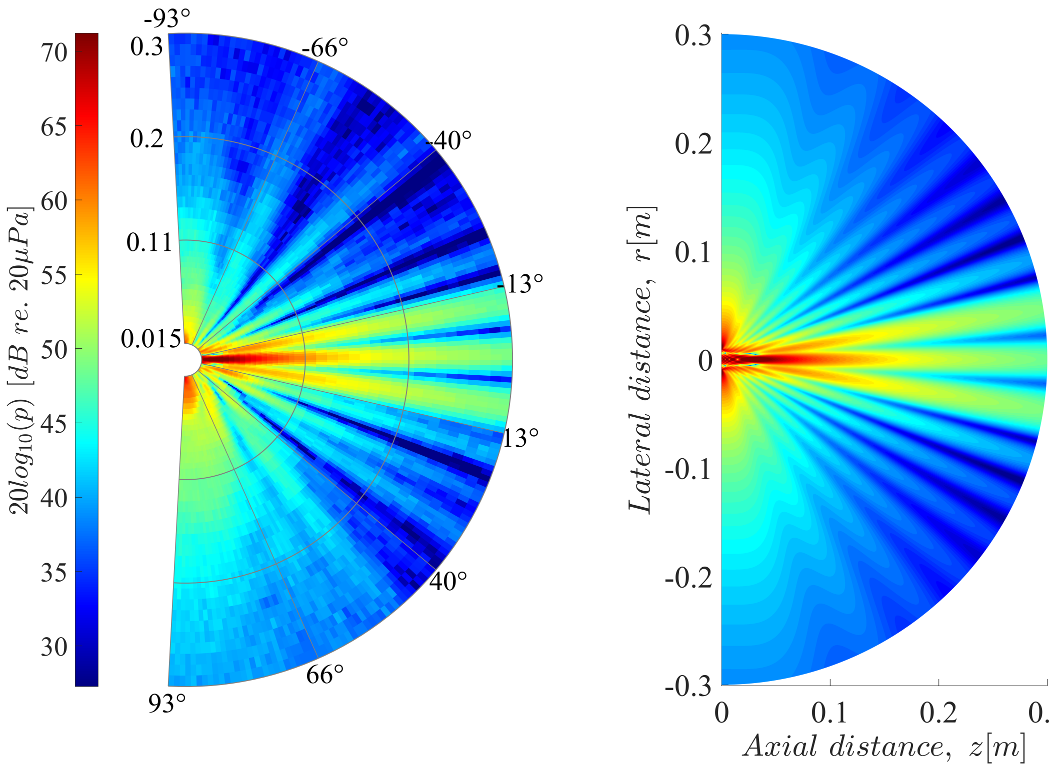



Figure 6.31: Measured (**left**) 2-D pressure field from 0.015-0.3 m with frequency 148290 Hz and simulated (**right**) from 0-0.3 m with frequency 147150 Hz with matching dB scale.

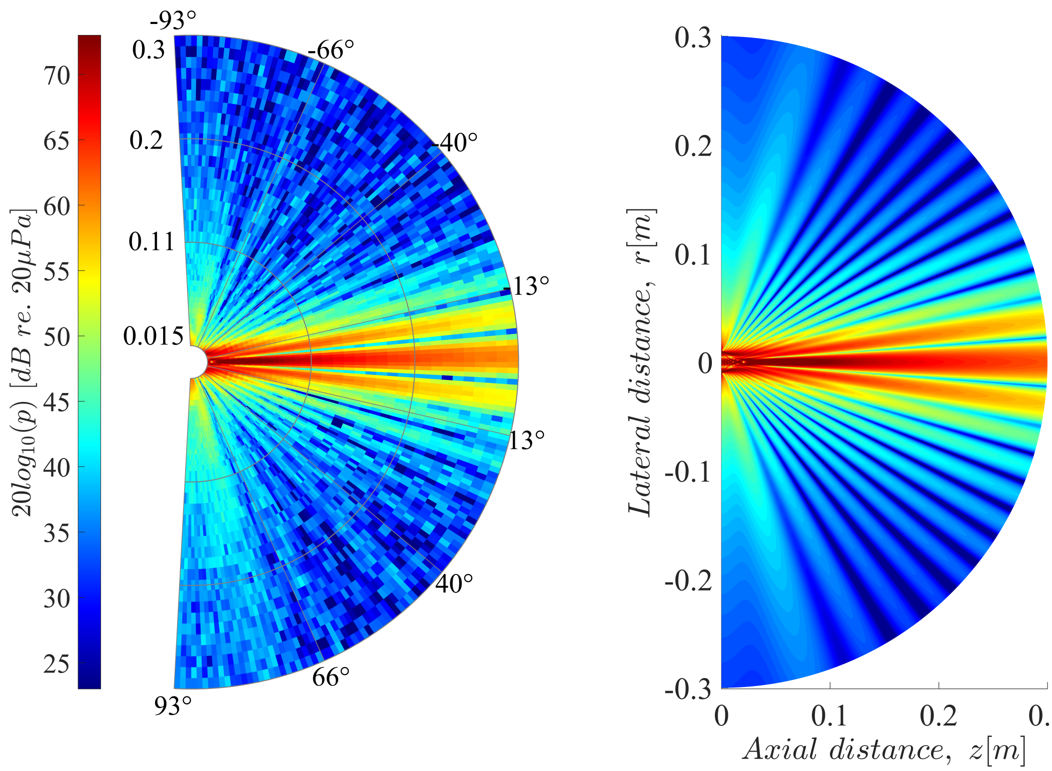



Figure 6.32: Measured (**left**) 2-D pressure field from 0.015-0.3 m with frequency 197720 Hz and simulated (**right**) from 0-0.3 m with frequency 196200 Hz with matching dB scale.

# Chapter 7

# Conclusions and further work

## 7.1 Conclusions

The measuring system, which consists of the piezoelectric disk Pz27, the receiver microphone Brüel & Kjær 4138, and the associated electronics for the measuring setup, has been studied when conducting measurements in the air in this work. All measurements performed with the measurement setup are compared with FE simulations. A MatLab app has also been created that has automated the measurement process, which has been very important to be able to carry out this work and will benefit everyone else who will continue to work with this measurement setup. Without automation, the large number of measurements performed in this work would not have been possible to conduct.

The electrical measurements and FE simulations gave good and comparable results. The radial extension modes R1 and R2 measurement performed on the wires soldered to the piezoelectric electrodes corresponded well with the FE simulation and deviated no more than 0.77 % and 0.58 % in frequency, respectively. The magnitude difference of the R1 and R2 modes deviated no more than 0.86 dB and -1.01 dB, respectively. Based on this, it can be said that the measurement and FE simulation of the electrical properties correspond well due to the good matching of the piezoelectric constants. However, there is room for improvements in the 110 kHz to 165 kHz range by adding solder lump in FE simulations to improve the results even more.

This work has studied two types of comparison between measurement and FE simulation of the directivity. The first type is the directivity measurements and FE simulations performed with a constant distance between the piezoelectric disk and microphone using different frequencies (Figs. 6.13 - 6.18). The second type is the directivity measurements and FE simulations performed with a constant frequency over several distances between the piezoelectric disk and microphone (Figs. 6.20 and 6.21). The first type of directivity measurement gave good results in all the main lobes, and, in general, the results are comparable to the FE simulations. After increasing the angle beyond the first side lobes, the measurements in Fig 6.13, 6.16 - 6.18 suffers from low SNR, which is concluded to be due to the low signal generator peak-to-peak voltage 1 V that was used when conducting the measurement. For all directivity measurements, asymmetry mirrored around the main lobe is present, which is concluded to be due to the solder lump on each side of the electrodes and a combination of not a perfect symmetry within the disk due to aging effects and changes in polarization after soldering. The second type of directivity measurement gave good results but not the same increase in the magnitude of the side lobes as for the FE simulations. The directivity measurements did more or less give similar results for all the different distances with small deviations in between the results. A conclusion about this behavior has not been drawn.

The number of measured pressure nodes agrees with simulations of the on-axis pressure. It is concluded that there are standing waves between the microphone and the piezoelectric disk in the on-axis measurements in Figs. 6.22 - 6.26, and these are difficult to avoid. It has been measured with a given number of pulse cycles that have been constant throughout all measurements performed in this work. This constant number of cycles has proven to be a good choice for achieving good measurement results but did cause problems with standing waves deep into the near field. The pressure amplitude differs in the near field regarding the measurement compared to FE simulation for the first radial mode, Fig. 6.24, and the cause is unknown. The larger deviation in pressure of the measurements compared to FE simulations in Figs 6.25 and 6.26 is concluded to be due to conducting measurements outside the Brüel & Kjær 1/8-inch pressure microphone's stated flat frequency response from 6.5 Hz to 140 kHz ±2 dB, the measurements results will entail larger uncertainties then expected. It can be concluded that $1/z$ dependency in the pressure is present for all on-axis measurements.

The 2-D pressure field measurements give good results compared to FE simulated 2-D pressure fields. However, it is clear that measurement and simulation results are not perfect matches, but it is still a substantial overlap between them. The most noticeable difference between measurement and simulation results is due to the SNR, as clearly seen in Figs. 6.31, 6.32, and noticeable in 6.28, and concluded to be due to the low signal generator peak-to-peak voltage of 1 V. For the 2-D pressure measurements in Figs. 6.30 and 6.31, it is clearly asymmetry mirrored around the main lobe is present, which is concluded to be due to the solder lump on each side of the electrodes and a combination of not a perfect symmetry within the disk due to aging effects and changes in polarization after soldering.

All measurement results conducted in this work and compared to FE simulation gave good agreements, and it can be concluded that using FE simulations is a good tool for approximating the admittance, directivity, on-axis pressure, and 2D pressure fields in the near and far fields.

## 7.2 Further work

Further work can be, exploring the effect of a solder lump on the edge of the electrodes of the piezoelectric disk with 3-D simulations in air to try recreating the asymmetric effects that occur in the present work.

Since it used a constant number of cycles throughout this work, further work can increase the number of cycles for frequencies higher than the first radial mode at 98860 Hz, e.g., 150 kHz to 90 cycles, 200 kHz to 120 cycles, and 250 kHz to 150 cycles and still avoid reflections from vertical rod and study if this can improve measurements results by obtaining a longer steady state area.

If possible, a suggestion for further work is to increase the accuracy of the microphone sensitivity beyond 140 kHz with a frequency-dependent factor.

Further work can be, performing measurements with a higher signal generator output voltage, e.g., 2-volt peak-to-peak at resonances and 20-volt peak-to-peak outside resonance, and studying improvements on the SNR ratio, which can further improve measurement results. It can also be suggested to improve the method of calculating the SNR.

More work can be conducting measurements on different-sized piezoelectric disks or other types of disks that can be interesting for gas measurements.

Finally, work regarding on-axis measurements in the near field to try to lower the effects of standing waves between the piezoelectric disk and microphone.

# References


[1] Aanes, M. (2009). Undersøkelser av piezokeramiske skiver. Målinger og endelig element analyser. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[2] Aanes, M. (2011). *A Guide to "inn-files" to FEMP 5.* University of Bergen, Department of Physics and Technology, Bergen, Norway.

[3] AGA-9 (1998). Measurement of Gas by Multipath Ultrasonic Meters. *Transmission Measurement Committee Report No.9, A.G.A, American Gas Association, Wilson Boulevard, Arlington, USA.*

[4] Agilent Technologies (2007). *Agilent 33220A 20 MHz Waveform Generator User's Guide.* 3501 Stevens Creek Blvd. Santa Clara, CA 95052 USA: Agilent Technologies Inc.

[5] American National Standard (ANSI S1.26-1995(ASA 113-1995)). *Method for Calculating of the Absorption of Sound by the Atmosphere.* New York, USA: Reaffirmed by ANSI 2004.

[6] Amundsen, Ø. S. (2011). Material constants determination for piezoelectric disks, and influence on source sensitivity. Measurements and simulations. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[7] Andersen, K. K. (2015). Reciprocity calibration of ultrasonic piezoelectric disks in air. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[8] ASL (1997). *F250 MKII precision Thermometer. Operator's Handbool.* 40 Tanners Drive, Blakelands. Milton Keynes, MK 14 5BN, England: Automatic Systems Laboratoties LTD.

[9] Astley, R. J. (2000). Infinite elements for wave problems: a review of current formulations and an assessment of accuracy. *International Journal for Numerical Methods in Engineering 49*(7), 951–976.

[10] Baby, S., T. Balasubramanian, and R. Pardikar (2003). Ultrasonic study for detection of inner diameter cracking in pipeline girth welds using creeping waves. *International Journal of Pressure Vessels and Piping 80*(2), 139–146.

[11] Benny, G., G. Hayward, and R. Chapman (2000). Beam profile measurements and simulations for ultrasonic transducers operating in air. *The Journal of the Acoustical Society of America 107*(4), 2089–2100.

[12] Bentley, J. P. (2005). *Principles of Measurement Systems* (Fourth Edition ed.). Harlow, England: Pearson Prentice Hall.

[13] Blackburn, John, F. and M. G. Cain (2006). Nonlinear piezoelectric resonance: A theoretically rigorous approach to constant IV measurements. *Journal of Applied Physics 100*(11), 114101.

[14] Brassier, P., B. Hosten, and F. Vulovic (2001). High-frequency transducers and correlation method to enhance ultrasonic gas flow metering. *Flow Measurement and Instrumentation 12*(3), 201–211.

[15] Brigham, E. O. (1988). *The fast Fourier transform and its applications.* Englewood Cliffs, USA: Prentice-Hall, Inc.

[16] Brüel & Kjær (1982 ). *Product Data, Pistonphone — Type 4228.* Nærum, Denmark: Brüel & Kjær, Sound & Vibration Measurement A/S.

[17] Brüel & Kjær (1982). *Condenser Microphones and Microphone Preamplifiers for acoustic measurements.* Nærum, Denmark: Brüel & Kjær, Sound & Vibration Measurement A/S.

[18] Brüel & Kjær (1985). *Instruction Manual, Measuring Amplifier Type 2636.* Nærum, Denmark: Brüel & Kjær, Sound & Vibration Measurement A/S.

[19] Brüel & Kjær (1995). *Calibration data for Type 4138 microphone serial no. 1832479.* Nærum, Denmark: Brüel & Kjær, Sound & Vibration Measurement A/S.

[20] Brüel & Kjær (1996). *Calibration data for Type 4228 pistonphone serial no. 1918465.* Nærum, Denmark: Brüel & Kjær, Sound & Vibration Measurement A/S.

[21] Burnett, D. S. (1987). *Finite Element Analysis, From Concepts To Application* (Reprinted with correction May, 1988 ed.). Whippany New Jersey, USA: Addison-Wesley Publishing Company.

[22] Chillara, V. K., E. S. Davis, C. Pantea, and D. N. Sinha (2019). Ultrasonic bessel beam generation from radial modes of piezoelectric discs. *Ultrasonics 96*, 140–148.

[23] Cocraft (2022). *Instruction manual Cocraft Cross-Line Laser Level HL 10-S.* Jungfernstieg, Hamburg: Cocraft.

[24] Finstad, S. K. (2021). Characterization and finite element modelling of piezoelectric ceramic discs vibrating in air, for a frequency range including the first two radial modes. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[25] Getman, I. and S. Lopatin (2000). Matching of series and parallel resonance frequencies for ultrasonic piezoelectric transducers. In *ISAF 2000. Proceedings of the 2000 12th IEEE International Symposium on Applications of Ferroelectrics (IEEE Cat. No.00CH37076)*, Volume 2, pp. 713–715 vol. 2.

[26] Grindheim, R. (2019). Ultrasonic measurement systems for gas. Experimental and theoretical characterization using piezoelectric elements at radial mode vibration in air. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[27] Hagen, A. (2017). Ultrasonic measurement systems with diffraction correction for gas. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[28] Hatano, H. and J. Oosumi (1983). Effect of bevelling on the thickness vibration of high-frequency Pb(ZrTi)O3 ceramic transducers. *Ultrasonics 21*(5), 205–210.

[29] Hauge, R. (2013). Finite element modeling of ultrasound measurement systems for gas. Comparison with experiments in air. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[30] Hauptmann, P., N. Hoppe, and A. Püttmer (2002). Application of ultrasonic sensors in the process industry. *Measurement Science and Technology 13*, R73.

[31] Hawlett-Packard (1982). *Operation and service manual. Model 4192A LF Impedance Analyzer.* Japan, Tokyo: Hawlett-Packard Company.

[32] IEEE Standard on Piezoelectricity (1987). *The Institute of Electrical and Electronics Engineers.* 345 East 47th Street, New York, NY 10017, USA.

[33] ISO 17089-1: (2019). Measurement of fluid flow in closed conduits - Ultrasonic meters for gas — Part 1: Meters for custody transfer and allocation measurement.

[34] Kenny, A., I. Cato, M. Desprez, G. Fader, R. Schüttenhelm, and J. Side (2003, 01). An overview of seabed-mapping technologies in the context of marine habitat classification. *ICES Journal of Marine Science 60*(2), 411–418.

[35] Keyence (2013). *High-speed, High-accuracy CCD Laser Displacement Sensor LK-G Series User's Manual.* 500 Park Boulevard, USA: Keyence.

[36] Kinsler, L. E., A. R. Frey, A. B. Coppens, and J. V. Sanders (2000). *Fundamentals of Acoustics* (Fourth Edition ed.). Hoboken, USA: John Wiley & Sons Inc.

[37] Knappskog, V. (2007). Radiellmode Svingninger i Piezoelektriske Ultralydtransdusere for Luft. Målinger og Endelig Element Analyser. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[38] Kocbach, J. (2000). *Finite Element Modeling of Ultrasonic Piezoelectric Transducers.* Ph. D. thesis, University of Bergen, Department of Physics, Bergen, Norway.

[39] Kocbach, J., P. Lunde, and M. Vestrheim (1999). Femp - finite element modeling of piezoelectric structures. theory and verification for piezoceramic disks. Technical report, No 1999-07 University of Bergen, Department of Physics, Christian Michelsen Reasearch, Bergen, Norway.

[40] Krohn-Hite (2014). *Operating anad Maintenance Manual KH Model 3940/3944.* Brockton, USA: Krohn-Hite Corporation.

[41] Kunkel, H., S. Locke, and B. Pikeroen (1990). Finite-element analysis of vibrational modes in piezoelectric ceramic disks. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 37*(4), 316–328.

[42] Lohne, K. D. (2005). Undersøkelse og Utnyttelse av Svingemoder i Ultralyd Transduserkonstruksjoner. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[43] Lunde, P., K.-E. Frøysa, V. Martinez, and Torvanger (2008). Pressure and temperature effects for Ormen Lange ultrasonic gas flow meters,-Results from a follow-up study. University of Bergen, Department of Physics and Technology, Christian Michelsen Research, CMR Prototech. Bergen, Norway.

[44] Lygre, A., M. Vestrheim, P. Lunde, and V. Berge (1987). Numerical simulation of ultrasonic flowmeters. pp. 196–201.

[45] Lynnworth, L. and Y. Liu (2007). Ultasonic flowmeters: Half-century progress report 1955-2005. *Ultrasonics 44 Suppl 1*, e1371–8.

[46] Meggitt (2018). *Data Sheet Type PZ27.* Porthusvej 4, DK3490, Kvistgaard, Denmark: Meggitt A/S.

[47] Mitutoyo (2018). *Outside Micrometer User's Manual.* Mitutoyo.

[48] Mosland, E. (2013). Reciprocity calibration method for ultrasonic piezoelectric transducers in air. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[49] Øyerhamn, R., E. Mosland, E. Storheim, P. Lunde, M. Vestrheim, and J. Kocbach (2018). Finite element modeling of ultrasound measurement systems for gas. comparison with experiments in air. *The Journal of the Acoustical Society of America, Department of Physics and Technology, University of Bergen, Bergen, Norway*.

[50] Paroscientific (2002). *Digiquartz Precision Pressure Instruments. Model 770.* Paroscientific, Inc.

[51] Physik Instrumente (1995). *Operating Manual MS44 E C-852 Signal Processor For Incremental Encoders.* D-763337 Waldbronn, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[52] Physik Instrumente (1996). *Product Instruction MP 30 E M-500 Linear Positioning Stages.* D-763337 Waldbronn, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[53] Physik Instrumente (1998). *Operating Manual MP 33E M-501.xx Series Linear Positioning Stages.* D-763337 Waldbronn, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[54] Physik Instrumente (2007). *MP 34E M-037/M-038 Worm-Gear Rotation Stages.* Auf der Romerstr. 1 76228 Karlsruhe, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[55] Physik Instrumente (2009). *MS77E User Manual, C-843 Motor Controller Card.* Auf der Romerstr. 1 76228 Karlsruhe, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[56] Physik Instrumente (2016). *Hydra Handbook.* Auf der Romerstr. 1 76228 Karlsruhe, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[57] Physik Instrumente (2018). *Linear Stage for Very High Loads, LS-270.* Auf der Romerstr. 1 76228 Karlsruhe, Germany: Physik Instrumente (PI) GmbH & Co. KG.

[58] Pickrill, R. A. and B. J. Todd (2003). The multiple roles of acoustic mapping in integrated ocean management, canadian atlantic continental margin. *Ocean & Coastal Management 46*(6), 601–614.

[59] Ricci, R. and A. Sona (2013). Experimental validation of an ultrasound-based measurement system for human motion detection and analysis. In *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pp. 300–305.

[60] Rohatgi, A. (2021). Webplotdigitizer: Version 4.5. https://automeris.io/WebPlotDigitizer , last accessed:11.04.2022.

[61] Ronold W. P. King, Harry Rowe Mimno, A. H. W. (1945). *Transmission Lines Antennas and Wave Guides* (First Edition ed.). New York, USA: McGraw-Hill Book Company, INC.

[62] Sanabria, S. J., T. Marhenke, R. Furrer, and J. Neuenschwander (2018). Calculation of volumetric sound field of pulsed air-coupled ultrasound transducers based on single-plane measurements. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control 65*, 72–84.

[63] Schafer, R. W. (2011). What is a savitzky-golay filter? [lecture notes]. *IEEE Signal Processing Magazine 28*(4), 111–117.

[64] Søvik, A. A. (2015). Ultrasonic measurement systems for gas. Finite element modelling compared with measurements in air. Master's thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[65] Storheim, E. (2015). *Diffraction effects in the ultrasonic field of transmitting and receiving circular piezoceramic disks in radial mode vibration FE modelling and comparison with measurements in air.* Ph. D. thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[66] Tektronix (2012). *Mixed Signal Oscilloscopes MSO3000 Series, DPO3000 Series Data Sheet.* Beaverton, Oregon, USA, Tektronix Inc.

[67] TESA Technology (2022). *Analogue dial gauges, metric, Standard model, resolution 0.01 mm, measuring range 30 mm.* Brown& Sharpe TESA, USA.

[68] The Mathworks, Inc. (2022). *MATLAB version (R2021b).* Natick, Massachusetts: The Mathworks, Inc.

[69] The Norwegian Petroleum Directorate (2001). Regulations relating to measurement of petroleum for fiscal purposes and for calculation of CO2-tax (the measurement regulations).

[70] Vaisala Oyj (2014). *User's Guide. Vaisala HUMICAP Humidity and Temperature Transmitter HMT310.* Vanha Nurmijärventie 21, FI-01670 Vantaa, Finland: Vaisala Oyj.

[71] Vervik, S. (2000). *Methods for characterization of gas-coupled ultrasonic sender-receiver measurement systems.* Ph. D. thesis, University of Bergen, Department of Physics and Technology, Bergen, Norway.

[72] Vestrheim, M. (2013 ). *PHYS 373 Akustiske Målesystemer, Lecture notes*. Bergen, Norway: University of Bergen, Department of Physics and Technology.

[73] Vestrheim, M. (2013). *PHYS 272 Akustiske Transdusere, Lecture notes*. Bergen, Norway: University of Bergen, Department of Physics and Technology.

[74] Wang, Z., Q. Cao, N. Luan, and L. Zhang (2008). Development of new pipeline maintenance system for repairing early-built offshore oil pipelines. In *2008 IEEE International Conference on Industrial Technology*, pp. 1–6.

[75] Whitworth M, Bricker L, M. C. (2015). Ultrasound for fetal assessment in early pregnancy.

[76] Wikipedia (2022). Angular distance. https://en.wikipedia.org/wiki/Angular_distance, last accessed:15.06.2022.

[77] Yu, Y., A. Safari, X. Niu, B. Drinkwater, and K. V. Horoshenkov (2021). Acoustic and ultrasonic techniques for defect detection and condition monitoring in water and sewerage pipes: A review. *Applied Acoustics 183*, 108282.

# Appendix A

# MatLab-scripts

## A.1 impanal.m

```
clear all
clc
close all
instrreset
vinfo = instrhwinfo('visa','agilent');
vinfo.ObjectConstructorName
obj1 = visa('agilent','GPIB0::17::INSTR');

% Connect to instrument object, obj1.
fopen(obj1);
fprintf(obj1, 'V1');
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Osc. level [V]
    amplitude = 0.3;
    f = [100:50:97450,97500:10:99500,99550:50:300000]/1000;  % Hz

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    ol = sprintf('%3.3f',amplitude);
    % Sett analysator i mode for admittans-måling
    fprintf(obj1, ['A2C3F1OL',ol,'EN']);

    % Tids-streng på format yyyymmddHHMMSS
    time = datestr(now, 'yyyymmddHHMMSS');
    % Tittelen som målingen blir lagra under

    i = 1;
```

```
    ii = 1;
    antal = length(f);
    g = ones(1,antal);
    b = ones(1,antal);
    fr = ones(1,antal);
    disp([num2str(antal),' frekvenser.'])
    disp('Starter måling...')
    for freq = f
        percent = i/antal*100;
        if percent >= ii*10
            disp([num2str(ii*10),' %'])
            ii = ii + 1;
        end
        s = sprintf('%3.3f',freq);
        fprintf(obj1, ['FR',s,'ENEX']);
        pause(0.1)
        data1 = fscanf(obj1);
        d=sscanf(data1,'%4c%f,%4c%f,%2c%f');
        g(i)=d(5);b(i)=d(10);fr(i)=d(13);
        i = i + 1;
    end
    disp('Måling ferdig.')
    disp('Lagrer data...')
    stoptime = datestr(now, 'yyyymmddHHMMSS');
    save(stoptime,'g','b','fr')
    disp('Ferdig!')
```

## A.2 positioninganalyze directivity.m

```
%%%%%%%%%%%%%%%%%%%%%%%%
% The script was created to analyze how much
% incorrect positioning affects the simulated
% directivity of the piezoelectric disk
%
% Created by Espen Fosse 2022
%%%%%%%%%%%%%%%%%%%%%%%%

clear all

% loading the values
d1 = load('PZ27_result_0.2m.mat');
d2 = load('PZ27_result_0.5m.mat');
```

```
d3 = load('PZ27_result_0.8m.mat');

sim = 1;
% frequency indeks
% 1963 - 98100Hz
% 2944 - 147150Hz
% 4982 = 249050Hz
mode = 4982;
% decimal distance and angle for interpolation functions
decdist = 3;
decang = 2;

disp(['simulation frequency ', num2str(d1.result.directivity_f{1}(mode))...
    ])

ang(1,:) = rad2deg(d1.result.directivity_theta{sim})+90;
Y_k(1,:) = abs(d1.result.directivity{sim}(:,mode).');
ang(2,:) = rad2deg(d2.result.directivity_theta{sim})+90;
Y_k(2,:) = abs(d2.result.directivity{sim}(:,mode).');
ang(3,:) = rad2deg(d3.result.directivity_theta{sim})+90;
Y_k(3,:) = abs(d3.result.directivity{sim}(:,mode).');

Y_org = [fliplr(abs(d1.result.directivity{sim}(:,mode).')),abs(...
    d1.result.directivity{sim}(:,mode).')];
ang_org = [-fliplr(rad2deg(d1.result.directivity_theta{sim})+90),rad2deg...
    (d1.result.directivity_theta{sim})+90];
% interpolated amplitude over new angles
angle = (0:10^-(decang):180);
p(1,:) = interp1(ang(1,:),Y_k(1,:),angle,'spline');
p(2,:) = interp1(ang(2,:),Y_k(2,:),angle,'spline');
p(3,:) = interp1(ang(3,:),Y_k(3,:),angle,'spline');

% original distances in mm
dist = [200,500,800];

% plotting angle in degrees
theta_R = (-90:0.1:90);
% tilt angle of piezoelectric disk in degrees
theta_T = 0;
% radius to center of from piezoelectric disk  to rotation stage in mm
sigma = 5;
% line going through rotation stage to center of microphone's
% start position and alpha is the angle from line to center of
% piezoelectric disk front in degrees
alpha = 180;
```

```
% start distance to in mm
z0 = 200;
% initial angle from microphone center to xz-plane
theta_M = 0;

beta = z0*sind(theta_M);
r0 = z0*cosd(theta_M);
R = sigma*cosd(alpha)+sqrt(r0^2+sigma^2*(cosd(alpha)^2-1));
r = sqrt(R^2-2*R*sigma*cosd(alpha-theta_R)+sigma^2);
theta_m = atand(beta./r);
costheta_r = (r.^2+r0^2-4*R^2*sind(theta_R/2).^2)./(2*r.*r0);
theta = acosd(sind(theta_m)*sind(theta_T)+cosd(theta_m)*cosd(theta_T).*...
    costheta_r );

% sound propagation distance
z = r./cosd(theta_m);

% interpolated amplitudes over distances
distance = (round(min(z),decdist):10^-(decdist):round(max(z),decdist));

for i = 1:length(p)
    p2(:,i) = interp1(dist,p(:,i),distance,'spline');
end

for i = 1:length(theta)
    col = find(round(angle,decang)==round(theta(i),decang));
    row = find(round(distance,decdist)==round(z(i),decdist));
    D(i) = p2(row,col);
end

Y_orgplot = 20*log10(Y_org./max(Y_org));
Y_orgplot2 = Y_org./max(Y_org);
D_plot = 20*log10(D./max(D));

% plot(ang_org,Y_orgplot,'r')
% plot(ang_org,Y_orgplot2,'r')

plot(ang_org,Y_orgplot,'r',theta_R,D_plot,'b')
xlim([-90 90])
xticks([-90 -60 -30 0 30 60 90])
ylim([-30 0])
```

## A.3 MeasParameters.m

```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Information about the calibration of the measurement microphone.
% Part of the software for acoustic measurements in air.
% Espen Storheim, 2011
% Based on work by Vidar Knappskog and Magne Aanes.
%
% Modified by Rune Hauge and Eivind Mosland, 2012/2013
% Modified by Espen Fosse 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function meas = MeasParameters()
    %% General measurement info.

    % Version of this software which was used to make the measurements. ...
       Should
    % be taken from elsewhere.

    % Name of the person performing the measurement.
    meas.name = 'E_F';
%     meas.name = 'ENM';

    % Information about the transmitting transducer.
    meas.source = 'Pz27 disk, D = 20.0 mm, T = 2.0 mm, Element No. 7 in ...
       batch 9/12.';

    % Information about the receiving transducer.
    meas.receiver = 'B&K Type 4138 mic';

    % Additional notes regarding the specific simulation.
%     meas.notes = 'Sensitivity';
    meas.notes = 'Directivity';
%     meas.notes = 'On Axis Pressure';

    meas.info = 'Måler direktiviteten over mangen distanser for å plotte...
        2d feltet';
    %% Measure distances for primary axis and secondary axis [mm]

    % Primary axis is z-axis, distance between transducer to
    % microphone/transducer
    meas.primary_axis = [120:10:300];
%     meas.primary_axis = [880];

    % Secondary axis can be your x, y or rotation axis, if no secoundary...
        is
    % used, set value to 0
```

```
    meas.secondary_axis = [-93:1:93];
%     meas.secondary_axis = [0];
    %% Measure frequency [Hz]
    meas.frequency = [98.86e3*3/2];
%     meas.frequency = [97e3:0.1e3:97.9e3];
    %% Define initial bandpass filter low and high cutoff frequency [kHz...
        ]

    for i = 1:length(meas.frequency)
        meas.cutoff_1(i) = (meas.frequency(i)/1000)/2;
        meas.cutoff_2(i) = (meas.frequency(i)/1000)*2;
    end
    %% Input waveform data.

    %      Vpp voltage out from the signal generator [V]. This is the ...
        actual voltage
    %      level of the function generator
    meas.voltage_inn = 1;

    % measured signal averaging(2,4,8,16,...,512)
    meas.average = 128;

    % scaling averaging (2,4,8,16,...,512), if set to 1, simple sampel ...
        is
    % taken insted of averaging, which is the smartest choice
    meas.average_scaling = 1;

    % Time in Hz between burst
    meas.burst_rate = 25;

    % Approximate time before the signal is averaged.
    meas.average_time = meas.average/meas.burst_rate;
    meas.average_time_scaling = meas.average_scaling/meas.burst_rate;

    % Estemated travel time from tranducer to microphone/transducer.
    % Normally faktor is 2, and then signal duration becomes
    % faktor*estemated_travle time
    meas.est_travel_time_faktor = 2;
    % If signal duration becomes less then ~ faktor*10 cm then min ...
        signal
    % duration is used insted of faktor*estemated_travle time
    meas.min_sig_duration =  meas.est_travel_time_faktor*2.9155e-04;

    % Normally, signal cycles changes based on distance from microphone,...
        but
```

```
    % this can be override with setting sig_cycles to not eaqual to 0
    meas.sig_cycles = 60;

    % A note on the input voltage: The signal generator claims that the ...
       voltage
    % specified above is the peak to peak voltage. This is the case when...
        the
    % generator is connected to a 50 Ohm load. However, the transmitting
    % transducer typically has an electrical impedance in the kilo Ohm ...
       range and is connected
    % directly to the generator. This causes a voltage division which ...
       depends
    % of the impedance of the transducer, and hence an impedance ...
       mismatch.
    %% Oscilloscope parameters.

    % Allowed values: 1e3, 10e3, 100e3, 1000e3, or 5000e3.
    meas.sample_count = 10e3;

    % Channel number where the signal generator is connected.
    meas.channel_electrical = 1;

    % Channel number where the oscilloscope is connected.
    meas.channel_acoustical = 2;
    %% Total input gain in the B&K 2636 measurement amplifier [dB].

    % Only recorded for later reference. Must be set manually.
    meas.gain_in = 40;
    meas.gain_out = 20;
    meas.gain = meas.gain_in + meas.gain_out;
end
```

## A.4 HVV_0m1.m

```
function HVV_0m1 = HVV_0m1(Frequency)
% HVV_0m1(f) calculates the transfer function HVV_0m1 from oscilloscope ...
   to
% transmitter for given frequency
% f:    Frequency [Hz]
f = Frequency;
% Cable length from oscilloscope to transducer in m
x = 3;
```

```
% Typical inductance per m values for RG58 coaxial cables
Lx = 250e-9;
% Typical capicitance per m values for RG58 coaxial cables
Cx = 100e-12;

Z0 = sqrt(Lx/Cx);
omega = 2*pi.*f;
kem = omega*sqrt(Lx*Cx);
Za = 1i*Z0*tan((kem*x)/2);
Zb = Z0./(1i*sin(kem*x));
load #7_admittance.mat fr b g
Z = 1./(g+1i*b);
ff = fr*1000;
ZT = interp1(ff, Z, f, 'spline');

HVV_0m1 = ZT.*Zb./(ZT.*(Za+Zb)+(Za+Zb).^2-Zb.^2);
```

## A.5 HVV_55m.m

```
function HVV_55m = HVV_55m(Frequency,Gain)
% HVV_55m(f) calculates the transfer function HVV_0m1 from microphone to
% oscilloscope for given frequency
% f:     Frequency [Hz]
f = Frequency;
% Cable length from amplifier to filter ch1 in m
x1 = 0.5;
% Cable length from filter ch1 to filter ch2 in m
x2 = 0.8;
% Cable length from filter ch2 to oscilloscope in m
x3 = 1.5;
% Typical inductance per m values for RG58 coaxial cables
Lx = 250e-9;
% Typical capicitance per m values for RG58 coaxial cables
Cx = 100e-12;

omega = 2*pi*f;

Zamp_out = 100;

Cfilt = 100e-12;
Rfilt = 1e6;
Zfilt_out = 50;
```

```
Zfilt = 1./(1i*omega*Cfilt+1/Rfilt);

Cosc = 11.5e-12;
Rosc = 1e6;
Zosc = 1./(1i*omega*Cosc+1/Rosc);

Lx = 250e-9;
Cx = 100e-12;
Z0 = sqrt(Lx/Cx);
kem = omega*sqrt(Lx*Cx);

Za1 = 1i*Z0*tan((kem*x1)/2);
Zb1 = Z0./(1i*sin(kem*x1));

Za2 = 1i*Z0*tan((kem*x2)/2);
Zb2 = Z0./(1i*sin(kem*x2));

Za3 = 1i*Z0*tan((kem*x3)/2);
Zb3 = Z0./(1i*sin(kem*x3));
HVV_55a = 10^(Gain/20);
HVV_5a51 =  Zfilt./((Zamp_out+Za1).*(1+(Zfilt+Za1)./Zb1)+Zfilt+Za1);
HVV_5152 =  Zfilt./((Zfilt_out+Za2).*(1+(Zfilt+Za2)./Zb2)+Zfilt+Za2);
HVV_525m =  Zosc./((Zfilt_out+Za3).*(1+(Zosc+Za3)./Zb3)+Zosc+Za3);

HVV_55m = HVV_55a*HVV_5a51.*HVV_5152.*HVV_525m;
```

## A.6 Vpp.m

```
function Vpp = Vpp(time,amplitude,start,stop,freq,filtorder,filtframelen...
    )
% Finding the DC component by taking the mean value of signal
DC = mean(amplitude);
% Subtract the DC value from signal
amp = amplitude-DC;
zeropadding = length(amplitude)*5;

% SampleInfoStruct.Both = 'IntCycles';     % bør brukes ved zeropad
SampleInfoStruct.Both = 'IntPeriods';    % bør brukes ved zeropad
SampleInfoStruct.StartVal = start;
SampleInfoStruct.EndVal = stop;
SigStruct.SigFreq = freq;
% Dersom 'intPeriods' eller 'IntCycles'   eller  bruk av peak_peak
```

```
SigStruct.periodFracJump = 1/16;
% Program starts
SigStruct.x = time;
SigStruct.y = amp;
CutOffIndxs = find_index_in_sig_advanced(SigStruct,SampleInfoStruct);
Sig_wa_reg = SigStruct.y(CutOffIndxs(2) : CutOffIndxs(3));

% framelenght must be odd
if rem(filtframelen,2) == 0
    filtframelen = filtframelen+1;
end

% filtering the measured signal with Savitzky-Golay method
filtered = sgolayfilt(Sig_wa_reg,filtorder,filtframelen);

% Y_k_tmp = fourier_transform(SigStruct.x,Sig_wa_reg,zeropadding);
Y_k_tmp = fourier_transform(SigStruct.x,filtered,zeropadding);

% finding the peak to peak voltage at given frequency
Vpp = 2*2*abs(interp1(abs(Y_k_tmp{1}), Y_k_tmp{2}, freq, 'cubic'));
```

## A.7 Receiver_Sensitivity.m

```
function Mv_f=Receiver_Sensitivity(Frequency)
% Linear interpolation of the dB correction from different ambient
% pressures
load Mv_measurment.mat

x = [685 800 940 990 1013 1060];
y = [-3.4 -2.05 -0.65 -0.2 0 0.39];
s = fit(x.',y.','linearinterp');

% Load volume correction, for DP 0774 = 0dB
Lv = 0;
% Correction for ambient pressure
Lp = s(DPO.pressure);
% Stated sound pressure level
sSPL = 124.11;
% Actual sound pressure level of pistophone
aSPL = sSPL + Lp + Lv;
% Amplifier gain
Gain = 20;
```

```
% Frequency from pistonphone
freq = 251.2;
V_ppmax = Vpp(DPO.x,DPO.wf,1,length(DPO.x),freq,5,10);
HVV = abs(HVV_55m(250,Gain));
Veff = V_ppmax/(HVV*2*sqrt(2));
peff = 10^(aSPL/20)*20*1e-6;

% Calculated calibrated reciver sensitivity. 0.4563mV/Pa in my case. The
% given sensitivity of the microphone is 0.822 mV/Pa.
Mv = Veff/peff;

% Open circuit pressure response
D1 = [20.058132685561468, -0.03159553295001594
21.05087511963177, -0.004975674480313333
22.09308910488738, -0.004699248120295429
23.075127239496453, -0.004450464396279052
24.100816983586117, -0.004201680672263563
25.050753654543616, -0.003980539584249598
26.038132196432052, -0.003759398496234745
27.06442838520751, -0.003538257408220779
28.131176164717257, -0.003317116320206814
29.09901530059949, -0.003123617868194372
30.100152460974538, -0.002930119416181931
30.9856398639198, -0.002764263600170125
32.05168540481925, -0.002570765148158571
33.15440771277152, -0.002377266696147018
34.29506863372019, -0.002183768244134576
35.64681298953859, -0.001962627156120611
37.05183651569491, -0.001741486068106645
38.141828818330985, -0.001575630252095727
39.64519370079917, -0.001354489164079986
41.207813895344884, -0.001133348076066909
43.03950182712216, -0.000884564352052308
44.952608314320514, -0.000635780628035931
47.17890095209987, -0.02670278637701848
49.514695074759025, -0.02642636001768483
51.966132717579626, -0.02614993365766782
54.80312407373893, -0.0258458646616484
57.795878408724505, -0.05188522777531457
60.65638501271208, -0.02526536930561107
63.65943986891338, -0.02498894294559317
66.48910290436899, -0.02474015922157679
69.44454450323498, -0.02449137549756130
72.53135552449893, -0.02424259177354493
```

```
75.75537534091245, -0.02399380804952944
79.5059705627214, -0.02371738168951154
83.44352977364937, -0.049784387439179234
87.5747601008906, -0.04950796107916222
91.91192875425654, -0.07557496682882991
97.39920832615756, -0.07524325519680808
102.71653131456982, -0.07493918620078865
108.84886181419361, -0.0746074745687677
116.46748492230017, -0.07422047766474282
124.01861390768617, -0.07386112339672035
132.69900855770388, -0.07347412649269547
141.98696725722905, -0.07308712958867059
153.40041900372466, -0.07264484741264177
162.55865338014362, -0.07231313578062082
172.2610165824735, -0.04563799203891428
182.545256834413, -0.0453062804068924 4
195.32208783488707, -0.04491928350286756
208.99320342673101, -0.04453228659884356
224.70441044155518, -0.04411764705881627
238.11959971942048, -0.04378593542679532
256.02040349742686, -0.04337129588676891
273.93995662570694, -0.04298429898274403
293.1137472285418, -0.042597302078719146
312.1176753565441, -0.042237947810695786
332.35371657275545, -0.041878593542673315
353.9017545018618, -0.041519239274649955
376.8468519956459, -0.041159885006627484
405.18276132827987, -0.06708867757628667
435.64930742610267, -0.09301747014594586
461.6582224554411, -0.09268575851392491
493.9709168008378, -0.09229876160990003
528.5452630893234, -0.09191176470587514
571.0317389443935, -0.09146948252984721
605.1231875570597, -0.09113777089782538
653.7652878517716, -0.09069548872179745
720.0916947063869, -0.06379920389207516
797.0012625520569, -0.06321870853603784
877.8727895316784, -0.06266585581600292
953.0337476511787, -0.06219593100397258
1029.6422202576316, -0.06175364882794465
1101.7096344831245, -0.06136665192391977
1190.2692395984482, -0.06092436974789095
1311.0455738980609, -0.060371517027856036
1389.3169549145211, -0.0600398053958342
1515.5722906662636, -0.059542237947802334
```

```
1629.5064742385878, -0.05912759840777593
1752.0325016325971, -0.085056390977436
1883.7427432984218, -0.0846417514374096
2015.5909386274632, -0.08425475453338471
2146.2710779771282, -0.08389540026536224
2318.7964995463685, -0.08345311808933431
2481.1331726557723, -0.10940955329499413
2654.7943226984075, -0.10902255639096925
2786.2313178527374, -0.10874613003095224
2952.5734934578413, -0.10841441839893129
3113.76358094979, -0.10811034940291187
3315.694012410975, -0.1340944272445741
3513.645902525506, -0.13376271561255315
3759.575381442612, -0.13337571870852827
4003.3261471992037, -0.1330163644405049
4283.529656781166, -0.13262936753648003
4561.2508162331815, -0.13227001326845755
4833.638147410243, -0.1582817337461231
5197.010201254747, -0.15786709420609668
5587.698997786313, -0.157452454660685
5949.975524079862, -0.15709310039804603
6335.740123291306, -0.15673374613002267
6845.030979856471, -0.15629146395399474
7359.723291062311, -0.18222025652365392
7874.850018159103, -0.18183325961962993
8304.76235267869, -0.1815291906236114
8886.034892537247, -0.18114219371958562
9462.157882459767, -0.18078283945156226
9978.726245725167, -0.18047877045554284
10371.880529349377, -0.1539141972578424
11044.337805733268, -0.15355484298981903
11535.082573171361, -0.12696262715611795
12106.176378560287, -0.12668620079610005
12583.15059193551, -0.10012162759840049
13205.931772154905, -0.0735017691286961
13859.74814335345, -0.0732253427686782
14758.114661653412, -0.04652255639096836
15714.951906534357, -0.04616320212294589
16977.920077169827, -0.019377487837231477
18342.389582936732, 0.007408226448482047
19816.51781154511, 0.03419394073419646
21305.91262297363, 0.06095201238390935
22796.82221585644, 0.08768244139762071
24988.582880231424, 0.11455108359134059
27523.327493266388, 0.16779080053074757
```

```
29448.859105935688, 0.22086466165414365
31662.21355962875, 0.24762273330385653
33069.59941653967, 0.24787151702787114
35382.60763185681, 0.3272888102609546
38040.776002059705, 0.40673374613003865
40507.13651879439, 0.407093100398062
43341.0152322141, 0.4601669615214581
45927.13546765157, 0.513185537372852
48432.95961514873, 0.5661764705882435
50341.950685981115, 0.5663976116762575
52582.864033026635, 0.4612726669615306
55456.913162824414, 0.35620300751880585
58206.10232391049, 0.2511057054400796
61091.57820945313, 0.14600840336135246
63197.26505637179, 0.040828173374621635
65063.36147790973, -0.14340999557716927
66984.56021837509, -0.3276481645289602
69294.42319019328, -0.4591718266253775
71334.01878699053, -0.48534940291905393
72721.8084799586, -0.3798651039363037
74859.99778107263, -0.3533558160106054
76322.21394529774, -0.3795886775762858
78192.3034605607, -0.45848076072533317
79719.6084756157, -0.48471362229101267
82064.80287834875, -0.48454776647500264
84481.5693191942, -0.5370687748783638
86969.5082947287, -0.5895897832817241
89530.71579967023, -0.6421107916850861
92608.14619754962, -0.5892304290137016
94870.03632559274, -0.536405351614321
97657.94290884356, -0.4835526315789389
100530.84790012054, -0.483386775762928
103488.26811712357, -0.48322091994691707
106529.43511934493, -0.43036819991153585
109658.29650014469, -0.3511720477664664
112879.05523803871, -0.2719758956213978
116192.63547874281, -0.1664363113666445
119026.92395984495, -0.06092436974789095
121928.48669724895, 0.07093100398054819
123699.48719847444, 0.2027310924369825
126718.82613948248, 0.2818996019460487
129815.82914620957, 0.30838124723574367
132337.32807373017, 0.4665524104378651
134909.86459873084, 0.5983801415303054
137532.4092678341, 0.730207872622743
```

```
140205.93420114485, 0.8620356037151797
142242.41301320156, 0.993835692171614
144306.26707734872, 1.1519792127377348
145692.10249991124, 1.3364385227775415
147088.99976687875, 1.5472412649270328
148499.290498152, 1.7580440070765242
149923.103110395, 1.9688467492260155
151358.2551025438, 2.2059929234851925
152804.81096584388, 2.469482529854056
153528.57439263677, 2.6539141972578606
155000.60782678, 2.8647169394073515
156486.75513151608, 3.0755196815568433
157987.15163077012, 3.286322423706334
159501.93394595556, 3.497125165855826
161033.699921934, 3.6815844758956318
164161.5642534129, 3.839755639097754
166548.54063330134, 3.9452122954445032
168165.95944322916, 3.945267580716507
169809.46128984625, 3.839949137549767
172304.87949111866, 3.681971472799656
173994.14714333607, 3.5239661654135435
175708.02838608803, 3.2869305616983735
177438.79174219005, 3.0498949579832035
179186.6035042316, 2.812859354268034
180076.5825961926, 2.602139540026546
180970.98201181978, 2.3914197257850605
182753.5866375774, 2.1543841220698905
184556.56956080245, 1.8910050862450332
185470.3868064449, 1.7066287041132338
187300.17272927362, 1.4432496682883773
188230.4500736161, 1.2325298540468905
189165.3478938763, 1.0218100398054037
190101.98514062806, 0.8374336576736052
191971.5996297703, 0.6267414860681209
192928.02601257144, 0.38967823971694937
193889.21743055253, 0.15261499336577788
195817.0164099142, -0.2424812030075083
197751.8993713959, -0.5322036709420512
199699.79978422992, -0.7692392746572221];

% Free field correction for mic B&K type 4138
D2 = [4007.553556653379, 0.16216216216216317
4171.523985222408, 0.16216216216216317
4361.593637668383, 0.16216216216215962
4560.34969849724, 0.193050193050194
```

```
4768.162992759082, 0.22393822393822482
5097.729374567116, 0.22393822393822482
5425.876500218985, 0.25482625482625565
5826.807002226876, 0.25482625482625565
6174.314057706561, 0.28571428571429003
6601.108787041513, 0.3166023166023155
7025.990009476682, 0.3166023166023173
7511.656186541617, 0.34749034749034813
8175.30676583339, 0.37837837837837895
8701.561343001555, 0.4092664092664098
9220.516914934291, 0.4401544401544406
9946.116316699288, 0.47104247104246966
10681.1802408922, 0.5328185328185313
11267.881754716329, 0.5637065637065639
12100.560859005827, 0.594594594594593
12822.304123128384, 0.6563706563706564
13466.479211351674, 0.6872586872586872
14333.415276075924, 0.7490347490347471
15120.815394438714, 0.8108108108108105
15951.562477890415, 0.903474903474903
16753.043310019864, 0.9652509652509647
17438.69982304954, 1.0270270270270263
18233.58344722897, 1.1196911196911188
19064.699128981494, 1.2123552123552113
19933.69838301448, 1.3050193050193037
20749.649446205003, 1.3976833976833962
21599.000038410624, 1.4903474903474887
22583.646254091753, 1.613899613899612
23718.761913320126, 1.7683397683397661
24800.042950269042, 1.8918918918918912
25700.570216429194, 2.0154440154440145
26633.796996808847, 2.1389961389961396
27478.205414054573, 2.262548262548263
28349.547855592133, 2.416988416988419
29379.131512011878, 2.5714285714285694
30446.107034819277, 2.7258687258687235
31411.562559999464, 2.8803088803088794
32407.818818171745, 3.0656370656370644
33435.48083424683, 3.2200772200772203
34342.37281678526, 3.3745173745173727
35431.78649771378, 3.590733590733592
36555.339469596234, 3.7451737451737444
37883.37113186014, 3.96138996138996
39084.888741774914, 4.146718146718147
40324.745193745155, 4.362934362934363
```

```
41603.6939783443, 4.548262548262544
42923.20619330782, 4.733590733590733
44087.69264795654, 4.9189189189189175
45284.03074875428, 5.135135135135135
46512.565266454076, 5.32046332046332
47774.42934948241, 5.505791505791505
49070.527213316214, 5.69111969111969
50401.78760437459, 5.876447876447875
51539.014520064986, 6.06177606177606
52701.598732028, 6.216216216216214
53890.407777012966, 6.370656370656368
54861.04856995523, 6.525096525096522
55849.17194630681, 6.679536679536676
57109.30942505863, 6.864864864864862
58397.8796703138, 7.050193050193046
59715.524217003884, 7.235521235521232
60791.43201484806, 7.420849420849418
62163.439290876755, 7.637065637065634
63850.270386668846, 7.85328185328185
65291.31285998187, 8.069498069498065
66764.87834999173, 8.285714285714283
67967.4061677525, 8.440154440154437
69191.98994649285, 8.625482625482622
70753.18444092202, 8.810810810810807
72350.01940002342, 9.027027027027025
73982.89347039249, 9.243243243243239
75652.62002196065, 9.459459459459456
77705.93130923978, 9.706563706563703
79459.22778529546, 9.891891891891888
81252.08428567818, 10.077220077220073
83456.41522816512, 10.262548262548261
86103.33786363956, 10.447876447876443
88833.70149212686, 10.602316602316598
91242.14849115594, 10.694980694980691
94134.38579384239, 10.787644787644783
97117.74570825178, 10.849420849420845
99749.07255594028, 10.849420849420845
102451.10558398512, 10.818532818532812
104757.92713762312, 10.75675675675675
107116.07557106025, 10.66409266409266
109040.3819306176, 10.57142857142857
110999.25784423781, 10.478764478764475
113497.2506241556, 10.355212355212352
115534.86772433866, 10.200772200772198
117086.88349409054, 10.077220077220073
```

```
119188.94530354132, 9.922779922779918
120788.66213038097, 9.737451737451732
122409.8498572236, 9.552123552123547
124052.79665978807, 9.366795366795362
125717.79458159857, 9.181467181467175
127405.13958589552, 8.996138996138992
129114.39121616929, 8.779922779922774
130845.82363460216, 8.532818532818528
132600.4746748429, 8.285714285714281
134378.65570012658, 8.038610038610035
135576.81927932706, 7.85328185328185
136784.09732192638, 7.606177606177602
139232.5974887442, 7.173745173745171
141096.47927815863, 6.803088803088798
143622.17312571823, 6.370656370656365
144899.43123038978, 6.061776061776056
146183.8568004814, 5.598455598455592
147482.20497857765, 5.227799227799224
149456.52278259964, 4.857142857142852
150783.07285878315, 4.45559845559845
152120.52484430978, 4.023166023166018
154153.39907145244, 3.5289575289575232
154834.6752811324, 3.2818532818532784
155518.0705690526, 3.0038610038610045
156902.92052218987, 2.7567567567567526
157596.34815742142, 2.509652509652506
158998.79289760426, 2.2316602316602285
160415.55770201754, 2.0154440154440127
161844.01856674667, 1.7683397683397644
162557.41889281085, 1.4594594594594579
164004.95250168373, 1.2123552123552095
165466.32488206084, 0.9961389961389937
167684.2775382433, 0.718146718146718
169178.43468459754, 0.5019305019305005
172212.73393191773, 0.2548262548262521
174521.11716896, -0.023166023166027117
177653.27752833022, -0.20849420849421207
180037.68437647808, -0.39382239382239526
183267.80058292698, -0.6100386100386093
188224.65855738753, -0.8571428571428577
193318.91070103558, -1.0115830115830153
197666.07300972778, -1.2277992277992293
200322.53149526555, -1.3204633204633254];
```

```
% cubic interpolation between points of open circuit pressure response
cubic1 = interp1(D1(:,1),D1(:,2),Frequency,'spline');

% cubic interpolation between points of free field correction for mic B&...
    K type 4138
cubic2 = interp1(D2(:,1),D2(:,2),Frequency,'spline');

% Open circuit responce including mikrophone free field correction
cubic3 = cubic1+cubic2;

% Open circuit responce including mikrophone free field correction with
% calibration
Y = 20*log10(Mv)+cubic3;
% Go from dB to mV/Pa
Mv_f = 10.^(Y/20)*1e3;
```

## A.8 plothorizontalpressurefield_basic.m

```
% plothorizontalpressurefield_basic.m
% Fargeplott av trykkfeltet. Fungerar både for PML og uendelege element.
% @author Espen Storheim, 2013(?)
%
% Skriptet er modifisert litt av Eivind Nag Mosland i desember 2020

% Simuleringsnummer, dersom parametrisk
sim = 1;

% Frekvens
f_ind =2;
FFr = result.nearfieldpressure_f{sim};
disp(['Viser trykkfeltet for f = ' num2str(FFr(f_ind)/1e3) ' kHz'])

% modene = find(FFr == f);

% Sett til ein dersom det er ei pml-køyring, eller ein berre vil sjÅ pÅ ...
    dei
% endelege elementa (ikkje det som er rekna ut i dei uendelege).
ispml = 0;

% Hentar ut koordinatane til nodane og det tilh?yrande komplekse ...
    trykkjet.
if ispml
```

```
    P_r = [result.nearfieldpressure_r{sim}.'];
    P_z = [result.nearfieldpressure_z{sim}.'];
    P = [result.nearfieldpressure{sim}(:,f_ind)];
else
    P_r = [result.nearfieldpressure_r{sim}.';result.farfieldpressure_r{...
        sim}];
    P_z = [result.nearfieldpressure_z{sim}.';result.farfieldpressure_z{...
        sim}];
    P = [result.nearfieldpressure{sim}(:,f_ind);result.farfieldpressure{...
        sim}(:,f_ind)];
end

% Ser på dB-magnituda til trykket. Kan også sjå på fasen.
P = 20*log10(abs(P)*mean(pressurefield_result.V1)/(2*20e-6));

NEWSIM_press = 0;

% Delaunay delar matrisa opp i nokre trekantar.
TRI = delaunay(P_z,P_r);

X = [];
Y = [];
C = [];

% Her er det endring frå FEMP originalt. Har bytta om P_z og P_r ...
    samanlikna
% med originalfila. Grunnen til dette er at det er enklare å plotte det
% slik enn å rotere labels og ticks, m.m.
for ii = 1:3
        X(1:size(TRI,1),ii) = P_z(TRI(:,ii));
        Y(1:size(TRI,1),ii) = P_r(TRI(:,ii));
        C(1:size(TRI,1),ii) = P(TRI(:,ii));
end

%% Plotting av feltet.
% Plottar feltet for r => 0.
fig = figure;
h1 = patch(X.',Y.',C.');
%h1 = contourf(X.',Y.',C.');
set(h1,'edgecolor','none');

% Plottar feltet for r <= 0. No har ein dobbelt opp i r = 0.
hold on
h2 = patch(X.',-Y.',C.');
set(h2,'edgecolor','none');
```

```
hold off

%% Colorbar og aksetekstar.
defaxis = gca;
h = colorbar;
ylabel(h, '20log |p| [dB re 1 Pa/V]')
axes(defaxis)
xlabel('Axial distance, z [m]')
ylabel('Lateral distance, r [m]')
axis image

%% Begrensar områda.
xlim([0 0.3])
ylim([-0.301 0.301])
```

## A.9 polarPcolor.m

```
function [varargout] = polarPcolor(R,theta,Z,varargin)
% [h,c] = polarPcolor1(R,theta,Z,varargin) is a pseudocolor plot of ...
    matrix
% Z for a vector radius R and a vector angle theta.
% The elements of Z specify the color in each cell of the
% plot. The goal is to apply pcolor function with a polar grid, which
% provides a better visualization than a cartesian grid.
%
%% Syntax
%
% [h,c] = polarPcolor(R,theta,Z)
% [h,c] = polarPcolor(R,theta,Z,'Ncircles',10)
% [h,c] = polarPcolor(R,theta,Z,'Nspokes',5)
% [h,c] = polarPcolor(R,theta,Z,'Nspokes',5,'colBar',0)
% [h,c] = polarPcolor(R,theta,Z,'Nspokes',5,'labelR','r (km)')
%
% INPUT
%   * R :
%         - type: float
%         - size: [1 x Nrr ] where Nrr = numel(R).
%         - dimension: radial distance.
%   * theta :
%         - type: float
%         - size: [1 x Ntheta ] where Ntheta = numel(theta).
%         - dimension: azimuth or elevation angle (deg).
```

```
%         - N.B.: The zero is defined with respect to the North.
%    * Z :
%         - type: float
%         - size: [Ntheta x Nrr]
%         - dimension: user's defined .
%    * varargin:
%         - Ncircles: number  of circles for the grid definition.
%         - autoOrigin: 'on' (the first circle of the plar grid has a ...
    radius
%           equal to the lowest value of R) or 'off'.
%         - Nspokes: number of spokes for the grid definition.
%         - colBar: display the colorbar or not.
%         - labelR: title for radial axis.
%         - RtickLabel: Tick label for the radial axis.
%         - colormap: Colormap for the pcolor function
%         - ncolor: Number of colors in the colorbar and pcolor
%         - circlesPos: position of the circles with respect to the ...
    origin
%         (it overwrites Ncircles if necessary)
%
%
% OUTPUT
% h: returns a handle to a SURFACE object.
% c: returns a handle to a COLORBAR object.
%
%% Examples
% R = linspace(3,10,100);
% theta = linspace(0,180,360);
% Z = linspace(0,10,360)'*linspace(0,10,100);
% figure
% polarPcolor(R,theta,Z,'Ncircles',3)
%
%% Author
% Etienne Cheynet, University of Stavanger, Norway. 23/10/2019
% see also pcolor
%
%%  InputParseer
p = inputParser();
p.CaseSensitive = false;
p.addOptional('Ncircles',5);
p.addOptional('autoOrigin','on');
p.addOptional('Nspokes',8);
p.addOptional('labelR','');
p.addOptional('RtickLabel',[]);
p.addOptional('colBar',1);
```

```
p.addOptional('Rscale','linear');
p.addOptional('colormap','parula');
p.addOptional('ncolor',[]);
p.addOptional('typeRose','meteo'); % 'meteo' or 'default'
p.addOptional('circlesPos',[]);
p.parse(varargin{:});
Ncircles = p.Results.Ncircles;
Nspokes = p.Results.Nspokes ;
labelR = p.Results.labelR ;
RtickLabel = p.Results.RtickLabel ;
colBar = p.Results.colBar ;
Rscale = p.Results.Rscale ;
autoOrigin = p.Results.autoOrigin ;
myColorMap = p.Results.colormap ;
ncolor = p.Results.ncolor ;
circPos = p.Results.circlesPos ;
typeRose = p.Results.typeRose ;
if ~isempty(circPos)
    Origin = max([min(circPos),min(R)]);
    circPos(circPos<min(R))=[];
    circPos(circPos>max(R))=[];
elseif strcmpi(autoOrigin,'on')
    Origin = min(R);
elseif strcmpi(autoOrigin,'off')
    Origin = 0;
else
    error(' ''autoOrigin'' must be ''on'' or ''of'' ')
end
if Origin==0 && strcmpi(Rscale,'log')
    warning(' The origin cannot be set to 0 if R is expressed on a ...
        logarithmic axis. The value ''Rmin'' is used instead')
    Origin = min(R);
end
if isempty(circPos)
    if ~isempty(RtickLabel)
        if numel(RtickLabel)~=Ncircles
            error(' The radial ticklabel must be equal to Ncircles');
        end
        if any(cellfun(@ischar,RtickLabel)==0)
            error(' The radial ticklabel must be a cell array of ...
                characters');
        end
    end
end
if ~isempty(circPos)
```

```
    circPos = unique([min(R),circPos,max(R)]);
end
%% Preliminary checks
% case where dimension is reversed
Nrr = numel(R);
Noo = numel(theta);
if isequal(size(Z),[Noo,Nrr]) && Noo~=Nrr,
    Z=Z';
end
% case where dimension of Z is not compatible with theta and R
if ~isequal(size(Z),[Nrr,Noo])
    fprintf('\n')
    fprintf([ 'Size of Z is : [',num2str(size(Z)),'] \n']);
    fprintf([ 'Size of R is : [',num2str(size(R)),'] \n']);
    fprintf([ 'Size of theta is : [',num2str(size(theta)),'] \n\n']);
    error(' dimension of Z does not agree with dimension of R and Theta'...
        )
end
%% data plot
rMin = min(R);
rMax = max(R);
thetaMin=min(theta);
thetaMax =max(theta);
if strcmpi(typeRose,'meteo')
    theta = theta;
elseif strcmpi(typeRose,'default')
    theta = 90-theta;
else
    error('"type" must be "meteo" or "default" ');
end
% Definition of the mesh
cax = newplot;
Rrange = rMax - rMin; % get the range for the radius
[rNorm] = getRnorm(Rscale,Origin,R,Rrange); % getRnorm is a nested ...
    function
YY = (rNorm)'*cosd(theta);
XX = (rNorm)'*sind(theta);
h = pcolor(XX,YY,Z,'parent',cax);
if ~isempty(ncolor)
    cmap = feval(myColorMap,ncolor);
    colormap(gca,cmap);
else
    colormap(gca,myColorMap);
end
% disp([max(R/Rrange),max(rNorm)])
```

```
shading flat
set(cax,'dataaspectratio',[1 1 1]);axis off;
if ~ishold(cax);
    % make a radial grid
    hold(cax,'on')
    % Draw circles and spokes
    createSpokes(thetaMin,thetaMax,Ncircles,circPos,Nspokes);
    createCircles(rMin,rMax,thetaMin,thetaMax,Ncircles,circPos,Nspokes)
end
%% PLot colorbar if specified
if colBar==1,
    c =colorbar('location','WestOutside');
    caxis([quantile(Z(:),0.01),quantile(Z(:),0.99)])
else
    c = [];
end
%% Outputs
nargoutchk(0,2)
if nargout==1,
    varargout{1}=h;
elseif nargout==2,
    varargout{1}=h;
    varargout{2}=c;
end
%...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...
% Nested functions
%...
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%...
    function createSpokes(thetaMin,thetaMax,Ncircles,circlesPos,Nspokes)

        spokeMesh = round(linspace(thetaMin,thetaMax,Nspokes));
        if isempty(circlesPos)
            circleMesh = linspace(rMin,rMax,Ncircles);
        else
            circleMesh  =  circlesPos;
        end
        contourD = abs((circleMesh - circleMesh(1))/Rrange+R(1)/Rrange);

        if strcmpi(typeRose,'meteo')
            cost = cosd(90-spokeMesh); % the zero angle is aligned with ...
                North
            sint = sind(90-spokeMesh); % the zero angle is aligned with ...
```

```
                North
        elseif strcmpi(typeRose,'default')
            cost = cosd(spokeMesh); % the zero angle is aligned with ...
                east
            sint = sind(spokeMesh); % the zero angle is aligned with ...
                east
        else
            error('"type" must be "meteo" or "default" ');
        end

        for kk = 1:Nspokes

            X = cost(kk)*contourD;
            Y = sint(kk)*contourD;

            if  Origin==0
                X(1)=Origin;
                Y(1)=Origin;
            end
            plot(X,Y,'color',[0.5,0.5,0.5],'linewidth',0.75,...
                'handlevisibility','off');
            % plot graduations of angles
            % avoid superimposition of 0 and 360
            if and(thetaMin==0,thetaMax == 360),
                if spokeMesh(kk)<360,

                    text(1.05.*contourD(end).*cost(kk),...
                        1.05.*contourD(end).*sint(kk),...
                        [num2str(spokeMesh(kk),3),char(176)],...
                        'horiz', 'center', 'vert', 'middle');
                end
            else
                text(1.05.*contourD(end).*cost(kk),...
                    1.05.*contourD(end).*sint(kk),...
                    [num2str(spokeMesh(kk),3),char(176)],...
                    'horiz', 'center', 'vert', 'middle');
            end

        end
    end
    function createCircles(rMin,rMax,thetaMin,thetaMax,Ncircles,...
        circlePos,Nspokes)

        if isempty(circlePos)
            if Origin ==0 % if the origin is set at rMin
```

```
            contourD = linspace(0,1+R(1)/Rrange,Ncircles);
        else % if the origin is automatically centered at 0
            contourD = linspace(0,1,Ncircles)+R(1)/Rrange;
        end
    else

        contourD = circlePos-circlePos(1);
        contourD = contourD./max(contourD)*max(R/Rrange);
        contourD =[contourD(1:end-1)./contourD(end),1]+R(1)/Rrange;
    end

    if isempty(circlePos)
        if strcmpi(Rscale,'linear')||strcmpi(Rscale,'lin'),
            tickMesh = linspace(rMin,rMax,Ncircles);
        elseif strcmpi(Rscale,'log')||strcmpi(Rscale,'logarithmic'),
            tickMesh  = logspace(log10(rMin),log10(rMax),Ncircles);
        else
            error('''Rscale'' must be ''log'' or ''linear'' ');
        end
    else
        tickMesh  = circlePos;
        Ncircles = numel(tickMesh);
    end

    % define the grid in polar coordinates


    if strcmpi(typeRose,'meteo')
        angleGrid = linspace(90-thetaMin,90-thetaMax,100);
    elseif strcmpi(typeRose,'default')
        angleGrid = linspace(thetaMin,thetaMax,100);
    else
        error('"type" must be "meteo" or "default" ');
    end

    xGrid = cosd(angleGrid);
    yGrid = sind(angleGrid);
    spokeMesh = linspace(thetaMin,thetaMax,Nspokes);

    % plot circles
    for kk=1:length(contourD)
        X = xGrid*contourD(kk);
        Y = yGrid*contourD(kk);
        plot(X,Y,'color',[0.5,0.5,0.5],'linewidth',1);
    end
```

```
        % radius tick label

        position = 0.51.*(spokeMesh(min(Nspokes,round(Ncircles/2)))+...
            spokeMesh(min(Nspokes,1+round(Ncircles/2))));
        if strcmpi(typeRose,'meteo'),position = 90-position; end
        if strcmpi(typeRose,'default') && min(90-theta)<5,position = 0; ...
            end
        if min(round(theta))==90 && strcmpi(typeRose,'meteo'),  position...
            = 0; end
        if max(round(theta))==90 && strcmpi(typeRose,'meteo'),  position...
            = 0; end

        for kk=1:Ncircles
            if isempty(RtickLabel),
                rtick = num2str(tickMesh(kk),2);
            else
                rtick = RtickLabel(kk);
            end

            % radial graduations
            t = text(contourD(kk).*cosd(position),...
                (contourD(kk)).*sind(position),...
                rtick,'verticalalignment','BaseLine',...
                'horizontalAlignment', 'right',...
                'handlevisibility','off','parent',cax);
            if min(round(abs(90-theta)))<5 && strcmpi(typeRose,'default'...
               ),
                t.Position =  t.Position - [0,0.1,0];
                t.Interpreter = 'latex';
                clear t;
            end
            if min(round(theta))==90 && strcmpi(typeRose,'meteo')
                t.Position =  t.Position + [0,0.02,0];
                t.Interpreter = 'latex';
                clear t;
            elseif max(round(theta))==90 && strcmpi(typeRose,'meteo')
                t.Position =  t.Position - [0,0.05,0];
                t.Interpreter = 'latex';
                clear t;
            end

            % annotate spokes
            if max(theta)-min(theta)>180,
                t = text(contourD(end).*1.3.*cosd(position),...
                    contourD(end).*1.3.*sind(position),...
```

```
                    [labelR],'verticalalignment','bottom',...
                    'horizontalAlignment', 'right',...
                    'handlevisibility','off','parent',cax);
            else
                t = text(contourD(end).*0.6.*cosd(position),...
                    contourD(end).*0.6.*sind(position),...
                    [labelR],'verticalalignment','bottom',...
                    'horizontalAlignment', 'right',...
                    'handlevisibility','off','parent',cax);
            end

            t.Interpreter = 'latex';
            if min(round(theta))==90 && strcmpi(typeRose,'meteo'),
                t.Position =  t.Position + [0,0.05,0];
                clear t;
            elseif max(round(theta))==90 && strcmpi(typeRose,'meteo'),
                t.Position =  t.Position + [0,0.05,0];
                clear t;
            end
            %                 if min(round(abs(90-theta)))<5 && strcmpi(...
                typeRose,'default'),
            %                     t.Position =  t.Position - [0,0.12,0];
            %                     t.Interpreter = 'latex';
            %                     clear t;
            %                 end
        end

    end
    function [rNorm] = getRnorm(Rscale,Origin,R,Rrange)
        if strcmpi(Rscale,'linear')||strcmpi(Rscale,'lin')
            rNorm = R-R(1)+Origin;
            rNorm = (rNorm)/max(rNorm)*max(R/Rrange);
        elseif strcmpi(Rscale,'log')||strcmpi(Rscale,'logarithmic')
            if rMin<=0
                error(' The radial vector cannot be lower or equal to 0 ...
                    if the logarithmic scale is used');
            end
            rNorm = log10(R); %normalized radius [0,1]
            rNorm =rNorm-rNorm(1);
            rNorm = (rNorm)/max(rNorm)*max(R/Rrange);
        else
            error('''Rscale'' must be ''log'' or ''linear'' ');
        end
    end
end
```

## A.10 absorption_in_air.m

```
function alpha = absorption_in_air(Frequency,Pressure,RH,Temperature)
%Calculating the absorption coefficient alpha dB/m

% Measure frequency
f=Frequency;
% Paroscientific reads pressure in hPa, coverting it to kPa
p=Pressure/10;
% Relative humidity read by Vaisala
h_rel=RH;

% Triple point isothermal temperature(0.01C)
T_01=273.16;
% Referance pressure in kPa (1atm)
p_ref=101.325;
%Referance temperature in Kelvin (20 degC)
T_ref=293.15;
% Temperature read by ASL f250 konverted to Kelvin
T=Temperature+273.15;

% Molar concentration of water in the air
V=10.79586*(1-(T_01/T))-5.02808.*log10(T/T_01)+1.50474*(10^-4)*(1-10^(-8...
    .29692*(T/T_01-1)))+0.42873*(10^-3)*(-1+10^(4.76955*(1-T_01/T)))-2...
    .2195983;

% Molar humidity
h=h_rel*(10^V)*(p/p_ref)^-1;

% Relaxation frequency for Oxygen
f_rO=(p/p_ref)*(24 + ((4.04*(10^4)*h)*(0.02+h)/(0.391+h)));

% Relaxation frequency for Nitrogen
f_rN=(p/p_ref)*(T/T_ref)^(-0.5)*(9+280*h*exp(-4.170*((T/T_ref)^(-1/3)-1)...
    ));

% Absorption coefficient i dB/m
alpha=8.686*f^2*( (1.84*(10^-11)*(p/p_ref)^-1*(T/T_ref)^(0.5)) + (T/...
    T_ref)^(-5/2)*(0.01275*(exp(-2239.1/T))*(f_rO/(f_rO^2+f^2)) + 0.1068*...
    exp(-3352/T)*(f_rN/(f_rN^2+f^2))));
```

## A.11 Admittance_plotting.m

```
% simulated admittence
load('PZ27_result.mat')
f = result.admittance_f{1}/1000;
YT = result.admittance{1};
YT_log = 20*log10(abs(YT));
% YT_ang = angle(YT);

GT = real(YT);
GT_log = 20*log10(GT);

BT = imag(YT);

load('PZ27_result_fluid.mat')
f_fluid = result.admittance_f{1}/1000;
YT_fluid = result.admittance{1};
YT_log_fluid = 20*log10(abs(YT_fluid));
% YT_ang_fluid = angle(YT_fluid);



GT_fluid = real(YT_fluid);
GT_log_fluid = 20*log10(GT_fluid);

BT_fluid = imag(YT_fluid);

% measured admittance
% load('20220203132417.mat')
% load('20220403155415.mat')
f_m = fr(2:end);
YT_m = g(2:end)+1i*b(2:end);
YT_m_log = 20*log10(abs(YT_m));
% YT_m_ang = angle(YT_m);

GT_m = g(2:end);
GT_m_log = 20*log10(GT_m);

BT_m = b(2:end);

figure
box on
hold on
```

```
plot(f,YT_log,'LineWidth',2,'Color','r')
plot(f_fluid,YT_log_fluid,'--','LineWidth',2,'Color','g')
plot(f_m,YT_m_log,'LineWidth',2,'Color','b')
xlabel('Frequency [kHz]')
ylabel('20log_{10}|Y_T| [dB re. 1 S]')
legend('Simuleted','Simuleted fluid','Measured')
set(gca,'FontSize',22,'FontName','Times New Roman')
ylim([-90 -25])
hold off

figure
hold on
box on
hold on
plot(f,GT_log,'LineWidth',2,'Color','r')
plot(f_fluid,GT_log_fluid,'--','LineWidth',2,'Color','g')
plot(f_m,GT_m_log,'LineWidth',2,'Color','b')
xlabel('Frequency [kHz]')
ylabel('20log_{10}(G_T) [dB re. 1 S]')
legend('Simuleted','Simuleted fluid','Measured')
set(gca,'FontSize',22,'FontName','Times New Roman')
ylim([-110 -25])
hold off

figure
hold on
box on
hold on
plot(f,BT,'LineWidth',2,'Color','r')
plot(f_fluid,BT_fluid,'--','LineWidth',2,'Color','g')
plot(f_m,BT_m,'LineWidth',2,'Color','b')
xlabel('Frequency [kHz]')
ylabel('B_T [S]')
legend('Simuleted','Simuleted fluid','Measured')
set(gca,'FontSize',22,'FontName','Times New Roman')
% ylim([-110 -25])
hold off
```

# Appendix B

# MatLab-app

## B.1 App's startup values

```
properties (Access = public)
    % color for app design
    red = [1,0,0];
    green = [0,1,0];
    black = [0,0,0];
    grey = [0.96,0.96,0.96];

    % indicates if a certain axis is homed or zeroed where rotation
    % axis has no induction sensor activated and therfore rotation axis
    % is always homed
    homed = [0,0,0,1];
    zeroed = [0,0,0,0];

    % the travell distance of y axis had to be callibrated because
    % y step = 10 was not 10mm; y calibration factor
    factor = 1.69027948373533;


    % values indicates if somthing is acitve or not
    initialized = 0;
    initializing = 0;
    connected = 0;
    connecting = 0;

    % if initialize or connecting fails
    initfail = 0;
    connectfail = 0;
```

```

    % other values app use
    stop = 0;
    start = 0;
    stage = 0;
    setup = 0;
    zeroz = 0;
    safety = 0;
    limits = 0;
    comp = 0;
    distance_transducer_microphone = 0;
    distance_transducer_rotaxis = 0;
    transducer_perpendicular_laser = 0;
    transducer_perpendicular_mainlobe = 0;
    tilt_angle_and_distance = 0;
    tilt_angle = 0;

    % empty values for app log and the path for the app
    text = {};
    path = '';


    % Measurements values
    meas = {};
    x = 0;
    wf = 0;
    timeDiv = '';
    maxV = 0;
    result = {};
    time = 0;
    selpath = '';
    measurement = 0;
    meas_el_sig = 0;
    filt_setting = 0;
    gen_setting = 0;
    environment = 0;
    angle = [];
    distance = [];

    % Number of bytes per word (8-bit if 1, 16-bit if 2, ...) when
    % reading oscilloscope
    noB = 2;

    % absolute position or absolute value for the zero position
    zeropos = {};
```

```
    abspos = {};
    field = {'xaxis','yaxis','zaxis','rotaxis'};
    name = {'X-axis','Y-axis','Z-axis','Rotation-axis'};
    linearslope = {};


    % handels that controll devices
    controller = {};
    instrument = {};
end
```

## B.2 Initialize machine function

```
methods (Access = public)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% following function handles the motion controllers and to each
% axis in the air setup
function InitMotor(app)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Skript InitMotor is used to connect to the controllers of X,
    % Y, Z and Rotation-axis in the air setup.
    % X, Y, Z and Rotation-axis can be used to move relatively or
    % absolute values.
    %
    % Espen Fosse, 2021/2022
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % clear handle for controllers
    app.controller = {};

    % Initialisation of LS-270 with HydraTT motor controller
    app.controller.HydraTT = serialport('COM1',115200);
    writeline(app.controller.HydraTT,'version')
    app.controller.HydraTT_idn = readline(app.controller.HydraTT);

    if isstring(app.controller.HydraTT_idn)
        app.controller.HydraTT_idn = strip(app.controller.HydraTT_idn);
        WriteTextInWindow(app,['Connected to: Physik Instrumente, Hydra ...
           TT, V' num2str(app.controller.HydraTT_idn)])
    else
        WriteTextInWindow(app,'Could not connenct to Hydra TT')
        app.initfail = 1;
```

```
        return
    end

    % z axis
    app.controller.HydraTT_LS270 = 1;
    % Initiate Motor restart
    writeline(app.controller.HydraTT,[num2str(...
        app.controller.HydraTT_LS270) ' init']);

    % Sets PID constants for motor
    app.controller.HydraTT_LS270_P = 0;
    app.controller.HydraTT_LS270_I = 0.001;
    app.controller.HydraTT_LS270_D = 0;
    writeline(app.controller.HydraTT,[num2str(...
        app.controller.HydraTT_LS270_P) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(...
        app.controller.HydraTT_LS270_I) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(...
        app.controller.HydraTT_LS270_D) ' 3 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);

    % Sets motor acceleration and velocity
    app.controller.HydraTT_LS270_aks = 2;
    app.controller.HydraTT_LS270_vel = 10;
    writeline(app.controller.HydraTT,[num2str(...
        app.controller.HydraTT_LS270_aks) ' ' num2str(...
        app.controller.HydraTT_LS270) ' sna']);
    writeline(app.controller.HydraTT,[num2str(...
        app.controller.HydraTT_LS270_vel) ' ' num2str(...
        app.controller.HydraTT_LS270) ' snv']);

    % Initialisation of M-535.22, M-531.DG and M-037.PD with C843.41 ...
        motor controller

    % Create Instance of controller class
    app.controller.C843 = C843_GCS_Controller();
    % rotation axis name
    app.controller.C843_M037PD_name = ('M-037.PD');
    % x axis name
    app.controller.C843_M531DG_name = ('M-531.DG');
    % y axis name
    app.controller.C843_M53522_name = ('M-535.22');

```

```
    % Connect using PCI
    app.controller.C843 = app.controller.C843.Connect(1);
    app.controller.C843 = app.controller.C843.InitializeController();

    % Query controller identification
    WriteTextInWindow(app,['Connected to: ' app.controller.C843.qIDN()])
    % Query axes
    app.controller.C843_availableaxes = app.controller.C843.qSAI_ALL();
    if(isempty(app.controller.C843_availableaxes))
        noax2 = 'No available axes on C.843.41 Motor Controller';
        WriteTextInWindow(app,noax2)
        app.initfail = 1;
        return
    end

    % x axis
    app.controller.C843_M531DG = app.controller.C843_availableaxes(4);
    % y axis
    app.controller.C843_M53522 = app.controller.C843_availableaxes(2);
    % rotation axis
    app.controller.C843_M037PD  = app.controller.C843_availableaxes(3);

    % Connect a stage
    % connecting to x axis
    app.controller.C843.CST(app.controller.C843_M531DG,...
       app.controller.C843_M531DG_name);
    % connecting to y axis
    app.controller.C843.CST(app.controller.C843_M53522,...
       app.controller.C843_M53522_name);
    % connecting to rotation axis
    app.controller.C843.CST(app.controller.C843_M037PD,...
       app.controller.C843_M037PD_name);

    % Initialize x and y rotation axis
    app.controller.C843.INI(app.controller.C843_M531DG);
    app.controller.C843.INI(app.controller.C843_M53522);
    app.controller.C843.INI(app.controller.C843_M037PD);

    % set to Acceleration, Deceleration and Velocity of rotation axis
    app.controller.C843.ACC(app.controller.C843_M037PD,0.5)
    app.controller.C843.DEC(app.controller.C843_M037PD,0.5)
    app.controller.C843.VEL(app.controller.C843_M037PD,10)

    %% Configuration values for LS-270

```

```
% Verdier som ikke skal endres. Konfigurasjonsinstillingene skal væ...
    re
% lagret på HYDRA-TT. Skal det være problemer med bevegelse av LS...
    -270
% er det en ide og 'uncomment' denne seksjonen og kjøre InitMotor og
% 'comment' alt i denne seksjonen igjen etterpå.
% (Etter erfaring, så er det greit å alltid ha disse 'uncomment')


% Sets the Vector velocity
writeline(app.controller.HydraTT,[num2str(1) ' sv']);

% Sets the Vector acceleration
writeline(app.controller.HydraTT,[num2str(1) ' sa']);

% Sets the Emergency switch configuration
writeline(app.controller.HydraTT,[num2str(1) ' setemsw']);

% Sets the Hardware limits
writeline(app.controller.HydraTT,[num2str(0) ' ' num2str(1016) ' ' ...
    num2str(app.controller.HydraTT_LS270) ' setnlimit']);

% Sets the Initial limits
writeline(app.controller.HydraTT,[num2str(0) ' ' num2str(1016) ' ' ...
    num2str(app.controller.HydraTT_LS270) ' setinilimit']);

% Sets the Pitch
writeline(app.controller.HydraTT,[num2str(5) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setpitch']);

% Sets the Motor form
writeline(app.controller.HydraTT,[num2str(0) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setmotor']);

% Sets the Motor current shift value
writeline(app.controller.HydraTT,[num2str(0.092540) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setMCShift']);

% Sets the Stop deceleration
writeline(app.controller.HydraTT,[num2str(74) ' ' num2str(...
    app.controller.HydraTT_LS270) ' ssd']);

% Sets the FRT parameters
writeline(app.controller.HydraTT,[num2str(0) ' 1 ' num2str(...
    app.controller.HydraTT_LS270) ' setfrtpara']);
```

```
    writeline(app.controller.HydraTT,[num2str(0) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setfrtpara']);

    % Sets the Calibration velocity
    writeline(app.controller.HydraTT,[num2str(10) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setncalvel']);
    writeline(app.controller.HydraTT,[num2str(0.1) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setncalvel']);

    % Sets the Range measure velocity
    writeline(app.controller.HydraTT,[num2str(10) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setnrmvel']);
    writeline(app.controller.HydraTT,[num2str(0.1) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setnrmvel']);

    % Sets the Reference velocity
    writeline(app.controller.HydraTT,[num2str(1) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setnrefvel']);
    writeline(app.controller.HydraTT,[num2str(0.1) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setnrefvel']);

    % Sets the Motion function
    writeline(app.controller.HydraTT,[num2str(7) ' ' num2str(...
        app.controller.HydraTT_LS270) ' setmotionfunc']);

    % Sets the Position origin configuration
    writeline(app.controller.HydraTT,[num2str(1) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setorgconfig']);

    % Sets Reference configuration
    writeline(app.controller.HydraTT,[num2str(0) ' ' num2str(...
        app.controller.HydraTT_LS270) ' setref']);

    % Sets the Calibration switch distance
    writeline(app.controller.HydraTT,[num2str(0) ' ' num2str(...
        app.controller.HydraTT_LS270) ' setncalswdist']);

    % Sets the Motor voltage minimum
    writeline(app.controller.HydraTT,[num2str(3.6) ' ' num2str(...
        app.controller.HydraTT_LS270) ' setumotmin']);

    % Sets the Motor voltage gradient
    writeline(app.controller.HydraTT,[num2str(7) ' ' num2str(...
        app.controller.HydraTT_LS270) ' setumotgrad']);

```

```
% Sets the Motor current limit
writeline(app.controller.HydraTT,[num2str(5) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setmaxcurrent']);

% Sets the Motor pole pairs
writeline(app.controller.HydraTT,[num2str(100) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setpolepairs']);

% Sets the Motor phase number
writeline(app.controller.HydraTT,[num2str(2) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setphases']);

% Sets the Positioning control mode
writeline(app.controller.HydraTT,[num2str(2) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setcloop']);

% Sets the Target window
writeline(app.controller.HydraTT,[num2str(0.0002) ' ' num2str(0.0002...
    ) ' ' num2str(app.controller.HydraTT_LS270) ' setclwindow']);

% Sets the Time on target
writeline(app.controller.HydraTT,[num2str(0.02) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setclwintime']);

% Sets the Scale period
writeline(app.controller.HydraTT,[num2str(-0.02) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setclperiod']);

% Sets the Position display selection
writeline(app.controller.HydraTT,[num2str(1) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setselpos']);

% Configures the Motor brake control function
writeline(app.controller.HydraTT,[num2str(0) ' 0 ' num2str(...
    app.controller.HydraTT_LS270) ' setbrakefunc']);
writeline(app.controller.HydraTT,[num2str(3) ' 1 ' num2str(...
    app.controller.HydraTT_LS270) ' setbrakefunc']);

% Enables or disables the Clock and direction function
writeline(app.controller.HydraTT,[num2str(0) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setcdfunc']);

% Sets the Clock and direction width
writeline(app.controller.HydraTT,[num2str(250) ' ' num2str(...
    app.controller.HydraTT_LS270) ' setcdwidth']);
```

```
    % Configures the Servo control
    writeline(app.controller.HydraTT,[num2str(100) ' 4 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(0) ' 5 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(24) ' 6 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(1) ' 7 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(1) ' 8 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);
    writeline(app.controller.HydraTT,[num2str(0.0000) ' 9 ' num2str(...
        app.controller.HydraTT_LS270) ' setsp']);

    % Configures the Adaptive positioning control
    writeline(app.controller.HydraTT,[num2str(0) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0.025) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0) ' 3 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0) ' 4 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0.1) ' 5 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0) ' 6 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0) ' 7 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0.0001) ' 8 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);
    writeline(app.controller.HydraTT,[num2str(0) ' 9 ' num2str(...
        app.controller.HydraTT_LS270) ' setadaptive']);

    % Configures the Auto commutation
    writeline(app.controller.HydraTT,[num2str(0) ' 1 ' num2str(...
        app.controller.HydraTT_LS270) ' setamc']);
    writeline(app.controller.HydraTT,[num2str(0) ' 2 ' num2str(...
        app.controller.HydraTT_LS270) ' setamc']);
    writeline(app.controller.HydraTT,[num2str(1) ' 3 ' num2str(...
        app.controller.HydraTT_LS270) ' setamc']);

    % Sets the specified Motor parameters entry
    writeline(app.controller.HydraTT,[num2str(0) ' 1 ' num2str(...
```

```
            app.controller.HydraTT_LS270) ' setmotorpara']);
        writeline(app.controller.HydraTT,[num2str(1) ' 2 ' num2str(...
            app.controller.HydraTT_LS270) ' setmotorpara']);
        writeline(app.controller.HydraTT,[num2str(0.9) ' 3 ' num2str(...
            app.controller.HydraTT_LS270) ' setmotorpara']);
        writeline(app.controller.HydraTT,[num2str(0.0025) ' 4 ' num2str(...
            app.controller.HydraTT_LS270) ' setmotorpara']);
        writeline(app.controller.HydraTT,[num2str(10) ' 5 ' num2str(...
            app.controller.HydraTT_LS270) ' setmotorpara']);
        writeline(app.controller.HydraTT,[num2str(1) ' 6 ' num2str(...
            app.controller.HydraTT_LS270) ' setmotorpara']);

        % Lagrer parameterene for opsettet som er i bruk
        writeline(app.controller.HydraTT,'csave');
        writeline(app.controller.HydraTT,'1 nsave');
end
```

## B.3 Homing function

```
%
function HomeMotor(app,a)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script HomeMotor is made so that each stage can find its
    % absolute zero position.
    %
    % a:     which stage is to be homed
    %        a = (1, 'x' or 'X') for X-axis              (M-531.DG)
    %        a = (2, 'y' or 'Y') for Y-axis              (M-535.22)
    %        a = (3, 'z' or 'Z') for Z-axis              (LS-270)
    %        a = (4, 'r' or 'R') for Rotation-axis       (M-037.PD)
    %
    % Espen Fosse, 2021/2022
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    if a == 1 || strcmp(a,'x') || strcmp(a,'X')
        ax = app.controller.C843_M531DG;
        a = 1;
    elseif a == 2 || strcmp(a,'y') || strcmp(a,'Y')
        ax = app.controller.C843_M53522;
        a = 2;
    elseif a == 3 || strcmp(a,'z') || strcmp(a,'Z')
        ax = app.controller.HydraTT_LS270;
```

```
        a = 3;
    elseif a == 4 || strcmp(a,'r') || strcmp(a,'R')
        ax = app.controller.C843_M037PD;
        a = 4;
    else
        noaxwtn = 'No axis with that name';
        WriteTextInWindow(app,noaxwtn)
        return
    end

    if ax == app.controller.HydraTT_LS270
        cmd = [num2str(ax) ' nrm'];
        writeline(app.controller.HydraTT,cmd);
        WriteTextInWindow(app,[app.name{a} ' is traveling Home'])
        MovingMotor(app,a)
        writeline(app.controller.HydraTT,['1016 ' num2str(ax) ' setnpos'...
            ])
        if app.stop == 1
            WriteTextInWindow(app,[app.name{a} ' stopped Home'])
        else
            WriteTextInWindow(app,[app.name{a} ' is now Home'])
        end
        PositionMotor(app,a);
    elseif  ax == app.controller.C843_M037PD
        app.controller.C843.MOV(ax,0)
        WriteTextInWindow(app,[app.name{a} ' is traveling Home'])
        MovingMotor(app,a)
        if app.stop == 1
            WriteTextInWindow(app,[app.name{a} ' stopped Home'])
        else
            WriteTextInWindow(app,[app.name{a} ' is now Home'])
        end
        PositionMotor(app,a);
    else
        app.controller.C843.FNL(ax)
        WriteTextInWindow(app,[app.name{a} ' is traveling Home'])
        MovingMotor(app,a)
        if app.stop == 1
            WriteTextInWindow(app,[app.name{a} ' stopped Home'])
        else
            WriteTextInWindow(app,[app.name{a} ' is now Home'])
        end
        PositionMotor(app,a);
    end
end
```

```
%
```

## B.4 Step function

```
function StepMotor(app,s,a,m)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Skript StepMotor that is used to move X, Y, Z og
    % Rotation-axis in the air setup.
    %
    % s:    amount of steps in [mm] or [degrees]; s = 1 is the same
    %       as 1[degree] or 1[mm].
    % a:    axis that is going to move is determed by
    %       a = (1, 'x' or 'X') for X-axis              (M-531.DG)
    %       a = (2, 'y' or 'Y') for Y-axis              (M-535.22)
    %       a = (3, 'z' or 'Z') for Z-axis              (LS-270)
    %       a = (4, 'r' or 'R') for Rotation-axis       (M-037.PD)
    % m:    movments is relative m = 0; movments is absolute m = 1
    %
    % Espen Fosse, 2021/2022
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if a == 1 || strcmp(a,'x') || strcmp(a,'X')
        ax = app.controller.C843_M531DG;
        a = 1;
    elseif a == 2 || strcmp(a,'y') || strcmp(a,'Y')
        ax = app.controller.C843_M53522;
        a = 2;
    elseif a == 3 || strcmp(a,'z') || strcmp(a,'Z')
        ax = app.controller.HydraTT_LS270;
        a = 3;
    elseif a == 4 || strcmp(a,'r') || strcmp(a,'R')
        ax = app.controller.C843_M037PD;
        a = 4;
    else
        noaxwtn = 'No axis with that name';
        WriteTextInWindow(app,noaxwtn)
        return
    end

    if a == 3
        if (m==0)
            cmd = [num2str(s) ' ' num2str(ax) ' nr'];
            writeline(app.controller.HydraTT,cmd);
```

```
            txt = [app.name{a} ' is moving ' num2str(s) '[mm]'];
            WriteTextInWindow(app,txt)
            MovingMotor(app,a)
            PositionMotor(app,a);
        elseif (m==1)
            cmd = [num2str(s) ' ' num2str(ax) ' nm'];
            writeline(app.controller.HydraTT,cmd);
            if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS')
                txt = [app.name{a} ' is moving to ' num2str(s) '[mm]: M ...
                    COORD''S'];
            else
                s2 = s-app.zeropos.(app.field{a});
                txt = [app.name{a} ' is moving to ' num2str(s2) '[mm]: U...
                     COORD''S'];
            end
            WriteTextInWindow(app,txt)
            MovingMotor(app,a)
            PositionMotor(app,a);
        else
            WriteTextInWindow(app,'Wrong value for movement type: ...
                relative movements m=0; absolute movements m=1')
            return
        end
    else
        if a == 2
            s = s*app.factor;
        end
        if (m==0)
            app.controller.C843.MVR(ax,s)
            if a == 2
                s = s/app.factor;
            end
            if a == 4
                WriteTextInWindow(app,[app.name{a} ' is moving ' num2str...
                    (s) '[degree]'])
            else
                WriteTextInWindow(app,[app.name{a} ' is moving ' num2str...
                    (s) '[mm]'])
            end
            MovingMotor(app,a)
            PositionMotor(app,a);
        elseif (m==1)
            app.controller.C843.MOV(ax,s)
            if a == 2
                s = s/app.factor;
```

```
            end
            if a == 4
                if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS...
                    ')
                    txt = [app.name{a} ' is moving to ' num2str(s) '[...
                        degree]: M COORD''S'];
                else
                    s2 = s-app.zeropos.(app.field{a});
                    txt = [app.name{a} ' is moving to ' num2str(s2) '[...
                        degree]: U COORD''S'];
                end
                WriteTextInWindow(app,txt)
            else
                if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS...
                    ')
                    txt = [app.name{a} ' is moving to ' num2str(s) '[mm...
                        ]: M COORD''S'];
                else
                    s2 = s-app.zeropos.(app.field{a});
                    txt = [app.name{a} ' is moving to ' num2str(s2) '[mm...
                        ]: U COORD''S'];
                end
                WriteTextInWindow(app,txt)
            end
            MovingMotor(app,a)
            PositionMotor(app,a);
        else
            WriteTextInWindow(app,'Wrong value for movement type: ...
                relative movements m=0; absolute movements m=1')
            return
        end
    end
end
%
```

## B.5 Position function

```
function PositionMotor(app,a)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script PositionMotor tells the absolute position of axis
    %
    % a:    which axis to read position
```

```
%       a = (1, 'x' or 'X') for X-axis              (M-531.DG)
%       a = (2, 'y' or 'Y') for Y-axis              (M-535.22)
%       a = (3, 'z' or 'Z') for Z-axis              (LS-270)
%       a = (4, 'r' or 'R') for Rotation-axis       (M-037.PD)
%
% Espen Fosse, 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if a == 1 || strcmp(a,'x') || strcmp(a,'X')
    ax = app.controller.C843_M531DG;
    a = 1;
elseif a == 2 || strcmp(a,'y') || strcmp(a,'Y')
    ax = app.controller.C843_M53522;
    a = 2;
elseif a == 3 || strcmp(a,'z') || strcmp(a,'Z')
    ax = app.controller.HydraTT_LS270;
    a = 3;
elseif a == 4 || strcmp(a,'r') || strcmp(a,'R')
    ax = app.controller.C843_M037PD;
    a = 4;
else
    noax = 'No axis with that name';
    WriteTextInWindow(app,noax)
    return
end

if a == 3
    pos = writeread(app.controller.HydraTT,[num2str(ax) ' np']);
    pos = str2double(strip(pos));
    if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS')
        txt = [app.name{a} ' is at ' num2str(pos) '[mm]: M COORD''S'...
            ];
    elseif app.zeroed(a) == 0
        txt = [app.name{a} ' is at ' num2str(pos) '[mm]: U COORD''S'...
            ];
    else
        pos2 = pos-app.zeropos.(app.field{a});
        txt = [app.name{a} ' is at ' num2str(pos2) '[mm]: U COORD''S...
            '];
    end
    WriteTextInWindow(app,txt)
    app.abspos.(app.field{a}) = pos;
elseif a == 4
    pos = app.controller.C843.qPOS(ax);
    if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS')
```

```
            txt = [app.name{a} ' is at ' num2str(pos) '[degree]: M COORD...
                ''S'];
        elseif app.zeroed(a) == 0
            txt = [app.name{a} ' is at ' num2str(pos) '[degree]: U COORD...
                ''S'];
        else
            pos2 = pos-app.zeropos.(app.field{a});
            txt = [app.name{a} ' is at ' num2str(pos2) '[degree]: U ...
                COORD''S'];
        end
        WriteTextInWindow(app,txt)
        app.abspos.(app.field{a}) = pos;
    else
        pos = app.controller.C843.qPOS(ax);
        if a == 2
            pos = pos/app.factor;
        end
        if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS')
            txt = [app.name{a} ' is at ' num2str(pos) '[mm]: M COORD''S'...
                ];
        elseif app.zeroed(a) == 0
            txt = [app.name{a} ' is at ' num2str(pos) '[mm]: U COORD''S'...
                ];
        else
            pos2 = pos-app.zeropos.(app.field{a});
            txt = [app.name{a} ' is at ' num2str(pos2) '[mm]: U COORD''S...
                '];
        end
        WriteTextInWindow(app,txt)
        app.abspos.(app.field{a}) = pos;
    end
end
%
```

## B.6 Moving function

```
function MovingMotor(app,a)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script makes MatLab/app wait until axis that moves is reach its
    % destination.
    %
    % a:    which axis that is moving
```

```
%       a = (1, 'x' or 'X') for X-axis              (M-531.DG)
%       a = (2, 'y' or 'Y') for Y-axis              (M-535.22)
%       a = (3, 'z' or 'Z') for Z-axis              (LS-270)
%       a = (4, 'r' or 'R') for Rotation-axis       (M-037.PD)
%
% Espen Fosse, 2021/2022
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if a == 1 || strcmp(a,'x') || strcmp(a,'X')
    ax = app.controller.C843_M531DG;
elseif a == 2 || strcmp(a,'y') || strcmp(a,'Y')
    ax = app.controller.C843_M53522;
elseif a == 3 || strcmp(a,'z') || strcmp(a,'Z')
    ax = app.controller.HydraTT_LS270;
elseif a == 4 || strcmp(a,'r') || strcmp(a,'R')
    ax = app.controller.C843_M037PD;
else
    noax = 'No axis with that name';
    WriteTextInWindow(app,noax)
    return
end

if a == 3
    while true
        pause(0.1)
        writeline(app.controller.HydraTT,[num2str(ax) ' nst']);
        beveger_seg = readline(app.controller.HydraTT);
        number = str2double(beveger_seg);
        if app.stop == 1
            return
        elseif (number ==32 || number ==36)
            return
        else
            continue
        end
    end

else
    while(app.controller.C843.IsMoving(ax))
        pause(0.1);
        if app.stop == 1
            return
        end
    end
    % pause set because IsMoving(ax) kan make matlab freeze
```

```
        pause(0.25);
    end

end
%
```

## B.7 Instrument connect

```
% following function handels the instruments and measurements
function InstrumentConnect(app)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Script InstrumentConnect creates a handle to the instruments
    % that is a part of the air setup
    %
    % Espen Storheim, 2011
    % Based on work by Vidar Knappskog and Magne Aanes.
    % Modified by Espen Fosse to be used in app, 2021/2022.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % clear handle of instruments
    app.instrument = {};
    ActionButtons(app)
    set(app.STOP_Button,'Enable','Off')
    set(app.INSTRUMENTCONNECT_Button,'Enable','On')

    %% Signal Generator Agilent 33220A. S/N:

    % Find a GPIB object.
    app.instrument.generator = instrfind('Type', 'gpib', 'BoardIndex', ...
       0, 'PrimaryAddress', 10, 'Tag', '');

    % Create the GPIB object if it does not exist
    % otherwise use the object that was found.
    if isempty(app.instrument.generator)
        app.instrument.generator = gpib('NI', 0, 10);
    else
        fclose(app.instrument.generator);
        app.instrument.generator = app.instrument.generator(1);
    end

    fopen(app.instrument.generator);
    app.instrument.generator_name = 'Agilent 33220A. S/N: ';
```

```
    app.instrument.generator_idn = query(app.instrument.generator,'*IDN?...
        ');

    % Test the connection. Should be a command where the response can be
    % verified.
    if isempty(app.instrument.generator_idn)
        WriteTextInWindow(app,'Warning: The signal generator is not ...
            connected or configured properly.')
        app.connectfail = 1;
        return
    else
        WriteTextInWindow(app,'1: The signal generator is connected and ...
            appears to be working.')
    end

    %% Digital Oscilloscope Tektronix DPO3012. S/N:

    app.instrument.scope = visadev("USB0::0x0699::0x0410::C024017::0::...
        INSTR");
%               app.instrument.scope = visadev("USB0::0x0699::0x0410::...
    C024018::0::INSTR");
%               app.instrument.scope = visadev("USB0::0x0699::0x0410::...
    C011044::0::INSTR");
%               app.instrument.scope = visadev('USB0::0x0699::0x0410::...
    C010246::0::INSTR'); %problems, it freezes matlab
    app.instrument.scope_name = 'Tektronix DPO3012. S/N: ';
    app.instrument.scope_idn = writeread(app.instrument.scope,'*IDN?');

    % Test the connection. Should be a command where the response can be
    % verified.
    if isempty(app.instrument.scope_idn)
        WriteTextInWindow(app,'Warning: The oscilloscope is not ...
            connected or configured properly.')
        app.connectfail = 1;
        return
    else
        WriteTextInWindow(app,'2: The oscilloscope is connected and ...
            appears to be working.')
    end

    %% Pressure sensor: Paroscientific DigiQuartz 740. S/N:

    app.instrument.pressure = visadev('COM13');
    app.instrument.pressure_name = 'Paroscientific DigiQuartz 740. S/N:'...
        ;
```

```
    app.instrument.pressure_idn = writeread(app.instrument.pressure,'...
        *0100P3');

    % Test the connection. Should be a command where the response can be
    % verified.
    if isempty(app.instrument.pressure_idn)
        WriteTextInWindow(app,'Warning: The paroscientific is not ...
            connected or configured properly.')
        app.connectfail = 1;
        return
    else
        WriteTextInWindow(app,'3: The paroscientific is connected and ...
            appears to be working.')
    end

    %% Relative humidity and temperature sensor: Vaisala HMT313. S/N:

    app.instrument.humidity = visadev('COM7');
    configureTerminator(app.instrument.humidity,"CR")
    app.instrument.humidity_name = 'Vaisala HMT313. S/N:';
    app.instrument.humidity_idn = writeread(app.instrument.humidity,'...
        send');

    % Test the connection. Should be a command where the response can be
    % verified.
    if isempty(app.instrument.humidity_idn)
        WriteTextInWindow(app,'Warning: The vasisala is not connected or...
             configured properly.')
        app.connectfail = 1;
        return
    else
        WriteTextInWindow(app,'4: The vasisala is connected and appears ...
            to be working.')
    end

    %% Bandpass filter: Krohn-Hite 3940A. S/N: AM2626.

    % Find a GPIB object.
    app.instrument.filter = instrfind('Type', 'gpib', 'BoardIndex', 0, '...
        PrimaryAddress', 25, 'Tag', '');

    % Create the GPIB object if it does not exist
    % otherwise use the object that was found.
    if isempty(app.instrument.filter)
        app.instrument.filter = gpib('NI', 0, 25);
```

```
    else
        fclose(app.instrument.filter);
        app.instrument.filter = app.instrument.filter(1);
    end
    fopen(app.instrument.filter);
    app.instrument.filter_name = 'Krohn-Hite 3940A. S/N: AM2626';
    app.instrument.filter_idn = query(app.instrument.filter,'*IDN?');

    % Test the connection. Should be a command where the response can be
    % verified.
    if isempty(app.instrument.filter_idn)
        WriteTextInWindow(app,'Warning: The filter is not connected or ...
            configured properly.')
        app.connectfail = 1;
        return
    else
        WriteTextInWindow(app,'5: The filter is connected and appears to...
             be working.')
    end

    %% Temperature sensor: ASL F250. S/N:
    % Find a GPIB object.
    app.instrument.temperature = instrfind('Type', 'gpib', 'BoardIndex',...
         0, 'PrimaryAddress', 3, 'Tag', '');

    % Create the GPIB object if it does not exist
    % otherwise use the object that was found.
    if isempty(app.instrument.temperature)
        app.instrument.temperature = gpib('NI', 0, 3);
    else
        fclose(app.instrument.temperature);
        app.instrument.temperature = app.instrument.temperature(1);
    end

    fopen(app.instrument.temperature);
    set(app.instrument.temperature,'EOSmode','read&write');
    set(app.instrument.temperature,'EOSCharCode',10); % Set terminator ...
       to LF.
    app.instrument.temperature_name = 'ASL F250 mk II. S/N: ';
    fprintf(app.instrument.temperature,'A0');
    app.instrument.temperature_idn = fscanf(app.instrument.temperature);

    % Test the connection. Should be a command where the response can be
    % verified.
    if isempty(app.instrument.temperature_idn)
```

```
            WriteTextInWindow(app,'Warning: The thermometer is not connected...
                 or configured properly.')
            app.connectfail = 1;
            return
        else
            WriteTextInWindow(app,'6: The thermometer is connected and ...
                appears to be working.')
        end
end
%
```

## B.8 Initialize instruments

```
function InitInstruments(app)
    % Initialize the bandpass filter.
    % There seems to be an overflow when the commands are combined, so ...
        they
    % have been separated and a pause of 100 ms is set between each
    % command.
    %
    % Set the input and output gain on both channels to 0 dB.
    txt = 'Initialize Bandpass Filter';
    WriteTextInWindow(app,txt)
    fprintf(app.instrument.filter,'AL;0IG;0OG;B');
    pause(0.1)
    % Set channel 1 to high pass mode.
    fprintf(app.instrument.filter,'CH1.1;M2');
    pause(0.1)
    % Set the cutoff frequency for channel 1.
    fprintf(app.instrument.filter,['F' num2str(app.result.cutoff_f1) 'K'...
        ]);
    pause(0.1)
    % Set channel 2 to low pass mode.
    fprintf(app.instrument.filter,'CH1.2;M1');
    pause(0.1)
    % Set the cutoff frequency for channel 2.
    fprintf(app.instrument.filter,['F' num2str(app.result.cutoff_f2) 'K'...
        ]);

    % Initialize the oscilloscope.
    txt = 'Initialize Oscilloscope';
    WriteTextInWindow(app,txt)
```

```
% Code for the Tektronix DPO3012.
% Set the acquisition mode to averaging.
writeline(app.instrument.scope,'ACQ:MOD AVE');
% Set the number of cycles to average.
writeline(app.instrument.scope,['ACQ:NUMAV ' num2str(...
    app.meas.average)]);
% Number of points which shall be read from the scope.
writeline(app.instrument.scope,['HOR:RECO ' num2str(...
    app.meas.sample_count)]);
% Start point for the recorded signal
writeline(app.instrument.scope,'DAT:START 1');
% Stop point for the recorded signal
writeline(app.instrument.scope,['DAT:STOP ' num2str(...
    app.meas.sample_count)]);
% Trigger specifications. Set to edge detection from external ...
    source.
writeline(app.instrument.scope,'TRIG:A:EDGE:SOU EXT');
% Set the trigger type to positive edge.
writeline(app.instrument.scope,'TRIG:A:TYP EDG');
% 2012.11.19 EM: Added additional initialization.
% CH1
% Set Offset to zero.
writeline(app.instrument.scope,'CH1:OFFS 0');
% Set position to zero.
writeline(app.instrument.scope,'CH1:POS 0');
% Set coupling to AC.
writeline(app.instrument.scope,'CH1:COUP AC');
% CH2
% Set Offset to zero.
writeline(app.instrument.scope,'CH2:OFFS 0');
% Set position to zero.
writeline(app.instrument.scope,'CH2:POS 0');
% Set coupling to AC.
writeline(app.instrument.scope,'CH2:COUP AC');
%  Number of bytes per word (8-bit if 1, 16-bit if 2, ...)
writeline(app.instrument.scope,['DATA:ENCDG SRIBINARY;WIDTH ' ...
    num2str(app.noB)])


% Initialize Signal Generator
txt = 'Initialize Signal Generator';
WriteTextInWindow(app,txt)
fprintf(app.instrument.generator,'BM:STATe on'); % burst mode
fprintf(app.instrument.generator,['BM:INT:RATE ' num2str(...
    app.meas.burst_rate)]);
```

```
    fprintf(app.instrument.generator,['BM:NCYC ', num2str(...
        app.result.sig_cycles)]);
    fprintf(app.instrument.generator,['FREQ ', num2str(...
        app.result.frequency)]);
    fprintf(app.instrument.generator,['VOLT ' num2str(...
        app.meas.voltage_inn)]);
end
%
```

## B.9 Measure function

```
function Measure(app)
    % Voltage scalings in oscilloscope, unit: V/div
    voltageScalings = [1e-3, 2e-3, 5e-3, 10e-3, 20e-3, 50e-3, 100e-3, ...
        200e-3, 500e-3 1 2 5 10];
    % The are 8 visible divisons on the screen and two above/below ...
        outside,
    % i.e., 10 in total. Maximum amplitude thus corresponds to
    % 5*verticalScalings, but is set to 4.5 to be on the save side
    NumVerDivs_max = 4.5;
    NumVerDivs_min = 1.7;

    % Time scalings in oscilloscope, unit: mus/div
    timeScalings = [40 100 200 400 1000]*1e-6;
    maxNumHorDivs = 10;

    % Set scope acquisition mode to SEQUENCE instead of RUNSTOP, so that...
         a
    % measurement is aquired when prompted, instead of continuously. See...
         page
    % 2-97 in programming manual for details.
    writeline(app.instrument.scope,'ACQ:STOPA SEQ');

    txt = ['Measure frequency ' num2str(app.result.frequency/1000) 'kHz'...
        ];
    WriteTextInWindow(app,txt)


    if app.gen_setting == 1
        txt = 'Adjusting signal generator settings';
        WriteTextInWindow(app,txt)
        % Common settings for electrical and acoustical signal
```

```
        % Update signal generator settings
        fprintf(app.instrument.generator,['BM:NCYC ', num2str(...
            app.result.sig_cycles)]);
        fprintf(app.instrument.generator,['FREQ ', num2str(...
            app.result.frequency)]);
        % Give little time to signal generator to set the setting
        pause(0.1)
    end

    % Find appropriate horizontal scaling & update time/Div
    tScaling = timeScalings(find(timeScalings*maxNumHorDivs >= ...
        app.result.sig_duration*1.5,1));
    writeline(app.instrument.scope,['HOR:SCA ',num2str(tScaling)]);

    if app.meas_el_sig == 1

        txt = 'Starting measurements of the electrical pulses';
        WriteTextInWindow(app,txt)

        %% Electrical signal (transmitter excitation)

        % Set the delay of acquisition data so that the resulting ...
            waveform is
        % centered tScaling*5 after the trigger occurs. (See 2-234 in ...
            manual)
        writeline(app.instrument.scope,['HOR:DEL:TIM ',num2str(tScaling...
            *5)]);

        % Find appropriate vertical scaling for transmitter excitation ...
            signal
        vScaling = voltageScalings(find(NumVerDivs_max*voltageScalings ...
            >= app.meas.voltage_inn,1));
        % Set voltage/div
        writeline(app.instrument.scope,['CH',num2str(...
            app.meas.channel_electrical),':SCA ' num2str(vScaling)]);

        % Set the number of cycles to average.
        writeline(app.instrument.scope,['ACQ:NUMAV ' num2str(...
            app.meas.average)]);
        % Number of points which shall be read from the scope.
        writeline(app.instrument.scope,['HOR:RECO ' num2str(...
            app.meas.sample_count)]);

        % Start aquisition.
        writeline(app.instrument.scope,'ACQ:STATE RUN');
```

```
        txt = 'Starting aquisition';
        WriteTextInWindow(app,txt)
        pause(app.meas.average_time)
        DPO_read(app,app.meas.channel_electrical);
        app.result.electric_t = app.x;
        app.result.electric = app.wf;
        app.result.electric_timescale = app.timeDiv;
        app.result.electric_maxV = app.maxV;
        app.result.electric_Vscale = str2double(writeread(...
            app.instrument.scope,['CH',num2str(...
            app.meas.channel_electrical),':SCA?']));
        app.result.electric_Termination = str2double(writeread(...
            app.instrument.scope,['CH',num2str(...
            app.meas.channel_electrical),':TER?']));
        txt = 'Finished reading the electrical pulses';
        WriteTextInWindow(app,txt)
    end
    %% Acoustical signal (receiving transducer)

    txt = 'Starting measurements of the acoustical pulses';
    WriteTextInWindow(app,txt)

    if app.filt_setting == 1
        txt = 'Adjusting filter settings';
        WriteTextInWindow(app,txt)
        % Adjust the bandwidth of the KH-filter
        % Set the cutoff frequency for filter channel 1. (Not working ...
            properly)
        fprintf(app.instrument.filter,['CH1.1;F' num2str(...
            app.result.cutoff_f1) 'K']);
        pause(0.2)
        % Set the cutoff frequency for channel 2.
        fprintf(app.instrument.filter,['CH1.2;F' num2str(...
            app.result.cutoff_f2) 'K']);
        % Give little time to filter to set the setting
        pause(0.1)
    end

    % Set the delay of acquisition data so that the resulting waveform ...
        is
    % centered [tScaling*5 + (under)estimated plane wave travel time] ...
        after
    % the trigger occurs. (See 2-234 in manual)
    writeline(app.instrument.scope,['HOR:DEL:TIM ',num2str(tScaling*5-0...
        .5*tScaling+app.result.est_travel_time)]);
```

```

    if app.meas.average_scaling == 1
        % Set aquisition mode to single sample instead of averaging
        % This speeds things up, and ensures that the noise-prior-to-...
            averaging
        % is not clipped, which could distort the averaged signals. See ...
            meas.
        % setup chapter in Hauge (2013) or Mosland (2013) for details
        writeline(app.instrument.scope,'ACQ:MOD SAM');
    else
        % Set the number of cycles to average.
        writeline(app.instrument.scope,['ACQ:NUMAV ' num2str(...
            app.meas.average_scaling)]);
    end
    % Number of points which shall be read from the scope.
    writeline(app.instrument.scope,['HOR:RECO ' num2str(...
        app.meas.sample_count)]);

    % Find appropriate vertical scaling for receiver signal
    writeline(app.instrument.scope,'ACQ:STATE RUN');
    pause(app.meas.average_time_scaling)
    DPO_read(app,app.meas.channel_acoustical);
    Scaling = str2double(writeread(app.instrument.scope,['CH',num2str(...
        app.meas.channel_acoustical),':SCA?']));

    ind = find(Scaling==voltageScalings);
    if isempty(ind)
        txt = 'ind is empty!';
        WriteTextInWindow(app,txt)
        ind = 1;
        writeline(app.instrument.scope,['CH',num2str(...
            app.meas.channel_acoustical),':SCA ',num2str(voltageScalings(...
            ind))]);
        % Start aquisition.
        writeline(app.instrument.scope,'ACQ:STATE RUN');
        txt = 'Checking scaling';
        WriteTextInWindow(app,txt)
        pause(app.meas.average_time_scaling)
        DPO_read(app,app.meas.channel_acoustical);
    end

    scaling_down = 0;
    finished = 0;
    while ~finished
        % Debugging, app.Vmax or ind is empty inside the while loop
```

```
        if isempty(app.maxV)
            % Start aquisition.
            txt = 'Max voltage not found. Trying to find max voltage...'...
                ;
            WriteTextInWindow(app,txt)
            writeline(app.instrument.scope,'ACQ:STATE RUN');
            pause(app.meas.average_time_scaling)
            DPO_read(app,app.meas.channel_acoustical);
            if ~isempty(app.maxV)
                txt = 'Max voltage found';
                WriteTextInWindow(app,txt)
            end
            continue
        elseif isempty(ind)
            txt = 'ind is empty!';
            WriteTextInWindow(app,txt)
            ind = 1;
            writeline(app.instrument.scope,['CH',num2str(...
                app.meas.channel_acoustical),':SCA ',num2str(...
                voltageScalings(ind))]);
            % Start aquisition.
            writeline(app.instrument.scope,'ACQ:STATE RUN');
            pause(app.meas.average_time_scaling)
            DPO_read(app,app.meas.channel_acoustical);

        end
        % Continue on finding the correct amplitude scaling
        if app.maxV >= voltageScalings(ind)*NumVerDivs_max
            txt = 'Increasing scaling';
            WriteTextInWindow(app,txt)
            Scaling = voltageScalings(ind+1);
            writeline(app.instrument.scope,['CH',num2str(...
                app.meas.channel_acoustical),':SCA ',num2str(Scaling)]);
            ind = ind +1;
            % Start aquisition.
            writeline(app.instrument.scope,'ACQ:STATE RUN');
            pause(app.meas.average_time_scaling)
            DPO_read(app,app.meas.channel_acoustical);

        elseif ind ~= 1 && app.maxV <= voltageScalings(ind)*...
            NumVerDivs_min
            txt = 'Decreasing scaling';
            WriteTextInWindow(app,txt)
            Scaling = voltageScalings(ind-1);
            writeline(app.instrument.scope,['CH',num2str(...
```

```
                app.meas.channel_acoustical),':SCA ',num2str(Scaling)]);
            ind = ind -1;
            % Start aquisition.
            writeline(app.instrument.scope,'ACQ:STATE RUN');
            pause(app.meas.average_time_scaling)
            DPO_read(app,app.meas.channel_acoustical);
            scaling_down = scaling_down + 1;
            % Prevent while loop to scale up and down over, and
            % over again.
            if scaling_down == 20
                if isempty(app.Vmax)
                    txt = 'Max Voltage is empty';
                    WriteTextInWindow(app,txt)
                    scaling_down = scaling_down-1;
                    continue
                end
                txt = 'Scaling caught in a loop. The lowest scaling ...
                    value is selected';
                WriteTextInWindow(app,txt)
                break
            end
        else
            txt = 'Correct scaling';
            WriteTextInWindow(app,txt)
            finished = 1;
        end
    end

    if app.meas.average_scaling == 1
        % Reset aquisition mode to averaging
        writeline(app.instrument.scope,'ACQ:MOD AVE');
    end

    % Set the number of cycles to average.
    writeline(app.instrument.scope,['ACQ:NUMAV ' num2str(...
        app.meas.average)]);
    % Number of points which shall be read from the scope.
    writeline(app.instrument.scope,['HOR:RECO ' num2str(...
        app.meas.sample_count)]);

    % Start aquisition.
    writeline(app.instrument.scope,'ACQ:STATE RUN');
    txt = 'Starting aquisition';
    WriteTextInWindow(app,txt)
    pause(app.meas.average_time)
```

```
    DPO_read(app,app.meas.channel_acoustical);

    app.result.acoustic_t = app.x;
    app.result.acoustic = app.wf;
    app.result.acoustic_timescale = app.timeDiv;
    app.result.acoustic_maxV = app.maxV;
    app.result.acoustic_Vscale = str2double(writeread(...
        app.instrument.scope,['CH',num2str(app.meas.channel_acoustical),'...
        :SCA?']));
    app.result.acoustic_Termination = str2double(writeread(...
        app.instrument.scope,['CH',num2str(app.meas.channel_acoustical),'...
        :TER?']));
    txt = 'Finished reading the acoustical pulses';
    WriteTextInWindow(app,txt)
end
%
```

## B.10 DPO read function

```
function DPO_read(app,ch)
    % Set data source
    writeline(app.instrument.scope,['DAT:SOUR CH' num2str(ch)]);

    % Set what samples to retrieve
    writeline(app.instrument.scope,'DAT:START 1');
    writeline(app.instrument.scope,['DAT:STOP ' num2str(...
        app.meas.sample_count)]);

    % Read the data
    writeline(app.instrument.scope,'CURV?');

    if app.noB == 2
        ydata = readbinblock(app.instrument.scope,'int16');
        txt = 'Reading ydata 16-bit';
        WriteTextInWindow(app,txt)
    elseif app.noB == 1
        ydata = readbinblock(app.instrument.scope,'int8');
        txt = 'Reading ydata 8-bit';
        WriteTextInWindow(app,txt)
    else
        WriteTextInWindow(app,'Unsupported word length');
    end
```

```

    % Flush the termination character from the scope
    flush(app.instrument.scope);
    txt = 'Reading additional data';
    WriteTextInWindow(app,txt)

    % Horizontal scaling
    app.timeDiv = str2double(writeread(app.instrument.scope,'HOR:SCA?'))...
        ;
    % Horizontal offset
    xze = str2double(writeread(app.instrument.scope,'WFMO:XZE?'));
    % Horizontal increment
    xin = str2double(writeread(app.instrument.scope,'WFMO:XIN?'));
    % Digital vertical offset
    YOF = str2double(writeread(app.instrument.scope,'WFMO:YOF?'));
    % Vertical multiplying factor
    YMU = str2double(writeread(app.instrument.scope,'WFMO:YMU?'));
    % Vertical offset
    YZE = str2double(writeread(app.instrument.scope,'WFMO:YZE?'));
    % Voltage/current vector
    app.wf = (ydata-YOF)*YMU + YZE;
    % Time vector
    app.x = (0:(length(ydata)-1))*xin + xze;
    % max voltage
    app.maxV = max(abs(app.wf));
end
%
```

## B.11 Environmental measurements

```
function VaisalaHMT313_read(app)
    txt = 'Reading Vaisala temperature and humidity';
    WriteTextInWindow(app,txt)
    app.result.vaisala = writeread(app.instrument.humidity,'send');
    app.result.vaisala = regexp(app.result.vaisala,'\d+.\d*','Match');
    app.result.vaisala_RH = str2double(app.result.vaisala(1));
    app.result.vaisala_T = str2double(app.result.vaisala(2));
end
%
```

```
function Paroscientific(app)
```

```
    txt = 'Reading Paroscientific pressure';
    WriteTextInWindow(app,txt)
    app.result.pressure = writeread(app.instrument.pressure,'*0100P3');
    app.result.pressure = str2double(app.result.pressure{1}(6:end-1));
end
%
```

```
function ASLF250(app)
    txt = 'Reading ASL F250 temperature';
    WriteTextInWindow(app,txt)
    fprintf(app.instrument.temperature,'A0');
    pause(0.1)
    app.result.temperature = fscanf(app.instrument.temperature);
    app.result.temperature = regexp(app.result.temperature,'\d+.\d*','...
       Match');
    app.result.temperature = str2double(app.result.temperature(1));
end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

## B.12 Setup functions

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% following function handles transducer and microphone posistion so
% you can find user/zero coordinates
function ATL(app,y)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Angle of transducer relativt to laser
    %
    % y = 0:    manually measure left and right side of transducer
    %           and make it perpendicular to laser.
    %
    % y = 1:    manually measure top and bottom of transducer to
    %           find angle of transducer tilted upp or downward.
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    system('C:\Program Files (x86)\KEYENCE\LK-Navigator\LK-Navigator.exe...
        &');
    system('Taskkill /IM cmd.exe');
    if y == 0
        inputvalues = inputdlg({'Front left side of transducer','Front ...
           right side of transducer','Radius of transducer'},'Measured ...
```

```
        average value [mm]',[1,50],{num2str(0);num2str(0);num2str(10)...
        });
        if isempty(inputvalues)
            return
        elseif cellfun('isempty',inputvalues(1)) || cellfun('isempty',...
            inputvalues(2)) || cellfun('isempty',inputvalues(3))
            return
        else
            inputvalues = cellfun(@str2num,inputvalues);
            theta = asind((inputvalues(2)-inputvalues(1))/(2*inputvalues...
                (3)));
            StepMotorFunc(app,theta,4,0)
            if app.stop == 1
                app.stop = 0;
                return
            end
            ZERO_R_ButtonPushed(app, matlab.ui.control.Button)
        end
    elseif y == 1
        inputvalues = inputdlg({'Top of transducer','Buttom of ...
            transducer','Radius of transducer'},'Measured average value [...
            mm]',[1,50],{num2str(0);num2str(0);num2str(10)});
        if isempty(inputvalues)
            return
        elseif cellfun('isempty',inputvalues(1)) || cellfun('isempty',...
            inputvalues(2)) || cellfun('isempty',inputvalues(3))
            return
        else
            inputvalues = cellfun(@str2num,inputvalues);
            app.angle = asind((inputvalues(2)-inputvalues(1))/(2*...
                inputvalues(3)));
        end
        if app.angle == 0
            txt = 'Transducer is perpendicular laser';
            WriteTextInWindow(app,txt)
        elseif app.angle < 0
            txt = ['Transducer is tilted ' num2str(abs(app.angle)) ' ...
                degrees downward'];
            WriteTextInWindow(app,txt)
        elseif app.angle > 0
            txt = ['Transducer is tilted ' num2str(abs(app.angle)) ' ...
                degrees uppward'];
            WriteTextInWindow(app,txt)
        end
    end
```

```
end
%
function DTRax(app)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Distance from transducer middlepoint to rotation axis
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    system('C:\Program Files (x86)\KEYENCE\LK-Navigator\LK-Navigator.exe...
         &');
    system('Taskkill /IM cmd.exe');
    inputvalues = inputdlg({'Front left side of transducer','Front right...
         side of transducer','Radius of transducer'},'Measured average ...
        value [mm]',[1,50],{num2str(0);num2str(0);num2str(10)});
    if isempty(inputvalues)
        return
    elseif cellfun('isempty',inputvalues(1)) || cellfun('isempty',...
        inputvalues(2)) || cellfun('isempty',inputvalues(3))
        return
    else
        inputvalues = cellfun(@str2num,inputvalues);
        theta = asind((inputvalues(1)-inputvalues(2))/(2*inputvalues(3))...
            );
    end

    inputvalues = inputdlg('Measure with laser in middle of transducer',...
        'Measured average value [mm]',[1,50]);
    if cellfun('isempty',inputvalues)
        return
    else
        inputvalues = cellfun(@str2num,inputvalues);
        d1 = inputvalues;
    end

    alpha = -10;
    StepMotorFunc(app,alpha,4,0)
    if app.stop == 1
        app.stop = 0;
        return
    end

    inputvalues = inputdlg({'Measure at point after move transducer','...
        Move and measure with laser in middle of transducer'},'Measured ...
        average value [mm]',[1,50]);
    if isempty(inputvalues)
        return
    elseif cellfun('isempty',inputvalues(1)) || cellfun('isempty',...
```

```
            inputvalues(2))
            return
        else
            inputvalues = cellfun(@str2num,inputvalues);
            d2 = inputvalues(1);
            d3 = inputvalues(2);
        end

        StepMotorFunc(app,-alpha,4,0)
        if app.stop == 1
            app.stop = 0;
            return
        end

        zpitch = 0.7;
        xpitch = 0.5;

        a = (d2-d3)/tand(theta+alpha);
        phi = atand(a/(d1-d3));
        h = (d1-d3)/cosd(phi);
        gamma = phi + alpha/2;
        r = (h/2)/sind(alpha/2);
        z_mov = sind(gamma)*r;
        x_mov = cosd(gamma)*r;
        txt = ['Transducer are at X =' num2str(x_mov) '[mm] and Z =' num2str...
            (z_mov) '[mm] from rotation axis'];
        WriteTextInWindow(app,txt)

        rotate_x = x_mov/xpitch;
        rotate_z = z_mov/zpitch;
        wholerot_x = fix(rotate_x);
        wholerot_z = fix(rotate_z);

        restrot_x = round(360*abs(rotate_x-wholerot_x));
        restrot_z = round(360*abs(rotate_z-wholerot_z));

        if rotate_x > 0
            if wholerot_x == 0
                txt_1 = ['Rotate X knob ' num2str(restrot_x) ' degrees CW.'...
                    ];
                WriteTextInWindow(app,txt_1)
            elseif abs(wholerot_x) == 1
                txt_1 = ['Rotate X knob ' num2str(abs(wholerot_x)) ' time CW...
                    , and aditional ' num2str(restrot_x) ' degrees.'];
                WriteTextInWindow(app,txt_1)
```

```
        else
            txt_1 = ['Rotate X knob ' num2str(abs(wholerot_x)) ' times ...
                CW, and aditional ' num2str(restrot_x) ' degrees.'];
            WriteTextInWindow(app,txt_1)
        end
    else
        if wholerot_x == 0
            txt_1 = ['Rotate X knob ' num2str(restrot_x) ' degrees CCW.'...
                ];
            WriteTextInWindow(app,txt_1)
        elseif abs(wholerot_x) == 1
            txt_1 = ['Rotate X knob ' num2str(abs(wholerot_x)) ' time ...
                CCW, and aditional ' num2str(restrot_x) ' degrees.'];
            WriteTextInWindow(app,txt_1)
        else
            txt_1 = ['Rotate X knob ' num2str(abs(wholerot_x)) ' times ...
                CCW, and aditional ' num2str(restrot_x) ' degrees.'];
            WriteTextInWindow(app,txt_1)
        end
    end

    if rotate_z > 0
        if wholerot_z == 0
            txt_2 = ['Rotate Z knob ' num2str(restrot_z) ' degrees CCW.'...
                ];
            WriteTextInWindow(app,txt_2)
        elseif abs(wholerot_z) == 1
            txt_2 = ['Rotate Z knob ' num2str(abs(wholerot_z)) ' time ...
                CCW, and aditional ' num2str(restrot_z) ' degrees.'];
            WriteTextInWindow(app,txt_2)
        else
            txt_2 = ['Rotate Z knob ' num2str(abs(wholerot_z)) ' times ...
                CCW, and aditional ' num2str(restrot_z) ' degrees.'];
            WriteTextInWindow(app,txt_2)
        end
    else
        if wholerot_z == 0
            txt_2 = ['Rotate Z knob ' num2str(restrot_z) ' degrees CW.'...
                ];
            WriteTextInWindow(app,txt_2)
        elseif abs(wholerot_z) == 1
            txt_2  = ['Rotate Z knob ' num2str(abs(wholerot_z)) ' time ...
                CW, and aditional ' num2str(restrot_z) ' degrees.'];
            WriteTextInWindow(app,txt_2)
        else
```

```
            txt_2 = ['Rotate Z knob ' num2str(abs(wholerot_z)) ' times ...
                CW, and aditional ' num2str(restrot_z) ' degrees.'];
            WriteTextInWindow(app,txt_2)
        end
    end
    questdlg({txt_1,txt_2},'Manually move the transducer','Ok','Ok')
end
%
function DTM(app)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Distance from transducer to microphone/transducer
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    system('C:\Program Files (x86)\KEYENCE\LK-Navigator\LK-Navigator.exe...
         &');
    system('Taskkill /IM cmd.exe');

    inputvalues = inputdlg({'Value one at transducer','Value two at ...
       microphone/transducer'},'Measured average value [mm]',[1,50]);
    if isempty(inputvalues)
        return
    elseif cellfun('isempty',inputvalues(1)) || cellfun('isempty',...
       inputvalues(2))
        return
    else
        inputvalues = cellfun(@str2num,inputvalues);
        app.distance = 30 + 30 + 182.5692 - inputvalues(1) - inputvalues...
            (2);
        value = app.abspos.(app.field{3})-app.distance;
        txt = ['Dictance from transducer to microphone/transducer is ...
           measured to be ' num2str(app.distance) '[mm]'];
        WriteTextInWindow(app,txt)

        choice = questdlg(['Do you want current position to ' app.name...
            {1} ' to be zero?'],'Zero Position','Yes','No','No');
        switch choice
            case 'Yes'
                SaveZero(app,1,'')
        end
        choice = questdlg(['Do you want current position to ' app.name...
            {2} ' to be zero?'],'Zero Position','Yes','No','No');
        switch choice
            case 'Yes'
                SaveZero(app,2,'')
        end
        SaveZero(app,3,value)
```

```
    end
end
%
function MLY(app)
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Main Lobe on y axis, linear slope
    % constants
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    if app.zeroed(3) == 0
        txt = 'Find zero z posistion with Setup 3, 4 or load zero.mat';
        WriteTextInWindow(app,txt)
    elseif app.zeroed(2) == 0
        txt = 'Find zero y posistion with Setup 3, 4 or load zero.mat';
        WriteTextInWindow(app,txt)
    else
        if isempty(app.angle)
            txt = 'No angle is found';
            WriteTextInWindow(app,txt)
            return
        elseif isempty(app.distance)
            txt = 'No distance from transducer to microphone/transducer ...
                is found';
            WriteTextInWindow(app,txt)
            return
        else
            delta_y = tand(app.angle)*app.distance;
            app.linearslope.a = delta_y/app.distance;
            app.linearslope.b = app.zeropos.(app.field{2});
            SaveLinearSlope(app)
        end
    end
end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

## B.13 Overall functions

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% following functions handles different needs in the app
function ResetButtons(app)
    buttons = [app.MACHINECOORDS_Button,app.GO_TO_ZERO_Button,...
        app.GO_TO_ZERO_X_Button,app.GO_TO_ZERO_Y_Button,...
```

```
        app.GO_TO_ZERO_Z_Button,app.GO_TO_ZERO_R_Button,...
        app.ZERO_ALL_Button,app.ZERO_X_Button,app.ZERO_Y_Button,...
        app.ZERO_Z_Button,app.ZERO_R_Button,app.POSITEIVESTEP_X_Button,...
        app.POSITEIVESTEP_Y_Button,app.POSITEIVESTEP_Z_Button,...
        app.POSITEIVESTEP_R_Button,app.NEGATIVESTEP_X_Button,...
        app.NEGATIVESTEP_Y_Button,app.NEGATIVESTEP_Z_Button,...
        app.NEGATIVESTEP_R_Button,app.HOME_ALL_Button,app.HOME_X_Button,...
        app.HOME_Y_Button,app.HOME_Z_Button,app.HOME_R_Button,...
        app.INITISLIZEMACHINE_Button,app.INSTRUMENTCONNECT_Button,...
        app.SETUP_Button,app.LOAD_Button,app.START_Button,app.STOP_Button...
        ,app.READ_DropDown,app.READ_Button];
    set(buttons,'Enable','Off')
    if app.initialized == 0
        set(buttons(25),'Enable','On')
    elseif ~all(app.homed)
        set(buttons,'Enable','On')
        movebuttons = [app.POSITEIVESTEP_X_Button,...
            app.NEGATIVESTEP_X_Button;app.POSITEIVESTEP_Y_Button,...
            app.NEGATIVESTEP_Y_Button;app.POSITEIVESTEP_Z_Button,...
            app.NEGATIVESTEP_Z_Button;app.POSITEIVESTEP_R_Button,...
            app.NEGATIVESTEP_R_Button];
        for i = 1:length(app.homed)
            if app.homed(i) == 0
                set(movebuttons(i,:),'Enable','Off')
            else
                set(movebuttons(i,:),'Enable','On')
            end
        end
    else
        set(buttons,'Enable','On')
    end
    if app.connected == 0
        set(buttons(31),'Enable','Off')
        set(buttons(32),'Enable','Off')
    end
end
%
function ResetColor(app)
    color = [app.ZERO_ALL_Button,app.ZERO_X_Button,app.ZERO_Y_Button,...
        app.ZERO_Z_Button,app.ZERO_R_Button];
    for i = 1:length(app.zeroed)
        if app.zeroed(i) == 0
            set(color(1,(i+1)),'BackgroundColor',app.red)
        else
            set(color(1,(i+1)),'BackgroundColor',app.grey)
```

```
        end
    end
    if all(app.zeroed == 1)
        set(color(1,1),'BackgroundColor',app.grey)
    else
        set(color(1,1),'BackgroundColor',app.red)
    end
end
%
function ActionButtons(app)
    buttons = [app.MACHINECOORDS_Button,app.GO_TO_ZERO_Button,...
       app.GO_TO_ZERO_X_Button,app.GO_TO_ZERO_Y_Button,...
       app.GO_TO_ZERO_Z_Button,app.GO_TO_ZERO_R_Button,...
       app.ZERO_ALL_Button,app.ZERO_X_Button,app.ZERO_Y_Button,...
       app.ZERO_Z_Button,app.ZERO_R_Button,app.POSITEIVESTEP_X_Button,...
       app.POSITEIVESTEP_Y_Button,app.POSITEIVESTEP_Z_Button,...
       app.POSITEIVESTEP_R_Button,app.NEGATIVESTEP_X_Button,...
       app.NEGATIVESTEP_Y_Button,app.NEGATIVESTEP_Z_Button,...
       app.NEGATIVESTEP_R_Button,app.HOME_ALL_Button,app.HOME_X_Button,...
       app.HOME_Y_Button,app.HOME_Z_Button,app.HOME_R_Button,...
       app.INITISLIZEMACHINE_Button,app.INSTRUMENTCONNECT_Button,...
       app.SETUP_Button,app.LOAD_Button,app.START_Button,app.STOP_Button...
       ,app.READ_DropDown,app.READ_Button];
    set(buttons,'Enable','Off')
    set(app.STOP_Button,'Enable','On')
end
%
function ActionLamp(app,onoff,a)
    lamp = [app.X_Lamp,app.Y_Lamp,app.Z_Lamp,app.R_Lamp];
    if onoff
        set(lamp(a),'Color',app.green);
    else
        set(lamp(a),'Color',app.black);
    end
end
%
function WritePosition(app)
    value = [app.X_EditField,app.Y_EditField,app.Z_EditField, ...
        app.R_EditField];
    if app.MACHINECOORDS_Lamp.Color == app.green
        for i = 1:4
            value(i).Value = app.abspos.(app.field{i});
        end
    else
        for i = 1:4
```

```
            if app.zeroed(i) == 1
                value(i).Value = app.abspos.(app.field{i})-app.zeropos.(...
                    app.field{i});
            else
                value(i).Value = app.abspos.(app.field{i});
            end
        end
    end
end
%
function WriteTextInWindow(app,txt)
    app.text = app.TextArea.Value;
    if cellfun(@isempty,app.text)
        app.text{1} = txt;
        app.TextArea.Value = app.text;
    else
        if length(app.text) >= 100
            app.text = app.text(2:end);
        end
        app.text{length(app.text)+1} = txt;
        app.TextArea.Value = app.text;
    end
    scroll(app.TextArea, 'bottom')


    if not(isfolder([app.path '\App_LOG']))
        mkdir([app.path '\App_LOG'])
    end
    celltext{1} = txt;
    if ~isfile([app.path '\App_LOG\TextWindow' app.time '.txt'])
        writecell(celltext, [app.path '\App_LOG\TextWindow' app.time '...
            .txt'], 'QuoteStrings',false);
    else
        writecell(celltext, [app.path '\App_LOG\TextWindow' app.time '...
            .txt'], 'QuoteStrings',false,'WriteMode','append')
    end
end
%
function HomeMotorFunc(app,a)
    if app.stop == 1
        WritePosition(app)
        return
    end
    ActionButtons(app)
    ActionLamp(app,true,a)
```

```
    app.safety = 1;
    while app.safety == 1
        HomeMotor(app,a)
        app.safety = 0;
    end
    if app.stop == 1
        WritePosition(app)
        return
    end
    ActionLamp(app,false,a)
    app.homed(a) = 1;
    WritePosition(app)
    ResetButtons(app)
end
%
function StepMotorFunc(app,s,a,m)
    if app.homed(a) == 0
        txt = [app.name{a} ' is not homed'];
        WriteTextInWindow(app,txt)
        return
    end

    if app.stop == 1
        WritePosition(app)
        return
    end

    softlim.xaxis = [0,300];
    softlim.yaxis = [0,300];
    softlim.zaxis = [0,1016];

    if a ~=4
        pos = app.abspos.(app.field{a});
        lowerlim = softlim.(app.field{a})(1);
        upperlim = softlim.(app.field{a})(2);
    end

    if a == 4
        ActionButtons(app)
        ActionLamp(app,true,a)
        app.safety = 1;
        while app.safety == 1
            StepMotor(app,s,a,m)
            app.safety =  0;
        end
```

```
        if app.stop == 1
            WritePosition(app)
            return
        end
        ActionLamp(app,false,a)
        WritePosition(app)
        ResetButtons(app)
    elseif m == 0
        if s < 0 && pos+s < lowerlim
            ActionButtons(app)
            ActionLamp(app,true,a)
            app.safety = 1;
            while app.safety == 1
                StepMotor(app,lowerlim,a,1)
                app.safety =  0;
            end
            if app.stop == 1
                WritePosition(app)
                return
            end
            ActionLamp(app,false,a)
            WritePosition(app)
            ResetButtons(app)
            txt = 'Reached lower limit';
            WriteTextInWindow(app,txt)
        elseif s > 0 && pos+s > upperlim
            ActionButtons(app)
            ActionLamp(app,true,a)
            app.safety = 1;
            while app.safety == 1
                StepMotor(app,upperlim,a,1)
                app.safety =  0;
            end
            if app.stop == 1
                WritePosition(app)
                return
            end
            ActionLamp(app,false,a)
            WritePosition(app)
            ResetButtons(app)
            txt = 'Reached upper limit';
            WriteTextInWindow(app,txt)
        else
            ActionButtons(app)
            ActionLamp(app,true,a)
```

```
                app.safety = 1;
                while app.safety == 1
                    StepMotor(app,s,a,m)
                    app.safety =  0;
                end
                if app.stop == 1
                    WritePosition(app)
                    return
                end
                ActionLamp(app,false,a)
                WritePosition(app)
                ResetButtons(app)
            end
        else
            if s <= lowerlim
                ActionButtons(app)
                ActionLamp(app,true,a)
                app.safety = 1;
                while app.safety == 1
                    StepMotor(app,lowerlim,a,1)
                    app.safety =  0;
                end
                if app.stop == 1
                    WritePosition(app)
                    return
                end
                ActionLamp(app,false,a)
                WritePosition(app)
                ResetButtons(app)
                txt = 'Reached lower limit';
                WriteTextInWindow(app,txt)
            elseif s >= upperlim
                ActionButtons(app)
                ActionLamp(app,true,a)
                app.safety = 1;
                while app.safety == 1
                    StepMotor(app,upperlim,a,1)
                    app.safety =  0;
                end
                if app.stop == 1
                    WritePosition(app)
                    return
                end
                ActionLamp(app,false,a)
                WritePosition(app)
```

```
                ResetButtons(app)
                txt = 'Reached upper limit';
                WriteTextInWindow(app,txt)
            else
                ActionButtons(app)
                ActionLamp(app,true,a)
                app.safety = 1;
                while app.safety == 1
                    StepMotor(app,s,a,m)
                    app.safety =  0;
                end
                if app.stop == 1
                    WritePosition(app)
                    return
                end
                ActionLamp(app,false,a)
                WritePosition(app)
                ResetButtons(app)
            end
        end
end
%
function SaveZero(app,a,value)
    if isempty(value)
        if a == 2
            app.linearslope.b = app.abspos.(app.field{a});
            SaveLinearSlope(app)
        end
        app.zeropos.(app.field{a}) = app.abspos.(app.field{a});
        zero = app.zeropos;
        if isfile([app.path '\zero.mat'])
            save([app.path '\zero.mat'],'-struct','zero','-append')
        else
            save([app.path '\zero.mat'],'-struct','zero')
        end
        app.zeroed(a) = 1;
        WritePosition(app)
        ResetColor(app)
    else
        if a == 2
            app.linearslope.b = value;
            SaveLinearSlope(app)
        end
        app.zeropos.(app.field{a}) = value;
        zero = app.zeropos;
```

```
            if isfile([app.path '\zero.mat'])
                save([app.path '\zero.mat'],'-struct','zero','-append')
            else
                save([app.path '\zero.mat'],'-struct','zero')
            end
            app.zeroed(a) = 1;
            WritePosition(app)
            ResetColor(app)
        end
    end
    %
    function SaveLinearSlope(app)
        slope = app.linearslope;
        if isfile([app.path '\slope.mat'])
            save([app.path '\slope.mat'],'-struct','slope','-append')
        else
            save([app.path '\slope.mat'],'-struct','slope')
        end
    end
    %
    function SaveMeasurements(app)
        if strcmpi(app.meas.notes,'Main Lobe on y-axis')
            filename_1 = [app.meas.date '_Main_Lobe_Y'];
        elseif strcmpi(app.meas.notes,'Main Lobe on x-axis')
            filename_1 = [app.meas.date '_Main_Lobe_X'];
        else
            filename_1 = [app.meas.date '_' app.meas.name];
        end
        newfolder_1 = [app.selpath '\' filename_1];
        if ~isfolder(newfolder_1)
            mkdir(newfolder_1)
        end
        if app.measurement == 1
            txt = 'Saving Measurement Parameters';
            WriteTextInWindow(app,txt)
            savepath_1 = [newfolder_1 '\measurement_parameters.mat'];
            m =  matfile(savepath_1,'Writable',true);
            m.parameters = app.meas;
        end
        if strcmpi(app.meas.notes,'Directivity')
            filename_2 = ['Primary_axis_' num2str(...
                app.result.primary_axis_pos) '[mm]'];
            newfolder_2 = [newfolder_1 '\' filename_2];
            if ~isfolder(newfolder_2)
                mkdir(newfolder_2)
```

```
        end
        filename_3 = ['Frequency_' num2str(app.result.frequency) '[Hz]'...
            ];
        newfolder_3 = [newfolder_2 '\' filename_3];
        if ~isfolder(newfolder_3)
            mkdir(newfolder_3)
        end
        txt = 'Saving Results';
        WriteTextInWindow(app,txt)
        if strcmpi(app.result.secondary_axis,'rotaxis')
        savepath_2 = [newfolder_3 '\' num2str(...
            app.result.secondary_axis_pos) '[degree].mat'];
        else
            savepath_2 = [newfolder_3 '\' num2str(...
                app.result.secondary_axis_pos) '[mm].mat'];
        end
        n =  matfile(savepath_2,'Writable',true);
        n.results = app.result;
    elseif strcmpi(app.meas.notes,'On Axis Pressure')
        filename_2 = ['Frequency_' num2str(app.result.frequency) '[Hz]'...
            ];
        newfolder_2 = [newfolder_1 '\' filename_2];
        if ~isfolder(newfolder_2)
            mkdir(newfolder_2)
        end
        if strcmpi(app.result.secondary_axis,'rotaxis')
            filename_3 = ['Secondary_axis' num2str(...
                app.result.secondary_axis_pos) '[degree]'];
        else
            filename_3 = ['Secondary_axis' num2str(...
                app.result.secondary_axis_pos) '[mm]'];
        end
        newfolder_3 = [newfolder_2 '\' filename_3];
        if ~isfolder(newfolder_3)
            mkdir(newfolder_3)
        end
        txt = 'Saving Results';
        WriteTextInWindow(app,txt)
        savepath_2 = [newfolder_3 '\' num2str(...
            app.result.primary_axis_pos) '[mm].mat'];
        n =  matfile(savepath_2,'Writable',true);
        n.results = app.result;
    elseif strcmpi(app.meas.notes,'Sensitivity')
        filename_2 = ['Primary_axis_' num2str(...
            app.result.primary_axis_pos) '[mm]'];
```

```
        newfolder_2 = [newfolder_1 '\' filename_2];
        if ~isfolder(newfolder_2)
            mkdir(newfolder_2)
        end
        if strcmpi(app.result.secondary_axis,'rotaxis')
            filename_3 = ['Secondary_axis' num2str(...
                app.result.secondary_axis_pos) '[degree]'];
        else
            filename_3 = ['Secondary_axis' num2str(...
                app.result.secondary_axis_pos) '[mm]'];
        end
        newfolder_3 = [newfolder_2 '\' filename_3];
        if ~isfolder(newfolder_3)
            mkdir(newfolder_3)
        end
        txt = 'Saving Results';
        WriteTextInWindow(app,txt)
        savepath_2 = [newfolder_3 '\' num2str(app.result.frequency) '[Hz...
            ].mat'];
        n =  matfile(savepath_2,'Writable',true);
        n.results = app.result;
    elseif strcmpi(app.meas.notes,'Main Lobe on y-axis')
        filename_2 = ['Primary_axis_' num2str(...
            app.result.primary_axis_pos) '[mm]'];
        newfolder_2 = [newfolder_1 '\' filename_2];
        if ~isfolder(newfolder_2)
            mkdir(newfolder_2)
        end
        filename_3 = ['Frequency_' num2str(app.result.frequency) '[Hz]'...
            ];
        newfolder_3 = [newfolder_2 '\' filename_3];
        if ~isfolder(newfolder_3)
            mkdir(newfolder_3)
        end
        txt = 'Saving Results';
        WriteTextInWindow(app,txt)
        if strcmpi(app.result.secondary_axis,'rotaxis')
        savepath_2 = [newfolder_3 '\' num2str(...
            app.result.secondary_axis_pos) '[degree].mat'];
        else
            savepath_2 = [newfolder_3 '\' num2str(...
                app.result.secondary_axis_pos) '[mm].mat'];
        end
        n =  matfile(savepath_2,'Writable',true);
        n.results = app.result;
```

```
    elseif strcmpi(app.meas.notes,'Main Lobe on x-axis')
        filename_2 = ['Primary_axis_' num2str(...
            app.result.primary_axis_pos) '[mm]'];
        newfolder_2 = [newfolder_1 '\' filename_2];
        if ~isfolder(newfolder_2)
            mkdir(newfolder_2)
        end
        filename_3 = ['Frequency_' num2str(app.result.frequency) '[Hz]'...
            ];
        newfolder_3 = [newfolder_2 '\' filename_3];
        if ~isfolder(newfolder_3)
            mkdir(newfolder_3)
        end
        txt = 'Saving Results';
        WriteTextInWindow(app,txt)
        if strcmpi(app.result.secondary_axis,'rotaxis')
        savepath_2 = [newfolder_3 '\' num2str(...
            app.result.secondary_axis_pos) '[degree].mat'];
        else
            savepath_2 = [newfolder_3 '\' num2str(...
                app.result.secondary_axis_pos) '[mm].mat'];
        end
        n =  matfile(savepath_2,'Writable',true);
        n.results = app.result;
    end
end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

## B.14 Button functions

```
% Callbacks that handle component events
methods (Access = private)

% Code that executes after component creation
function startupFcn(app)
    % At start up, resets all buttons
    app.path = fileparts(mfilename('fullpath'));
    addpath(app.path)
    app.time = strcat(datestr(now,'yyyy_mm_dd_HH_MM'));
    ResetButtons(app)
    ResetColor(app)
```

```
end

% Button pushed function: INITISLIZEMACHINE_Button
function INITISLIZEMACHINE_ButtonPushed(app, event)
    % If you accedentaly hit INIT button, you have the ability to
    % stop
    if app.initializing == 1
        return
    elseif app.initialized == 1
        choice = questdlg('WARNING: Do you want to initialize again?','...
            WARNING','Yes','No','No');
        switch choice
            case 'No'
                return
        end
    end

    app.TextArea.Value = '';

    % Set up connection to the motor controllers with InitMotor and
    % finds initial position of with PositionMotor.
    % app.initializing sets everything on pause to initializing and
    % finding position is done.
    app.initializing = 1;
    while app.initializing == 1
        InitMotor(app)
        if app.initfail == 1
            txt = 'Initialize Machine failed';
            WriteTextInWindow(app,txt)
            app.initfail = 0;
            app.initializing = 0;
            return
        end
        for i = 1:4
            PositionMotor(app,i);
            app.zeroed(i) = 0;
            if i == 4
                app.homed(i) = 1;
            else
                app.homed(i) = 0;
            end
        end
        app.initialized = 1;
        app.initializing = 0;
    end
```

```

    % Set stop lamp to green to tell user that app is ready to and
    % set machine coordinates/absolute positions to correct state
    app.STOP_Lamp.Color = app.green;
    app.MACHINECOORDS_Lamp.Color = app.green;
    app.MACHINECOORDS_Button.Text = 'MACHINE COORDS';
    app.SETUP_Button.Text = 'SETUP';
    app.USERCOORDS_Lamp.Color = app.black;
    % Write position to position fields and reset buttons and
    % colors to corret states.
    WritePosition(app)
    ResetButtons(app)
    ResetColor(app)
end

% Button pushed function: INSTRUMENTCONNECT_Button
function INSTRUMENTCONNECT_ButtonPushed(app, event)
    % If you accedentaly hit instr button, you have the ability to
    % stop
    if app.connected == 1
        choice = questdlg('WARNING: Do you want to connect to ...
            instruments again?','WARNING','Yes','No','No');
        switch choice
            case 'No'
                return
        end
    end
    InstrumentConnect(app)
    if app.connectfail == 1
        txt = 'Instrument Connect failed';
        WriteTextInWindow(app,txt)
        app.connectfail = 0;
        return
    end
    app.connected = 1;
    set(app.STOP_Button,'Enable','On')
    ResetButtons(app)
end

% Button pushed function: HOME_ALL_Button
function HOME_ALL_ButtonPushed(app, event)
    txt = 'Home All';
    WriteTextInWindow(app,txt)
    home_order = [4,3,1,2];
    for i = 1:length(home_order)
```

```
        HomeMotorFunc(app,home_order(i))
        if app.stop == 1
            app.stop = 0;
            return
        end
    end
end

% Button pushed function: HOME_X_Button
function HOME_X_ButtonPushed(app, event)
    txt = ['Home ' app.name{1}];
    WriteTextInWindow(app,txt)
    HomeMotorFunc(app,1)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: HOME_Y_Button
function HOME_Y_ButtonPushed(app, event)
    txt = ['Home ' app.name{2}];
    WriteTextInWindow(app,txt)
    HomeMotorFunc(app,2)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: HOME_Z_Button
function HOME_Z_ButtonPushed(app, event)
    txt = ['Home ' app.name{3}];
    WriteTextInWindow(app,txt)
    HomeMotorFunc(app,3)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: HOME_R_Button
function HOME_R_ButtonPushed(app, event)
    txt = ['Home ' app.name{4}];
    WriteTextInWindow(app,txt)
```

```
    HomeMotorFunc(app,4)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: POSITEIVESTEP_X_Button
function POSITEIVESTEP_X_ButtonPushed(app, event)
    if app.STEP_X_DIR_EditField.Value < 0
        return
    end
    step = app.STEP_X_DIR_EditField.Value;
    StepMotorFunc(app,step,1,0)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: POSITEIVESTEP_Y_Button
function POSITEIVESTEP_Y_ButtonPushed(app, event)
    if app.STEP_Y_DIR_EditField.Value < 0
        return
    end
    step = app.STEP_Y_DIR_EditField.Value;
    StepMotorFunc(app,step,2,0)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: POSITEIVESTEP_Z_Button
function POSITEIVESTEP_Z_ButtonPushed(app, event)
    if app.STEP_Z_DIR_EditField.Value < 0
        return
    elseif app.setup == 1
        questdlg('WARNING: Exit setup!','WARNING','Ok','Ok')
        return
    end
    step = app.STEP_Z_DIR_EditField.Value;
    StepMotorFunc(app,step,3,0)
    if app.stop == 1
        app.stop = 0;
```

```
            return
        end
    end

    % Button pushed function: POSITEIVESTEP_R_Button
    function POSITEIVESTEP_R_ButtonPushed(app, event)
        if app.STEP_R_DIR_EditField.Value < 0
            return
        end
        step = app.STEP_R_DIR_EditField.Value;
        StepMotorFunc(app,step,4,0)
        if app.stop == 1
            app.stop = 0;
            return
        end
    end

    % Button pushed function: NEGATIVESTEP_X_Button
    function NEGATIVESTEP_X_ButtonPushed(app, event)
        if app.STEP_X_DIR_EditField.Value < 0
            return
        end
        step = -app.STEP_X_DIR_EditField.Value;
        StepMotorFunc(app,step,1,0)
        if app.stop == 1
            app.stop = 0;
            return
        end
    end

    % Button pushed function: NEGATIVESTEP_Y_Button
    function NEGATIVESTEP_Y_ButtonPushed(app, event)
        if app.STEP_Y_DIR_EditField.Value < 0
            return
        end
        step = -app.STEP_Y_DIR_EditField.Value;
        StepMotorFunc(app,step,2,0)
        if app.stop == 1
            app.stop = 0;
            return
        end
    end

    % Button pushed function: NEGATIVESTEP_Z_Button
    function NEGATIVESTEP_Z_ButtonPushed(app, event)
```

```
    if app.STEP_Z_DIR_EditField.Value < 0
        return
    elseif app.setup == 1
        questdlg('WARNING: Exit setup!','WARNING','Ok','Ok')
        return
    elseif app.zeroed(3) == 1 && app.zeroz == 1
        newpos = app.abspos.zaxis-app.STEP_Z_DIR_EditField.Value;
        if newpos < app.zeropos.zaxis
            choice = questdlg('WARNING: You are crossing your zero point...
                , this may cause you to crash with the microphone/...
                transducer.','WARNING','Don''t cross', 'Ignore','Don''t ...
                cross');
            switch choice
                case 'Don''t cross'
                    return
                case ''
                    return
                case 'Ignore'
                    app.zeroz = 0;
            end
        end
    end
    step = -app.STEP_Z_DIR_EditField.Value;
    StepMotorFunc(app,step,3,0)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: NEGATIVESTEP_R_Button
function NEGATIVESTEP_R_ButtonPushed(app, event)
    if app.STEP_R_DIR_EditField.Value < 0
        return
    end
    step = -app.STEP_R_DIR_EditField.Value;
    StepMotorFunc(app,step,4,0)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: STOP_Button
function STOP_ButtonPushed(app, event)
```

```
    if strcmpi(app.STOP_Button.Text,'STOP')
        txt = 'Stop';
        WriteTextInWindow(app,txt)
        app.stop = 1;
        ActionButtons(app)
        app.STOP_Lamp.Color = app.red;
        app.controller.C843.STP();
        writeline(app.controller.HydraTT,'1 nabort')
        for i = 1:4
            ActionLamp(app,false,i)
        end
        app.STOP_Button.Text='RESTART';
    elseif strcmpi(app.STOP_Button.Text,'RESTART') && app.safety == 0
        txt = 'Restart';
        WriteTextInWindow(app,txt)
        app.STOP_Lamp.Color = app.green;
        app.STOP_Button.Text='STOP';
        ResetButtons(app)
    end
end

% Button pushed function: ZERO_ALL_Button
function ZERO_ALL_ButtonPushed(app, event)
    txt = 'Zero All';
    WriteTextInWindow(app,txt)
    for i = 1:4
        SaveZero(app,i,'')
    end
end

% Button pushed function: ZERO_X_Button
function ZERO_X_ButtonPushed(app, event)
    txt = ['Zero ' app.name{1}];
    WriteTextInWindow(app,txt)
    SaveZero(app,1,'')
end

% Button pushed function: ZERO_Y_Button
function ZERO_Y_ButtonPushed(app, event)
    txt = ['Zero ' app.name{2}];
    WriteTextInWindow(app,txt)
    SaveZero(app,2,'')
end

% Button pushed function: ZERO_Z_Button
```

```
function ZERO_Z_ButtonPushed(app, event)
    txt = ['Zero ' app.name{3}];
    WriteTextInWindow(app,txt)
    SaveZero(app,3,'')
end

% Button pushed function: ZERO_R_Button
function ZERO_R_ButtonPushed(app, event)
    txt = ['Zero ' app.name{4}];
    WriteTextInWindow(app,txt)
    SaveZero(app,4,'')
end

% Button pushed function: GO_TO_ZERO_Button
function GO_TO_ZERO_ButtonPushed(app, event)
    txt = 'All Go To Zero';
    WriteTextInWindow(app,txt)

    gotozero_order = [4,1,2,3];
    for i = 1:length(gotozero_order)
        if app.homed(gotozero_order(i)) == 0
            txt = [app.name{gotozero_order(i)} ' is not homed.'];
            WriteTextInWindow(app,txt)
            continue
        elseif app.zeroed(gotozero_order(i)) == 0
            txt = [app.name{gotozero_order(i)} ' have no zero ...
                coordinates'];
            WriteTextInWindow(app,txt)
            continue
        elseif app.setup == 1 && gotozero_order(i) == 3
            txt = ['Exit Setup to move ' app.name{gotozero_order(i)} ' ...
                to zero'];
            WriteTextInWindow(app,txt)
            continue
        elseif app.zeroed(gotozero_order(i)) == 1
            pos = app.zeropos.(app.field{gotozero_order(i)});
            if gotozero_order(i) == 3
                distmic = 10;
                choice = questdlg('WARNING: Possible collision can occur...
                     with the microphone/transducer.','WARNING',['Stop at...
                     ' num2str(distmic) '[mm]'], 'Ignore',['Stop at ' ...
                    num2str(distmic) '[mm]']);
                switch choice
                    case ['Stop at ' num2str(distmic) '[mm]']
                        StepMotorFunc(app,(pos + distmic),gotozero_order...
```

```
                            (i),1)
                        if app.stop == 1
                            app.stop = 0;
                            return
                        end
                    case ''
                        return
                    case 'Ignore'
                        StepMotorFunc(app,pos,gotozero_order(i),1)
                        if app.stop == 1
                            app.stop = 0;
                            return
                        end
                end
            else
                StepMotorFunc(app,pos,gotozero_order(i),1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
            end
        end
    end
end

% Button pushed function: GO_TO_ZERO_X_Button
function GO_TO_ZERO_X_ButtonPushed(app, event)
    if app.homed(1) == 0
        txt = [app.name{1} ' is not homed.'];
        WriteTextInWindow(app,txt)
        return
    elseif app.zeroed(1) == 0
        txt = [app.name{1} ' have no zero coordinates'];
        WriteTextInWindow(app,txt)
        return
    end

    txt = [app.name{1} ' Go To Zero'];
    WriteTextInWindow(app,txt)

    pos = app.zeropos.(app.field{1});
    StepMotorFunc(app,pos,1,1)
    if app.stop == 1
        app.stop = 0;
        return
```

```
    end
end

% Button pushed function: GO_TO_ZERO_Y_Button
function GO_TO_ZERO_Y_ButtonPushed(app, event)
    if app.homed(2) == 0
        txt = [app.name{2} ' is not homed.'];
        WriteTextInWindow(app,txt)
        return
    elseif app.zeroed(2) == 0
        txt = [app.name{2} ' have no zero coordinates'];
        WriteTextInWindow(app,txt)
        return
    end

    txt = [app.name{2} ' Go To Zero'];
    WriteTextInWindow(app,txt)

    pos = app.zeropos.(app.field{2});
    StepMotorFunc(app,pos,2,1)
    if app.stop == 1
        app.stop = 0;
        return
    end
end

% Button pushed function: GO_TO_ZERO_Z_Button
function GO_TO_ZERO_Z_ButtonPushed(app, event)
    if app.setup == 1
        txt = 'Exit Setup';
        WriteTextInWindow(app,txt)
        return
    elseif app.homed(3) == 0
        txt = [app.name{3} ' is not homed.'];
        WriteTextInWindow(app,txt)
        return
    elseif app.zeroed(3) == 0
        txt = [app.name{3} ' have no zero coordinates'];
        WriteTextInWindow(app,txt)
        return
    end

    txt = [app.name{3} ' Go To Zero'];
    WriteTextInWindow(app,txt)

```

```
    pos = app.zeropos.(app.field{3});
    distmic = 10;
    choice = questdlg('WARNING: Possible collision can occur with the ...
       microphone/transducer.','WARNING',['Stop at ' num2str(distmic) '[...
       mm]'], 'Ignore',['Stop at ' num2str(distmic) '[mm]']);
    switch choice
        case ['Stop at ' num2str(distmic) '[mm]']
            StepMotorFunc(app,(pos + distmic),3,1)
            if app.stop == 1
                app.stop = 0;
                return
            end
        case ''
            return
        case 'Ignore'
            StepMotorFunc(app,pos,3,1)
            if app.stop == 1
                app.stop = 0;
                return
            end
    end
end

% Button pushed function: GO_TO_ZERO_R_Button
function GO_TO_ZERO_R_ButtonPushed(app, event)
    if app.homed(4) == 0
        txt = [app.name{4} ' is not homed.'];
        WriteTextInWindow(app,txt)
        return
    elseif app.zeroed(4) == 0
        txt = [app.name{4} ' have no zero coordinates'];
        WriteTextInWindow(app,txt)
        return
    end

    txt = [app.name{4} ' Go To Zero'];
    WriteTextInWindow(app,txt)

    pos = app.zeropos.(app.field{4});
    StepMotorFunc(app,pos,4,1)
    if app.stop == 1
        app.stop = 0;
        return
    end
end
```

```matlab

% Button pushed function: MACHINECOORDS_Button
function MACHINECOORDS_ButtonPushed(app, event)
    if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS')
        txt = 'User Coordinates';
        WriteTextInWindow(app,txt)
        app.MACHINECOORDS_Button.Text = 'USER COORDS';
        app.MACHINECOORDS_Lamp.Color = app.black;
        app.USERCOORDS_Lamp.Color = app.green;
        for i = 1:4
            WritePosition(app)
        end
    elseif strcmpi(app.MACHINECOORDS_Button.Text,'USER COORDS')
        txt = 'Machine Coordinates';
        WriteTextInWindow(app,txt)
        app.MACHINECOORDS_Button.Text = 'MACHINE COORDS';
        app.MACHINECOORDS_Lamp.Color = app.green;
        app.USERCOORDS_Lamp.Color = app.black;
        for i = 1:4
            WritePosition(app)
        end
    end
end

% Button pushed function: LOAD_Button
function LOAD_ButtonPushed(app, event)
    txt = 'Load';
    WriteTextInWindow(app,txt)
    [file,selpathh] = uigetfile({'*m';'*mat'});
    selectedfile = fullfile(selpathh,file);
    if isequal(file,0)
        txt = 'Canceled Loading';
        WriteTextInWindow(app,txt)
        return
    else
        txt = ['Loaded File: ', selectedfile];
        WriteTextInWindow(app,txt)
    end
    if strcmpi(file,'zero.mat')
        app.zeropos = load(selectedfile);
        app.zeroed(1) = 1;
        app.zeroed(2) = 1;
        app.zeroed(3) = 1;
        app.zeroed(4) = 1;
        app.zeroz = 1;
```

```
    elseif strcmpi(file,'MeasParameters.m')
        app.meas = MeasParameters;
        app.start = 1;
    elseif strcmpi(file,'MainLobeX.m')
        app.meas = MainLobeX;
        app.start = 1;
    elseif strcmpi(file,'MainLobeY.m')
        app.meas = MainLobeY;
        app.start = 1;
    elseif strcmpi(file,'slope.mat')
        app.linearslope = load(selectedfile);
        app.comp = 1;
    else
        txt = 'File not supported by app';
        WriteTextInWindow(app,txt)
    end
    WritePosition(app)
    ResetColor(app)
end

% Button pushed function: SETUP_Button
function SETUP_ButtonPushed(app, event)
    if ~all(app.homed)
        txt = 'Not all axis is homed';
        WriteTextInWindow(app,txt)
        return
    elseif strcmpi(app.SETUP_Button.Text,'SETUP')
        txt = 'Setup';
        WriteTextInWindow(app,txt)
        set = {'SETUP';'';'L = Laser stage is needed, take it up before ...
            start.'};
        list = {'Setup 1: Tilt angle of transducer: (L)';
            'Setup 2: Distance from transducer to rotationaxis: (L)';
            'Setup 3: Transducer perpendicular to laser: (L)';
            'Setup 4: Distance from transducer to microphone/transducer:...
                (L)';
            'Setup 5: Tilt angle of transducer and distance to ...
               microphone/transducer: (L)';
            'Turn off linear slope compensation';
            'Exit Setup'};
        choice = listdlg('PromptString',set,'ListSize',[400,150],'...
           ListString',list,'SelectionMode','single');
        if isempty(choice)
            choice = 0;
        end
```

```
        switch choice
            case 0
                txt = 'Canceled Setup';
                WriteTextInWindow(app,txt)
                return
            case 1
                app.setup = 1;
                txt = 'Setup 1';
                WriteTextInWindow(app,txt)
                StepMotorFunc(app,360,3,1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
                txt = 'Ready to measure? Click measure';
                WriteTextInWindow(app,txt)
                questdlg(txt,'Choose','Ok','Ok');
                app.SETUP_Button.Text = 'MEASURE';
                app.tilt_angle = 1;
            case 2
                app.setup = 1;
                txt = 'Setup 2';
                WriteTextInWindow(app,txt)
                StepMotorFunc(app,360,3,1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
                txt = 'Ready to measure? Click measure';
                WriteTextInWindow(app,txt)
                questdlg(txt,'Choose','Ok','Ok');
                app.SETUP_Button.Text = 'MEASURE';
                app.distance_transducer_rotaxis = 1;
            case 3
                app.setup = 1;
                txt = 'Setup 3';
                WriteTextInWindow(app,txt)
                StepMotorFunc(app,360,3,1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
                txt = 'Ready to measure? Click measure';
                WriteTextInWindow(app,txt)
                questdlg(txt,'Choose','Ok','Ok');
```

```
                app.SETUP_Button.Text = 'MEASURE';
                app.transducer_perpendicular_laser = 1;
            case 4
                app.setup = 1;
                txt = 'Setup 4';
                WriteTextInWindow(app,txt)
                StepMotorFunc(app,360,3,1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
                txt = 'Adjust laser such laser point is in the center of...
                     transducer and then adjust the X and Y-axis such ...
                    laser point is in center microphone/transducer';
                WriteTextInWindow(app,txt)
                questdlg(txt,'Choose','Ok','Ok');
                app.SETUP_Button.Text = 'MEASURE';
                app.distance_transducer_microphone = 1;
            case 5
                app.setup = 1;
                txt = 'Setup 5';
                WriteTextInWindow(app,txt)
                StepMotorFunc(app,360,3,1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
                txt = 'Adjust laser such laser point is in the center of...
                     transducer and then adjust the X and Y-axis such ...
                    laser point is in center microphone/transducer';
                WriteTextInWindow(app,txt)
                questdlg(txt,'Choose','Ok','Ok');
                app.SETUP_Button.Text = 'MEASURE';
                app.tilt_angle_and_distance = 1;
            case 6
                app.comp = 0;
            case 7
                app.setup = 0;
                txt = 'Exit Setup';
                WriteTextInWindow(app,txt)
                app.distance_transducer_microphone = 0;
                app.distance_transducer_rotaxis = 0;
                app.transducer_perpendicular_laser = 0;
                app.tilt_angle_and_distance = 0;
                app.tilt_angle = 0;
```

```
                app.SETUP_Button.Text = 'SETUP';
                if app.abspos.(app.field{3}) < 500
                    StepMotorFunc(app,500,3,1)
                    if app.stop == 1
                        app.stop = 0;
                        return
                    end
                end
        end
    else
        txt = 'Measure';
        WriteTextInWindow(app,txt)
        if app.distance_transducer_microphone == 1
            DTM(app)
            app.distance_transducer_microphone = 0;
        elseif app.distance_transducer_rotaxis == 1
            DTRax(app)
            app.distance_transducer_rotaxis = 0;
        elseif app.transducer_perpendicular_laser == 1
            ATL(app,0)
            app.transducer_perpendicular_laser = 0;
        elseif app.tilt_angle_and_distance == 1
            DTM(app)
            ATL(app,1)
            MLY(app)
            app.tilt_angle_and_distance = 0;
        elseif app.tilt_angle == 1
            ATL(app,1)
            app.tilt_angle = 0;
        end
        app.SETUP_Button.Text = 'SETUP';
    end
end

% Button pushed function: START_Button
function START_ButtonPushed(app, event)
    notes = {'Directivity','Sensitivity','On Axis Pressure','Main Lobe ...
        on y-axis','Main Lobe on x-axis'};

    if app.start == 0
        txt = 'No measurement parameters loaded';
        WriteTextInWindow(app,txt)
        return
    elseif ~all(app.homed)
        txt = 'Not all axis is homed';
```

```
        WriteTextInWindow(app,txt)
        return
    elseif ~all(app.zeroed)
        txt = 'Not all axis have zero coordinates';
        WriteTextInWindow(app,txt)
        return
    elseif app.connected == 0
        txt = 'Not connected to instruments';
        WriteTextInWindow(app,txt)
        return
    elseif ~any(strcmp(notes,app.meas.notes))
        txt = 'No meas.notes of that type, you have to modify ...
           SaveMeasurements(app) or choose meas.notes:';
        WriteTextInWindow(app,txt)
        for i = 1:length(notes)
            txt = notes{i};
            WriteTextInWindow(app,txt)
        end
        return
    end

    if strcmpi(app.MACHINECOORDS_Button.Text,'MACHINE COORDS')
        MACHINECOORDS_ButtonPushed(app, ...
           matlab.ui.eventdata.ButtonPushedData)
    end

    primary_axis = 3;
    txt = [app.name{primary_axis} ' are the primary axis'];
    WriteTextInWindow(app,txt)

    mainlobex = 0;
    if strcmpi(app.meas.notes,'Main Lobe on y-axis')
        secondary_axis = 2;
    elseif strcmpi(app.meas.notes,'Main Lobe on x-axis')
        secondary_axis = 4;
        mainlobex = 1;
    else
        set = {'Secondary axis';'';'Choose your secondary axis'};
        list = {'X-axis';'Y-axis';'Rotation-axis'};
        choice = listdlg('PromptString',set,'ListSize',[150,80],'...
           ListString',list,'SelectionMode','single');
        if isempty(choice)
            choice = 0;
        end
        switch choice
```

```
            case 0
                return
            case 1
                secondary_axis = 1;
            case 2
                secondary_axis = 2;
            case 3
                secondary_axis = 4;
        end
    end
    txt = [app.name{secondary_axis} ' selected as secondary axis'];
    WriteTextInWindow(app,txt)

    app.selpath = uigetdir(app.path);
    if isempty(app.selpath)
        app.selpath = app.path;
    end

    axis = [1,2,3,4];
    axis = axis(axis~=primary_axis);
    axis = axis(axis~=secondary_axis);

    app.meas.zeropos = app.zeropos;
    app.meas.date = strcat(datestr(now,'yyyy_mm_dd_HH_MM'));

    app.measurement = 0;
    started = 0;
    init = 0;

    freq_change = zeros(length(app.meas.primary_axis),width(...
        app.meas.secondary_axis)*length(app.meas.frequency));
    for ii = 1:length(app.meas.primary_axis)
        ll = 0;
        for jj = 1:width(app.meas.secondary_axis)
            for kk = 1:length(app.meas.frequency)

                % Calculate current measurement and total
                % measurements
                app.measurement = app.measurement + 1;
                totmeas = length(app.meas.primary_axis)*width(...
                    app.meas.secondary_axis)*length(app.meas.frequency);
                measurementleft = totmeas-app.measurement;
                app.result.measurement = ['Measurement ' num2str(...
                    app.measurement) ' of ' num2str(totmeas)];

```

```
            % Needed to reduce measurement time, and following
            % script measure only the output signal when
            % distance in z direction changes, the last signal
            % before moving in z direction and measure with a
            % frequency of 10 when no changes in z direction.
            ll = ll+1;
            if ~ismember(kk,freq_change(ii,:))
                if strcmpi(app.meas.notes,'On Axis Pressure')
                    % measure el signal for first or every 10n
                    % distance
                    if ii == 1
                        app.meas_el_sig = 1;
                    elseif mod(ii,10) == 0
                        app.meas_el_sig = 1;
                    else
                        app.meas_el_sig = 0;
                    end
                    freq_change(ii,ll) = kk;
                    z_dist = 1;
                else
                    % measure el signal because of new z distance
                    freq_change(ii,ll) = kk;
                    app.meas_el_sig = 1;
                    z_dist = 1;
                end
            elseif mod(sum(freq_change(ii,:)==kk),10) == 0
                % measure el signal because of 10n frequency
                freq_change(ii,ll) = kk;
                app.meas_el_sig = 1;
                z_dist = 0;
            elseif width(freq_change) < ll+length(app.meas.frequency...
               )
                % measure el  signal because of last secoundary ...
                    movment
                freq_change(ii,ll) = kk;
                app.meas_el_sig = 1;
                z_dist = 0;
            else
                % measure only acoutical
                freq_change(ii,ll) = kk;
                app.meas_el_sig = 0;
                z_dist = 0;
            end

            % Prevent to change filter settings if frequency is
```

```
        % not changed
        if app.measurement == 1
            % Set filter settings for first time
            app.filt_setting = 1;
        elseif ll == 1 && ii-1 ~= 0 && freq_change(ii-1,width(...
            freq_change)) ~= freq_change(ii,ll)
            % Changes filter settings because frequency
            % changed whith z movment
            app.filt_setting = 1;
        elseif ll-1 ~= 0 && freq_change(ii,ll) ~= freq_change(ii...
            ,(ll-1))
            % Changes filter settings because of frequency
            % changed
            app.filt_setting = 1;
        else
            % Frequency is the same and filter stays
            % unchanged
            app.filt_setting = 0;
        end

        % Prevent generator settings to change if frequency
        % or z distance is unchanged
        if app.filt_setting == 1
            app.gen_setting = 1;
        elseif z_dist == 1
            app.gen_setting = 1;
        else
            app.gen_setting = 0;
        end

        % Measure enviroment signal with a frequency of 30
        if app.measurement == 1
            app.environment = 1;
        elseif mod(app.measurement,30) == 0
            app.environment = 1;
        else
            app.environment = 0;
        end

        step_primary = app.zeropos.(app.field{primary_axis})+...
            app.meas.primary_axis(ii);
        if mod(ii,2) == 0
            stepdirrev = fliplr(app.meas.secondary_axis);
            if height(app.meas.secondary_axis) > 1
                step_secondary = app.zeropos.(app.field{...
```

```
                                secondary_axis})+stepdirrev(ii,jj);
                        else
                            step_secondary = app.zeropos.(app.field{...
                                secondary_axis})+stepdirrev(jj);
                        end
                    else
                        if height(app.meas.secondary_axis) > 1
                            step_secondary = app.zeropos.(app.field{...
                                secondary_axis})+app.meas.secondary_axis(ii,...
                                jj);
                        else
                            step_secondary = app.zeropos.(app.field{...
                                secondary_axis})+app.meas.secondary_axis(jj);
                        end
                    end

                    for tt = 1:length(axis)
                        if axis(tt) == 2 && app.comp == 1
                            app.result.lin_slope_const_a = app.linearslope.a...
                                ;
                            app.result.lin_slope_const_b = app.linearslope.b...
                                ;
                            compensation = app.linearslope.a*...
                                app.meas.primary_axis(ii)+app.linearslope.b;
                            if round((app.abspos.(app.field{axis(tt)})- ...
                                compensation),4) ~= 0
                                StepMotorFunc(app,compensation,axis(tt),1)
                                if app.stop == 1
                                    app.stop = 0;
                                    return
                                end
                            end
                        elseif axis(tt) == 1 && mainlobex == 1
                            position = app.zeropos.(app.field{axis(tt)})+...
                                app.meas.xaxis_offset;
                            if round((app.abspos.(app.field{axis(tt)})- ...
                                position),4) ~= 0
                                StepMotorFunc(app,position,axis(tt),1)
                                if app.stop == 1
                                    app.stop = 0;
                                    return
                                end
                            end
                        elseif started == 0 || round((app.abspos.(app.field{...
                            axis(tt)})-app.zeropos.(app.field{axis(tt)})),4) ...
```

```
                    ~= 0
                    step = app.zeropos.(app.field{axis(tt)});
                    StepMotorFunc(app,step,axis(tt),1)
                    if app.stop == 1
                        app.stop = 0;
                        return
                    end
                end
            end

            if length(app.meas.primary_axis) == 1
                if started == 0 || round((app.abspos.(app.field{...
                    primary_axis})-app.zeropos.(app.field{...
                    primary_axis})),4) ~= app.meas.primary_axis(ii)
                    StepMotorFunc(app,step_primary,primary_axis,1)
                    if app.stop == 1
                        app.stop = 0;
                        return
                    end
                end
            elseif round((app.abspos.(app.field{primary_axis})-...
                app.zeropos.(app.field{primary_axis})),4) ~= ...
                app.meas.primary_axis(ii)
                StepMotorFunc(app,step_primary,primary_axis,1)
                if app.stop == 1
                    app.stop = 0;
                    return
                end
            end

            if mod(ii,2) == 0
                if height(app.meas.secondary_axis) > 1
                    if round((app.abspos.(app.field{secondary_axis})...
                        -app.zeropos.(app.field{secondary_axis})),4) ...
                        ~= stepdirrev(ii,jj)
                        StepMotorFunc(app,step_secondary,...
                            secondary_axis,1)
                        if app.stop == 1
                            app.stop = 0;
                            return
                        end
                    end
                elseif round((app.abspos.(app.field{secondary_axis})...
                    -app.zeropos.(app.field{secondary_axis})),4) ~= ...
                    stepdirrev(jj)
```

```
                    StepMotorFunc(app,step_secondary,secondary_axis...
                        ,1)
                    if app.stop == 1
                        app.stop = 0;
                        return
                    end
                end
            else
                if height(app.meas.secondary_axis) > 1
                    if round((app.abspos.(app.field{secondary_axis})...
                        -app.zeropos.(app.field{secondary_axis})),4) ...
                        ~= app.meas.secondary_axis(ii,jj)
                        StepMotorFunc(app,step_secondary,...
                            secondary_axis,1)
                        if app.stop == 1
                            app.stop = 0;
                            return
                        end
                    end
                elseif round((app.abspos.(app.field{secondary_axis})...
                    -app.zeropos.(app.field{secondary_axis})),4) ~= ...
                    app.meas.secondary_axis(jj)
                    StepMotorFunc(app,step_secondary,secondary_axis...
                        ,1)
                    if app.stop == 1
                        app.stop = 0;
                        return
                    end
                end
            end

            app.result.primary_axis = app.field{primary_axis};
            app.result.secondary_axis = app.field{secondary_axis};
            app.result.primary_axis_pos = app.meas.primary_axis(ii);
            if mod(ii,2) == 0
                if height(app.meas.secondary_axis) > 1
                    app.result.secondary_axis_pos = stepdirrev(ii,jj...
                        );
                else
                    app.result.secondary_axis_pos = stepdirrev(jj);
                end
            else
                if height(app.meas.secondary_axis) > 1
                    app.result.secondary_axis_pos = ...
                        app.meas.secondary_axis(ii,jj);
```

```
            else
                app.result.secondary_axis_pos = ...
                    app.meas.secondary_axis(jj);
            end
        end
        app.result.pos.(app.field{1}) = app.abspos.(app.field...
            {1})-app.zeropos.(app.field{1});
        app.result.pos.(app.field{2}) = app.abspos.(app.field...
            {2})-app.zeropos.(app.field{2});
        app.result.pos.(app.field{3}) = app.abspos.(app.field...
            {3})-app.zeropos.(app.field{3});
        app.result.pos.(app.field{4}) = app.abspos.(app.field...
            {4})-app.zeropos.(app.field{4});
        app.result.abs_pos.(app.field{1}) = app.abspos.(...
            app.field{1});
        app.result.abs_pos.(app.field{2}) = app.abspos.(...
            app.field{2});
        app.result.abs_pos.(app.field{3}) = app.abspos.(...
            app.field{3});
        app.result.abs_pos.(app.field{4}) = app.abspos.(...
            app.field{4});

        distance_to_microphone = app.result.pos.(app.field{3})*1...
            e-3;
        if app.environment == 1
            VaisalaHMT313_read(app)
            Paroscientific(app)
            ASLF250(app)
        end

        app.result.sound_speed = 331*sqrt((...
            app.result.temperature+273.15)/273.15);
        app.result.est_travel_time = (distance_to_microphone/...
            app.result.sound_speed);
        % override with own signal cycles if not set to 0
        if app.meas.sig_cycles == 0
            sig_duration_nominal = max((...
                app.meas.est_travel_time_faktor*...
                app.result.est_travel_time),...
                app.meas.min_sig_duration);
            app.result.sig_cycles = floor(app.meas.frequency(kk)...
                *sig_duration_nominal);
        else
            app.result.sig_cycles = app.meas.sig_cycles;
        end
```

```
        app.result.sig_duration = app.result.sig_cycles/...
            app.meas.frequency(kk);
        app.result.frequency = app.meas.frequency(kk);
        app.result.cutoff_f1 = app.meas.cutoff_1(kk);
        app.result.cutoff_f2 = app.meas.cutoff_2(kk);

        % Start time
        if started == 0
            start_time = datetime('now');
            started = 1;
        end

        % Initialize first time
        if init == 0
            InitInstruments(app)
            init = 1;
        end
        Measure(app)

        app.result.time = datetime('now');
        time_used = app.result.time-start_time;
        average_time = (time_used)/app.measurement;
        time_finished = average_time*measurementleft;

        SaveMeasurements(app)

        if measurementleft >= 1
            txt = ['Measurement ' num2str(app.measurement) ' is ...
                complete'];
            WriteTextInWindow(app,txt)
            txt = ['There are ' num2str(measurementleft) ' ...
                measurements left'];
            WriteTextInWindow(app,txt)
            txt = ['The current average time for each ...
                measurement is ' datestr(average_time,'MM:SS')];
            WriteTextInWindow(app,txt)
            txt = ['Expected to be complete at ' datestr(...
                app.result.time+time_finished)];
            WriteTextInWindow(app,txt)
        else
            txt = ['The measurements are complete and took ' ...
                datestr(time_used,'dd:HH:MM:SS')];
            WriteTextInWindow(app,txt)
        end

```

```
                % Reset scope acquisition mode to RUNSTOP, so that ...
                    realtime changes is
                % visible on the oscilloscope.
                writeline(app.instrument.scope,'ACQ:STOPA RUNST');
            end
        end
    end
end

% Button pushed function: READ_Button
function READ_ButtonPushed(app, event)
    read = app.READ_DropDown.Value;
    switch read
        case 'DPO CH1'
            app.meas = {};
            app.meas.sample_count = app.SAMPLE_COUNT_Spinner.Value;
            app.meas.average = app.AVERAGE_Spinner.Value;
            app.meas.voltage_inn = app.VOLT_Spinner.Value;
            app.meas.burst_rate = app.BURST_RATE_Spinner.Value;
            app.meas.average_time = app.meas.average/app.meas.burst_rate...
                +1;
            app.result.frequency = app.FREQUENCY_Spinner.Value;
            app.result.cutoff_f1 = (app.result.frequency/1000)/2;
            app.result.cutoff_f2 = (app.result.frequency/1000)*2;
            app.result.sig_cycles = app.CYCLES_Spinner.Value;

            InitInstruments(app)

            % Set scope acquisition mode to SEQUENCE instead of RUNSTOP,...
                 so that a
            % measurement is aquired when prompted, instead of ...
                continuously. See page
            % 2-97 in programming manual for details.
            writeline(app.instrument.scope,'ACQ:STOPA SEQ');

            % Reset aquisition mode to averaging
            writeline(app.instrument.scope,'ACQ:MOD AVE');
            % Set the number of cycles to average.
            writeline(app.instrument.scope,['ACQ:NUMAV ' num2str(...
                app.meas.average)]);
            % Number of points which shall be read from the scope.
            writeline(app.instrument.scope,['HOR:RECO ' num2str(...
                app.meas.sample_count)]);

            % Start aquisition.
```

```
            writeline(app.instrument.scope,'ACQ:STATE RUN');
            txt = 'Starting aquisition';
            WriteTextInWindow(app,txt)
            pause(app.meas.average_time)

            DPO_read(app,1)
            dporesult.x = app.x;
            dporesult.wf = app.wf;

            Paroscientific(app)
            dporesult.pressure = app.result.pressure;
            VaisalaHMT313_read(app)

            dporesult.vaisala_RH = app.result.vaisala_RH;
            dporesult.vaisala_T = app.result.vaisala_T;

            ASLF250(app)
            dporesult.temperature = app.result.temperature;

            for fn = fieldnames(app.result)'
                app.meas.(fn{1}) = app.result.(fn{1});
            end
            dporesult.settings = app.meas;
            txt = 'Saving Measurement Parameters';
            WriteTextInWindow(app,txt)
            savepath = [app.path '\DPO_read_CH1.mat'];
            m = matfile(savepath,'Writable',true);
            m.DPO = dporesult;
            txt = 'Finished';
            WriteTextInWindow(app,txt)
            % Reset scope acquisition mode to RUNSTOP, so that realtime ...
                changes is
            % visible on the oscilloscope.
            writeline(app.instrument.scope,'ACQ:STOPA RUNSTop');
        case 'DPO CH2'
            app.meas = {};
            app.meas.sample_count = app.SAMPLE_COUNT_Spinner.Value;
            app.meas.average = app.AVERAGE_Spinner.Value;
            app.meas.voltage_inn = app.VOLT_Spinner.Value;
            app.meas.burst_rate = app.BURST_RATE_Spinner.Value;
            app.meas.average_time = app.meas.average/app.meas.burst_rate...
                +1;
            app.result.frequency = app.FREQUENCY_Spinner.Value;
            app.result.cutoff_f1 = (app.result.frequency/1000)/2;
            app.result.cutoff_f2 = (app.result.frequency/1000)*2;
```

```
            app.result.sig_cycles = app.CYCLES_Spinner.Value;

            InitInstruments(app)

            % Set scope acquisition mode to SEQUENCE instead of RUNSTOP,... so that a
            % measurement is aquired when prompted, instead of ... continuously. See page
            % 2-97 in programming manual for details.
            writeline(app.instrument.scope,'ACQ:STOPA SEQ');

            % Reset aquisition mode to averaging
            writeline(app.instrument.scope,'ACQ:MOD AVE');
            % Set the number of cycles to average.
            writeline(app.instrument.scope,['ACQ:NUMAV ' num2str(...
                app.meas.average)]);
            % Number of points which shall be read from the scope.
            writeline(app.instrument.scope,['HOR:RECO ' num2str(...
                app.meas.sample_count)]);

            % Start aquisition.
            writeline(app.instrument.scope,'ACQ:STATE RUN');
            txt = 'Starting aquisition';
            WriteTextInWindow(app,txt)
            pause(app.meas.average_time)

            DPO_read(app,2)
            dporesult.x = app.x;
            dporesult.wf = app.wf;

            Paroscientific(app)
            dporesult.pressure = app.result.pressure;
            VaisalaHMT313_read(app)

            dporesult.vaisala_RH = app.result.vaisala_RH;
            dporesult.vaisala_T = app.result.vaisala_T;

            ASLF250(app)
            dporesult.temperature = app.result.temperature;

            for fn = fieldnames(app.result)'
                app.meas.(fn{1}) = app.result.(fn{1});
            end
            dporesult.settings = app.meas;
            txt = 'Saving Measurement Parameters';
```

```
            WriteTextInWindow(app,txt)
            savepath = [app.path '\DPO_read_CH2.mat'];
            m = matfile(savepath,'Writable',true);
            m.DPO = dporesult;
            txt = 'Finished';
            WriteTextInWindow(app,txt)
            % Reset scope acquisition mode to RUNSTOP, so that realtime ...
               changes is
            % visible on the oscilloscope.
            writeline(app.instrument.scope,'ACQ:STOPA RUNST');
        case 'DPO CH2 Read Only'
            % Reading only from screen, only changeable value is
            % samplecount. Useful when calibrating microphone.
            app.meas = {};
            app.meas.sample_count = app.SAMPLE_COUNT_Spinner.Value;

            DPO_read(app,2)
            dporesult.x = app.x;
            dporesult.wf = app.wf;

            Paroscientific(app)
            dporesult.pressure = app.result.pressure;
            VaisalaHMT313_read(app)

            dporesult.vaisala_RH = app.result.vaisala_RH;
            dporesult.vaisala_T = app.result.vaisala_T;

            ASLF250(app)
            dporesult.temperature = app.result.temperature;

            dporesult.sample_count = app.meas.sample_count;

            txt = 'Saving Measurement Parameters';
            WriteTextInWindow(app,txt)
            savepath = [app.path '\DPO_read_CH2_Read.mat'];
            m = matfile(savepath,'Writable',true);
            m.DPO = dporesult;
            txt = 'Finished';
            WriteTextInWindow(app,txt)
        case 'PAROSCIENTIFIC'
            Paroscientific(app)
            txt = num2str(app.result.pressure);
            WriteTextInWindow(app,txt)
        case 'VAISALA'
            VaisalaHMT313_read(app)
```

```
            txt = ['Relative Humidity = ' num2str(app.result.vaisala_RH)...
                ];
            WriteTextInWindow(app,txt)
            txt = ['Temperature = ' num2str(app.result.vaisala_T)];
            WriteTextInWindow(app,txt)
        case 'ASLF250'
            ASLF250(app)
            txt = ['Temperature = ' num2str(app.result.temperature)];
            WriteTextInWindow(app,txt)
    end
end
```

# Appendix C

# FEMP-scripts

## C.1 read_inn_project.m (vacuum)

```
function [read]=read_inn_project(read,commands);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read .inn-file. Note that this function calls a project specific
% read_inn_project.m which should be in the working directory
%
% Part of FEMP (Finite Element Modeling of Piezoelectric structures)
% Programmed by Jan Kocbach (jan@kocbach.net)
% (C) 2000 Jan Kocbach. This file is free software;  you can ...
    redistribute
% it and/or modify it only under the the terms of the GNU GENERAL PUBLIC
% LICENSE which should be included along with this file.
% (C) 2000-2010 Christian Michelsen Research AS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Put a file read_inn_project.m in your project directory to define ...
    local
% FEMP input commands. Also include init_const_project.m in this ...
    directory
% and define the commands there.
global glob;
read=read;


%% piezodiskfluid_egen
    if ~isempty(read.piezodiskfluidtest)

        read.points=[]; read.areas=[]; read.materials=[]; read.dof=[]; ...
```

```
            read.restraints=[];

        r=read.piezodiskfluidtest(1,1,:);
        t=read.piezodiskfluidtest(1,2,:);
        rfluid=read.piezodiskfluidtest(1,3,:);
        elfluid=read.piezodiskfluidtest(1,4,:);
        matnum=read.piezodiskfluidtest(1,5,:);
        elr=read.piezodiskfluidtest(1,6,:);
        elt=read.piezodiskfluidtest(1,7,:);
        matnumfluid=read.piezodiskfluidtest(1,8,:);
        theta=read.piezodiskfluidtest(1,9,:);

        for s=1:size(r,3)
        rfluidtemp=0+rfluid(s);
        rinffluid=rfluidtemp*2;
        read.points(:,:,s)=[
        1 0 -t(s)/2;
        2 r(s) -t(s)/2;
        3 0 t(s)/2;
        4 r(s) t(s)/2];

        read.areas(:,:,s)=[1 1 2 4 3 elr(s) elt(s) 0 0];

        read.materials(:,:,s)=[1 glob.globvariables.piezo matnum(s)];

        read.dof(:,:,s)=[-1 1 t(s)/2-1e-9 t(s)/2+1e-9 glob.free.ep];
        read.restraints(:,:,s)=[-1 1 -t(s)/2-1e-9 -t(s)/2+1e-9 ...
            glob.free.ep 1];
        glob.tfront(s)=t(s)/2;
        end
    end
```

## C.2 read_inn_project.m (fluid)

```
function [read]=read_inn_project(read,commands);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read .inn-file. Note that this function calls a project specific
% read_inn_project.m which should be in the working directory
%
% Part of FEMP (Finite Element Modeling of Piezoelectric structures)
% Programmed by Jan Kocbach (jan@kocbach.net)
% (C) 2000 Jan Kocbach. This file is free software;  you can ...
```

```
    redistribute
% it and/or modify it only under the the terms of the GNU GENERAL PUBLIC
% LICENSE which should be included along with this file.
% (C) 2000-2010 Christian Michelsen Research AS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Put a file read_inn_project.m in your project directory to define ...
    local
% FEMP input commands. Also include init_const_project.m in this ...
    directory
% and define the commands there.
global glob;
read=read;


%% piezodiskfluid_egen
    if ~isempty(read.piezodiskfluidtest)

        read.points=[]; read.areas=[]; read.materials=[]; read.dof=[]; ...
            read.restraints=[];

        r=read.piezodiskfluidtest(1,1,:);
        t=read.piezodiskfluidtest(1,2,:);
        rfluid=read.piezodiskfluidtest(1,3,:);
        elfluid=read.piezodiskfluidtest(1,4,:);
        matnum=read.piezodiskfluidtest(1,5,:);
        elr=read.piezodiskfluidtest(1,6,:);
        elt=read.piezodiskfluidtest(1,7,:);
        matnumfluid=read.piezodiskfluidtest(1,8,:);
        theta=read.piezodiskfluidtest(1,9,:);

        for s=1:size(r,3)
        rfluidtemp=0+rfluid(s);
        rinffluid=rfluidtemp*2;
        read.points(:,:,s)=[
        1 0 -t(s)/2;
        2 r(s) -t(s)/2;
        3 0 t(s)/2;
        4 r(s) t(s)/2;
        5 0 rfluid(s);
        6 rfluid(s)*sin(theta(s)) rfluid(s)*cos(theta(s));
        7 rfluid(s)*sin(theta(s)) -rfluid(s)*cos(theta(s));
        8 0 -rfluid(s);
        9 0 rinffluid;
        10 rinffluid*sin(theta(s)) rinffluid*cos(theta(s));
```

```
        11 rinffluid*sin(theta(s)) -rinffluid*cos(theta(s));
        12 0 -rinffluid;
        13 0 0];

        read.areas(:,:,s)=[1 1 2 4 3 elr(s) elt(s) 0 0;
        2 3 4 6 5 elfluid(s) elfluid(s) 0 13;
        2 4 2 7 6 elfluid(s) elfluid(s) 0 13;
        2 2 1 8 7 elfluid(s) elfluid(s) 0 13;
        3 5 6 10 9 1 1 13 13;
        3 6 7 11 10 1 1 13 13;
        3 7 8 12 11 1 1 13 13];

        read.materials(:,:,s)=[1 glob.globvariables.piezo matnum(s);
        2 glob.globvariables.fluid matnumfluid(s);
        3 glob.globvariables.infinitefluid matnumfluid(s)];

        read.dof(:,:,s)=[-1 1 t(s)/2-1e-9 t(s)/2+1e-9 glob.free.ep];
        read.restraints(:,:,s)=[-1 1 -t(s)/2-1e-9 -t(s)/2+1e-9 ...
           glob.free.ep 1];
        glob.tfront(s)=t(s)/2;
        end
    end
```

## C.3 init_const_project.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialization of constants for FEMP
% Part of FEMP (Finite Element Modeling of Piezoelectric structures)
% Programmed by Jan Kocbach (jan@kocbach.net)
% (C) 2000 Jan Kocbach. This file is free software;  you can ...
   redistribute
% it and/or modify it only under the the terms of the GNU GENERAL PUBLIC
% LICENSE which should be included along with this file.
% (C) 2000-2010 Christian Michelsen Research AS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% INIT_CONST_PROJECT: Initialize constants - project specifix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Make a copy of this file in your project folder to
```

```
% make project specific definitions

%'piezodiskfront1',

commands = [commands,'piezodiskfluidtest'];
```

## C.4 Pz27.inn

```
set
Radius_P,           10e-3
Thickness_P,        2e-3
Radius_Infel,       30e-3
Elements_FL,        7
Materialnumber_P,   77
Elements_R_P,       7
Elements_T_P,       7
Materialnumber_FL,  10101
Theta,              1.3
end

materialfile
5
end

meshingtype
elementsperwavelength,300e3
end

viewmesh
0
end

# The order of the finite elements is 2 - i.e. 8 node isoparametric ...
    elements are applied
order
2
end

# The order of the infinite elements is set to 10.
infiniteorder
10
end
```

```

piezodiskfluidtest
Radius_P,Thickness_P,Radius_Infel,Elements_FL,Materialnumber_P,...
    Elements_R_P,Elements_T_P,Materialnumber_FL,Theta,q_DampDist_r,...
    q_DampDist_z
end

#directharmonicanalysis
#0,50,300e3,complex_loss
#end

#admittance
#0,0,0
#end

#save
#admittance,admittance_f
#end

#nearfieldpressure
#0,0,0,-1,1,-1,1
#end

# Calculate far-field pressure for the frequencies used in the time-...
    harmonic
# analysis. Calculate out to 3 times the distance at which the infinite
# elements are applied (10*7.0 mm= 70.0 mm), with 20 divisions per 7.0 ...
    mm.

#farfieldpressure
#0,0,0,10,20
#end

#save
#farfieldpressure,farfieldpressure_f,farfieldpressure_r,...
    farfieldpressure_z,nearfieldpressure,nearfieldpressure_z,...
    nearfieldpressure_r,nearfieldpressure_f
#end

#onaxispressure
#0,0,0,10,20
#end

#directivity
#0,0,0,1
```

```
#end

#sensitivity
#0,0,0,1
#end

#save
#sensitivity,sensitivity_f,directivity,directivity_theta,directivity_f
#end

#save
#sensitivity,sensitivity_f,onaxispressure,onaxispressure_r,...
    onaxispressure_f
#end

#save
#sensitivity,sensitivity_f,directivity,directivity_theta,directivity_f,...
    onaxispressure,onaxispressure_r,onaxispressure_f
#end

#save('test_res.mat','result','-v7.3');
```

## C.5 material5.dat

```
       77     piezo      pz27 (Lohne/Knappskog)
# mechanical terms
  1.1875E+11 7.43000E+10 7.42500E+10 0.00000E+00 0.00000E+00 0.00000E+00
  7.43000E+10 1.1875E+11 7.42500E+10 0.00000E+00 0.00000E+00 0.00000E+00
  7.42500E+10 7.42500E+10 1.12050E+11 0.00000E+00 0.00000E+00 0.00000E...
      +00
  0.00000E+00 0.00000E+00 0.00000E+00 2.11000E+10 0.00000E+00 0.00000E...
      +00
  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 2.11000E+10 0.00000E...
      +00
  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 2.22250E...
      +10
# coupling terms
  0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 1.12000E+01 0.00000E...
      +00
  0.00000E+00 0.00000E+00 0.00000E+00 1.12000E+01 0.00000E+00 0.00000E...
      +00
-5.40000E+00 -5.40000E+00 1.60389E+01 0.00000E+00 0.00000E+00 0.00000E...
```

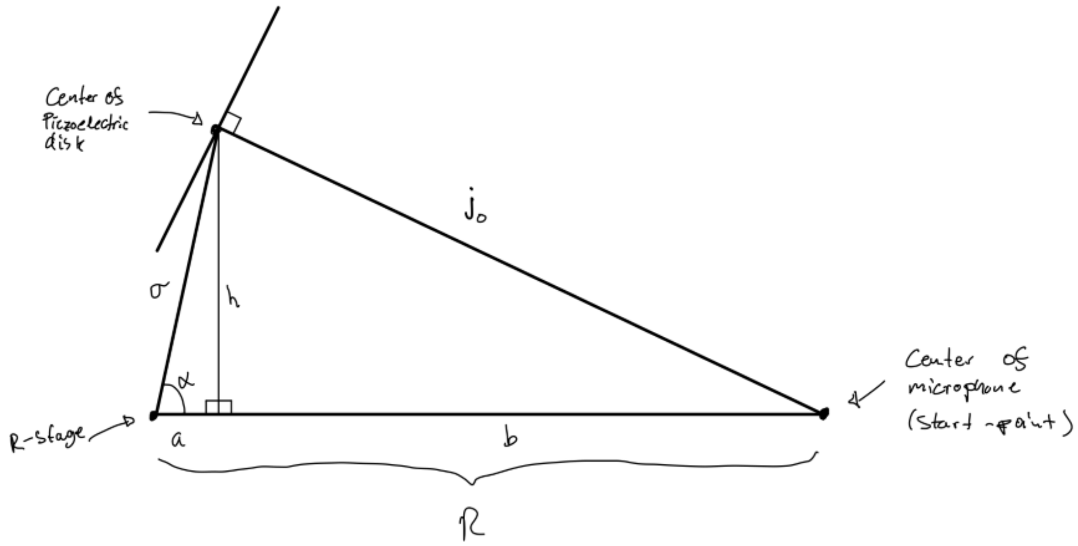
```
   +00
# dielectric terms
  8.11043e-09 0.00000E+00 0.00000E+00
  0.00000E+00 8.11043e-09 0.00000E+00
  0.00000E+00 0.00000E+00 8.14585e-09
# density and damping coefficients
  7.70000E+03 9.99000e+02 9.99000e+02
# mechanical Q-factors
  9.57500e+01 7.12400e+01 1.20190e+02 0.00000e+00 0.00000e+00 0.00000e...
     +00
  7.12400e+01 9.57500e+01 1.20190e+02 0.00000e+00 0.00000e+00 0.00000e...
     +00
  1.20190e+02 1.20190e+02 1.77990e+02 0.00000e+00 0.00000e+00 0.00000e...
     +00
  0.00000e+00 0.00000e+00 0.00000e+00 7.50000e+01 0.00000e+00 0.00000e...
     +00
  0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 7.50000e+01 0.00000e...
     +00
  0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 2.25342e...
     +02
# piezoelectric Q-factors
  0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 -2.00000e+02 0.00000e...
     +00
  0.00000e+00 0.00000e+00 0.00000e+00 -2.00000e+02 0.00000e+00 0.00000e...
     +00
  -1.66000e+02 -1.66000e+02 -3.23770e+02 0.00000e+00 0.00000e+00 0...
     .00000e+00
# dielectric Q-factors
  5.00000e+01 0.00000e+00 0.00000e+00
  0.00000e+00 5.00000e+01 0.00000e+00
  0.00000e+00 0.00000e+00 8.62800e+01
# end of material data

     10100     fluid air
 1.20500E+00 1.41767E+05 0.00000E+00 0.00000E+00
     10101     fluid air20grader
 1.21000E+00 1.42355e+05 0.00000E+00 0.00000E+00
```

# Appendix D

# Additional information

## D.1 Derivation of R



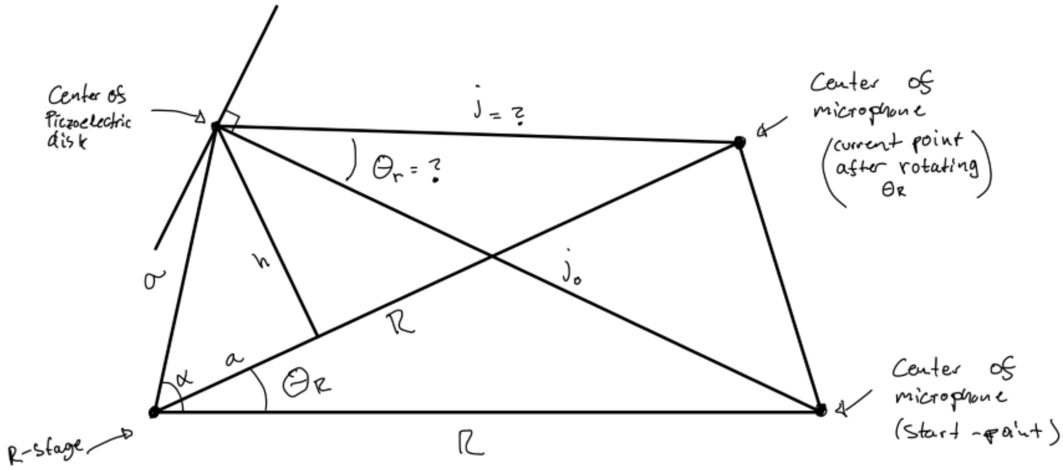

$$R = a + b$$

$$a = \sigma \cos(\alpha)$$

$$\left.\begin{aligned} \sigma^2 &= a^2 + h^2 \\ j_o &= b^2 + h^2 \end{aligned}\right\} \Rightarrow \quad b = \sqrt{j_o^2 + a^2 - \sigma^2}$$

$$R = \sigma \cos(\alpha) + \sqrt{j_o^2 + \sigma^2(\cos^2(\alpha) - 1)}$$

## D.2 Derivation of j




$$a = \sigma \cos(\alpha - \theta_R)$$

$$h = \sigma \sin(\alpha - \theta_R)$$

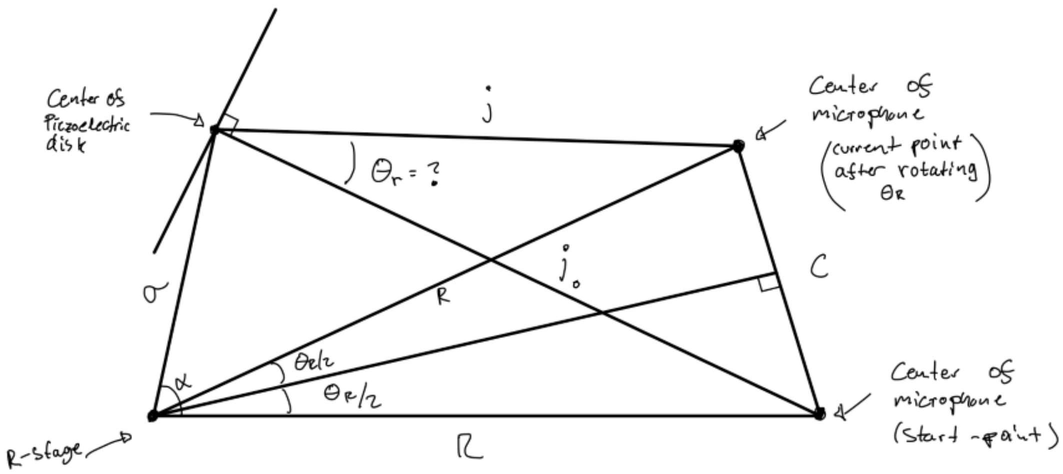
$$j = \sqrt{(R-a)^2 + h^2}$$

$$= \sqrt{(R - \sigma \cos(\alpha - \theta_R))^2 + \sigma^2 \sin^2(\alpha - \theta_R)}$$

$$= \sqrt{R^2 - 2R\sigma \cos(\alpha - \theta_R) + \sigma^2 \left(\cos^2(\alpha - \theta_R) + \sin^2(\alpha - \theta_R)\right)}$$

$$j = \sqrt{R^2 - 2R\sigma \cos(\alpha - \theta_R) + \sigma^2}$$

## D.3 Derivation of $\theta_r$




$$\frac{C}{2} = R\sin\left(\frac{\theta_R}{2}\right) \Rightarrow C = 2R\sin\left(\frac{\theta_R}{2}\right)$$

$$C^2 = j^2 + j_o^2 - 2 j j_o \cos(\theta_r)$$

$$C^2 = 4R^2\sin^2\left(\frac{\theta_R}{2}\right)$$

$$\theta_r = \cos^{-1}\left(\frac{j^2 + j_o^2 - 4R^2\sin^2\left(\frac{\theta_R}{2}\right)}{2 j j_o}\right)$$