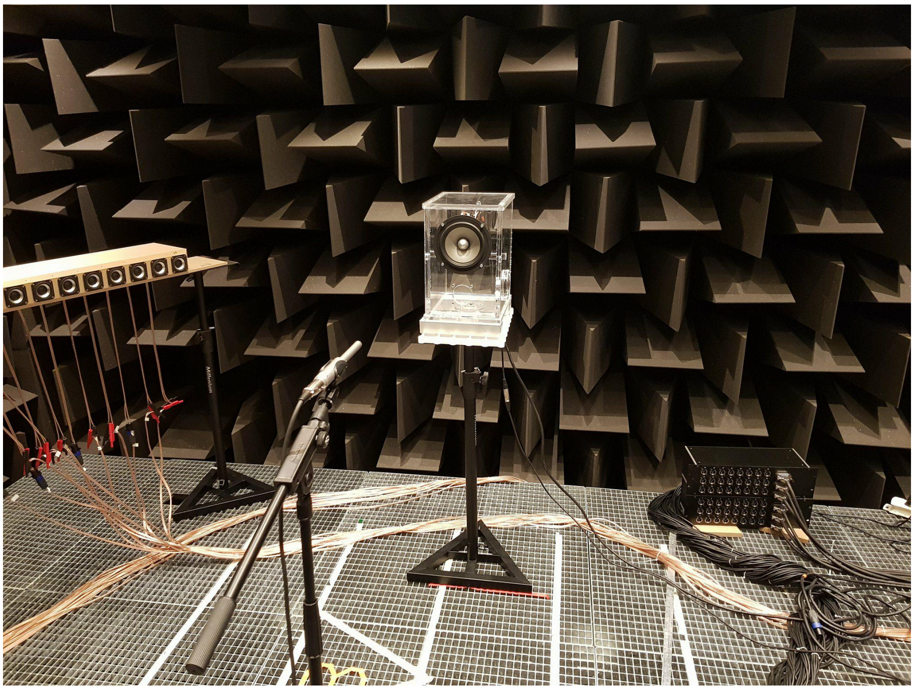
Speaker modelling

F18-ETLYAK



Jonas Lind
201507296

Lasse Østerberg Sangild
20114274

Advisor
Lars G. Johansen

Abstract

This project has aimed to create a Matlab framework able to predict frequency responses of speakers, with and without bass reflexes, based on Thiele/Small parameters and the size of the cabinet and bass reflex.

The models for the speaker, based on electrical, mechanical and acoustical analogies, are implemented in Matlab classes, split up into the entities speaker, cabinet, drive unit, crossover filter and bass reflex.

Testing the models was done by comparing them to measurements of real speakers with known Thiele/Small values and volumes.

The models were easy to work with and proved able to predict the cut-off frequency and slope of a speaker placed in a cabinet to a satisfying degree.

Contents

Chapter 1

Introduction

1.1 Project Description

A way of producing a great loud speaker, is to have a great model to test before the actual production. Our project will aim to create such a model, taking into account as many of the following variables as time allows:

- Shape of speaker
- Size of speaker
- Number of units
- Size of units
- Placement of units
- Crossover filter

This model will be used to compare the frequency response of a known speaker, measured in the anechoic room, to the modelled speaker response.

If time is, an optimization for the flattest frequency response will be done within realistic bounds for the optimized values.

1.2 Project delimitations

The Project Description is the groups vision of the ideal solution to the problem outlined in the Introduction. To give the best possible view of the groups capabilities in developing such a solution, the most core parts of the project will have the highest priority, hence some parts will be excluded from this version of the project.

Must
- Use Matlab to model a speaker with one drive unit in a closed cabinet.
- Measure the speaker response of a speaker with one drive unit in a closed cabinet and compare with the Matlab model.

Should
- Add a crossover filter to the Matlab model.
- Add a crossover filter to the measurement setup and compare with the Matlab model.

Could
- Add a bass reflex to the Matlab model.
- Compare the Matlab bass reflex model with the plexi glass speaker (from AudioLab) with bass reflex.

Won't
- Model multiple drive units.

The project will focus on creating a working model of a speaker with one drive unit in a closed box.

If time is, the addition of a bass reflex will take priority over e.g. multiple drive units.

Chapter 2
Analysis

This section describes the analysis of the system; how the system is envisioned to be created, which subsystems it will consist of and the tasks each subsystem has to be able to perform to meet the specified requirements.

2.1 Loudspeaker Modelling

The loudspeaker system can consist of multiple drive units reproducing sound in different frequency spectra. The drive unit can be parted in three equivalent circuits representing the electrical, the mechanical and the acoustical element. Each of these elements can be converted to its electrical equivalent and be represented with an Ohm's Law analogy [3, p. 115].

The electrical element of the drive unit is specified by the voice coil with its DC resistance R_e and self inductance L_e. The coupling between the electrical and the mechanical elements is specified as the force factor Bl. This is the product of the magnetic field in the voice coil gap and the length of the voice coil in the magnetic field. [3, p. 34]

The mechanical element of the drive unit is specified by the mass M_{MD} of the diaphragm, the mechanical damping R_{MS} and the compliance C_{MS}. The mechanical element will introduce a resonance frequency f_S and is described by its quality factor Q_{MS}. A total quality factor Q_{TS} is found when combining the Q-factor for the electrical and mechanical elements, $\frac{Q_{ES}Q_{MS}}{Q_{ES}+Q_{MS}}$.

The acoustical element of the drive unit is specified by an acoustical impedance in front and behind the diaphragm. The acoustical impedance in front is given by the relation between sound pressure p and volume velocity q which is the velocity of air that is moved by the diaphragm.

These parameters are known as Thiele/Small parameters and are used to specify the performance of a drive unit and is derived by A.N. Thiele [5] and Richard H. Small [4]. The parameters can be used to decide the volume of the loudspeaker cabinet and the length of the bass-reflex port. The best performance often includes improving the bass response and to obtain a flatness in general of the frequency response. The physical parameters of the drive unit can be found in the datasheet.

R_E is the DC resistance of the voice coil.

L_E is the inductance of the voice coil.

f_s is the resonance frequency.

Q_{TS} is the combined electric and mechanical damping of the drive unit.

M_{MS} is the mass of the drive unit's moving parts including acoustic load.

C_{MS} is the mechanical compliance of the drive unit's suspension.

R_{MS} is the mechanical resistance of the drive unit's suspension.

Bl is the force factor determined by the product of the magnetic flux density in the air gap and the length of wire in the air gap.

S_D is the surface area of the drive unit's diapraghm.

2.2 Transfer Functions

The loudspeaker being modelled in this project has been parted in 4 subsystems; a drive unit, a cross-over filter, a cabinet and a bass reflex. Each subsystem have a transfer function that can be derived from the Thiele/Small parameters.

2.2.1 Cabinet

The cabinet is a sealed enclosure, which separates the front and back side of the diaphragm, thus avoiding acoustic short circuiting. Placing a drive unit in a sealed enclosure prevents the pressure at the front and back to interact with each other. [3, p. 44]. The volume of the closed cabinet is characterized by the air acting like a spring (a capacity) which affects the drive unit's compliance, see eq. (2.1), where V_B is the volume of the cabinet, ρ is the density of air and c is the speed of sound.

$$C_{AB} = \frac{V_B}{\rho\, c^2} \tag{2.1}$$

The transfer function for a drive unit placed in an enclosure is shown in eq. (2.2). The sound pressure is specified at a distance $r = 1\,\text{m}$.

$$p = \frac{\rho S_D Bl U_G}{2\pi r M_{MS} R_E} \frac{s^2}{s^2 + s\left(\frac{(Bl)^2}{R_E M_{MS}} + \frac{R_{MS}}{M_{MS}}\right) + \frac{1}{M_{MS} C_{MS}}\left(1 + \frac{C_{MS} {S_D}^2}{C_{AB}}\right)} \tag{2.2}$$

2.2.2 Drive Unit

The drive unit can be modelled as mounted in an infinite baffle, which also separates the front and back side of the diaphragm. The transfer function for a drive unit placed in an infinite baffle is shown in eq. (2.3). This equation contains the diffraction and reflection seen in the two last terms.

$$\begin{aligned} p = &\frac{\rho S_D Bl U_G}{2\pi r M_{MS} R_E} \left| \frac{s^2}{s^2 + \frac{\omega_s}{Q_{TS}} s + \omega_s^2} \right| \\ &\left[1 - \frac{r_F}{r_B} D(ka) \exp(-jk(r_B - r_F)) \right] \left[1 - \frac{r_F}{r_R} D(ka) \exp(-jk(r_R - r_F)) \right] \end{aligned} \tag{2.3}$$

The constant term in eq. (2.3) transforms the voltage U_G to the sound pressure p while the complex term represent a second order high pass filter. The slope of the high pass filter will decrease by 12 dB per octave below the resonance frequency.

In this project the drive unit is being modelled as mounted in an infinite enclosure, a very large closed cabinet. Doing this will avoid the diffraction and reflection terms in eq. (2.3). Placing the drive unit in enclosure will prevent acoustic short circuiting as well. If the drive unit is placed in a very large sealed enclosure it will make the model behave like the drive unit was mounted in an infinite baffle. When the volume becomes very large the compliance as well becomes very large seen in eq. (2.1). The term containing the C_{AB} variable goes towards zero as $C_{AB} \to \infty$ in eq. (2.2) and the transfer function for a drive unit placed in an very large enclosure is now as shown in eq. (2.5).

$$p_\infty = \lim_{C_{AB} \to \infty} (p) \tag{2.4}$$

$$\Rightarrow p_\infty = \frac{\rho S_D Bl U_G}{2\pi r M_{MS} R_E} \frac{s^2}{s^2 + s\left(\frac{(Bl)^2}{R_E M_{MS}} + \frac{R_{MS}}{M_{MS}}\right) + \frac{1}{M_{MS} C_{MS}}} \tag{2.5}$$

2.2.3 Crossover Filter

Different drive units are differently designed to reproduce the sound in specific frequency spectra. When a loudspeaker system consist of multiple drive units a crossover filter is used to separate these frequency spectrums. For a drive unit (tweeter) reproducing the high frequency spectrum, a second order high pass filter is used to filter the low frequency spectrum, see eq. 2.6 [3, p. 82-83].

$$H_T = \frac{s^2}{\omega_0^2 + 2\zeta\,\omega_0 s + s^2} \tag{2.6}$$

For a drive unit (woofer) reproducing the low frequency spectrum, a second order low pass filter is used to filter the high frequency spectrum, see eq. (2.7).

$$H_W = \frac{\omega_0^2}{\omega_0^2 + 2\zeta\,\omega_0 s + s^2} \tag{2.7}$$

A quality factor is introduced for the crossover filter, as it was for the drive unit itself. With a Q-factor of 0.7 a maximal flat pass-band filter is made. This is known as a butterworth filter. The sharper the transition band needs to be, the higher the Q-factor, given the damping factor ζ which is the inverse of Q.

2.2.4 Bass Reflex

A bass reflex is a port placed in the cabinet, also called a vented cabinet. By creating a port, the pressure from the rear side of the diaphragm can increase the low frequency spectrum. The air in the port will resonate with the volume of the cabinet and this will introduce an additional resonance frequency [3, p. 53]. The transfer function for a drive unit placed in a vented cabinet is shown in eq. (2.8).

$$p = \frac{\rho S_D Bl U_G}{2\pi r M_{MS} R_E}\, H_{BR}(s) \tag{2.8}$$

The constant term in eq. (2.8) transforms the voltage U_G to the sound pressure p while the $H_{BR}(s)$ represent a fourth order high pass filter and the slope will decrease by 24 dB per octave below the resonance frequency.

$$H_{BR}(s) = \frac{s^4}{s^4 + a_3\omega_0 s^3 + a_2\omega_0^2 s^2 + a_1\omega_0^3 s + \omega_0^4} \tag{2.9}$$

The parameters a_1, a_2 and a_3 in eq. (2.9) can be defined according to the Thiele/Small parameters to output the desired frequency response [3, p. 55].

2.3 Frequency Response

The sound pressure is measured in dB SPL and is a ratio of the sound pressure p and the threshold of hearing $p_{ref} = 20\,\mu\text{Pa}$, see eq. (2.10).

$$L = 20\log_{10}\left(\frac{p}{p_{ref}}\right) \tag{2.10}$$

The transfer functions applies in an area between the drive unit's resonance frequency f_s and approximately the frequency $f_1 \approx \frac{c}{2\pi a}$ where a is the radius of the drive unit's baffle and c is the velocity of sound [3, p. 41].

Chapter 3

MATLAB Framework

To simulate the complete speaker, a MATLAB framework based on OOP is made [1]. This approach is chosen to be able to keep the implementations of the speaker parts separate, should you wish to alter the implementation later on.

3.1 Analysis

The framework must include a speaker consisting of a drive unit and a closed cabinet, should include a crossover filter and could include a bass reflex, according to the Project delimitations. The frequency response of each unit should be exposed for investigation and the frequency response of the entire system must be available, to compare it to the measurements of real speakers.

3.2 Design

The design of the MATLAB framework is shown in fig. 3.1.

The TransferFunction class implements the `plotResponse(f)` method, and the abstract method `transform(x)`. `plotResponse(f)` plots the amplitude spectrum for the frequency values given by the argument `f`. It uses the, implemented by any subclass, `transform(x)` method to access the frequency response.

The Speaker class contains the objects necessary to calculate the transfer function for the complete speaker. It's purpose is to collect the other units, to test them together, and to provide the frequency response to compare with the measured response.

The DriveUnit class describes the drive unit on the basis of Thiele/Small parameters [6] and eq. (3.1) [3, p. 51]. The `plotResponse(f)` function plots

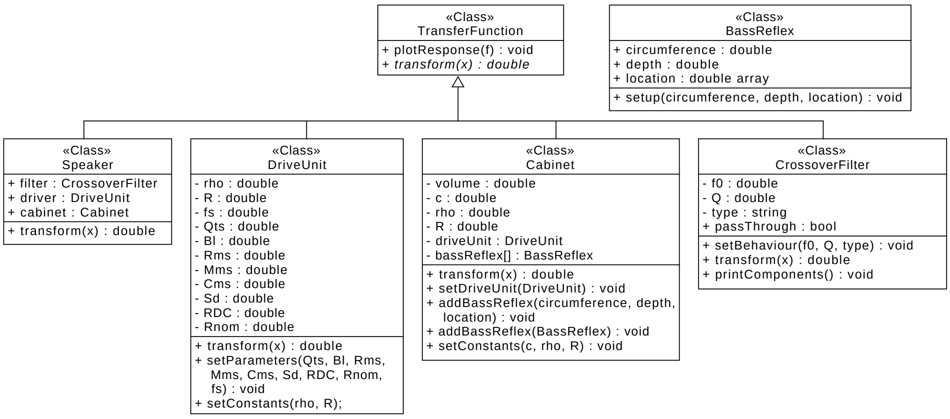


Figure 3.1: Class diagram of the MATLAB framework.

the response of the drive unit in an infinitely large closed box.

$$p = \frac{\rho S_D Bl U_G}{2\pi r M_{MS} R_E} \frac{s^2}{s^2 + s\left(\frac{(Bl)^2}{R_E M_{MS}} + \frac{R_{MS}}{M_{MS}}\right) + \frac{1}{M_{MS} C_{MS}}} \tag{3.1}$$

The Cabinet class describes a closed box cabinet, with the possibility to add a number of bass reflexes and check if their locations are valid. Without bass reflexes `plotResponse(f)` plots the response in a cabinet of the specified volume, according to eq. (2.2). If bass reflexes are present, it should plot the frequency response according to eq. (2.8).

The CrossoverFilter class is meant to filter the input before it reaches the physical parts of the speaker. It is meant to either be used to filter the signal in an appropriate way if several drive units are installed, or to try to refine the signal, if the speaker response is not satisfactory.

The BassReflex class is a bass reflex tube placed somewhere on the cabinet. It cannot itself show a frequency response, but must be placed in a cabinet.

3.3 Implementation

The `DriveUnit` class is shown below for clarification. The overall structure is a standard Matlab class inheriting from TransferFunctions, as shown in fig. 3.1. Following are the private and public properties and the public methods, including the implementation of the `transform(f)` method.

Matlab evaluates the property definition section only once and assigns the same value to the property of every instance in the class. [1] The default values are specified individually as private or public. The private properties include constants so these properties gets a default value that other instances can not access. The datasheet values for the drive unit is public so other instances, e.g. `Cabinet`, can access these values.

The implementations of all the classes in the MATLAB framework can be found in appendix A.

```
classdef DriveUnit < TransferFunctions
    properties (Access = private)
        %% Defaults
        % Density of air (kg/m^3)
        rho     = 1.1839;
        % Distance to microphone (m)
        R       = 1;
        end
    properties (Access = public)
        %% Data sheet values
        fs
        Qts
        Bl
        Rms
        Mms
        Cms
        Sd
        Re
        Rnom
        %% Derived values
        UG
    end
```

The methods are public so `TransferFunctions` can access the `transform(f)` method to plot the frequency response of the drive unit. See appendix A for implemtation of `TransferFunctions`. The implementation of the `transform(f)` method create the transfer function for a drive unit using the properties specified. The transfer function for a drive unit can be found in section 2.2.2, eq. (2.2).

```
    methods (Access = public)
        function p = transform(obj, f)
            % Create the transfer function for a drive unit
            % mounted in an infinite sealed enclosure.
            setDerivedParameters(obj);
            s = 1i .* 2 .* pi .* f;
            k0 = (obj.rho * obj.Bl * obj.Sd * obj.UG) / (2 * pi
            ↪  * obj.R * obj.Re * obj.Mms);
```

```
            k1 = ((obj.Bl^2) / (obj.Re * obj.Mms)) + (obj.Rms / obj.Mms);
            k2 = 1 / (obj.Mms * obj.Cms);
            p = k0 .* (s.^2 ./ (s.^2 + s .* k1 + k2));
        end
```

The `setParameters(Qts, Bl, Rms, Mms, Cms, Sd, Re, Rnom, fs)` method is used to set the parameters for the drive unit. The Thiele/Small parameters is found in the datasheet for the given drive unit. A input validation is performed for the Sd parameter as it is specified either in m^2 or cm^2 in the datasheets.

```
        function setParameters(obj, Qts, Bl, Rms, Mms, Cms, Sd, Re, Rnom, fs)
            % Sets the parameters for the drive unit.
            % The parameters should be found in the datasheet
            % for the given drive unit.
            if  (Sd > 1) && (Sd < 1000)
                obj.Sd = Sd/10000;
            else
                warning('Sd is specified in cm^2')
            end
            obj.Qts = Qts;
            obj.Bl = Bl;
            obj.Rms = Rms;
            obj.Mms = Mms;
            obj.Cms = Cms;
            obj.Re = Re;
            obj.Rnom = Rnom;
            obj.fs = fs;
            setDerivedParameters(obj);
        end
```

The `setDerivedParameters()` method sets the U_G voltage given on the nominel resistance R_{nom} specified in the `setParameters` method. The U_G voltage is set so the input for the drive unit is 1 W since the sensitivity is specified in dB per 1 W in a distance of 1 m.

```
        function setDerivedParameters(obj)
            % Sets the derived parameters
            % dependent on the given drive unit.
            % UG Voltage for 1 W electric power in nominel resistance ohm
```

```
            obj.UG = sqrt(1*obj.Rnom);
        end
```

The `setConstants()` method make it possible for the user to change parameters who usually are constants as density of air. If the method is not called the default values specified in the property definition section is applied.

```
        function setConstants(obj, rho, R)
            % Change default values of rho, pRef and R.
            % Check for correct number of input arguments
            if ~(any([1, 3] == nargin))
                error(' Call setConstants(rho, r) with 2 parameters or\n%s',...
                'with 0, setConstants(), to reset to default.');
            end
            if nargin == 1
                obj.rho = 1.1839;
                obj.R = 1;
            else
                obj.rho = rho;
                obj.R = R;
            end
        end
    end
end
```

An instance of the `DriveUnit` is created and the parameters of the Fountek FW168 drive unit is set with the method `setParameters()`. The method `plotResponse(logspace(1,4,1000))` from the `TransferFunctions` class plots the response between 10 Hz and 10 kHz. In fig. 3.2 the simulated frequency response of the Fountek FW168 drive unit placed in an infinite sealed enclosure is seen.

```
k = DriveUnit();
% setParameters(Qts, Bl, Rms, Mms, Cms, Sd, Re, Rnom, fs)
k.setParameters(0.397, 8.2, 1.3036, 14.7e-3, 0.821e-3, 119, 7.2, 8, 45);
k.plotResponse(logspace(1,4,1000));
```

An instance of the `Cabinet` is created with the volume of the cabinet as an argument. The Fountek FW168 drive unit is set with the method `setDriveUnit()`. The method `plotResponse(logspace(1,4,1000))` from the `TransferFunctions` class plots the response between 10 Hz and 10 kHz. In fig. 3.3 the simulated frequency response of the Fountek FW168 drive unit placed in an cabinet with a volume of 17.9 l is seen.

```
% 26.8cm x 20.2cm x 33.0cm
V = (26.8e-2*20.2e-2*33.0e-2);
u = Cabinet(V);
u.setDriveUnit(k);
u.plotResponse(logspace(1,4,100));
```

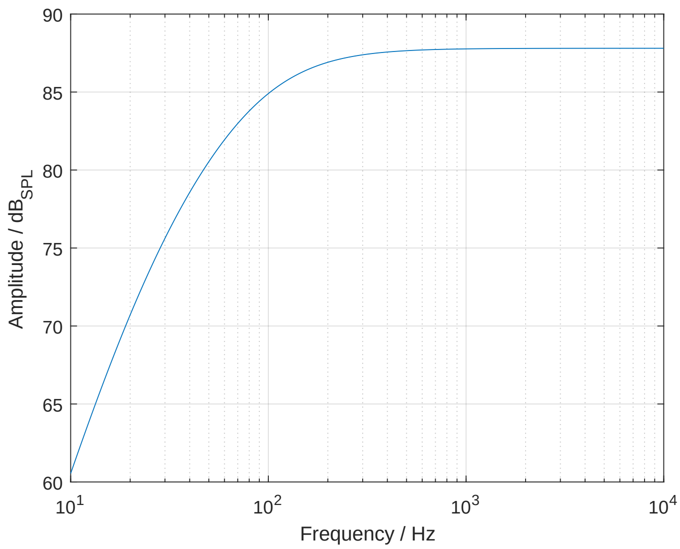


Figure 3.2: Simulated output of the FW168 Fountek drive unit.

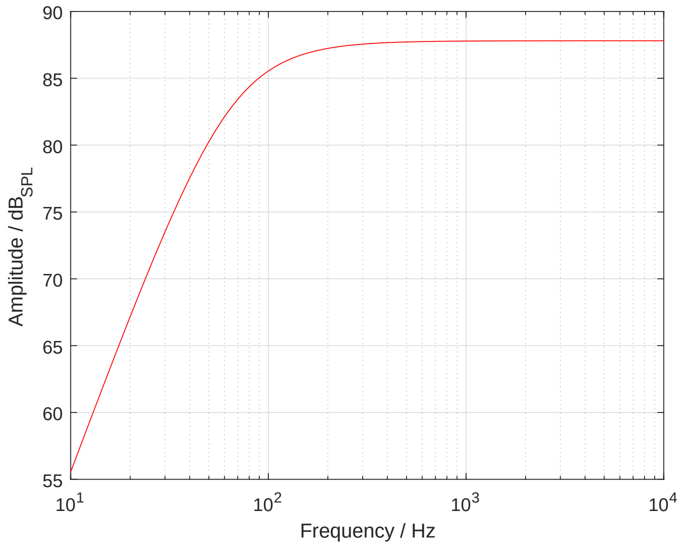


Figure 3.3: Simulated output of the FW168 Fountek drive unit placed in a cabinet (≈17 l).

Chapter 4

Speaker measurement

To compare the MATLAB model with a real speaker, several measurements have been carried out using a PC, the Argon DA-1 amplifier, the Behringer Xenyx 302USB mixer (for PC to microphone connection), a Behringer ECM8000 microphone and alternating between the two speakers shown in fig. 4.1.

4.1 Setup

The measurement setup is shown in fig. 4.2, showing the see-through speaker and the microphone placed 1 m apart on-axis.

The setup has a distance of 1 m to be able to look at the joined output, when the bass reflex is inserted into the see-through speaker.

(a) A see-through speaker with the possibility to remove or insert bass reflexes. Approximate volume: 18 l. Drive Unit: Fountek FW168.

(b) Wooden speaker. Approximate volume: 1.5 l. Drive Unit: Vifa C11WG-09.

Figure 4.1: The two speakers used for measurements.

	See-through	Wooden
S_D/cm^2	119	55.0
M_{ms}/g	14.7	4.9
$C_{ms}/\mathrm{mm\,N^{-1}}$	0.821	0.997
L_e/mH	-	0.4
R_e/Ω	7.2	3.2
$Bl/\mathrm{T\,m}$	8.2	3.8
V_{as}/l	16.5	4.2
f_s/Hz	45	72
Q_{ts}	0.397	0.40
R_{nom}/Ω	8	4

Table 4.1: Thiele/Small parameters for the used drive units according to [7] and [2].

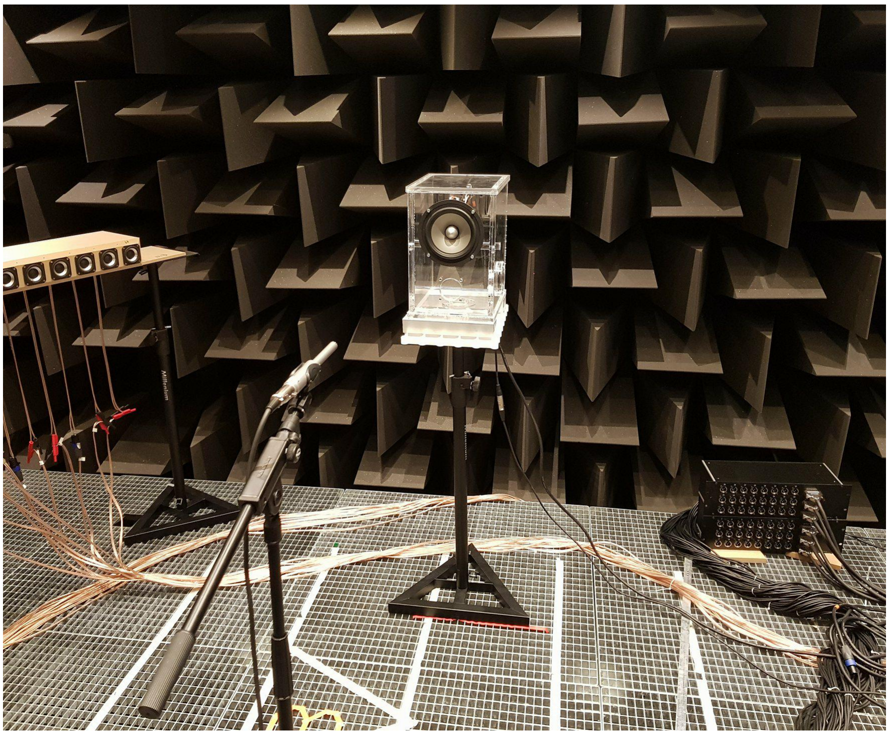
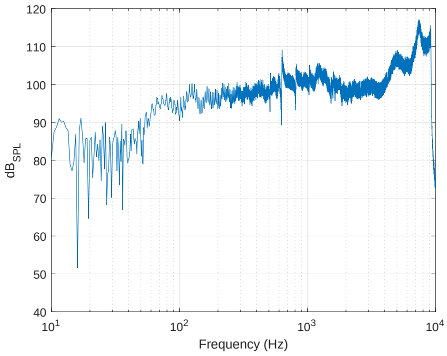
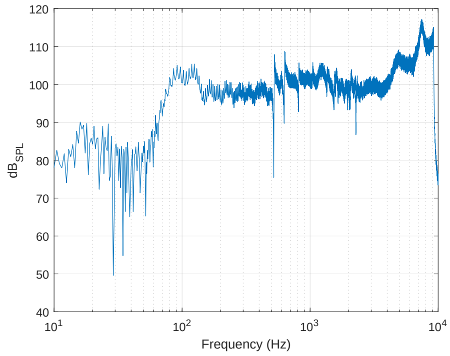


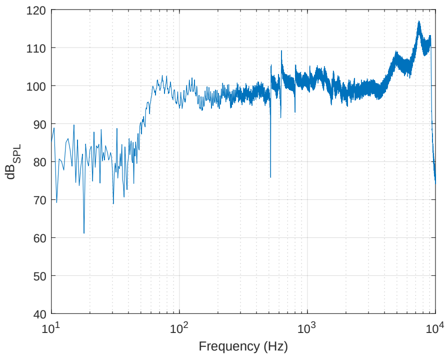
Figure 4.2: Measurement setup showing the microphone and speaker with $r = 1\,\mathrm{m}$.



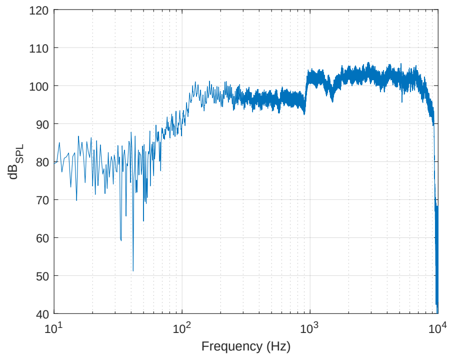
(a) See-through speaker. All openings sealed.



(b) See-through speaker. Bottom opening open.



(c) See-through speaker. Bass reflex with length 7 cm and diameter 5 cm inserted at bottom.



(d) Wooden speaker measurement.

Figure 4.3: Measurements of the speakers.

4.2 Measurement

Three measurements were made with the see-through speaker; closed box, bottom-hole open and bottom 0.27 l bass reflex. One closed box measurement was made with the wooden speaker. The results are shown in fig. 4.3.

Comparing the closed box approach of both speakers, fig. 4.4, they differ mostly in the regions 50 Hz to 100 Hz and 400 Hz to 10 000 Hz. This is most likely due to different Thiele/Small parameters.

When comparing the see-through speaker setups, fig. 4.5a, the responses are very similar, except for in the range 50 Hz to 150 Hz, fig. 4.5b. This is the range at which the response goes from a slope of 12 dB/octave for the closed setup, and 24 dB/octave for the open and bass reflex setups, to a more linear frequency response. Here the gain from using a bass reflex gives approximately 3 dB to 5 dB in the range 60 Hz to 90 Hz.

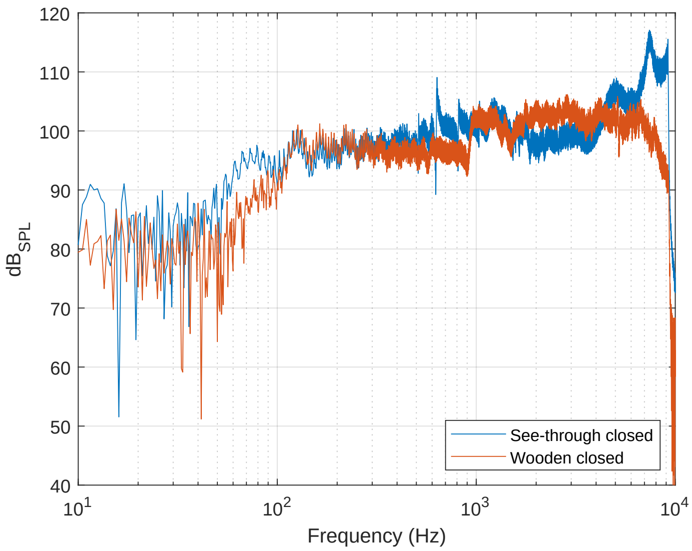
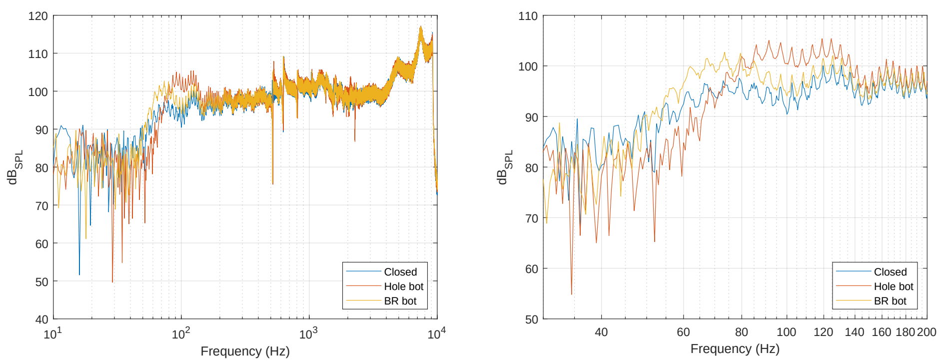


Figure 4.4: Comparison of see-through closed and wooden.



(a) See-through speaker.

(b) Zoom of fig. 4.5a.

Figure 4.5: Comparison of the different setups of the see-through speaker.

Chapter 5

Results

This chapter compares the created models with the measurements.

Figures 5.1 and 5.2 shows the comparisons of the drive unit model, cabinet model and the measurements of the see-through and wooden speakers. The first striking observation is that generally the models have a lower amplitude than the measured data. Secondly the models are very linear after the cut-off frequency, whereas the measurements seems to contain some overtones. A clear similarity in behaviour of the Cabinet Model and the measurements is in the slope, especially when looking at the wooden comparison fig. 5.2 it is clear that the slope of the Drive Unit Model is not as steep as the measurement, whereas the slope of the Cabinet Model matches the measurement very well. Both the speaker measurements and their models exhibit matching cut-off frequencies.

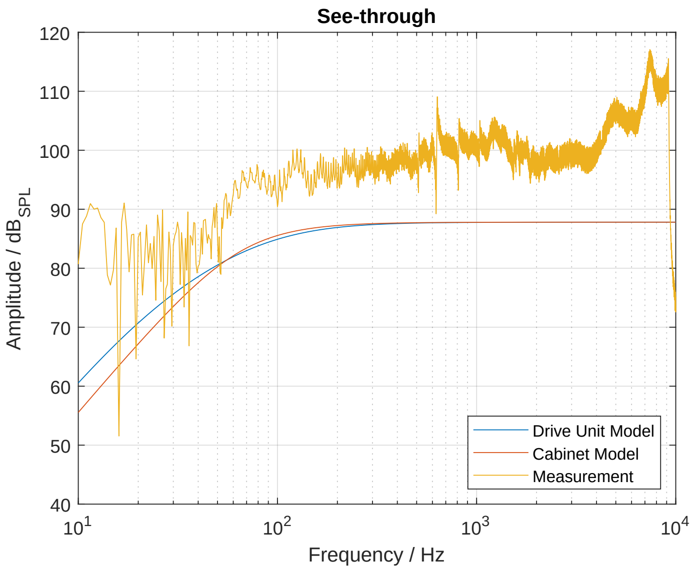


Figure 5.1: The two models for the Drive Unit (blue), Cabinet (red) and the measurement of the see-through speaker (orange).

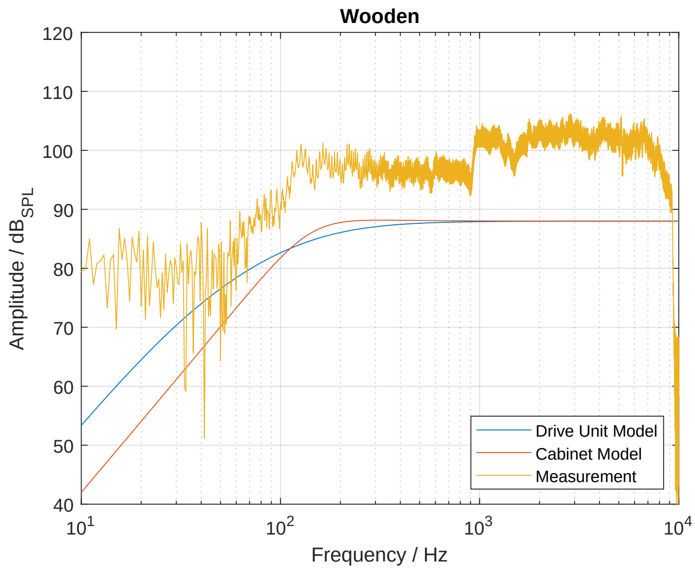


Figure 5.2: The two models for the Drive Unit (blue), Cabinet (red) and the measurement of the wooden speaker (orange).

Chapter 6

Discussion & Conclusion

The goal of this project has been to implement Matlab models of speaker parts in a modulized framework, allowing for easy investigations of speaker frequency responses. The models have been compared to measurements of two speakers in different configurations.

The models implemented are useful for visualizing the effects of changing the Thiele/Small parameters and the volume of the cabinet. The framework has not implemented bass reflexes or the ability to filter an arbitrary signal using the Matlab `filter` function, though the basis for doing so is presented in chapter 2.

The measurements, shown and compared in chapter 4, exhibit the behaviour expected. The fitting of a bass reflex into the cabinet, gives a higher output of lower frequencies, fig. 4.5b. The measurements where made with a linear chirp, which in hindsight did not give as much information in the low frequency area as desired. This could be remedied by using a logarithmic chirp in stead.

Comparing the models and the measurements revealed that the models were able to predict the cut-off frequency of the speaker, and the slope of the high-pass filter. The amplitude of the models though, was approximately 10 dB too low.

On the basis of this we can conclude that the models can be useful for predicting cut-off frequency and slope of a speaker in a cabinet without bass reflexes.

Chapter 7

Bibliography

Articles

[4] Richard H. Small. “Closed-Box Loudspeaker Systems-Part 1: Analysis”. In: **J. Audio Eng. Soc** 20.10 (1972), pp. 798–808. URL: http://www.aes.org/e-lib/browse.cfm?elib=2022.

[5] Neville Thiele. “Loudspeakers in Vented Boxes: Part 1”. In: **J. Audio Eng. Soc** 19.5 (1971), pp. 382–392. URL: http://www.aes.org/e-lib/browse.cfm?elib=2173.

Books

[3] Tore Arne Skogberg. **Elektroakustik, Version 2.6**.

Datasheets

[2] **Fountek FW168**. URL: https://www.parts-express.com/pedocs/specs/296-729-fountek-fw168-specifications.PDF.

Webpages

[1] **Class Definition**. URL: https://se.mathworks.com/help/matlab/object-oriented-programming-in-matlab.html.

[6] **Thiele/Small parameters**. URL: https://en.wikipedia.org/wiki/Thiele/Small_parameters.

[7] **Vifa Speakers C11WG-09-04**. URL: http://www.oregondv.com/Vifa_Speakers_C11WG-09-04.htm.

Appendix A

Code

A.1 TransferFunctions

```
classdef TransferFunctions < matlab.mixin.SetGet
  properties (Access = protected)
    pRef = 20e-6;
  end

  methods (Abstract, Access = public)
    L = transform(obj, f)
  end

  methods
      function handle = plotResponse(obj, f, H)
      if nargin == 1
        f = logspace(2, 4, 100);
      elseif nargin >= 4
        error('Supply a list of frequencies to plot or none to plot from 100 Hz to 10000 Hz');
      end
      p = transform(obj, f);
      L = 20 * log10(abs(p) ./ obj.pRef);

      if nargin == 3
        figure(H);
      else
        figure;
      end
      semilogx(f, L);
      hold on
      grid on
      xlabel('Frequency / Hz');
      ylabel('Amplitude / dB_{SPL}');
```

```

      handle = gcf;
    end
  end
end
```

A.2 DriveUnit

```
classdef DriveUnit < TransferFunctions
  properties (Access = private)
    %% Defaults
    % Density of air (kg/m^3)
    rho     = 1.1839;
    % Distance to microphone (m)
    R       = 1;
  end
  properties (Access = public)
    %% Data sheet values
    fs
    Qts
    Bl
    Rms
    Mms
    Cms
    Sd
    Re
    Rnom
    %% Derived values
    UG
  end

  methods (Access = public)
      function p = transform(obj, f)
          % TRANSFORM Create the transfer function for a drive unit
          % mounted in an infinite sealed inclosure.
          %
          % L = TRANSFORM(f) Returns the sound pressure.
          setDerivedParameters(obj);
          % p = rho*Sd*Bl*UG/2*pi*Mms*Re
```

```
        s = 1i .* 2 .* pi .* f;
        k0 = (obj.rho * obj.Bl * obj.Sd * obj.UG) / (2 * pi * obj.R * obj.Re * obj.Mms);
        k1 = ((obj.Bl^2) / (obj.Re * obj.Mms)) + (obj.Rms / obj.Mms);
        k2 = 1 / (obj.Mms * obj.Cms);
        p = k0 .* (s.^2 ./ (s.^2 + s .* k1 + k2));
    end

    function setParameters(obj, Qts, Bl, Rms, Mms, Cms, Sd, Re, Rnom, fs)
        % SETPARAMETERS Sets the parameters for the drive unit.
        % The parameters should be found in the datasheet for the
        % given drive unit.
        %
        % SETPARAMETERS(Qts, Bl, Rms, Mms, Cms, Sd, Re, Rnom, fs)
        % Qts  Total Q of the drive unit [0.0 < Qts < 1.0]
        % Bl   Product of magnet field strength and length of wire in the magnetic field (Tm)
        % Rms  Mechanical resistance of the drive unit's suspension (Ns/m)
        % Mms  Mass of the diaphragm/coil (kg)
        % Cms  Compliance of the drive unit's suspension (m/N)
        % Sd   Effective surface area of driver diaphragm (cm^2) [1.0 < Sd < 1000.0]
        % Re   DC resistance of the voice coil (ohm)
        % Rnom Nominel resistance of drive unit (ohm)
        % fS   Resonance frequency of the drive unit (Hz)
        if  (Sd > 1) && (Sd < 1000)
            obj.Sd = Sd/10000;
        else
            warning('Sd is specified in cm^2')
        end
        obj.Qts = Qts;
        obj.Bl = Bl;
        obj.Rms = Rms;
        obj.Mms = Mms;
        obj.Cms = Cms;
        obj.Re = Re;
```

```
          obj.Rnom = Rnom;
          obj.fs = fs;
          setDerivedParameters(obj);
      end

      function setDerivedParameters(obj)
          % SETDERIVEDPARAMETERS Sets the derived parameters
          % dependent on the given drive unit.
          %
          % UG Voltage for 1 W electric power in nominel resistance ohm
          obj.UG = sqrt(1*obj.Rnom);
      end

      function setConstants(obj, rho, R)
        % SETCONSTANTS Change default values of rho, pRef and rf.
        %
        % SETCONSTANTS(rho, pRef, rf)
        % rho  Density of air (kg/m^3)
        % R   Distance to microphone (m)

        % Check for correct number of input arguments
        if ~(any([1, 3] == nargin))
          error(' Call setConstants(rho, r) with 2 parameters or\n%s',...
          'with 0, setConstants(), to reset to default.');
        end
        if nargin == 1
          obj.rho = 1.1839;
          obj.R = 1;
        else
          obj.rho = rho;
          obj.R = R;
        end
      end
  end
end
```

A.3 Speaker

```
classdef Speaker < TransferFunctions
  properties (Access = public)
    filter
    driver
    cabinet
  end
  methods
    function p = transform(obj, f)
      % Insert test for if cabinet already has a driver, else input the
      % specified driver from obj.driver
      p = obj.filter.transform(f) .* obj.cabinet.transform(f);
    end
  end
end
```

A.4 Cabinet

```
classdef Cabinet < TransferFunctions
  properties (Access = private)
    %% Defaults
    % Speed of sound (m/s) (default for 25 ??C)
    c = 346.13;
    % Density of air (kg/m^3) (default for 25 ??C)
    rho = 1.1839;
    % Distance to microphone (m)
    R = 1;

    %% Box parameters
    % Box volume (m^3)
    volume
    % Any bass reflexes
    bassReflex

    % Drive unit
    driveUnit

    % Derived parameters
    CAB
```

```
    end

    methods (Access = public)
    % Returns the sound pressure  in dB SPL
      function pF = transform(obj, f)
          setDerivedParameters(obj);
          s = 1i .* 2 .* pi .* f;
          k0 = (obj.rho / (2 * pi * obj.R)) * ((obj.driveUnit.Bl * obj.driveUnit.Sd * obj.driveUnit.UG) / (obj.driveUnit.Re * obj.driveUnit.Mms));
          k1 = (obj.driveUnit.Bl^2 / (obj.driveUnit.Re * obj.driveUnit.Mms)) + obj.driveUnit.Rms / obj.driveUnit.Mms;
          k2 = (1 / (obj.driveUnit.Mms * obj.driveUnit.Cms)) * (1 + ((obj.driveUnit.Cms * obj.driveUnit.Sd^2) / obj.CAB));
          pF = k0 .* (s.^2 ./ (s.^2 + s .* k1 + k2));
          %qF = obj.FA ./ (obj.RAE + s .* obj.MAS + 1 ./ (s .* obj.CAS) + obj.RAS + 1 ./ (s .* obj.CAB));
          %pF = obj.rho .* s .* qF ./ (2 * pi * obj.R);
      end

      % Sets the derived parameters dependent on the given Drive Unit
      function setDerivedParameters(obj)
        obj.CAB = obj.volume / (obj.rho * obj.c.^2);
      end
    end

    methods (Access = public)
      % Constructor
      function obj = Cabinet(volume)
        try
          narginchk(1, 1);
          obj.volume = volume;
        catch
          warning('Cabinet created without volume. Volume set to 1 m^3.');
          obj.volume = 1;
        end
      end

      % Setting the drive unit
```

```
    function setDriveUnit(obj, driveUnit)
      obj.driveUnit = driveUnit;
    end

    function setConstants(obj, c, rho, R);
      % Check for correct number of input arguments
      if ~any([1, 4] == nargin)
        error(' Call setConsants(c, rho, r) with 3 parameters or\n%s',...
        'with 0, setConstants(), to reset to default.');
      end

      if nargin == 1
        obj.c = 346.13;
        obj.rho = 1.1839;
        obj.R = 1;
      else
        obj.c = c;
        obj.rho = rho;
        obj.R = R;
      end
    end

    function setBassReflex(circumference, depth)
    end

  end
end
```

A.5 CrossoverFilter

```
classdef CrossoverFilter < TransferFunctions
    properties (Access = public)
    passthrough = 0;

    properties (Access = private)
        f0
        Q
        type

```

```
        % Derived
        d
    end
    methods (Access = public)
        function passThrough()
        end

        function A = transform(obj, f)
            setDerivedParameters(obj);
            s = 1i * 2 * pi .* f;
            switch obj.type
                case 'low'
                    %A = 1 ./ (1 + s / (2 * pi * obj.f0)); % first order
                    A = (((2 * pi * obj.f0)^2) ./ (((2 * pi * obj.f0)^2)...
                        + 2 * obj.d * (2 * pi * obj.f0) * s + s.^2)); % second order
                    %L = 20 .* log10(abs(lp) ./ obj.pRef);
                case 'high'
                    %A = (s / (2 * pi * obj.f0)) ./ (1 + (s / (2 * pi * obj.f0))); % first order
                    A = (s.^2 ./ (((2 * pi * obj.f0)^2) + 2 * obj.d * ...
                        (2 * pi * obj.f0) * s + s.^2)); % second order
                    %L = 20 .* log10(abs(hp) ./ obj.pRef);
                otherwise
                    warning('Type must either be "high" or "low"');
            end
        end
        function setDerivedParameters(obj)
        % Sets the derived parameter dependent on the Q-value
          obj.d  = 1/(2.*obj.Q);
        end
        function printComponents()
        end
    end
    methods (Access = public)
        function setBehaviour(obj, f0, Q, type)
            obj.f0 = f0;
            obj.Q = Q;
```

```
            if (ischar(type) == 1)
                obj.type = type;
            else
                error('Type must be a char');
            end
        end
    end
end
```

A.6 BassReflex

```
classdef BassReflex
  properties
    circumference
    depth
    location
  end
  methods
    function setup(circumference, depth, location)
      obj.circumference = circumference;
      obj.depth = depth;
      obj.location = location;
    end
  end
end
```