IMPROVING THE SOUND ABSORPTION OF CROSS-LAMINATED TIMBER PANELS USING RESONANT ABSORBER LAYERS

by

Banda Logawa

B.A.Sc., The University of British Columbia, 2013

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

June 2017

Abstract

Developed in the mid-1990s in Austria and Germany, Cross Laminated Timber (CLT) is an innovative wood product known for its strength in both orthogonal directions, and its dimensional stability, making it a sustainable alternative to concrete slabs. CLT is created through the cross-lamination process, which glues together odd number of layers of wood planks placed in orthogonally alternating directions.

With the growing interest in the application of CLT in North America, numerous studies has been conducted to characterize the acoustical properties of CLT panels. However, most of them focused on the sound-transmission aspect of CLT, very few on the sound absorption. This thesis will explore the sound-absorption characteristics of CLT, the effect on overall room-acoustical conditions, the utilization of resonant sound-absorbing layers on CLT to make it more sound-absorptive, and proposed solutions to improve this performance aspect.

To demonstrate the low sound absorption and poor acoustical conditions in rooms with exposed and untreated CLT panels, several in-situ reverberation-time (RT) measurements were conducted in multiple buildings in British Columbia. Average sound-absorption coefficients and estimated Speech Intelligibility Indices (SII) were calculated as baseline performance measures for this study. Based on the results from five different buildings, involving 8 rooms configurations, average sound-absorption coefficients for exposed CLT panels are approximately between 0.02 to 0.13, resulting in barely acceptable conditions for verbal communication.

To optimize the sound-absorption characteristics of prototype CLT panels, a transfer-matrix model has been developed to predict the performance of multi-layered CLT panels. This theoretical model was then validated by using three different sound-absorption measurement methods (impedance tube, spherical decoupling, and reverberation chamber) for multiple HR array configurations.

After identifying the important parameters of an HR system and their effects on performance, a final prototype configuration with Helmholtz Resonator Array was then created with the goal of improving the room- acoustical performance of CLT, as well as responding to input from the CLT manufacturers and experts. Both the theoretical and experimental results confirmed that the proposed solution has the required sound-absorption performance and achieves all research objectives.

Lay Summary

Cross-Laminated Timber (CLT) is generally considered as a sustainable alternative to concrete slabs. Due to its cross-lamination process, CLT is known for its strength in both orthogonal directions and dimensional stability. This research is conducted to study the often-overlooked aspect of CLT which is its sound absorption property. By measuring reverberation time, speech intelligibility index, and sound absorption coefficient of CLT surfaces in 5 different CLT buildings in British Columbia, the author found that CLT surfaces has very low sound absorption and may lead to excessive noise and low speech intelligibility especially in the middle frequencies. In this thesis, prediction tool to calculate the acoustical properties of a multi-layered system consisting of CLT surfaces were developed and validated. It was later utilized to design a prototype multi-layered CLT system which solve these undesirable room acoustical properties.

Preface

This thesis is an original, unpublished, and independent work of the author performed under the direct supervision of Dr. Murray Hodgson, UBC Acoustics and Noise Research Group, and in collaboration with Dr. Ciprian Pirvu, FPInnovations Vancouver, as well as Dr. Frank Lam, UBC Wood Science Department.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Murray Hodgson, for his endless support and encouragement during my study. Without his guidance, I would not be able achieved all that I aimed for during my research.

I would also like to thank Dr. Ciprian Pirvu and Dr. Frank Lam for being the collaborators on this research project. They have provided invaluable inputs on how to tackle my research objectives. Furthermore, I couldn’t have procured all materials required for my prototypes without their help.

I’m also grateful for the various support that the NEWBuildS network has provided me during my research. Without their connection and networking opportunity, I would not be able to meet Bill Downing and other people from Structurlam who had provided essential feedbacks during the early stage of the research.

Last but not least, I would like to thank my exam committee, Dr. James Olson and Dr. Chris Waltham, and all members of UBC Acoustics and Noise Research Group, UBC Sustainable Building Science Program, and NEWBuildS network.

Dedication

To the loving memory of my father, Gunawan Logawa, who had supported and encouraged me to pursue my goals in life.

To my dear mother, Sandrayani Setiadarma, and brother, Garda Logawa, who have always been there for me during my toughest moment.

1 Introduction

1.1 Project Motivation

Applications of Cross-Laminated Timber (CLT) as a building material in North America have grown significantly during the last few years. Compared to the typical light-frame wood-construction method, it is possible to build mid-to-high-rise wood-based buildings with CLT due to its relative in-plane and out-of-plane strengths and stiffness in both longitudinal and transversal directions. Furthermore, due to the prefabricated nature of CLT, it offers relatively quick building erection time, as opposed to other construction methods, such as concrete slabs.

FPInnovations has played a major role in providing guidelines and key technical information on CLT construction in North America by releasing handbooks for Canada in 2011 [11] and the United States in 2013 [12]. These handbooks cover various aspects of CLT, such as its design, manufacturing, and performance in several areas such as: structural design, seismic performance, duration of load and creep factors, vibration performance, acoustic (sound-transmission) performance, enclosure design, etc.

Even though extensive studies have been done to characterize the sound-transmission and impact-insulation performance of CLT and its assemblies, as seen in the "**CLT Handbook**" and other publications [13], there are very few studies dedicated to the sound absorption of CLT. The low sound-absorption of entreated and exposed CLT may result in excessive reverberation time. This may cause detrimental effects to the speech intelligibility and the acoustical conditions in some room applications.

This study was conducted to characterize and improve the sound-absorption performance of CLT and its assemblies, and provide sufficient information and guidelines to calculate and design CLT assemblies with resonant sound-absorber layers and high sound absorption.

1.2 Objectives and Scopes

This study is conducted to achieve several objectives such as:

- Review the sound-absorption of existing untreated Cross-Laminated Timber (CLT) Panels
- Review different methods to measure sound-absorption coefficient
- Study the theory of Helmholtz Resonators, the factors that affect their acoustical performance, and different ways of integrating them into CLT panels
- Establish sound-absorption performance targets for various room configurations
- Develop a prediction model by using the transfer-matrix method to assist the design of multi-layered CLT panels with an additional Helmholtz-Resonator layer
- Develop and test prototype panels as proofs of concept and refine them for various room applications.

The scope of the study is limited to the sound-absorption aspects of CLT panels. Because of this, other acoustical aspects of the panels, such as their sound transmission and surface scaterring, are only discussed briefly in this thesis. However, it would be possible to use the proposed transfer-matrix model of CLT to predict the sound-transmission aspect of multi-layered CLT panels. Furthermore, the focus of the study is on the implementation of an additional acoustical layer in addition to the existing CLT panel, such that the sound-transmission performance of the resulting product would mostly be equal to or better than the existing product.

1.3 Research Methodology

Due to the limited amount of study which had been conducted to determine and improve the sound absorption of CLT, it was crucial to measure this performance by using proven methods, commonly used in the industry. Three different measurement methods were utilized in the study: the impedance-tube method [7] to measure the normal-incidence sound absorption, the spherical-decoupling method [8] [9] to determine sound absorption at specific angles of incidence, and the reverberation-room method [10] to measure the random-incidence sound absorption coefficient which is most applicable to sound in rooms. Using these values as the baseline sound-absorption performance of the CLTs, desired sound-absorption performance characteristics were then identified based on the optimal room reverberation-time and speech-intelligibility criteria.

While the basic theory and understanding of simple Helmholtz-Resonator sound-absorption performance (frequency and magnitude) has long been established, it was in the interest of the study to develop a robust prediction model to take into account various layer compositions and materials in a multi-layer resonant sound-absorber system, especially for a relatively new materials such as Cross-Laminated Timber (CLTs). This would enable designers and product manufacturers to combine multiple materials (perforated panels, porous sound absorbers, isotropic and orthotropic solids) and create CLT assemblies with the desired sound-absorption performance.

After developing the mathematical model, several prototypes were then created and tested using the same measurement techniques which were used to test the baseline sound-absorption performance. The resulting sound-absorption performance was then compared to the predicted value.

1.4 Brief History of Cross-Laminated Timber

Cross-Laminated Timber (CLT) were first introduced in Austria and Germany in the mid-1990s. However, CLT building construction was not significant until the early 2000s in Europe. Compared to the European market, CLT applications in North America have just recently begun in the early 2010s. CLT was introduced as a green alternative to the typical 'non-light' construction methods, such as concrete and masonry.

Applications of Cross-Laminated Timber (CLT) as a building material in North America have grown significantly during the last few years. Compared to the typical light-frame wood-construction method, it is possible to build mid to high-rise wood-based buildings with CLT due to its relative in-plane and out-of-plane strengths and stiffness in both longitudinal and transversal directions. Furthermore, due to the prefabricated nature of CLT, it offers relatively quick building-erection times, as compared to other construction methods such as concrete slabs. Finally, CLT products are very sustainable, especially because they are mostly made using beetle-kill pines which are abundant in Canada (about 12.7 million hectares in 2012) [14].

As a revolutionary material, boundaries of mid-high rise wood building are constantly being pushed back with the introduction of sustainable building materials such as CLT products. Due to its strength and exotic appearance, many building developers are currently considering this sustainable option. This sustainable movement towards mid-to-high-rise buildings is further expanded through the efforts of FPInnovations, which provides the construction guidelines for CLT in the CLT Handbook [11]. During the time of writing of this report, an 18-storey-high CLT building is under construction at the University of British Columbia and scheduled to be finished in 2017 [15].

Cross-Laminated Timber (CLT) typically consists of multiple layers of wooden boards glued in a crosswise manner (typically orthogonal to one another). Due to this cross-lamination process, CLT panels has relatively high in-plane and out-of-plane strength and stiffness in both orthogonal directions, which makes it comparable to reinforced concrete slabs. The typical construction of CLTs panels consists of odd numbers of layers (typically varying between 3 and 9), depending on the required mechanical performance. The orientations of the outer layers of CLT panels are typically specified such that they are parallel to the vertical loads for wall application, and parallel to the major span direction for floor and roof systems. Typical constructions of CLT panels can be seen in **Figure 1** and **Figure 2** [11].

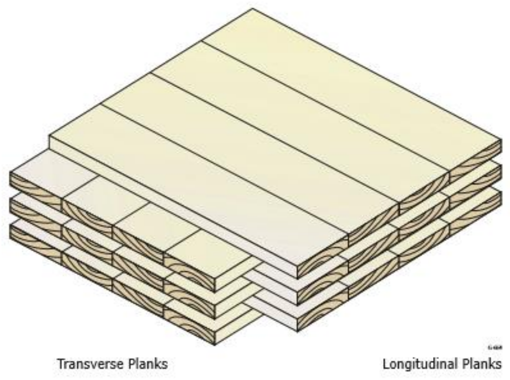


Figure 1. CLT Panel Configuration [11].

To standardize the performance of CLT panels, **ANSI/APA PRG320-2012** standard [16] is commonly adopted by manufacturers in North America to provide guidelines on the panel dimensions and dimensional tolerances, component requirements, CLT performance criteria, qualification and product marking, and quality assurance. A similar process can also be found in Europe, as seen in the **European Technical Approval: ETA-06/0138** document for CLT products from KLH, one of the major CLT manufacturer in Europe [17].

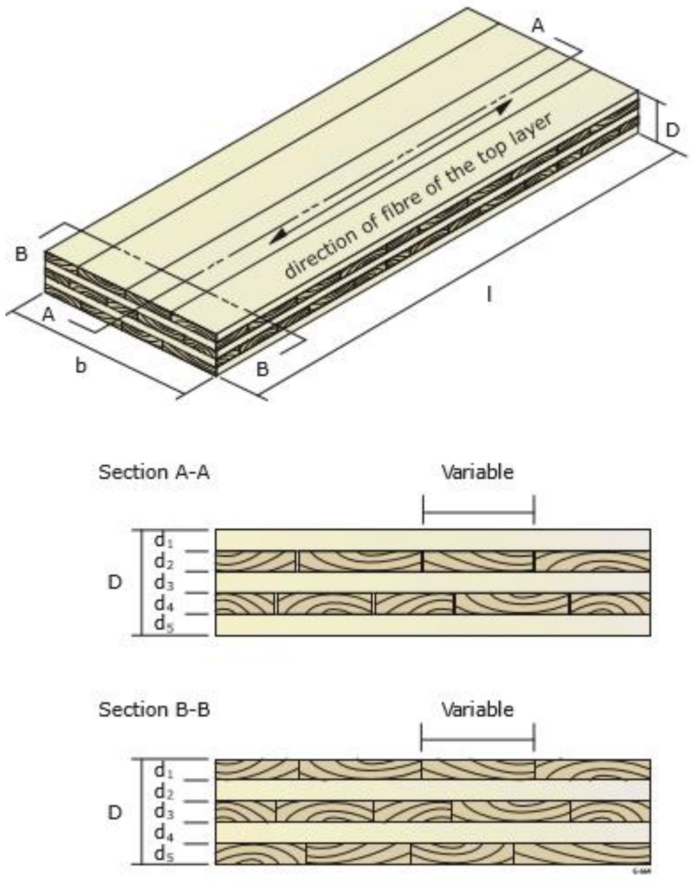


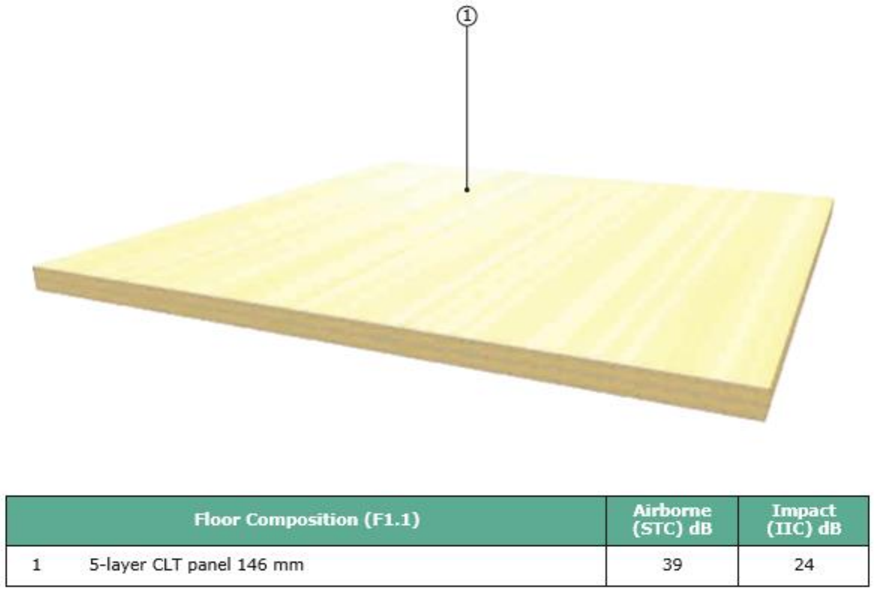
Figure 2. CLT Panel Cross-sections [11].

1.5 Literature Review on Acoustic Properties of Cross-Laminated Timber

1.5.1 Sound-Transmission Class and Impact-Insulation Class

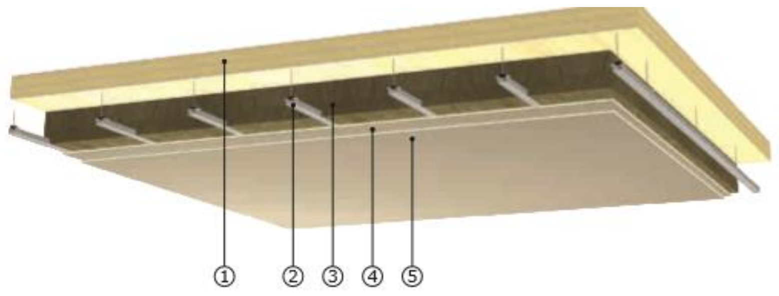
With the growing popularity of CLT as an innovative green building material in North America, numerous studies were conducted to determine various performance aspects of CLTs, such as their seismic and vibration, structural, sound-insulation and fire performances. Most of these results are presented in the "**CLT Handbook",** which was published by FPInnovations in 2011 for Canada [11] and 2013 for United States [12]. In the handbook, airborne Sound

Transmission Class (STC) and Impact Insulation Class (IIC) were studied for various floor assemblies for the North American market. Based on the study done for FPInnovations by the French Institute of Technology for Forest-Based and Furniture Sectors (FCBA) in 2009, a basic 5-ply CLT panel would provide an STC value of 39 dB and an IIC of 24 dB (see **Figure 3**). A CLT floor assembly with suspended ceiling configuration may have STC and IIC values of 63 dB and 62 dB, respectively (see **Figure 4**). Furthermore, it was also shown that floor configurations made from wood topping subfloor and suspended ceiling performed the best, with STC and IIC values of 66 dB and 69 dB, respectively [11] (**Figure 5**).



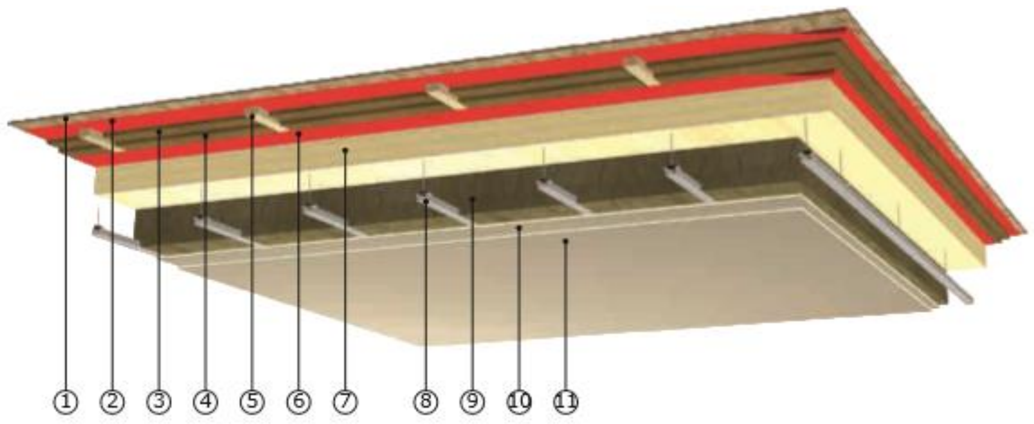
	Floor Composition (F1.1)	Airborne (STC) dB	Impact (IIC) dB
1	5-layer CLT panel 146 mm	39	24

Figure 3. STC and IIC values of 5-layer CLT [11].



	Floor Composition (F4.4)	Airborne (STC) dB	Impact (IIC) dB
1	5-layer CLT panel 146 mm	63	62
2	Resilient supports and rails (200 mm)		
3	Sound insulation material (fibre glass) (200 mm)		
4	Gypsum board 15 mm		
5	Gypsum board 15 mm		

Figure 4. STC and IIC values of 5-CLT Panel with Suspended Ceiling Configuration [11].



	Floor Composition (F4.5)	Airborne (STC) dB	Impact (IIC) dB
1	OSB panel 15 mm	66	69
2	Flooring underlayment ROBERTS		
3	Low-density fibre board THERMISOREL 20 mm		
4	Low-density fibre board THERMISOREL 20 mm		
5	Lumber sleepers 40 mm x 40 mm		
6	Flooring underlayment ROBERTS		
7	5-layer CLT panel 146 mm		
8	Resilient supports and rails (200 mm)		
9	Sound insulation material (fibre glass) (200 mm)		
10	Gypsum board 15 mm		
11	Gypsum board 15 mm		

Figure 5. STC and IIC Values for 5-layer CLT Panel with Both Wood Topping Subfloor and Suspended Ceiling [11].

Comparable measurement results were also found in a laboratory test conducted by Pérez and Fuente in 2013 on 3-ply (81 mm thickness) and 5-ply (135 mm thickness) EGO CLT products [18]. In the study, it was found that the Equivalent Sound Transmission Class (ESTC) of 3-ply and 5-ply Bare CLT were approximately 30 dBA and 37 dBA, respectively. Improvement by adding an anti-impact element and plasterboard plus a suspended ceiling to the 5-ply CLT increased the R value by up to an additional of 21 dBA.

In-situ STC and IIC values of various CLT assemblies have also been measured by Pagnoncelli et al. as seen in [13]. In the study, 8 different residential dwellings in Milan North Area, which were built entirely using CLT systems, were chosen. ESTC of the CLT systems tested were found to be approximately 12 dB higher than typical double-leaf cavity systems used in light-frame timber construction.

In December 2014, the National Research Council of Canada (NRC) released a technical report on sound insulation in mid-rise wood buildings, which included various CLT floor and wall assemblies [19]. STC of bare 3-ply and 5-ply CLT panels used in the test were found to be 33 dB and 38 dB, respectively. To achieve the sound-insulation requirement of STC 50 or higher, as indicated in the 2010 version of National Building Code of Canada (NBCC), it was found that a 5-ply CLT needed to be coupled with a gypsum membrane on wood furring with a spacing of at least 400 mm. For thinner constructions, such as the 3-ply CLT, STC 50 or higher was achieved with gypsum board mounted on resilient channel with 600 mm spacing on 38 mm wood furring with 400 mm spacing. Furthermore, it was also found that double 3-ply CLT performed the best compared to the traditional 3-ply and 5-ply CLT. In addition to the airborne sound-transmission measurement, various flanking-transmission measurements of the CLT systems were also discussed in the report.

1.5.2 Sound-Absorption Coefficient and Reverberation Time

Unlike the sound-transmission aspect of CLT panels, there have been very few studies conducted on the sound absorption. This might cause future complications, as most untreated internal building surfaces, such as CLT panels, are likely very sound reflective, leading to highly reverberant and poor quality acoustical conditions. In 2012, Nore et al. from Norwegian Institute of Technology studied the effects of integrating Helmholtz Resonators into CLT panels by introducing internal cavities inside of the panels [1]. These cavities were created by manually placing the wood planks of the CLT panels such that they would have gaps in between elements, as seen in **Figure 6** and **Figure 7**.

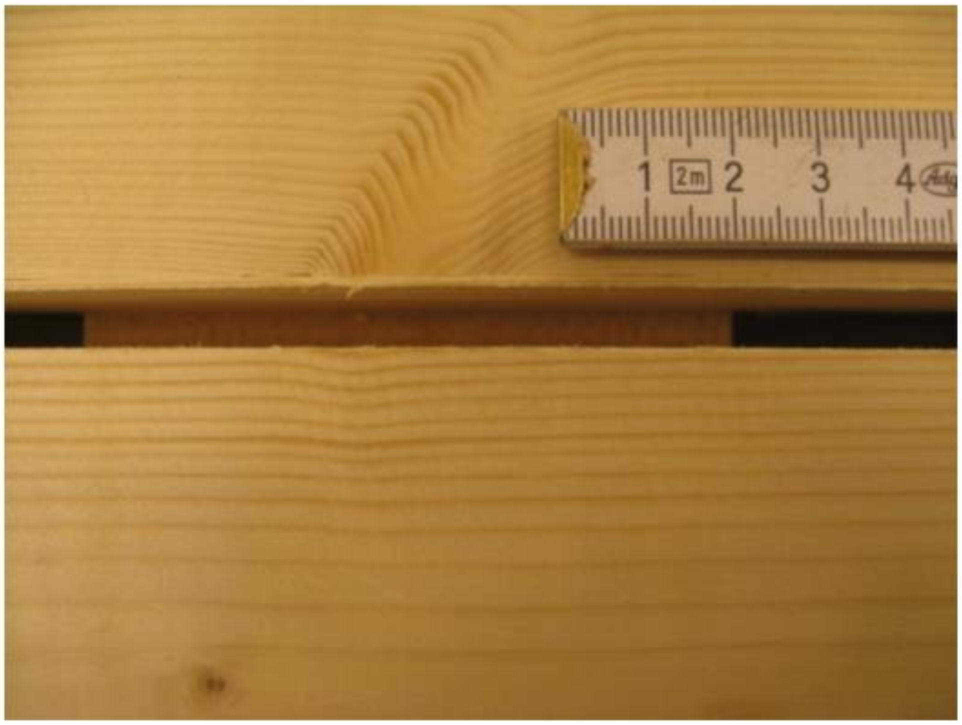


Figure 6. CLT panels with Helmholtz Resonator Created by Norwegian Institute of Technology [1].

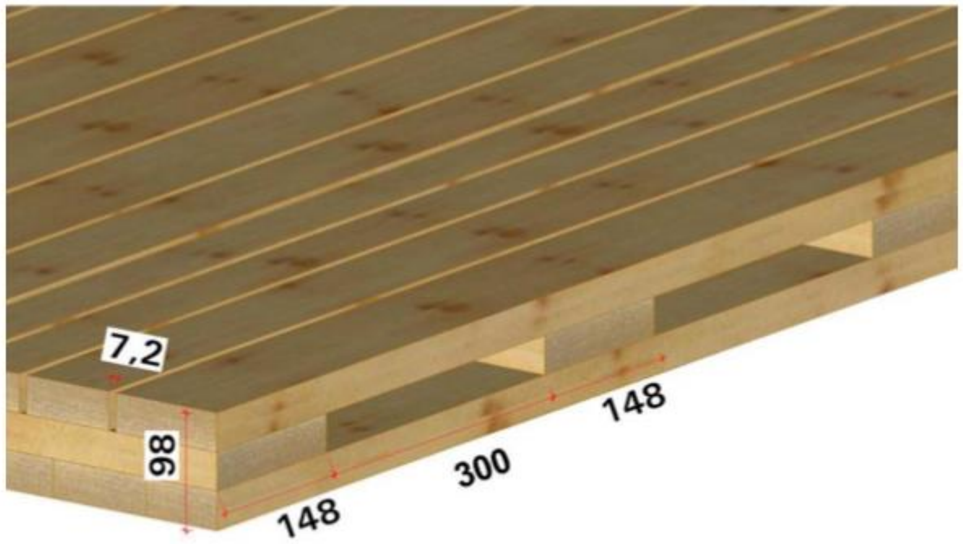


Figure 7. Side View of HR- CLT Panels by Norwegian Institute of Technology [1].

In the study, Nore et al. tested various configurations of Helmholtz Resonator Integrated CLT (HR-CLT) panels with different perforation ratios (ratio between the air gap at the first layer and the panel frontal area), cavity volumes and also porous-absorber materials (felt, mineral wool, acoustic cloth). It was found that the highest weighted absorption coefficient (α_w) of 0.25, with maximum absorption coefficient of 1 at 315 Hz, was achieved using a perforation ratio of 7%, cavity volume of 66%, with felt or mineral wool and acoustic cloth combination as the damping material. Even though these results had shown relatively promising sound-absorption performance, most of the resulting panels had comparably lower (more than 50% reduction) mechanical strength compared to standard CLT panels [1].

1.6 Current CLT Manufacturing Process and Design Considerations

During the period of this study, CLT panels were manufactured by only two companies in Canada: Structurlam Products in Penticton, British Columbia, and Nordic Structures in Montreal, Quebec. Due to the close proximity to University of British Columbia (UBC), Structurlam was chosen as a site-visit location to learn the current manufacturing process of CLT products. Furthermore, it was also essential to gather input from the industry and identify the design limitations and concerns which might affect the research.

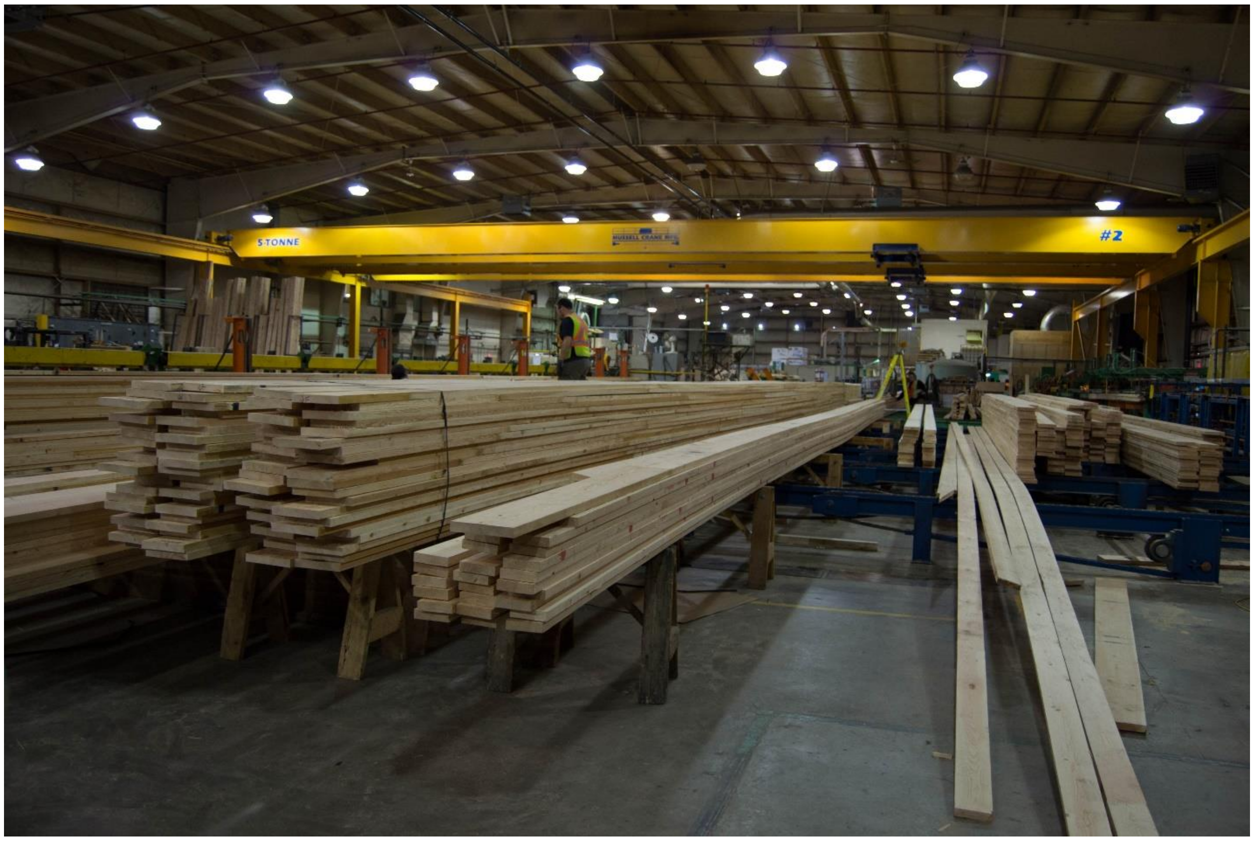


Figure 8. Material Preparation Station for CLT and Glulam in Structurlam.

According to Bill Downing, president of Structurlam, there were two main concerns with integrating Helmholtz-Resonator mechanisms into the CLT itself: severe reduction in mechanical strength (as seen in the previous study by Nore et al. [1]) and complications in the manufacturing process. To create products similar to those described in [1], manual placement of alternating wood planks was required which would exceed the adhesive open time, currently set to be maximum of 30 minutes. While the added sound absorption might be beneficial, it would increase the manufacturing cost significantly.

Similar concerns about the reductions of the mechanical strength of CLT with the introduction of internal cavities and air channels (typical in Helmholtz Resonator construction) were expressed by Dr. Frank Lam from UBC Wood Science and Dr. Ciprian Pirvu from FPInnovations Vancouver. In addition to the reduction in mechanical strength, these types of constructions may also cause detrimental effects on the fire resistance of CLT panels.

Keeping all of these feedbacks in mind, the author decided to pursue a different approach to adding sound absorption to CLT, compared to the one proposed by Nore et al. [1]. Instead of studying the effects of adding internal cavities into CLT panels, this study explored the acoustic-structure interaction between CLT and an additional HR layer. Furthermore, mathematical model and design tool for creating this multi-layered CLT assembly was developed.

2 Room Acoustics in CLT Buildings

Architectural Acoustics is typically divided into two general categories: room acoustics and buildings acoustics. While building acoustics mainly deals with sound interaction between rooms inside of a building due to sound transmission through the common wall, room acoustics deals with the sound behavior inside of one enclosed room. Therefore, the sound absorption properties of the inner surfaces of a room are very important in room acoustics. Insufficient sound absorption will create a build-up of reverberant sound, or what most people refer to as "echo" inside of the room.

While substantial reverberation may be sought for some specific room purposes, such as for music in concert halls and churches, excessive reverberation may lead to undesirable acoustic conditions for other purposes, such as for speech in offices, educational spaces, and healthcare buildings. In these types of environment, it will lead to poor speech intelligibility, highly 'noisy' environments and overall-room occupant dissatisfaction, as seen in a previous study by Hodgson on the acoustical evaluation of multiple green buildings in Canada [20].

For the purpose of evaluating the room-acoustical properties in CLT buildings, three closely-related aspects of room acoustics which characterize the acoustical conditions in rooms were considered: sound absorption, reverberation time and speech intelligibility. These three metrics were used to determine the quality of the acoustical conditions in the rooms with untreated and exposed CLT surfaces, ranging from an office workspace to a gym facility, and determine the design goal in this study.

2.1 Sound Absorption

Whenever a sound wave strikes a flat surface or sound-transmitting partition, there are several interactions which can occur at the boundary between the partition and the medium in which the sound is travelling. These interactions result in three different phenomena: reflection, transmission, and absorption. The general interaction between a sound wave and a partition can be seen in **Figure 9**.

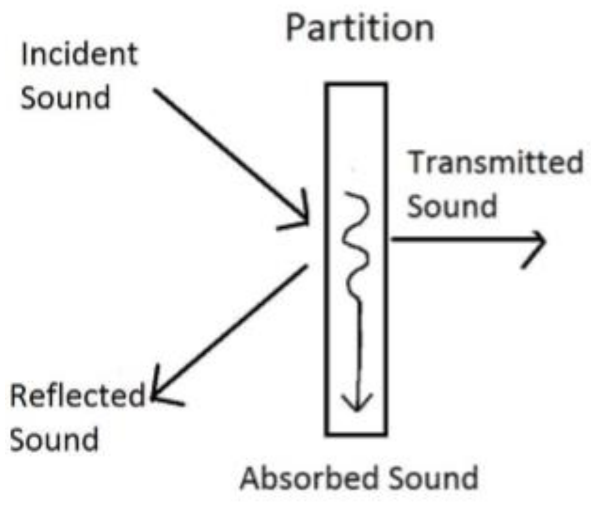


Figure 9. Interaction Between Sound Wave and Partition.

The acoustic properties of the partition will determine the proportion of energy which is reflected, absorbed, and transmitted. Depending on the design scenario of the partition, different amounts of sound transmission, sound absorption, and sound reflection may be desired. For example, in building acoustics, where we are concerned about the sound propagation between rooms, we typically want to minimize the sound transmission of the building partitions. On the other hand, in room acoustics, we are mainly interested in the sound inside of a specific room, especially with regards to the noise, speech intelligibility, and reverberation time. These three factors are closely related to the sound reflection and absorption properties of the internal building surfaces.

The sound-absorption property of a material is typically quantified by its sound-absorption coefficient, the proportion of incident sound energy that is absorbed by the material. It ranges from 0 (all incident sound energy is being reflected and there is no absorption) to 1 (all sound is absorbed and none is reflected). On some occasions, in which sound absorption is not necessarily restricted to one frontal surface of a sound absorber, such as in the case of hanging acoustic baffles [21] or thick porous materials [22], these values may exceed 1. This is mainly because sound absorption coefficient is normalized to the frontal area of a material.

2.2 Reverberation Time

To quantify how much reverberation there is in a certain room or environment, many acousticians typically use a room-acoustical parameter called the reverberation time. Reverberation time is defined as the time that it takes for sound to decay 60 dB in a room [23]. High reverberation time strongly correlates to insufficient sound absorption inside of the room such that sound will linger for a considerable amount of time due to reflections between the reflective surfaces of the room.

Reverberation can be typically measured by using a sound source to generate sound inside of a room. By turning off the sound source after a certain time, the decay rate of the sound can be measured, as illustrated in **Figure 10**.

As a measurable room-acoustic parameter, reverberation time is commonly used as the design parameter. Since the 1950s, multiple authors have made suggestions as to the optimal reverberation time based on the room volume and purpose. Long has compiled various suggestions into one graph as seen in **Figure 11** [23].

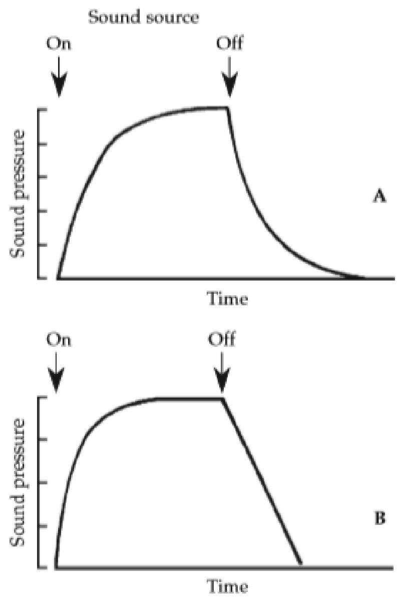


Figure 10. Reverberation Time Measurement Diagram. (A) in linear scale. (B) in logarithmic scale [24].

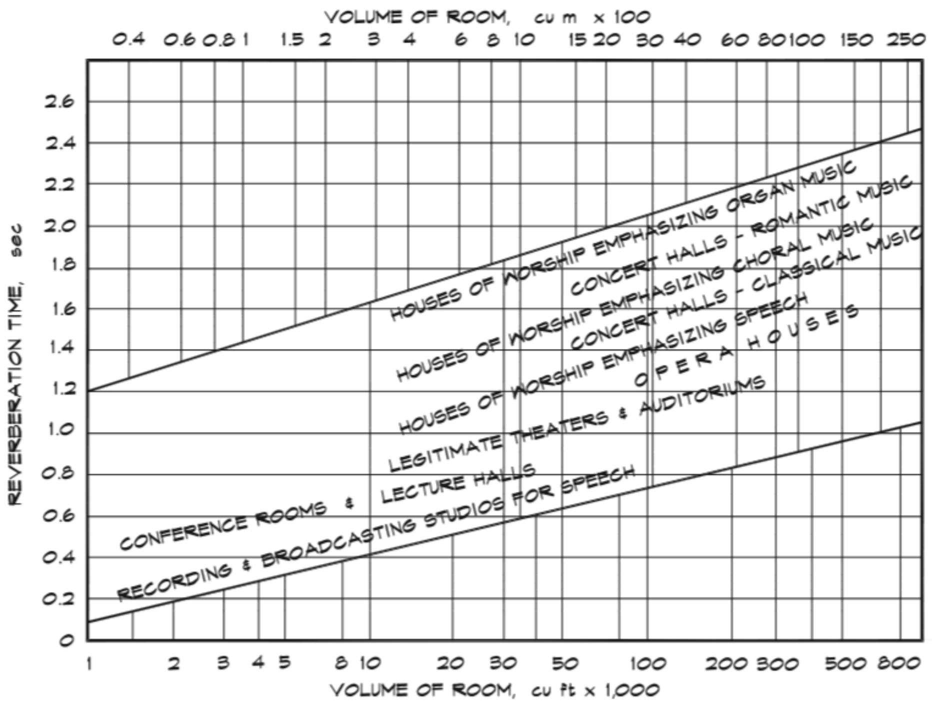


Figure 11. Recommended Reverberation Time vs Room Volume [23].

Everest also compiled some suggestions for the optimal reverberation time for various room configurations, as seen in **Figure 12** to **Figure** 14 [24]. Similar to the earlier recommended values, these values are proposed by experts based on their practical design experiences. However, it's still relatively unclear how these values are related to room-occupant satisfaction. Therefore, experts may not always agree on these recommended reverberation times.

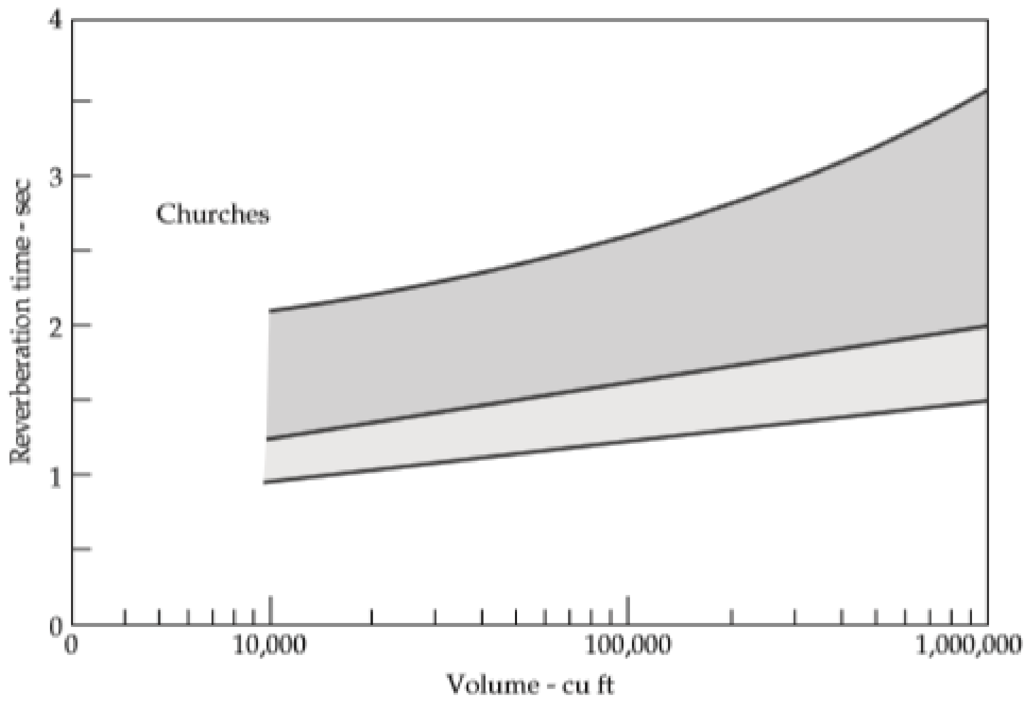


Figure 12. Recommended Reverberation Time for Churches [24].

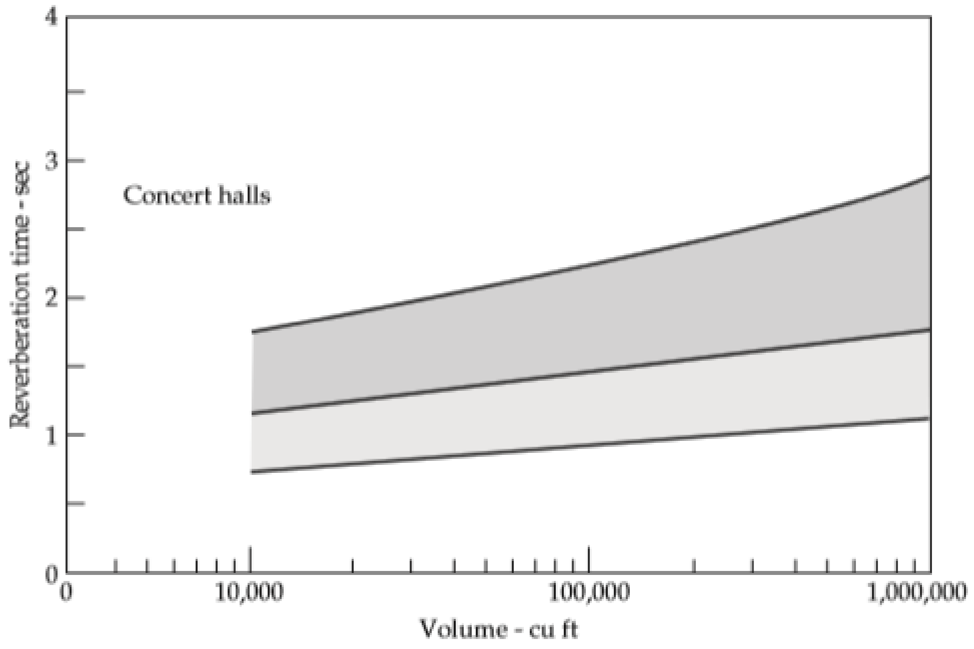


Figure 13. Recommended Reverberation Time for Concert Halls [24].

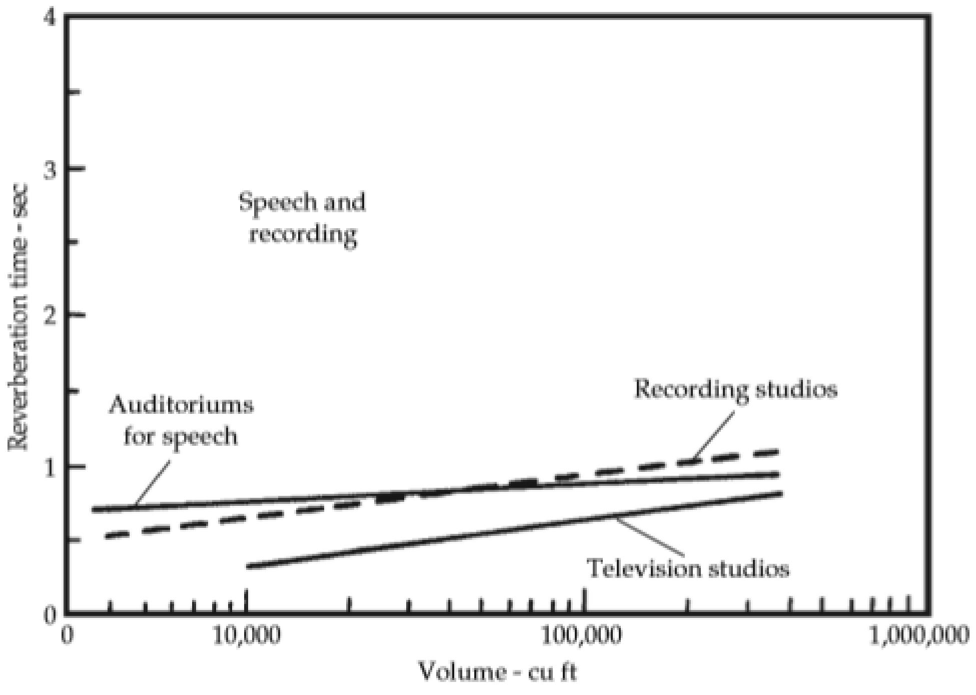


Figure 14. Recommended Reverberation Time for Speech and Music Recording [24].

2.3 Speech Intelligibility

In addition to the increase in noise levels, which may be unpleasant for room occupants, a lack of sound absorption and excessive reverberation may also hinder the verbal communication or speech intelligibility inside of a room. Houtgast, Steeneken, and Plomp proposed a Modulation Transfer Function to calculate Speech Intelligibility Index (SII) based on the following parameters: room volume, reverberation time, ambient noise level, talker vocal output, and talker listener distance [25].

According to Long, SII can be defined as," a direct measure of the fraction of words or sentences understood by a listener [23]." In ANSI S3.5 the speech-intelligibility quality of a room can be categorized into 'bad', 'poor', 'fair', 'good', and 'excellent' [26], according to the prevailing SII value (see **Table 1**)

Table 1. Speech Intelligibility Quality based on ANSI S3.5 Standard [26].

SII Range	Speech Intelligibility Quality
SII<0.3	'bad'
0.3<SII<0.45	'poor'
0.45<SII<0.6	'fair'
0.6<SII<0.75	'good'
SII>0.75	'excellent'

2.4 In-Situ Measurement in CLT Buildings

2.4.1 Experimental Methodology

To determine the baseline room-acoustical performance of untreated and exposed CLT panels in rooms, and the resulting room- acoustical quality, measurements were conducted in eight different room configurations in five different buildings in British Columbia: Churchill 4 plex apartment, Dr. Harrington Dental, Parkinson Recreation Centre, UBC Okanagan Hangar Fitness Centre, and Van Kam Freightway office. These buildings were selected based on one main requirement. The buildings in this study have one or more empty room which has 75% or more of its inner surfaces covered with untreated CLT panels.

During the measurement, the integrated impulse-response method was utilized to measure the reverberation time of the room based on the ISO 3382 [27] and ASTM E2235 [28] standards (refer to **Table 2**). In this method, a Norsonic dodecahedral sound source was selected to generate equal sound energy in all directions. The impulse response of the room was then captured using a free-field-half-inch microphone and preamplifier, which was connected to a Type 1 sound-level meter RION NA-28. This signal was acquired by the WinMLS system, which was also employed to produce the maximum length sequence signal for room excitation.

Assuming that the captured complex sound signal is reasonably continuous, Fourier Theorem stated that the signal can be regarded as a sum of single frequency sinusoidal components with certain amplitude and phase. By utilizing this theorem and applying Fast Fourier Transform (FFT) to the measured time data, the frequency response of the room was then calculated. This frequency result is then filtered into several octave bands, where the upper band frequency is twice of the lower band frequency.

Reverberation time measured in this study was based on the interpolation result of the decay curve for a range of 20 dB from 5 dB below stationary level (commonly referred as the T_{20}) at each octave band. This was chosen based on the achieved Signal-to-Noise Ratio (SNR) for the current equipment and setup which is typically above 30 dB for most of the measurements taken, which satisfy the conditions laid out in the two standards [27] [28]. For each room of interest, 6 measurement points were chosen with 16 averages at each point. Outliers were then identified by using the Modified Thomson-Tau method before taking the average reverberation time value for each room at each octave band frequency.

Given the measured reverberation times, the Diffuse Field Theory, proposed by Sabine in [29] ,was then used to find the average surface absorption per surface area, commonly referred to as the sound absorption coefficient. It can be derived using the following formula:

$$\overline{\overline{\alpha_d}} = \frac{V}{S}\left(\frac{0.16}{T_{60}} - 4m\right) \qquad (1)$$

in which, $\overline{\overline{\alpha_d}}$ = Average diffuse-field sound-absorption coefficient

V = Room volume (m^3)

S = Room surface area (m^2)

T_{60} = Reverberation time (s)

m = Ambient air absorption (Np/m)

By calculating the average diffuse-field absorption coefficient in octave bands from 125-8000 Hz, it is possible to estimate how much absorption the CLT surfaces have, as CLT is the major construction material used in the buildings which were measured in this study.

Table 2. Comparison between ISO 3382 and ASTM E2235.

Specifications	ISO 3382	ASTM E2235
Type of Method	Integrated Impulse Response and Interrupted Noise Method	Interrupted Noise Method
Sound Source	Omnidirectional Speaker	Omnidirectional Speaker
Sound Receiver	Type 1 Sound Level Meter (EC 651) with maximum microphone diameter of 13 mm	Omnidirectional Microphone with ±1 dB random-incidence amplitude response
Measurement Positions (Stationary Microphones)	Minimum distance between receiver and nearest surface is at least a quarter wavelength or approximately 1 m Minimum distance between source and receiver is: $d_{min} = \sqrt{\frac{V}{cT}}$	Minimum distance between receiver and nearest surface is at least 0.75 m
Number of Locations and Averages Required	Minimum of three measurements on each receiver position	Minimum of three mic. locations Product of the mic. positions, number of decays and number of sound source is at least 15
Other Requirements	No overloading allowed Signal to Noise Ratio (SNR) of at least 45 dB for T_{30} or 35 dB for T_{20}	Signal to Noise Ratio (SNR) must be at least 10 dB

Lastly, the reverberation-time data were also used to estimate the SII in the measured locations, based on the Modulation Transfer Method [25]. There were several assumptions which were used to calculate the SII under a best-case scenario. A low background noise level of NC-30 was used in this calculation, as per recommendation in the ASHRAE Handbook [30] for maximum allowable HVAC-related background sound in living areas, conference rooms, and office spaces. The talker's vocal output was determined based on the sound power level for casual and normal speech effort as outlined in the ANSI S3.5 [31] standard.

2.4.2 Results and Discussions

2.4.2.1 Reverberation Time

Reverberation times found in the measured CLT buildings have a relatively similar trend with most of them plateauing at around 1000 Hz as seen in **Table 3** as well as **Figure 15**. When this trend in reverberation time occurs, it is mostly due to panel absorption at low frequency and ambient air absorption at high frequency. In terms of the length of the reverberation time, 1 to 2 s reverberation time in the 1000 Hz octave is found for smaller room volumes (<400 m^3) and 2 to 2.4 s for larger room (>400 m^3).

Table 3. Reverberation Time Measurement Results

Location	Volume (m3)	Reverberation Time (s)						
		125Hz	250Hz	500Hz	1000Hz	2000Hz	4000Hz	8000Hz
Churchill 4 plex Apartment, Shower Room	14.75	0.50	0.67	0.82	0.98	0.98	0.74	0.65
Dr. Harrington Dental, Doctor's Office	32.30	0.81	1.01	1.16	1.51	1.45	1.07	0.86
Parkinson Recreation Centre, Activity Room	423.48	1.51	2.17	2.25	2.50	2.16	1.49	0.87
UBCO Hangar Fitness Centre, Storage Room	23.12	0.72	0.97	1.10	1.35	1.04	0.79	0.71
UBCO Hangar Fitness Centre, Studio 1	414.97	1.53	2.02	2.37	2.58	2.43	1.93	1.26
Van Kam Freightway, 1st Room	55.87	0.96	1.23	1.36	1.85	1.78	1.51	1.16
Van Kam Freightway, 2nd Room	28.38	0.94	1.15	1.45	1.98	1.77	1.38	1.11
Van Kam Freightway, 3rd Room	102.47	0.95	1.31	1.86	2.47	2.27	1.69	1.20

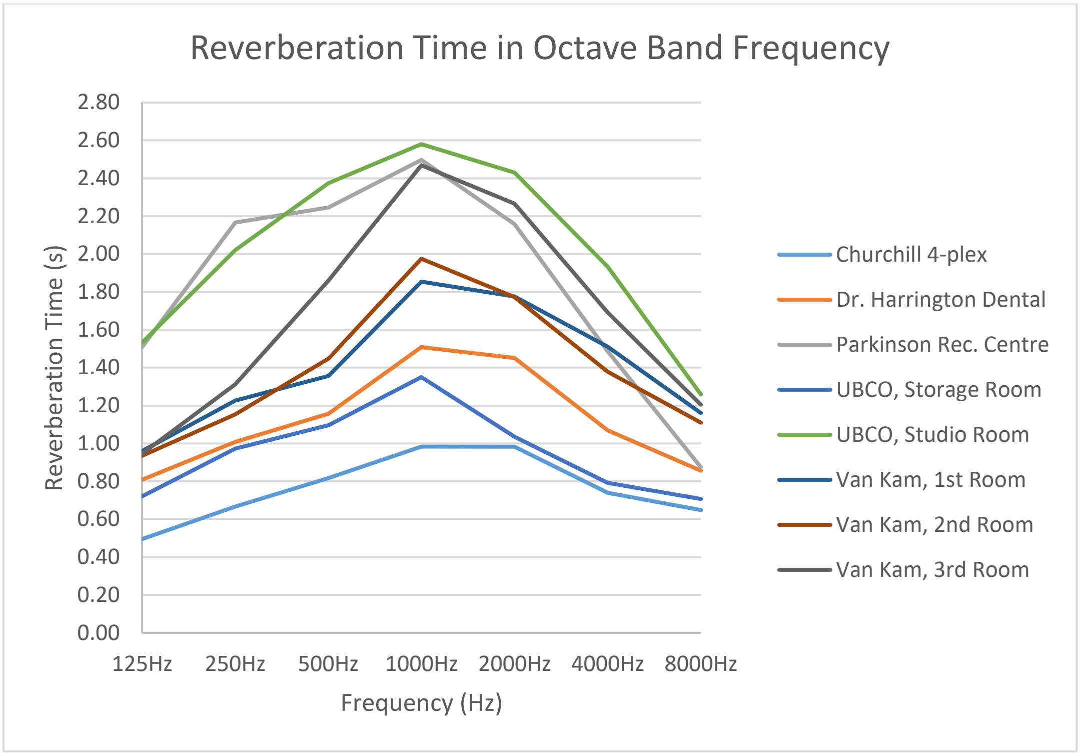


Figure 15. Measured Reverberation Times in Octave Bands in Rooms with Exposed CLT Panels.

2.4.2.2 Sound-Absorption Coefficient

Figure 15 shows that in rooms with internal surfaces, which mainly consist of untreated CLT panels, reverberation time is relatively high, especially at around 1000 Hz frequency. This result suggests that CLT has very low absorption. By calculating the average diffuse-field absorption coefficient in octave bands from 125-8000 Hz, we can estimate how much absorption the CLT surfaces have as CLT is the major construction material used in the buildings which were measured in this study. According to the results shown in **Table 4** and **Figure 16**, the sound-absorption coefficient typically starts from 0.13 at 125 Hz and then decreases to about 0.02 at high frequency for most of the rooms. This means that approximately 98.7% to 99.8% of sound is being reflected by the room surfaces across the frequency range of interest. There are also noticeable dips in sound-absorption performance at around 1000 Hz for most cases.

Table 4. Diffuse Field Sound Absorption Coefficient Results.

Location	Average Sound Absorption per Surface Area						
	125Hz	250Hz	500Hz	1000Hz	2000Hz	4000Hz	8000Hz
Churchill 4 plex Apartment, Shower Room	0.13	0.09	0.08	0.06	0.06	0.07	0.06
Dr. Harrington Dental, Doctor's Office	0.10	0.08	0.07	0.05	0.05	0.06	0.04
Parkinson Recreation Centre, Activity Room	0.10	0.07	0.07	0.06	0.06	0.07	0.05
UBCO Hangar Fitness Centre, Storage Room	0.10	0.07	0.06	0.05	0.07	0.08	0.06
UBCO Hangar Fitness Centre, Studio 1	0.12	0.09	0.08	0.07	0.07	0.06	0.03
Van Kam Freightway, 1st Room	0.10	0.08	0.07	0.05	0.05	0.05	0.02
Van Kam Freightway, 2nd Room	0.09	0.07	0.06	0.04	0.04	0.04	0.02
Van Kam Freightway, 3rd Room	0.12	0.08	0.06	0.04	0.04	0.05	0.02

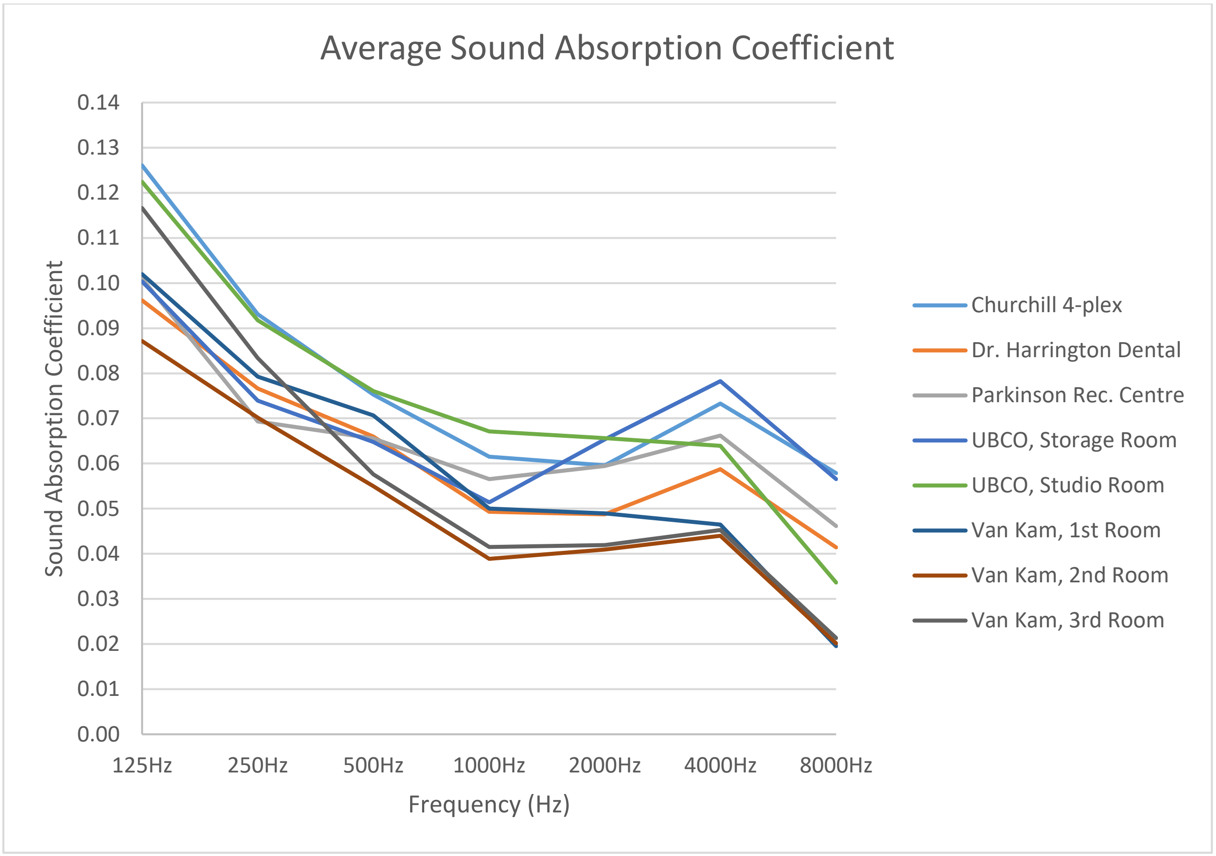


Figure 16. Average diffuse Field Sound Absorption Coefficient Measured in CLT Buildings.

2.4.2.3 Speech Intelligibility

Table 5 and **Figure 17** show the Speech Intelligibility Index (SII) in these buildings is mostly in the “fair” category for the 1 m and 2 m distances in smaller room (<300m^3) and at the 2 m distance for larger rooms (>300m^3) based on the measurement condition laid out in section 2.1. Only the 1 m talker-listener distance in larger rooms has speech intelligibility in the “good” category.

Increasing the speech vocal effort from casual to normal only increases SII by about 0.01 to 0.03. This is to be expected since the background noise that is used in this calculation is relatively low (NC-30) such that the effect of the different speech efforts is almost negligible. With the increase in background noise level, worse speech intelligibility can be expected for the same speech effort.

Table 5. Speech Intelligibility Measurement Result (NC-30 Background Noise).

Location	Casual		Normal	
	1m talker-listener distance	2m talker-listener distance	1m talker-listener distance	2m talker-listener distance
	SII	SII	SII	SII
Dr. Harrington Dental Office	0.57	0.54	0.57	0.54
Parkinson Recreation Centre, Activity Room	0.69	0.56	0.69	0.54
UBCO Hangar Fitness Centre, Studio 1	0.66	0.51	0.67	0.52
Van Kam Freightway, 1st Room	0.55	0.49	0.55	0.50
Van Kam Freightway, 2nd Room	0.47	0.44	0.47	0.44
Van Kam Freightway, 3rd Room	0.54	0.46	0.55	0.47

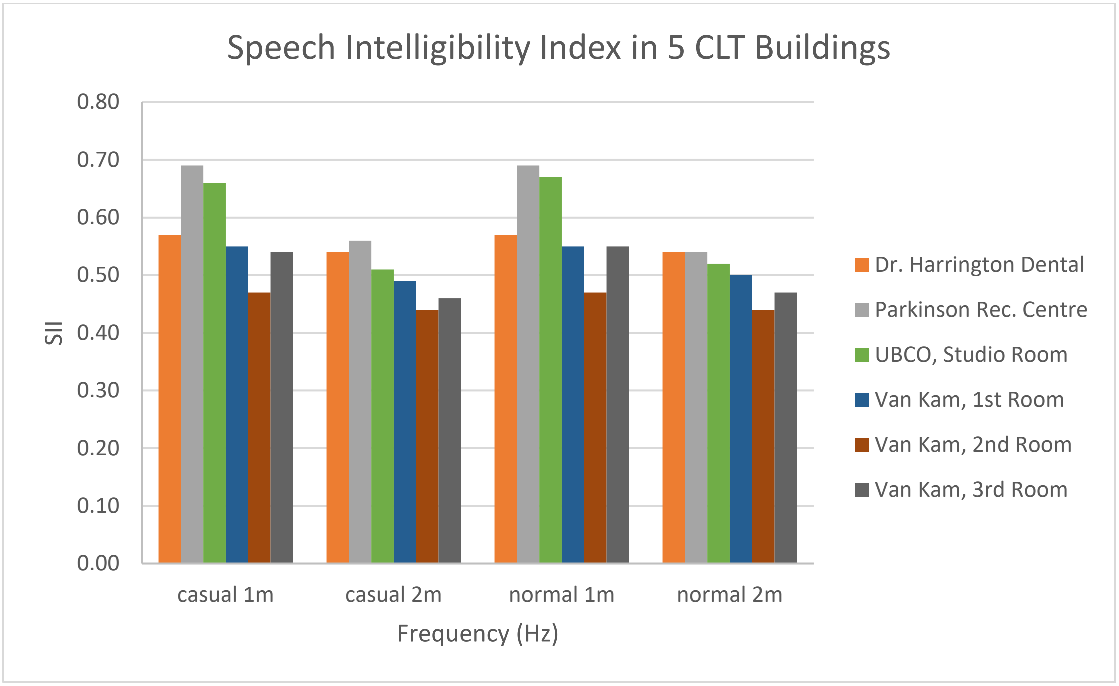


Figure 17. Speech Intelligibility Index (SII) in Five CLT Buildings.

Based on the measurement results gathered. It can be concluded that the sound absorption of Cross-Laminated Timber (CLT) panels is very low, resulting in high noise and reverberation as well as low speech intelligibility. It is clearly of interest to investigate ways to make CLT panels more sound-absorptive.

3 Theory

3.1 Introduction to Helmholtz Resonators

Depending on the sound-absorption mechanism and the frequency of interest, there are three main types of sound absorber: porous, panel, and resonant sound absorbers. Porous sound absorbers are typically used for their high-frequency sound absorption, panel sound absorbers for their low-frequency absorption and, lastly, resonant absorbers for their mid-high frequency sound absorption. In this study, we are mainly interested in a specific type of resonant absorber called a Helmholtz Resonator (HR).

A Helmholtz Resonator, named after a famous physician and physicist Hermann von Helmholtz, can generally be described as a resonator consisting of an enclosed air cavity connected to the outside sound field through a small opening or neck, as seen in **Figure 18**. This small air volume in the neck of the resonator behaves similarly to the mass in a typical mass-spring-damper system, while the enclosed air in the cavity behaves as the spring. The sound absorption in a Helmholtz Resonator is mainly due to the viscous damping in the neck area, in which a small air volume oscillates back and forth due to sound-wave excitation. Maximum sound absorption occurs at the natural frequency of the system, while its quality factor depends on its damping factor.

One of the classical examples of a Helmholtz Resonator is a beer bottle. When we blow air over the opening of a beer bottle, we produce a sharp tone which corresponds to its resonant frequency. This same resonance phenomenon is responsible for the sound absorption in a Helmholtz Resonator.

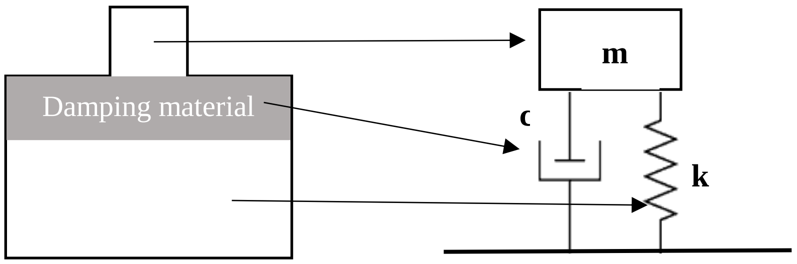


Figure 18. Helmholtz Resonator and its Mechanical System Analogy.

If we take this beer bottle and excite it with a sound wave propagating towards its opening area, assuming that it behaves as a resonant absorber, it will theoretically absorb some sound energy, especially that which has a frequency near the resonant frequency. This phenomenon can be demonstrated by way of an impedance tube measurement as described in ISO/CD 10534-2 standard [7].

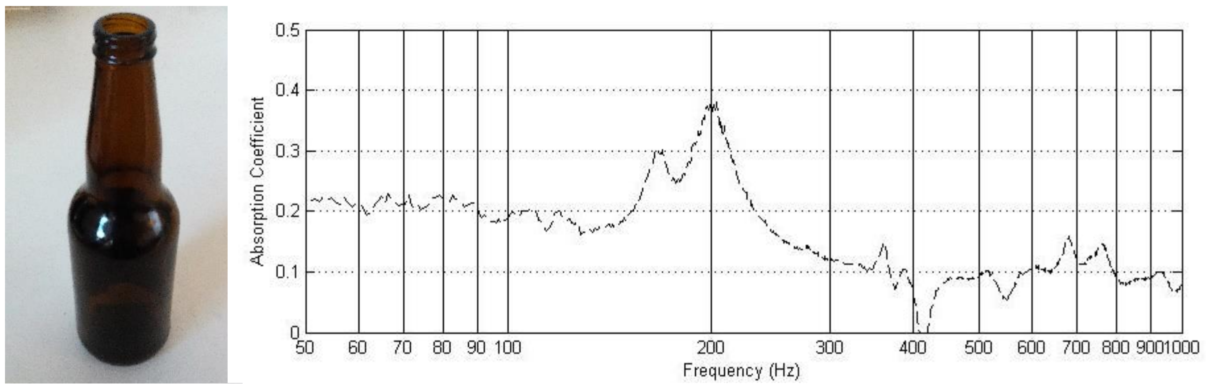


Figure 19. Resonance Frequency of a 441ml Beer Bottle and Its Sound Absorption Measured in Impedance Tube.

As seen in **Figure 19**, a 441 ml beer bottle with cavity height and diameter of about 120 and 60 mm, respectively, neck length of 90 mm, and neck radius of 12.5 mm has a normal-incident sound-absorption coefficient of slightly below 0.38 at 200 Hz (its presumed resonant frequency). This result is very interesting considering that there is no additional damping material inside of the beer bottle.

3.2 Classical Theory of Helmholtz Resonators

3.2.1 Natural Frequency and Impedance of an Ideal Helmholtz Resonator

The natural frequency of a Helmholtz Resonator can be calculated if we know the governing equation of the resonator. Taking a control volume in the neck region of the resonator, as seen in **Figure 20**, we can derive the unbalanced force equations, as seen in **Eqs. 2** and **3**:

$$P_{out}S - P_{in}S = \rho l S \frac{du}{dt} \tag{2}$$

$$P_{out} = P_{in} + \frac{\rho l}{S}\frac{dU}{dt} \tag{3}$$

in which,	Pin	= pressure inside cavity (Pa)
	Pout	= pressure outside (Pa)
	S	= surface area (m2)
	ρ	= mass density (kg/m3)
	l	= length of the neck (m)
	u	= particle velocity (m/s)
	U	= volume velocity ($m^3/2$).

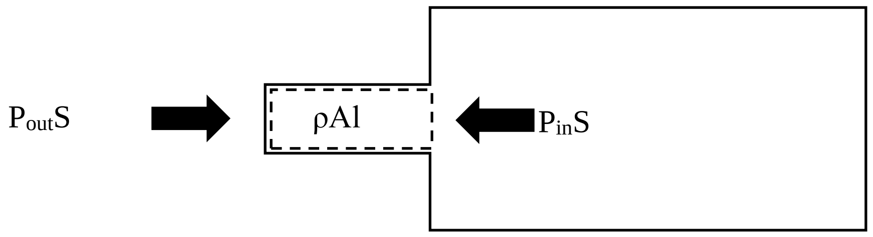


Figure 20. Unbalanced Forces Acting on the Control Volume at the Neck of Helmholtz Resonator.

In these two equations, the difference in forces due to the pressures just outside of the Helmholtz Resonator (P$_{out}$) and inside the cavity (P$_{in}$) induces particle acceleration $\left(\frac{du}{dt}\right)$ or volumetric flow acceleration $\left(\frac{dU}{dt}\right)$ of the mass of air inside the control volume, which is the product of the air density (ρ), length of the neck (l) and area of the neck (S).

The two terms on the right-hand side of **Eq. 3** can be correlated by looking at the relations between pressure, change in volume inside of the cavity, and the bulk modulus of the medium inside the cavity (K) (**Eqs. 4** and **5)**:

$$P_{in} = \frac{K}{V}\delta V \tag{4}$$

$$P_{in} = \frac{K}{V}\int Udt. \tag{5}$$

Plugging the value of P_{in} in **Eq. 5 in**to **Eq. 3**, and using the definition of speed of sound in the Newton-Laplace equation, where speed of sound (c) equals $\left(\sqrt{\frac{K}{\rho}}\right)$, we obtain a familiar 2nd order differential equation relating the input pressure (P_{out}) to the volume of the air column at the neck of the Helmholtz Resonator (v):

$$P_{out} = \frac{\rho c^2}{V}\int Udt + \frac{\rho l}{S}\frac{dU}{dt} \tag{6}$$

$$P_{out} = \frac{\rho c^2}{V}v(t) + \frac{\rho l}{S}\frac{d^2 v}{dt^2}. \tag{7}$$

The coefficient of the second term in **Eq. 7** $\left(\frac{\rho l}{S}\right)$ is commonly referred to as the acoustic inertance of the system, while the inverse of the coefficient of the first term $\left(\frac{\rho c^2}{V}\right)$ is referred to as the acoustic compliance. Comparing this to a mechanical mass-spring-damper system, the acoustic inertance can be regarded as a mass, while the inverse of acoustic compliance can be regarded as the spring stiffness. Using this relation, we can calculate the resonance frequency of the system (f_n, or ω_n for angular frequency) as follows:

$$\omega_n = \sqrt{\frac{1}{m_A C_A}} \tag{8}$$

$$\omega_n = c\sqrt{\frac{S}{Vl}} \tag{9}$$

Converting **Eq. 7** to the frequency domain, we can derive the acoustical impedance (Z) and specific impedance $(z = ZS)$ at the opening of an ideal Helmholtz Resonator; these correlate input pressure to volumetric flow rate or particle velocity, respectively. It should be noted that these two impedances have different dimensions compared to mechanical impedance $(Z_m = zS)$ which correlates input force to particle velocity:

$$P_{out} = \frac{\rho c^2}{Vj\omega}U + \frac{\rho l}{S}j\omega U \tag{10}$$

$$Z = \frac{\rho c^2}{Vj\omega} + \frac{\rho l}{S}j\omega \tag{11}$$

$$z = \frac{\rho c^2 S}{Vj\omega} + \rho l j\omega. \tag{12}$$

3.2.2 Radiation Impedance for Helmholtz Resonator with Circular Opening

It should be noted that **Eqs. 11** and **12** are valid for a very ideal case in which the air particles inside of the control volume, as seen in **Figure 20,** oscillate uniformly and there is no resistance or losses in the neck region. In fact, this control volume of air doesn't stay uniform throughout the oscillation process, as there will be radiation at the opening of the neck of the Helmholtz Resonator. Furthermore, viscous damping, caused by this radiation process, as well as friction from the walls, will also occur in the neck region. To account for these two phenomena, we need to introduce both real (radiation resistance) and imaginary terms (radiation reactance) in **Eqs. 11** and **12**.

Kinsler et al. suggested that, for a Helmholtz Resonator with a circular opening such that the wavelength of the sound wave is very large compared to the radius of the opening (a), the opening could be regarded as a baffled piston [32]. Mechanical impedance and acoustic impedance for flanged and unflanged openings can then be calculated as follows:

For Helmholtz Resonators with circular flanged openings:

$$Z_m = \frac{1}{2}\rho c S(ka)^2 + j\rho c S\left(\frac{8}{3\pi}\right)ka \tag{13}$$

$$Z = \frac{1}{2}\frac{\rho c}{S}(ka)^2 + j\left(\frac{\rho c}{S}\right)\left(\frac{8}{3\pi}\right)ka. \tag{14}$$

For Helmholtz Resonators with circular unflanged openings:

$$Z_m = \frac{1}{4}\rho c S(ka)^2 + j\rho c S(0.6)ka \tag{15}$$

$$Z = \frac{1}{4}\frac{\rho c}{S}(ka)^2 + j\left(\frac{\rho c}{S}\right)(0.6)ka. \tag{16}$$

Assuming that the neck of a Helmholtz Resonator behaves similarly to a circular tube with two open ends (with or without flanges on the side connecting to the outside and with flanges on the side connecting the tube to the cavity) and that there are negligible losses due to wall friction, **Equation 11** can be modified to include the radiation-impedance terms, as follows:

$$Z = \frac{\rho c^2}{Vj\omega} + \frac{\rho l}{S}j\omega + \frac{\rho(0.85 + 0.85)a}{S}j\omega + \frac{1}{2}\frac{\rho}{Sc}(\omega a)^2 \quad (flanged) \tag{17}$$

$$Z = \frac{\rho c^2}{Vj\omega} + \frac{\rho l}{S}j\omega + \frac{\rho(0.6 + 0.85)a}{S}j\omega + \frac{1}{4}\frac{\rho}{Sc}(\omega a)^2 \quad (unflanged) \tag{18}$$

The third terms on the right-hand sides of **Eqs. 17** and **18** consist of end-correction factors (δ), such that the effective length of the neck of a Helmholtz Resonator (l') equals $l + \delta a$ or $l + 1.7a$ for flanged resonators and $l + 1.45a$ for unflanged resonators. The fourth terms of the right-hand sides of the equations in **Eqs. 17** and **18** come from the radiation resistance (R_r) derived for an open-ended circular pipe.

Typically, there is also a loss of energy due to viscous loss and heat-conduction losses in the wall. While this thermoviscous resistance (R_w) is relatively small compared to the resistance provided by a damping material, this needs to also be included in the impedance calculation, especially if damping material is absent. According to Kinsler et.al. [32], the thermoviscous resistance for a Helmholtz Resonator with cylindrical neck can be calculated as follows:

$$R_w = \frac{2\rho_0 l' c \alpha_w}{S} \tag{19}$$

$$\alpha_w = \alpha_{w\eta} + \alpha_{w\kappa} = \frac{1}{ac}\left(\frac{\eta\omega}{2\rho_0}\right)^{1/2}\left(1 + \frac{\gamma - 1}{\sqrt{Pr}}\right) \tag{20}$$

As seen in **Eq. 20**, sound absorption due to viscous loss ($\alpha_{w\eta}$) and thermal conduction ($\alpha_{w\kappa}$) are functions of neck radius (a), speed of sound (c), medium density (ρ_0), coefficient of shear viscosity (η), frequency (ω), heat capacity ratio (γ), and Prandtl number (Pr). It should be noted that the viscous-loss term in **Eq. 21** is very similar to the more commonly used resistance term, which was derived by Rayleigh [33]:

$$R_w = \frac{\sqrt{2\rho\mu\omega}}{2\varepsilon S} \tag{21}$$

Ingard has previously noticed that this resistance is considerably low compared to those measured from experiments. He suggested that a better approximation can be achieved by using double the value [34]. During the experiments (**Section 4.4**), the author has found that the resistance value for HR array is approximately triple the value in **Eq. 21** for low- damping system and quadruple the value for high-damping system.

An alternative viscous resistance term can be calculated using the formula derived by Guess [35], as follows:

$$R_w = \frac{\rho}{\varepsilon S}\sqrt{8\upsilon\omega}\left(1 + \frac{l}{2a}\right) \tag{22}$$

in which, υ = Kinematic viscosity of air (10^{-5} kg/m s)

μ = Dynamic viscosity of air (10^{-5} m^2/s)

ε = Ratio between area of opening to area of cavity.

3.2.3 Higher Mode Consideration for Cavities with Various Shapes

The simplest form of radiation impedance and the resulting end-correction factors, shown in **Eq. 17** and **18,** are most likely sufficient for predicting the performance of an HR Resonator with an opening which is very small compared to its cavity's dimensions. However, for less extreme cases, sometimes it is necessary to look at the contribution of higher modes of sound-wave propagation inside of the HR cavity. Ingard [34] derived the end-correction formula for multiple HR configurations as seen in the following equations:

For a circular opening with a circular cavity:

$$\delta a = \sqrt{A_0}\frac{4}{\sqrt{\pi}}\frac{1}{\xi}\sum_{m=0}^{\infty}\sum_{n=1}^{\infty}\frac{\mathrm{J}_1{}^2(q_{mn}\xi)\mathrm{J}_m{}^2\left(\frac{q_{mn}a}{R}\right)}{q_{mn}{}^3\left(1-\frac{m^2}{q_{mn}{}^2}\right)\mathrm{J}_m{}^2(q_m)} \tag{23}$$

in which, A_0 = Area of opening (m^2)

ξ = Ratio between radius of opening and radius of cavity

J = Bessel function of first kind

a = Spacing between center of opening and cavity (m)

m, n = Wave propagation modes.

For a circular opening with a rectangular cavity:

$$\delta a = \sqrt{A_0}\frac{4}{\pi^{3/2}}(\xi\eta)^{-1/2}\sum_{m=0}^{\infty}\sum_{n=1}^{\infty} v_{mn}\frac{\mathrm{J}_1{}^2\left(\sqrt{\xi^2 m^2 + \eta^2 n^2}\right)}{\left(\frac{m^2 b}{a} + \frac{n^2 a}{b}\right)^{3/2}} \tag{24}$$

in which, ξ = Ratio between radius of opening and length of cavity

η = Ratio between radius of opening and width of cavity.

For a rectangular opening with a rectangular cavity:

$$\delta a = \sqrt{A_0}\frac{2}{\pi}\sum_{m}^{\infty}\sum_{n}^{\infty} v_{mn}\frac{G_{mn}}{\left(\frac{m^2 b_1}{a} + \frac{n^2 a_1}{b}\right)^{1/2}} \tag{25}$$

in which, G_{mn} = $\left[\frac{sin(\pi m\xi)}{\pi m\xi}\frac{sin(\pi n\eta)}{\pi n\eta}\right]^2 \xi\eta$

ξ = Ratio between length of opening and length of cavity

η = Ratio between width of opening and width of cavity

v_{mn} = 0 if m=0 and n=0, ½ if m or n=0, 1 otherwise.

For a porosity or fraction of open area (ε) of less than 0.16, Ingard's correction factor in **Eqs. 24** and **25** can be approximated as follows [34]:

$$\delta = 0.8(1 - 1.14\sqrt{\varepsilon}) \tag{26}$$

$$\delta = 0.85(1 - 1.25\sqrt{\varepsilon}) \tag{27}$$

An alternative approximation for a circular aperture was derived by Rschevkin and reported in [36], which includes a broader range of porosities (including ε=1):

$$\delta = 0.8(1 - 1.47\sqrt{\varepsilon} + 0.47\varepsilon^{3/2}) \tag{28}$$

3.2.4 Pressure-Reflection Factor and Sound-Absorption Coefficient

If we assume that the resonator behaves as a termination with specific impedance z_{HR} at the opening of the Helmholtz Resonator (x=0), we can determine the sound-absorption coefficient of the resonator by looking at the pressure and particle velocity, which satisfy boundary conditions at x=0.

At x=0, due to pressure continuity, we find the relations between the magnitude of the incident, reflected and transmitted sound waves:

$$P_i + P_r = P_t \tag{29}$$

$$\boldsymbol{A}e^{-jkx} + \boldsymbol{B}e^{jkx} = \boldsymbol{C}e^{-jkx} \tag{30}$$

$$\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{C} \tag{31}$$

in which, P_i = Incident wave, P=$\boldsymbol{A}e^{-jkx}$

P_r = Reflected wave, P=$\boldsymbol{B}e^{jkx}$

P_t = Transmitted wave, P=$\boldsymbol{C}e^{-jkx}$

x = Distance to the top surface of the neck opening.

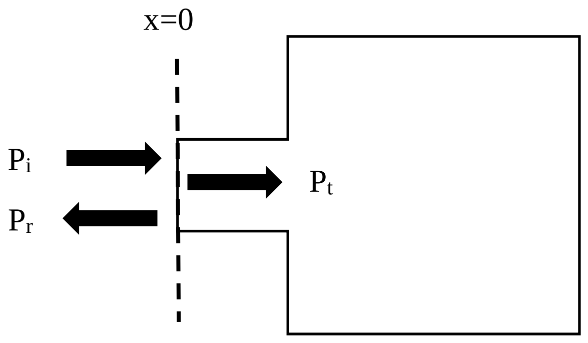


Figure 21. Pressure and Velocity at Boundary x=0.

Similarly, the particle velocity along the x-direction also needs to be continuous at the boundary. Using the relation between pressure, velocity and impedance, we obtain the following equations:

$$u|_{x=0^-} = u|_{x=0^+} \tag{32}$$

$$\frac{\boldsymbol{A}e^{-jkx} - \boldsymbol{B}e^{jkx}}{z_0} = \frac{\boldsymbol{C}e^{-jkx}}{z_{HR}} \tag{33}$$

$$\frac{\boldsymbol{A} - \boldsymbol{B}}{z_0} = \frac{\boldsymbol{C}}{z_{HR}} \tag{34}$$

If we use both **Eqs. 31** and **34** to eliminate C, we can get the formula for the pressure-reflection coefficient R, which is simply the ratio between the pressure magnitude of the reflected wave and the pressure magnitude of the incident wave:

$$R = \frac{\boldsymbol{B}}{\boldsymbol{A}} = \frac{z_{HR} - z_0}{z_{HR} + z_0} \tag{35}$$

Since sound energy is related to the square of sound pressure, the sound-absorption coefficient can be simply derived as follows:

$$\alpha = 1 - |R|^2. \tag{36}$$

3.2.5 Arrays of Helmholtz Resonators

In an actual room acoustical application, it is often necessary to use arrays of Helmholtz Resonators, as seen in **Figure 22,** to increase the sound absorption of the building surfaces. For a simple HR array with identical resonator units and connected cavity, we can assume that the opening of each Helmholtz Resonator is connected by means of a perforated panel. Introducing porosity (ε) and cavity depth (d) to **Eq. 10**, we obtain the resonant frequency and surface impedance of an HR array:

$$\omega_n = c\sqrt{\frac{\varepsilon}{dl'}} \tag{37}$$

$$Z = \frac{\rho(l + \delta a)}{S} j\omega + R_w + R_r - j\rho c S(\cot(k * d)) \tag{38}$$

The last term in **Eq. 38** is simply the acoustic impedance of a plane wave, travelling in an air column with a rigid backing. For more general configurations of HR array, it is often easier to look at the HR array as a multi-layered system and use the transfer-matrix method to predict the performance of the overall system.

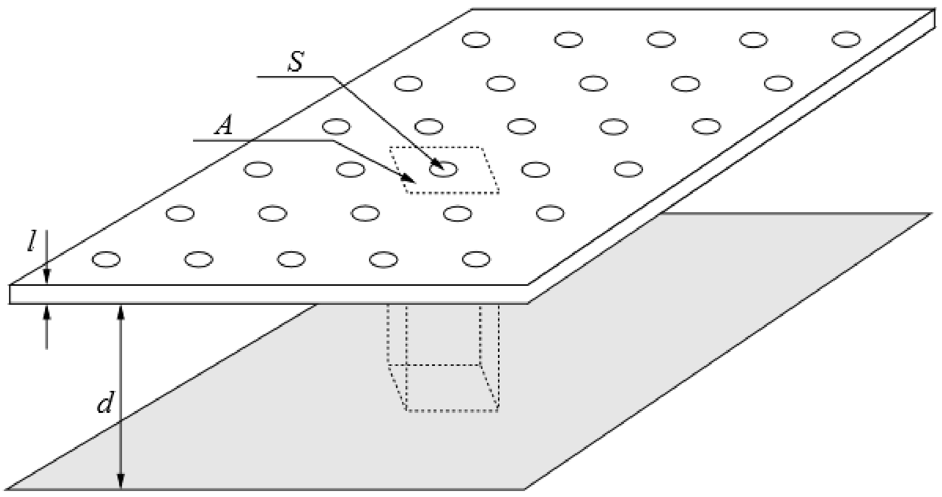


Figure 22. Helmholtz Resonator Array with Connected Cavity [37].

3.3 Transfer-Matrix Method

While the classical formula is often sufficient to predict the performance of a simple HR array, as shown in the previous section, it is considerably limited in application, especially for non-ideal HR arrays, such as the ones with non-rigid backing, porous absorbers near its backing, multiple arrays stacked in series, etc. To cover more general configurations of HR arrays, it is often beneficial to divide the system into multiple layers and employ the transfer- matrix model to relate the pressure and velocity of each layer to the others, as shown in **Figure 23**.

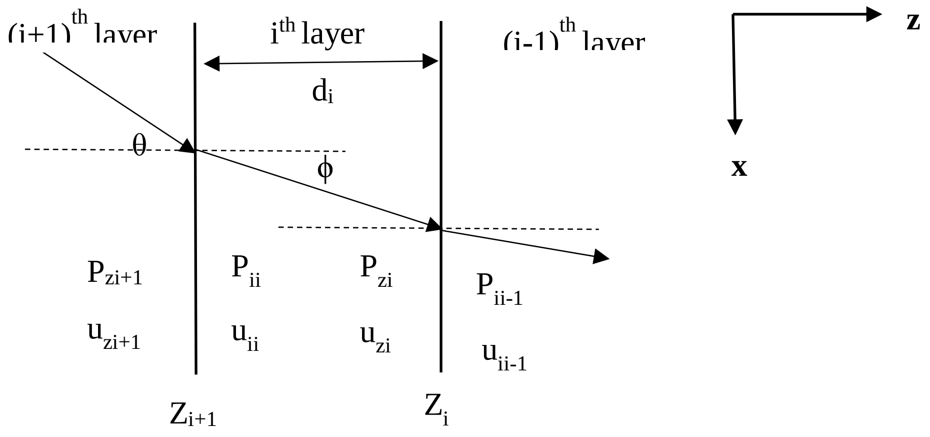


Figure 23. Transfer Matrix Model Geometry for Multi-Layered System.

The transfer matrix of layer ith (T) as seen in **Figure 23** is a matrix relating **V**(ii), a column vector which describes the acoustic field near the boundary Z_{i+1}, to **V**(zi) such that:

$$V_{ii} = [T]\, V_{zi} \tag{39}$$

Allard and Atalla [2] formulated the transfer-matrix model to predict the sound absorption, transmission, and reflection of a multi-layered homogenous and isotropic material consisting of a fluid layer, isotropic solid layer, elastic porous layer, and a thin plate layer. In this study, this transfer-matrix model is further expanded to include a perforated panel (with macro and micro-perforations) and an orthotropic material, such as Cross-Laminated Timber (CLT).

3.3.1 Fluid Media

Assuming that the **V** matrix in a fluid medium consists of pressure (p) and particle velocity (u), the transfer matrix of the ith layer can be derived by looking at the pressure and particle velocity near the first boundary, between the ith and i+1th layers and the second boundary, between the ith and i-1th layers. Assuming a plane-wave propagation mode with wave number in the z-direction (k_z) equal to $\sqrt{k^2 - k^2 sin\theta}$, pressure and particle velocity can be calculated as follows [2]:

$$p(z) = A_1 e^{-jk_z z} + A_2 e^{jk_z z} \tag{40}$$

$$u(z) = \frac{k_z}{\omega\rho}\left(A_1 e^{-jk_z z} - A_2 e^{jk_z z}\right) \tag{41}$$

If we set $z = 0$ at the second boundary and $z = -d_i$ at the first boundary, and use **Eqs. 40** and **41** to get the acoustic fields at the two point of interest indicated in **Figure 23**, we obtain the following equations:

$$p_{ii}(z) = A_1 e^{jk_z d_i} + A_2 e^{-jk_z d_i} \tag{42}$$

$$u_{ii}(z) = \frac{k_z}{\omega\rho}\left(A_1 e^{jk_z d_i} - A_2 e^{-jk_z d_i}\right) \tag{43}$$

$$p_{zi}(z) = A_1 + A_2 \tag{44}$$

$$u_{zi}(z) = \frac{k_z}{\omega\rho}(A_1 - A_2) \tag{45}$$

Combining the four equations above, the transfer matrix for a fluid layer is simply:

$$[T] = \begin{bmatrix} \cos(k_z d_i) & \mathrm{j}\frac{\omega\rho}{k_z}\sin(k_z d_i) \\ \mathrm{j}\frac{k_z}{\omega\rho}\sin(k_z d_i) & \cos(k_z d_i) \end{bmatrix} \tag{46}$$

3.3.2 Isotropic Solid Layer

Based on the previous work done on analyzing muli-layered solid media by Folds and Loggins [38], wave propagation in an elastic solid consists of four different waves: incident and reflected longitudinal waves, and incident and reflected shear waves. Introducing the displacement potential term φ for the longitudinal wave and ψ for the shear wave, Allard [2] calculated the velocity (v) and stresses in tangential and normal directions as follows:

$$u_x = j\omega\left(\frac{\partial\varphi}{\partial x} - \frac{\partial\psi}{\partial z}\right) \tag{47}$$

$$u_z = j\omega\left(\frac{\partial\varphi}{\partial z} + \frac{\partial\psi}{\partial x}\right) \tag{48}$$

$$\sigma_{zz} = \lambda\left(\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2}\right) + 2\mu\left(\frac{\partial^2\varphi}{\partial z^2} + \frac{\partial^2\psi}{\partial x\partial z}\right) \tag{49}$$

$$\sigma_{xz} = \mu\left(\frac{\partial^2\varphi}{\partial x\partial z} + \frac{\partial^2\psi}{\partial x^2} - \frac{\partial^2\psi}{\partial z^2}\right) \tag{50}$$

As seen in **Eqs. 49** and **50**, tensile and shear stresses in the material are related to the Lamé coefficients λ and μ, which are defined as follows:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{51}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{52}$$

in which, E = Young's Modulus (Pa)

ν = Poisson's ratio.

The two displacement potentials in **Eqs. 49** and **50** are written as follows:

$$\varphi = e^{j\omega t - jk_x x}\left[A_1 e^{-jk_{xz}z} + A_2 e^{jk_{xz}z}\right] \tag{53}$$

$$\psi = e^{j\omega t - jk_x x}\left[A_3 e^{-jk_{zz}z} + A_4 e^{jk_{zz}z}\right] \tag{54}$$

The two wave numbers, k_{xz} and k_{zz}, are calculated as follows:

$$k_{xz} = \sqrt{\left(\frac{\omega^2\rho}{\lambda + 2\mu}\right) = k^2 sin^2\theta} \tag{55}$$

$$k_{zz} = \sqrt{\left(\frac{\omega^2\rho}{\mu}\right) = k^2 sin^2\theta.} \tag{56}$$

Using the four values calculated in **Eqs. 47-50** as the components of the acoustic-field matrix (**V**), and introducing a new vector $\mathbf{A} = [A_1 + A_2,\ A_1 - A_2,\ A_3 + A_4,\ A_3 - A_4]^T$, an intermediary matrix (Υ) is calculated such that $\mathbf{V} = [\Upsilon(\mathrm{z})]\mathbf{A}$. At the two boundaries seen in **Figure 23**, we obtain $\mathbf{V}_{ii} = [\Upsilon(-\mathrm{h})]\mathbf{A}$ and $\mathbf{V}_{zi} = [\Upsilon(0)]\mathbf{A}$ if we set the origin point at the second boundary. The transfer matrix for an isotropic solid layer is then calculated as follows:

$$\boldsymbol{T} = [\Upsilon(-\mathrm{h})][\Upsilon(0)]^{-1} \tag{57}$$

3.3.3 Orthotropic Solid Layer

Unlike isotropic solids, orthotropic solids have three perpendicular planes of elastic symmetry. By using Hooke's law, the stiffness matrix of an orthotropic material relating the stresses (σ) and strain (ε) in the material has 12 non-zero terms such that:

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{yz} \\ \sigma_{xz} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{21} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{31} & C_{32} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ 2\varepsilon_{yz} \\ 2\varepsilon_{xz} \\ 2\varepsilon_{xy} \end{bmatrix} \tag{58}$$

These non-zero terms are calculated as follows:

$$C_{11} = \frac{(1 - \nu_{yz}\nu_{zy})}{E_y E_z \Delta}, C_{22} = \frac{(1 - \nu_{xz}\nu_{zx})}{E_x E_z \Delta}, C_{33} = \frac{(1 - \nu_{xy}\nu_{yx})}{E_x E_y \Delta} \tag{59}$$

$$C_{12} = \frac{\nu_{yx} + \nu_{zx}\nu_{yz}}{E_y E_z \Delta}, C_{13} = \frac{\nu_{zx} + \nu_{yx}\nu_{yz}}{E_y E_z \Delta}, C_{21} = \frac{\nu_{xy} + \nu_{xz}\nu_{zy}}{E_x E_z \Delta}$$

$$C_{23} = \frac{\nu_{zy} + \nu_{zx}\nu_{xy}}{E_x E_z \Delta}, C_{31} = \frac{\nu_{xz} + \nu_{xy}\nu_{yz}}{E_y E_x \Delta}, C_{32} = \frac{\nu_{yz} + \nu_{xz}\nu_{yx}}{E_y E_x \Delta}$$

$$C_{44} = G_{yz}, C_{55} = G_{xz}, C_{66} = G_{xy}$$

$$\Delta = \frac{1 - \nu_{xy}\nu_{yx} - \nu_{yz}\nu_{zy} - \nu_{zx}\nu_{xz} - 2\nu_{xy}\nu_{yz}\nu_{zx}}{E_x E_y E_z} \tag{60}$$

This Hooke's law can be written in a generalized form as:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{61}$$

$$\varepsilon_{kl} = \frac{1}{2}\left(\frac{\partial s_k}{\partial x_l} + \frac{\partial s_l}{\partial x_k}\right) \tag{62}$$

$$\frac{\partial \sigma_{ij}}{\partial x_j} = \rho \frac{\partial^2 s_i}{\partial t^2} \tag{63}$$

in which, s_l = displacement vector in the i-direction

x_l = position vector in the i-direction.

Combining **Eqs. 61-63**, we obtain the wave equation:

$$\rho \frac{\partial^2 s_i}{\partial t^2} - C_{ijkl} \frac{\partial^2 s_k}{\partial x_i \partial x_j} = 0 \tag{64}$$

Assuming plane wave propagation of the form $s_i = A_i e^{jk_j x_j - \omega t}$ and wave number $k = \frac{\omega}{v_{phase}} n$, where n is a directional unit vector, Bucur [6] rearrange the wave equation into:

$$\left(C_{ijkl} n_j n_k - \delta_{ik} \rho {v_{phase}}^2\right) P_m = 0 \tag{65}$$

in which, δ_{ik} = Kronecker tensor; $\delta_{ik} = 1$ for i=k and $\delta_{ik} = 0$ for i≠k

P_m = polarization or components of unit vector tangential to the displacement.

The first term in **Eq. 65** ($C_{ijkl}n_j n_k$) is the Kelvin-Christoffel tensor (Γ_{ik}). For wave propagation in the xz-axis, as seen in **Figure 23**, there are three main tensors of interest:

$$\Gamma_{xx} = C_{11}{n_x}^2 + C_{55}{n_z}^2, \quad \Gamma_{zz} = C_{33}{n_z}^2 + C_{55}{n_x}^2$$
$$\Gamma_{xz} = (C_{13} + C_{55}\ ) n_x n_z. \tag{66}$$

Wave propagation in the xz-plane consists of three different waves: quasi-longitudinal (QL) and quasi-transverse (QT) waves in plane of propagation and transverse waves (T) in the orthogonal direction. Phase velocities (V) of each wave are calculated as follows [6]:

$$V_{QL,QT} = \frac{\sqrt{\Gamma_{xx} + \Gamma_{zz} \pm \sqrt{(\Gamma_{xx} - \Gamma_{zz})^2 + 4\Gamma_{xz}}}}{2\rho} \tag{67}$$

$$V_T = \frac{\sqrt{C_{66}{n_x}^2 + C_{44}{n_z}^2}}{\rho}. \tag{68}$$

Displacement potentials of the orthotropic solid are then calculated as follows:

$$\varphi_{QL} = e^{j\omega t - jk_x X}\left[A_1 e^{-jk_{QL}} + A_2 e^{jk_{QL}}\right] \tag{69}$$

$$\psi_{QT} = e^{j\omega t - jk_x X}\left[A_3 e^{-jk_{QT}} + A_4 e^{jk_{QT}}\right] \tag{70}$$

$$\psi_T = e^{j\omega t - jk_x X}\left[A_5 e^{-jk_T} + A_6 e^{jk_T}\right]. \tag{71}$$

Using a similar method as outlined in Biot theory, total displacement can be calculated as $s = \nabla\varphi + \nabla\times\psi$. Velocity is just $\frac{\partial s}{\partial t}$ and the normal strain are $\frac{\partial s_x}{\partial x}, \frac{\partial s_z}{\partial z}$ in x and z-axis. Shear strain is defined as $\frac{1}{2}\left(\frac{\partial s_z}{\partial x} + \frac{\partial s_x}{\partial z}\right)$. Using those relations, the acoustic field matrix can be calculated as follows:

$$u_x^s = j\omega\left(\frac{\partial \varphi_{QL}}{\partial x} - \frac{\partial \psi_T}{\partial z}\right) \tag{72}$$

$$u_z^s = j\omega\left(\frac{\partial \varphi_{QL}}{\partial z} + \frac{\partial \psi_T}{\partial x}\right) \tag{73}$$

$$\sigma_{zz}^{s} = C_{31}\left(\frac{\partial\varphi^{2}{}_{QL}}{\partial x^{2}} - \frac{\partial\psi_{T}}{\partial x\partial z}\right) + C_{33}\left(\frac{\partial\varphi^{2}{}_{QL}}{\partial z^{2}} + \frac{\partial\psi_{T}}{\partial x\partial z}\right) \tag{74}$$

$$\sigma_{xz}^{s} = C_{55}\left(2\frac{\partial\varphi_{QL}}{\partial x\partial z} - \frac{\partial^{2}\psi_{T}}{\partial z^{2}} + \frac{\partial^{2}\psi_{T}}{\partial x^{2}}\right) \tag{75}$$

Using the wavenumber calculated from the three velocity components, a similar method as the one employed in **Eq. 57** can be used to calculate the transfer matrix.

3.3.4 Porous Media

Compared to other types of materials, porous media have been studied extensively for their acoustic properties. This is mostly due to their ease of access and their usefulness as acoustic materials, especially as a sound absorber or an impact-insulation material. Typically, there are two major categories of models which are used to predict the sound-propagation behavior in a porous media: empirical and analytical models. Analytical models are then divided further into rigid-frame and elastic-frame models.

3.3.4.1 Empirical Models

Most empirical models are derived using a curve-fitting approach based on existing experimental results. One of the most commonly-used, yet relatively simple, empirical models was developed by Delaney and Bazley in 1970 [39]. This model used the airflow resistivity of the material to determine the characteristic impedance and sound absorption at any given frequency. Further improvements to this model were made by Miki in 1990 to improve the prediction model accuracy by modifying the equation such that the positive-real property could be achieved in the low-frequency region [40]. Lastly, in 2008, Komatsu improved these models to further increase prediction accuracy for high-density and low-density fibrous materials in which the ratio of frequency to airflow resistivity is less than 0.01 m^3/kg and more than 0.1 m^3/kg, respectively [41]. These three models can be seen as follows:

Delaney-Bazley Model [39]

$$Z_c = \rho_0 c_0 \left(1 + 0.0571 \left(\frac{\rho_0 f}{\sigma_R}\right)^{-0.754} - j0.087 \left(\frac{\rho_0 f}{\sigma_R}\right)^{-0.732}\right) \tag{76}$$

$$k = \frac{\omega}{c_0}\left(1 + 0.0978 \left(\frac{\rho_0 f}{\sigma_R}\right)^{-0.7} - j0.189 \left(\frac{\rho_0 f}{\sigma_R}\right)^{-0.595}\right) \tag{77}$$

Miki Model [40]

$$Z_c = \rho_0 c_0 \left(1 + 0.070 \left(\frac{f}{\sigma_R}\right)^{-0.632} - j0.107 \left(\frac{f}{\sigma_R}\right)^{-0.632}\right) \tag{78}$$

$$k = \frac{\omega}{c_0}\left(1 + 0.109 \left(\frac{f}{\sigma_R}\right)^{-0.618} - j0.160 \left(\frac{f}{\sigma_R}\right)^{-0.618}\right) \tag{79}$$

Komatsu Model [41]

$$Z_c = \rho_0 c_0 \left(1 + 0.00027 \left(2 - log\left(\frac{f}{\sigma_R}\right)\right)^{6.2} - j0.0047 \left(2 - log\left(\frac{f}{\sigma_R}\right)\right)^{4.1}\right) \tag{80}$$

$$k = \frac{\omega}{c_0}\left(0.0069 \left(2 - log\left(\frac{f}{\sigma_R}\right)\right)^{4.1} + j(1 + 0.0004) \left(2 - log\left(\frac{f}{\sigma_R}\right)\right)^{6.2}\right) \tag{81}$$

in which, Z_c= Characteristic impedance (Rayl or Pa.s.m^{-1})

k = Propagation constant (m^{-1})

ω = Angular velocity (rad/s)

σ_R = Airflow resistivity (Pa.s.m^{-2})

f = Frequency (Hz)

ρ_0 = Ambient air density (kg/m^3)

c_0 = Speed of sound in air (m/s).

The transfer matrix for this empirical model for porous media can be derived by assuming that the porous media behave similarly to an equivalent fluid with characteristic impedance Z_c, such that:

$$[T] = \begin{bmatrix} \cos(k_z d_i) & \mathrm{j} Z_c \sin(k_z d_i) \\ \mathrm{j}\dfrac{1}{Z_c}\sin(k_z d_i) & \cos(k_z d_i) \end{bmatrix}. \tag{82}$$

3.3.4.2 Analytical Model – Rigid-Frame Assumption

In contrast to the three empirical models, analytical models are typically developed to predict the absorption values based on sound-propagation theories. They are divided into two different categories based on the underlying assumption: rigid-frame model and elastic-frame model. With the rigid-frame model assumption, the porous medium is typically assumed to be an equivalent fluid in which there is only an air-borne wave propagating inside the medium. On the other hand, elastic-frame models also include the propagation of multiple stress waves for the fluid-saturated poro-elastic media.

One of the most commonly used rigid-frame prediction models was introduced by Attenborough in 1983 [42]. This model uses various material properties: porosity, flow resistivity, tortuosity, and also shape factors. It was found that, by using these parameters, better accuracy was achieved in predicting the sound absorption of high flow resistivity materials, especially soils and sands at low frequency, compared to the Delaney-Bazley model. In this model, sound propagation in each pore in the rigid medium is analyzed and then extended into a bulk medium to get the characteristic impedance and propagation constant.

Attenborough Model [42]

Complex density (ρ_x) and compressibility functions (C_x) in each pore of a rigid frame air-filled porous materials are calculated as follows:

$$\rho_x = \rho_0 \left(1 - \frac{2T\left(\lambda\sqrt{-j}\right)}{\lambda\sqrt{-j}}\right)^{-1}, \quad T = \frac{J_1(\zeta)}{J_{0(\zeta)}} \tag{83}$$

$$C_x = \frac{1}{\gamma P_0}\left(1 + (\gamma - 1)\frac{2T\left(\lambda\sqrt{-jN_{Pr}}\right)}{\lambda\sqrt{-jN_{Pr}}}\right), \quad \lambda = \frac{1}{2S_p}\sqrt{\frac{8\rho_0\omega\alpha_\infty}{\sigma\Phi}} \tag{84}$$

in which, $J_x(\zeta)$	= Bessel function of the xth order
γ	= Specific heat ratio of air (1.4)
P_0	= Atmospheric air pressure (Pa)
N_{Pr}	= Prandtl number
S_p	= Pore shape factor
Φ	= Porosity
α_∞	= Tortuosity

From these complex density and compressibility functions, the effective density (ρ_e) and bulk modulus (K_e) of the porous material can be derived. These two values are then used to calculate the characteristic impedance (Z_c) and propagation constant (k) of the material:

$$\rho_e = \left(\frac{\alpha_\infty}{\Phi}\right)\rho_x, \quad K_e = \frac{1}{C_e} = \frac{1}{\Phi C_x}, \qquad Z_c = \sqrt{\rho_e K_e}, \quad k = \omega\sqrt{\frac{\rho_e}{K_e}} \tag{85}$$

Johnson-Allard-Champoux Model [43] [44]

In addition to the model proposed by Attenborough, there is an alternative model which was proposed by Johnson, Koplik, and Dashen in 1987 which used four parameters: open porosity, static airflow resistivity, high-frequency limit of tortuosity, and viscous characteristic length [43]. In 1991, Allard and Champoux [44] introduced two more parameters to take into account the thermal effects and frequency-dependent bulk modulus of saturating fluids:

$$\rho_e = \alpha_\infty \rho_0 \left(1 + \frac{\sigma \Phi}{j\omega \alpha_\infty \rho_0} \sqrt{1 + \frac{4j{\alpha_\infty}^2 \eta \rho_0 \omega}{{\sigma_R}^2 \Lambda^2 \Phi^2}} \right) \tag{86}$$

$$K_e = \frac{\gamma P_0}{\gamma - (\gamma - 1)(1 + \frac{8\eta}{j\Lambda'^2 N_{Pr} \omega \rho_0} \sqrt{1 + \frac{j\Lambda'^2 N_{Pr} \omega \rho_0}{16\eta}}} \tag{87}$$

in which, Λ = Viscous characteristic length (m)

Λ' = Thermal characteristic length (m).

Similar to the Attenborough formula, the characteristic impedance (Z_c) and propagation constant (k) can be derived from the effective density and bulk modulus of the medium. The transfer-matrix model for rigid-frame porous media has the same form as in **Eq. 82**.

3.3.4.3 Analytical Model – Elastic-Frame Assumption

A transfer-matrix model for porous media with elastic frame has been developed by Allard [2] using Biot wave theory. According to Biot, wave propagation in porous media consists of two compressional/longitudinal waves and one shear wave. Allard defined the acoustic-field matrix **V** as a vector consisting of the velocity of the frame (solid) in the x and z-directions (u_x^s and u_z^s), the velocity of the fluid (u_z^f), the compressional stress of the frame (σ_{zz}^s), the shear stress of the frame(σ_{xz}^s), and the compressional stress of the fluid (σ_{zz}^f). These six values can be calculated as follows [2]:

$$u_x^s = j\omega\left(\frac{\partial\varphi_x^s}{\partial x} + \frac{\partial\varphi_y^s}{\partial x} - \frac{\partial\psi_y^s}{\partial z}\right) \tag{88}$$

$$u_z^s = j\omega\left(\frac{\partial\varphi_x^s}{\partial z} + \frac{\partial\varphi_y^s}{\partial z} - \frac{\partial\psi_y^s}{\partial x}\right) \tag{89}$$

$$u_z^f = j\omega\left(\frac{\partial\varphi_x^f}{\partial z} + \frac{\partial\varphi_y^f}{\partial z} - \frac{\partial\psi_y^f}{\partial x}\right) \tag{90}$$

$$\begin{aligned}\sigma_{zz}^s = {} & (P-2N)\left(\frac{\partial^2\left(\varphi_x^s+\varphi_y^s\right)}{\partial x^2} + \frac{\partial^2\left(\varphi_x^s+\varphi_y^s\right)}{\partial z^2}\right) \\ & + Q\left(\frac{\partial^2\left(\varphi_x^f+\varphi_y^f\right)}{\partial x^2} + \frac{\partial^2\left(\varphi_x^f+\varphi_y^f\right)}{\partial z^2}\right) + 2N\left(\frac{\partial^2\left(\varphi_x^s+\varphi_y^s\right)}{\partial z^2} + \frac{\partial^2\psi_y^s}{\partial x\partial z}\right)\end{aligned} \tag{91}$$

$$\sigma_{xz}^s = N\left(2\frac{\partial^2\left(\varphi_x^s+\varphi_y^s\right)}{\partial x\partial z} + \frac{\partial^2\psi_y^s}{\partial x^2} - \frac{\partial^2\psi_y^s}{\partial z^2}\right) \tag{92}$$

$$\sigma_{zz}^f = R\left(\frac{\partial^2\left(\varphi_x^f+\varphi_y^f\right)}{\partial x^2} + \frac{\partial^2\left(\varphi_x^f+\varphi_y^f\right)}{\partial z^2}\right) + Q\left(\frac{\partial^2\left(\varphi_x^s+\varphi_y^s\right)}{\partial x^2} + \frac{\partial^2\left(\varphi_x^s+\varphi_y^s\right)}{\partial z^2}\right) \tag{93}$$

Based on **Eqs. 88** to **93**, stress components in an elastic- frame porous material have several coefficients, which are the three Biot coefficients (P,Q,R) and its shear modulus (N). The three Biot coefficients are calculated as follows [2]:

$$P = \frac{4}{3}N + K_b + \frac{(1-\emptyset)^2}{\emptyset} \tag{94}$$

$$Q = K_f(1-\emptyset) \tag{95}$$

$$R = \emptyset K_f \tag{96}$$

in which,	$\emptyset$	= Porosity
	K_b	= Bulk modulus of the frame

K_f = Bulk modulus of the fluid (refer to **Eq. 87**).

Assuming an isotropic frame model, bulk modulus of the frame is simply $\lambda + \frac{2}{3}\mu$ or:

$$K_b = \frac{2N(\nu + 1)}{3(1 - 2\nu)}. \tag{97}$$

The six displacement potentials in **Eqs. 88- 93** are written as follows [2]:

$$\varphi_x^s = A_1 e^{j(\omega t - k_{xz}z - k(sin\theta)x} + A_2 e^{j(\omega t + k_{xz}z - k(sin\theta)x} \tag{98}$$

$$\varphi_y^s = A_3 e^{j(\omega t - k_{yz}z - k(sin\theta)x} + A_4 e^{j(\omega t + k_{yz}z - k(sin\theta)x} \tag{99}$$

$$\psi_y^s = A_5 e^{j(\omega t - k_{zz}z - k(sin\theta)x} + A_6 e^{j(\omega t + k_{zz}z - k(sin\theta)x} \tag{100}$$

$$\varphi_x^f = \vartheta_x \varphi_x^s \tag{101}$$

$$\varphi_y^f = \vartheta_y \varphi_y^s \tag{102}$$

$$\psi_y^f = \vartheta_z \psi_y^s. \tag{103}$$

The six wave numbers, k_{ij} for compressional waves and k'_{ij} for the shear waves, are calculated as follows:

$$k_{iz} = \sqrt{\varsigma_i{}^2 - (ksin\theta)^2}, \qquad i = x, y, z \tag{104}$$

$$k'_{iz} = -k_{iz}, \qquad i = x, y, z \tag{105}$$

Similar to the isotropic-solid case, using the six values calculated in **Eqs. 98-103** as the components of the acoustic-field matrix (**V**), and introducing a new vector$\mathbf{A} = [A_1 + A_2,\ A_1 - A_2,\ A_3 + A_4,\ A_3 - A_4,\ A_5 + A_6,\ A_5 - A_6\ ]^T$, an intermediary matrix (Υ) is calculated such that $\mathbf{V} = [\Upsilon(\mathrm{z})]\mathbf{A}$. The transfer matrix can then be calculated using **Eq. 57**.

3.3.5 Perforated Panel

A transfer-matrix model for perforated panels with holes relatively large compared to its viscous boundary layer was developed using the method derived by Maa [4] for micro-perforated panels, and later by Attala and Sgard [5] for micro- and macro-perforated panels. These formulas were developed specifically for circular perforations. However, by using the end-correction factor derived by Ingard [34], these prediction models can be extended to include both circular and slot perforations.

Maa's Prediction Model for Micro-perforated Panels

As previously mentioned in **Section 3.2.2**, resistance from the thermoviscous effect in the neck region of an HR unit or perforated panel is relatively small compared to the resistance from a typical acoustic damping material. This is valid for most cases except for perforations which are significantly small such that the hole dimension is smaller than the viscous boundary layer at the wall. This type of perforated panel is often called micro-perforated panels, which mostly comes from their submillimeter hole size. In this type of panel, it is possible to achieve wide-band absorption without using any damping material.

One of the most well-known prediction models for micro-perforated panels was developed by Maa in 1998 [4]. According to Maa, the relative acoustic impedance of a micro-perforated panels is calculated as follows:

$$\frac{z}{\rho_0 c\varepsilon} = r + j\omega m \tag{106}$$

$$r = \frac{32\mu t}{\rho_0 c\varepsilon d^2} k_r, \qquad k_r = \sqrt{1 + \frac{k^2}{32} + \frac{\sqrt{2}}{32} k \frac{d}{l}} \tag{107}$$

$$\omega m = \frac{\omega t}{c\varepsilon} k_m, \qquad k_m = 1 + \frac{1}{\sqrt{1 + \frac{k^2}{2}}} + 0.85 \frac{d}{l} \tag{108}$$

Assuming that micro-perforated panels behave as a locally-reacting material, wavenumber (k) can be calculated as follows:

$$k = d\sqrt{\omega \rho_0 / 4\mu}. \tag{109}$$

Based on **Eq. 106,** and using the control-volume assumption, such that the particle velocity just before and after the perforated panel layer are the same, the transfer matrix for micro-perforated panel is simply:

$$[T] = \begin{bmatrix} 1 & (r + j\omega m)cos\theta \\ 0 & 1 \end{bmatrix}. \tag{110}$$

<u>Attala and Sgard's Equivalent-Fluid Model</u>

For a more general perforated panel, Attala and Sgard [5] suggested that the perforated panel behaves similarly to a rigid-frame porous media, such that the equivalent-fluid model developed by Johnson, Allard, and Champoux [44] for porous media can be utilized. Using the high-frequency approximation for the effective density in **Eq. 86**, the effective density and surface impedance can be obtained as follows:

$$\rho_e = \alpha_\infty \rho_0 \left(1 + \frac{4}{\rho_0 \omega d} R_w\right) - j\alpha_\infty \rho_0 \frac{4}{\rho_0 \omega d} R_w \tag{111}$$

$$z = \alpha_\infty \rho_0 \frac{4l}{d} R_w + j\alpha_\infty \rho_0 \omega t + j\alpha_\infty \frac{4l}{d} R_w. \tag{112}$$

In the above equations, R_w is simply the resistance term derived previously in **Eqs. 21** and **22**. In Attala and Sgard's work, the tortuosity of the panel is approximated as 1+ 2 δ, in which δ = corrected length. To include more general cases of perforated panels with various cavity and neck sizes, Ingard's model in S**ection 3.2.3** can be used to calculate the tortuosity. Using similar assumptions as in micro-perforated panels, the transfer matrix can be simply derived as follows:

$$[T] = \begin{bmatrix} 1 & zcos\theta \\ 0 & 1 \end{bmatrix}. \tag{113}$$

3.3.6 Assembling the Global Transfer Matrix

In a multi-layered system model, Allard introduces interface matrices to connect two different layers with different natures [2]. These interface matrices ($[I]$ and $[J]$) are defined such that at boundary Zi in **Figure 23**:

$$[I]\boldsymbol{V_{zi}} + [I]\boldsymbol{V_{ii-1}} = 0 \tag{114}$$

Both $[I]$ and $[J]$ matrices are derived based on the continuity of velocity and stress components at the boundary. Transfer-matrix model for fluid media, simplified and rigid-frame porous media, as well as perforated panels are considered very similar and have the same interface matrices. The interface matrices for various layers are described as follows [2]:

<u>**Solid-Fluid Interface**</u>

$$[I_{SF}] = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad [J_{SF}] = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix}, \tag{115}$$

<u>**Porous-Fluid Interface**</u>

$$[I_{PF}] = \begin{bmatrix} 0 & (1-\emptyset) & \emptyset & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad [J_{PF}] = \begin{bmatrix} 0 & -1 \\ (1-\emptyset) & 0 \\ 0 & 0 \\ \emptyset & 0 \end{bmatrix}, \tag{116}$$

Solid-Porous Interface

$$[I_{SP}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad [J_{SP}] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \tag{117}$$

The interface matrices are interchangeable depending on the order of the layer at each interface. For a multilayered system where the incident wave travels from a fluid media (typically air in most acoustical applications), the transfer and interface matrices can then be combined into [2]:

$$[D_0] = \begin{bmatrix} [I_{f1}] & [J_{f1}][T_1] & [0] & \cdots & [0] & [0] \\ [0] & [I_{12}] & [J_{12}][T_2] & \cdots & [0] & [0] \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ [0] & [0] & [0] & \cdots & [J_{(n-2)(n-1)}][T_{n-1}] & [0] \\ [0] & [0] & [0] & \cdots & [I_{(n-1)(n)}] & [J_{(n-1)(n)}][T_n] \end{bmatrix} \tag{118}$$

Termination conditions should be considered when assembling the global transfer matrix for a multi-layered system. Two type of termination are considered in this study: hard-backing and semi-infinite air layer.

Rigid-Wall Termination

In the case of the rigid wall termination, velocity components must be zero at the final layer. Depending on the nature of the previous layer, $[D_0]$ matrix is then modified as follows [2]:

$$[D] = \begin{bmatrix} & [D_0] & & \\ [0] & \cdots & [0] & [Y] \end{bmatrix} \tag{119}$$

in which,

$$[Y] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \textit{ for porous layer}$$

$$[Y] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \textit{ for solid layer} \tag{120}$$

$$[Y] = [0 \quad 1] \textit{ for fluid and perforated panel layer.}$$

<u>**Semi-Infinite Air Layer**</u>

For semi-infinite air termination, $[D_0]$ matrix is modified into [2]:

$$[D] = \begin{bmatrix} & & [D_0] & & & [0] \\ [0] & \cdots & [0] & [I_{(n)f}] & & [J_{(n)f}] \\ [0] & \cdots & \cdots & [0] & -1 & Z_0/cos\theta \end{bmatrix} \tag{121}$$

3.3.7 Reflection, Absorption and Transmission Coefficients

Surface impedance of a multi-layered system can be calculated by using the following formula [2]:

$$Z_s = -\frac{det[D_1]}{det[D_2]} \tag{122}$$

in which, $[D_1]$ = matrix $[D]$ with the first column removed

$[D_2]$ = matrix $[D]$ with the second column removed.

Similar to **Eq. 35** and **36**, pressure-reflection and sound-absorption coefficients can be calculated as follows:

$$R = -\frac{Z_s - Z_0/cos\theta}{Z_s - Z_0/cos\theta} \tag{123}$$

$$\alpha(\theta) = 1 - |R|^2. \tag{124}$$

For diffuse-field excitation, sound-absorption coefficient can be found by:

$$\alpha_{diffuse} = \frac{\int_0^{\pi/2} \alpha(\theta) cos\theta sin\theta d\theta}{\int_0^{\pi/2} cos\theta sin\theta d\theta} \tag{125}$$

Pressure transmission coefficient and sound transmission loss are calculated as follows:

$$T = -(1 + R)\frac{det[D_{n+1}]}{det[D_1]} \tag{126}$$

$$TL(\theta) = -20log\,|T| \tag{127}$$

$$TL_{diffuse} = -10log\left[\frac{\int_0^{\frac{\pi}{2}} |T|^2(\theta) cos\theta sin\theta d\theta}{\int_0^{\frac{\pi}{2}} cos\theta sin\theta d\theta}\right]. \tag{128}$$

4 Experimental Methods

To aid the prototype-development process and validate the proposed transfer-matrix model (TMM), multiple methods of measuring sound-absorption coefficient of a material were considered. While the idea of using the transfer-matrix model (TMM) to predict the acoustic performance of a multi-layer material isn't necessarily new, as seen in [2], its usage in predicting the performance of an HR-array system hasn't been fully explored. Because of this, it is essential to check the validity of this prediction model by comparing it to the results obtained by using the conventional experimental methods (impedance tube, spherical decoupling method, and reverberation chamber method). These three methods were chosen based on the sound-wave excitation which is used in the measurement: impedance tube for normal incidence, spherical decoupling for specific angle of incidence, and reverberation chamber for diffuse-field excitation.

4.1 Impedance Tube

4.1.1 Theory

Impedance-tube measurement is typically conducted to measure the normal-incidence reflection, absorption and transmission coefficients of a small sample of a material. In this measurement, a plane wave is sent from one end of a relatively long and rigid-walled tube, made of a material with low sound-absorption, towards the specimen of interest where reflection and absorption will occur. Depending on the chosen method, multiple microphones are used to capture the incident wave, reflected wave and, sometimes, the transmitted wave. There are three main measurement methods which use this principle: moving-microphone or standing-wave method, two-microphone or transfer-function method, and four-microphone method [45]. While the first two methods are used to measure normal-incidence sound-absorption coefficients, the third method is typically used to measure the normal-incidence sound-transmission coefficient. However, this last method is only accurate for porous materials which support lateral wave propagation, unlike solid panels such as CLT panels. Because of this, only the first two methods will be discussed in detail.

4.1.1.1 Moving-Microphone Method (Standing-Wave Method)

In this method, a single-frequency tone is generated at the source end of the tube by using a loudspeaker such that there is a standing wave is created in the tube due to the reflection by the specimen, as seen in **Figure 24**. By using a movable microphone, measurements are made at two locations in the standing wave, one at the maxima and the other one at the minima of the standing wave. Details of the standing-wave method and the tube requirements can be found in ASTM C384 standard [46].

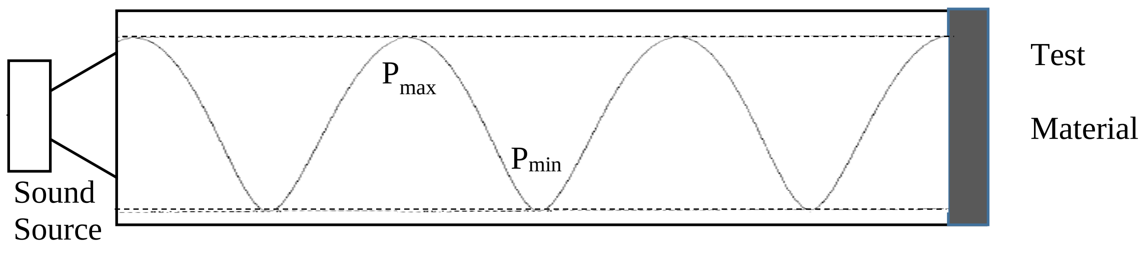


Figure 24. Pressure Level Distribution inside of an Impedance Tube Excited with a Single Frequency Sound Wave.

Assuming plane-wave propagation, the sound pressure inside the tube at any given point is:

$$P = A\left(e^{jkz} + Re^{-jkz}\right) \tag{129}$$

in which, P = Pressure inside the impedance tube (Pa)

z = Distance between the sound source and receiver (m)

k = Wave number in z-direction (1/m)

R = Pressure reflection coefficient

By moving the microphone along the length of the tube, the ratio between the maximum sound pressure and minimum sound pressure also called the standing wave ratio (SWR) can be calculated as follows:

$$SWR = \frac{P_{max}}{P_{min}} = \frac{1+|R|}{1-|R|} \tag{130}$$

By rearranging **Eq. 130**, the pressure reflection (R) and sound-absorption coefficient (α) can be simply calculated as follows:

$$|R| = \frac{SWR-1}{SWR+1} \tag{131}$$

$$\alpha = 1 - |R|^2. \tag{132}$$

4.1.1.2 Two-Microphone Method (Transfer-Function Method)

Even though the standing-wave method is relatively simple and accurate, it is very time-consuming and tedious. This is mainly due the repetition that is involved to get the sound-absorption coefficient over a wide range of frequencies. The transfer-function method and the standing-wave method use the similar tube and specimen holder construction as the standing-wave method; however, the sound source and microphone configurations are different. Instead of a single-frequency tone, a broad-band signal (typically white noise) is used as the sound signal. This sound signal is then captured at two fixed locations at certain distances from the specimen by using two microphones. This is very advantageous compared to the standing-wave method since all frequencies can be measured at the same time. The requirements for the impedance-tube apparatus, as specified in ASTM E1050 and ISO/CD 10534-2 standards, can be seen in **Table 6**.

Table 6. ASTM E1050 and ISO/CD 10534-2 Impedance Tube Specifications.

Specifications	**ASTM E1050 [47]**	**ISO/CD 10534-2 [7]**
Tube Cross-Section	Rectangular or circular with uniformity from end to end.	Rectangular or circular with uniformity from end to end.
Tube Wall	Non- Absorptive material with massive and rigid wall with negligible sound and vibration transmission	Non- Absorptive material with massive and rigid wall with negligible sound and vibration transmission. Wall thickness should be about 5% diameter (circular) or 10% cross-dimension (square)
Tube Diameter	$f_{upper} < 0.586\frac{c}{d}\ (circular)$ $f_{upper} < 0.500\frac{c}{d}\ (square)$ in which, f_{upper} = upper frequency limit (Hz) and d = diameter or larger cross-section dimension (m).	$f_{upper} < 0.58\frac{c}{d}\ (circular)$ $f_{upper} < 0.500\frac{c}{d}\ (square)$ in which, f_{upper}= upper frequency limit (Hz) and d = diameter or larger cross-section dimension(m).
Tube Length	At least three tube diameters between the sound source and nearest microphone	At least three tube diameters between the sound source and nearest microphone
Specimen Holder	Holder must have an airtight fit with the specimen. Petroleum jelly (Vaseline) or Silicone Grease can be used as sealant	Holder must have an airtight fit with the specimen. Vaseline can be used as sealant

Specimen Backing	Solid metal plate with thickness of more than 2cm is recommended.	Solid metal plate with thickness of more than 2cm is recommended.
Sound Source	Sound source shall be sealed and isolated from the tube.it should also have a uniform power response over the frequency range of interest.	Sound source must be contained in an insulating box. Surface of the loudspeaker membrane covers at least 2/3 of the tube's cross-sectional area
Microphone	Microphone diameter should be less than 20% of the wavelength for the highest frequency of interest.	Microphone diameter should be less than 20% of the spacing between the two mics. It should also be less than $\frac{c}{f_{upper}}$
Microphone Spacing	$0.01\frac{c}{f_{lower}} < spacing < \frac{1}{2}\frac{c}{f_{upper}}$	$0.05\frac{c}{f_{lower}} < spacing < 0.45\frac{c}{f_{upper}}$

If the microphones used for this measurement are phase-matched, the transfer function between microphones 2 and 1 (H_{12}) has the following form:

$$H_{12} = \frac{P_2}{P_1} = \frac{e^{jkz_2} + Re^{-jkz_2}}{e^{jkz_1} + Re^{-jkz_1}} \tag{133}$$

By rearranging **Eq. 133**, the sound pressure reflection coefficient can be calculated as follows:

$$R = \frac{H_{12} - e^{-jks}}{e^{jks} - H_{12}} e^{2jkz_1} \tag{134}$$

In addition to the real component, $2\pi\frac{f}{c}$, wave number (k) should also include an imaginary term which corresponds to the losses within the tube (k''), such that:

$$k = 2\pi\frac{f}{c} - k'' \tag{135}$$

Tube attenuation is typically determined by measuring the pressures at two subsequent minima in the tube. However, it can also be approximated by using the following formula [7]:

$$k'' = 0.0194\frac{d\sqrt{f}}{c} \tag{136}$$

An alternative formula was developed by Kirchoff and modified by Beranek [48]:

$$k'' = 0.02203\frac{d\sqrt{f}}{c} \tag{137}$$

While **Eqs. 133** and **134** are valid for phase-matched microphones, a correction can be applied to a pair of non-phase-matched microphones to get the similar result. To calculate the correction factor (H_c), the measurement needs to be done twice. The first measurement is done with the regular microphone configuration while the second measurement is conducted with the microphones' positions interchanged. It should be noted that the connections between each microphone to its preamplifier and signal analyzer should be kept the same even though they are in the interchanged position. The corrected transfer function can then be calculated [7]:

$$H_{12} = \frac{\dot{H}_{12}}{H_c} \tag{138}$$

$$H_c = \sqrt{\frac{\dot{H}_{12}}{\ddot{H}_{12}}} \tag{139}$$

in which, $\dot{H}_{12}$ = Transfer function at the initial position

$\ddot{H}_{12}$ = Transfer function at the interchanged position

H_c = Correction factor.

After the corrected transfer function has been calculated to account for phase mismatch between the two microphones, the pressure-reflection and sound-absorption coefficients can be calculated using **Eqs. 134** and **Eq. 132**, respectively.

4.1.2 Experiment Setup

For this study, the transfer-function method was used with a pair of ¼ inch B&K 4178 phase-matched condenser microphones. Each microphone is connected to a B&K 2269 preamp, which supplies the required 200 V polarization voltage, and to a signal analyzer (Soundbook by SINUS GmbH). This signal analyzer is also used to generate a white noise signal which is sent to a 4” woofer. For low-frequency analysis, the experiment is conducted inside a 101.5-cm-long circular steel tube with an internal diameter of 9.90 cm. For the high-frequency case, a 30-cm-long tube with a diameter of 2.90 cm is used instead.

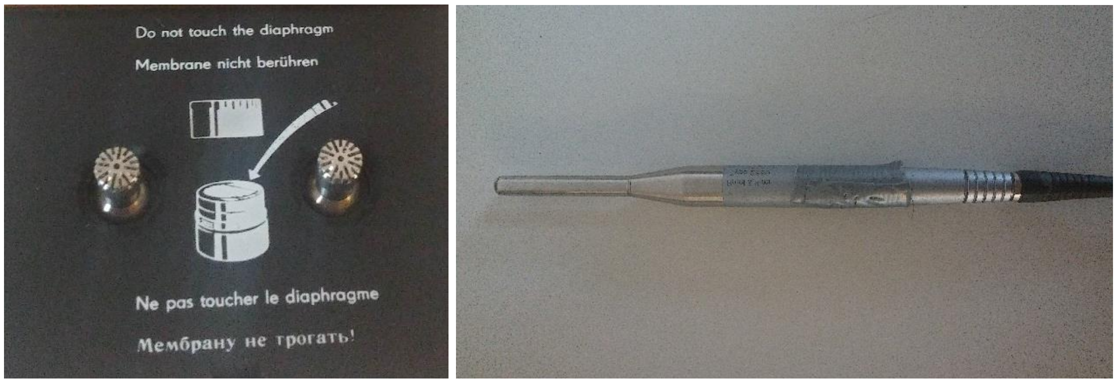


Figure 25. A pair of Phase Matched B&K 4178 Microphones and B&K 2269 Preamplifier with ½ inch to ¼ inch Adapter..

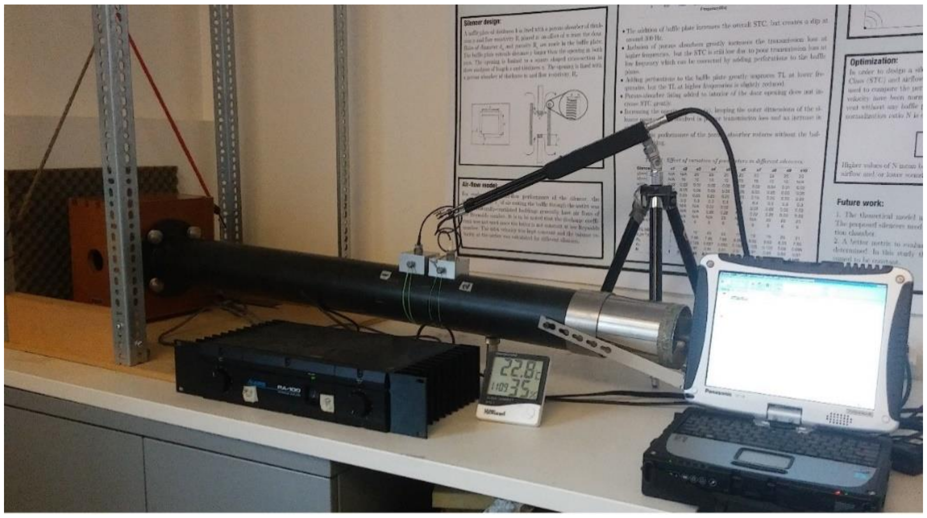


Figure 26. Low-Frequency Impedance Tube and Power Amplifier.

4.2 Spherical Decoupling

4.2.1 Theory

Impedance-tube measurement is very straight-forward and typically sufficient in measuring the normal-incidence sound-absorption coefficient of materials. However, in a real environment, a normal-incidence assumption is not really practical, and in most cases, the diffuse-field sound absorption is required instead. A diffuse field is defined as a field in which the energy density propagating in all directions inside of a room is uniform and equally probable [45]. Even though this is a special case which may not be realistic in most applications, it is considered to be the best assumptions in rooms. To get the diffuse-field result, approximation can be made by integrating the sound-absorption coefficient over a range of angles of incidence. If a material is normally-reacting, where normal- incidence acoustic impedance is independent of angle of incidence, diffuse field sound absorption coefficient can also be calculated from the normal incidence value [49].

The spherical-decoupling method was introduced by Allard as a method to measure the acoustic impedance and sound-absorption coefficient of a material when it is exposed to a sound wave incident at a specific angle of incidence [44]. While the basic concept behind this method is very similar to that of a regular impedance-tube measurement, there are several main differences, such as: spherical-wave propagation instead of plane-wave, arbitrary angle of incidence instead of normal incidence, and a hemi-anechoic chamber as the test platform instead of a long acoustic waveguide. A simplified diagram of a spherical-decoupling experimental setup can be seen in **Figure 27**.

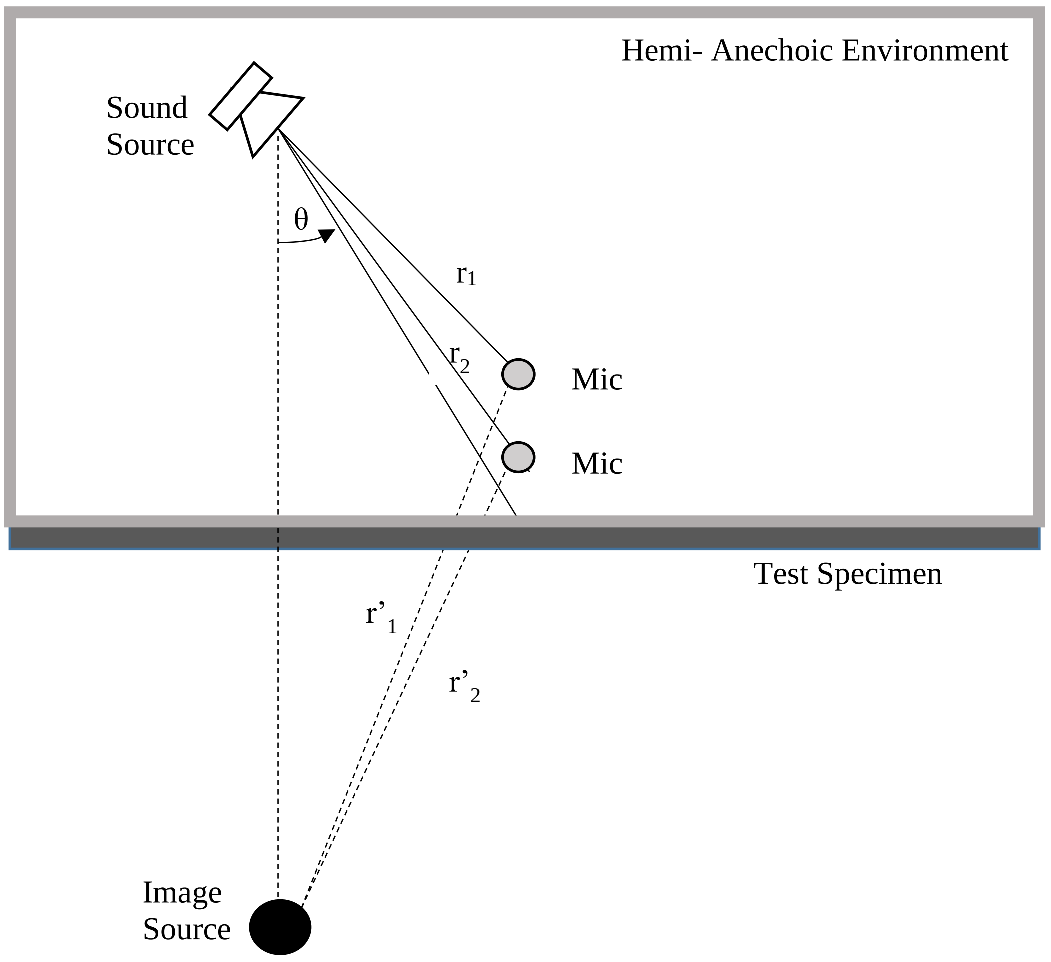


Figure 27. Spherical Decoupling Method Experimental Setup.

Assuming that the total sound pressure at each microphone in **Figure 27** comes from the superposition of two spherical waves, which are generated by the sound source and image source, pressures at Mic 1 (M_1) and Mic 2 (M_2) can be expressed as follows:

$$P(M_1) = P_0\left(\frac{e^{jkr_1}}{r_1} + R_\theta \frac{e^{jkr_1'}}{r_1'}\right) \tag{140}$$

$$P(M_2) = P_0\left(\frac{e^{jkr_2}}{r_2} + R_\theta \frac{e^{jkr_2'}}{r_2'}\right) \tag{141}$$

It should be noted that **Eqs. 133** and **134** are only valid for phase-matched microphones. Similar to the impedance-tube method, a correction factor can be used to account for phase mismatch between the two microphones. The transfer function between the two microphones (H_{12}), can be calculated as follows:

$$H_{12} = \frac{P_2}{P_1} = \frac{\frac{e^{jkr_2}}{r_2} + R_\theta \frac{e^{jkr_2'}}{r_2'}}{\frac{e^{jkr_1}}{r_1} + R_\theta \frac{e^{jkr_1'}}{r_1'}} \tag{142}$$

By rearranging **Eq. 142**, the pressure reflection can be calculated using the following formula:

$$R_\theta = \frac{\frac{e^{jkr_2}}{r_2} - H_{12}\frac{e^{jkr_1}}{r_1}}{H_{12}\frac{e^{jkr_1'}}{r_1'} - \frac{e^{jkr_2'}}{r_2'}} \tag{143}$$

The sound-absorption coefficient for a specific angle of incidence, and the diffuse-field excitation, can then be calculated using the following equations:

$$\alpha_\theta = 1 - |R_\theta|^2 \tag{144}$$

$$\alpha_{diffuse} = \frac{\int_0^{\pi/2} \alpha_\theta cos\theta sin\theta d\theta}{\int_0^{\pi/2} cos\theta sin\theta d\theta} \tag{145}$$

4.2.1 Experimental Setup

For spherical-decoupling measurement, a Norsonic dodecahedral omni-directional loudspeaker was used as the sound source. On the receiver side, a GRAS 50AI-B intensity probe with phase-matched 40AK ½-inch microphones was chosen, as seen in **Figure 28**. The two microphones were set-up such that the spacing between the first microphone and the test sample was 1 cm while the spacing between the two microphones was 2.5 cm. The sound-intensity probe is then connected to the same signal analyzer as the one used for the impedance-tube measurement. Six angles of incidence were chosen for this experiment (15°, 30°, 45°, 60°, 75°,

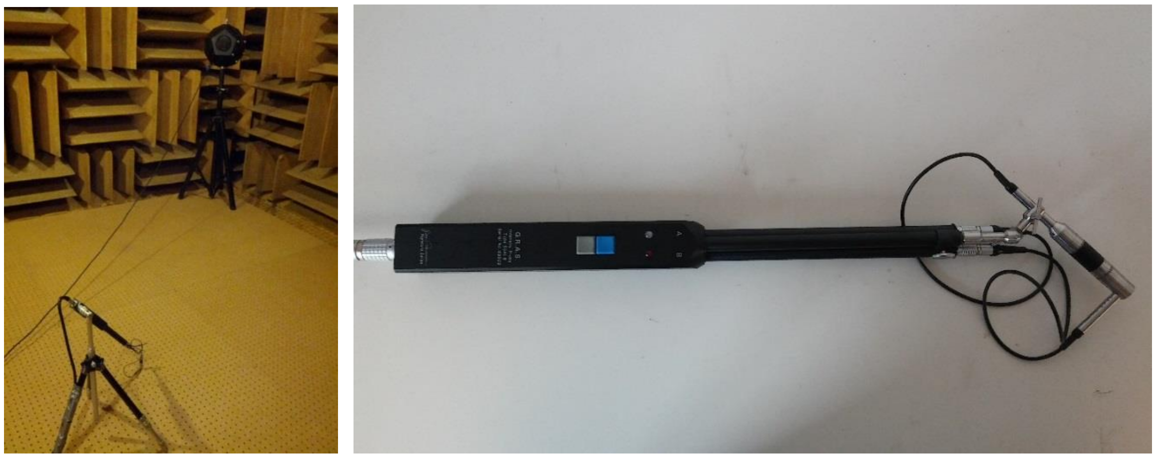


Figure 28. Spherical-Decoupling Experimental Setup with GRAS Intensity Probe inside an Anechoic Chamber.

and 90°). Based on these six angles of incidence, the diffuse-field sound absorption coefficient was then calculated by using **Eq. 145**.

4.3 Reverberation-Chamber Method

4.3.1 Theory

Instead of calculating the diffuse-field sound absorption by integrating specific-angle sound-absorption coefficients as seen in the Spherical Decoupling Method, it is possible to measure this value directly in a special laboratory environment, called a reverberation chamber. Based on the ASTM C423 standard, a reverberation chamber is a room designed such that the enclosed reverberant sound field closely approximate the diffuse-field condition both in the steady-state and during sound decay [50]. The complete specification of the reverberation chamber can be seen in **Table 7.**

To measure sound absorption inside of a reverberation chamber, two sets of measurements must be conducted. In the first experiment, the room-average reverberation time of the empty chamber at multiple locations is calculated from the measured sound-decay rate when a sound source is suddenly switched off. According to Long [23], reverberation time (RT) is defined as the time that it takes for sound to decay 60 dB in a room. In the next experiments, similar measurements are made with the specimen of interest inside of the chamber. If the specimen adds sound absorption, it will reduce the measured RT. Based on the two RT measurements, the absorption coefficient of the material can be calculated using the following formula:

$$\alpha_{diffuse} = \frac{1}{S}\left[0.16V\left(\frac{1}{T'_{60}} - \frac{1}{T_{60}}\right)\right] \tag{146}$$

in which, S = Specimen surface area (m^2)

V = Room volume (m^3)

T'_{60} = Reverberation time of the room with the specimen inside (s)

T_{60} = Reverberation time of empty room (s).

4.3.2 Experiment Setup

Due to the lack of access to a reverberation chamber at University of British Columbia, this measurement was done in a sound-transmission test facility owned by VanAir Design, in Vancouver. As seen in **Figure 29**, this test facility consists of two reverberation rooms which are adjacent to one another with opening between the two chambers fully blocked off during this experiment.

During the measurement, the test specimen with an area of 5.95 m^2 was placed in the middle of the room such that it was at an angle and not parallel to any side walls, and at distances of more than 1 m from those wall surfaces. Using WinMLS system, maximum-length-sequence (MLS) signals were then sent to the loudspeaker at the corner of the room (please refer to **Figure 7**), to generate sound waves to excite the room. By using a deterministic signal such as the MLS signal, impulse responses of the room could be measured for a broad-band frequency range of interest in one measurement. To capture the impulse response, a free-field half-inch microphone, connected to a Type 1 sound level meter RION NA-28, is chosen for this measurement. Fast Fourier Transform (FFT) was then used to calculate the frequency response of the impulse response and analyze the decay rate of sound energy in each frequency band. Lastly, the reverberation time (RT) and sound absorption of the specimen can then be calculated based on the decay rate, as seen in **Eq. 146**. This process is repeated until 16 averages were taken for each microphone location and then again for 6 different locations, which are chosen randomly at 1.5 m distance apart.

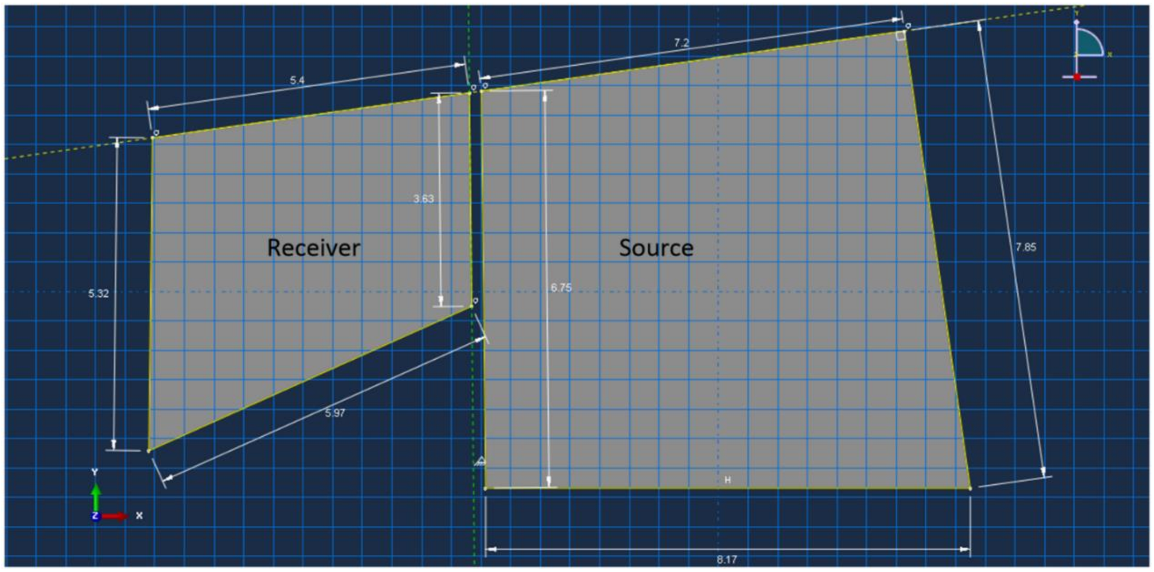


Figure 29. VanAir Sound Transmission Suite Floor Plan [51].

Table 7. ASTM C423 Reverberation Chamber Measurement Requirement.

Specifications	**ASTM C423 Requirements**
Room Size and Shape	Room volume should be larger than 125 m^3 (larger than 200 m^3 is recommended). No two room dimensions are equal.
Room Sound Absorption	Average sound absorption is less than or equal to 0.05 for one-third octave bands centered from 250 to 2500 Hz.
Temperature and Humidity	Temperature is higher than 10°C and average relative humidity is larger than 40%.
Signal to Noise Ratio	Signal to Noise Ratio (SNR) is at least 45 dB.
Test Specimen Size	Specimen area is larger than 5.57 m^2 (area of 6.69 m^2 is recommended).
Test Specimen Placement	No part of the specimen is located within 0.75 m of any reflective surface other than the one backing it.
Sound Source	One or more loudspeakers facing trihedral corner of the room should be used.
Microphone	Omnidirectional microphone with a flat random-incidence amplitude response (±2 dB within one-third octave band) should be used.
Number of Measurements	Measurements should be made at five or more locations which are at least 1.5 m apart and 0.75 m from any surface. At least 50 decays in each room condition shall be measured (empty and with specimen)

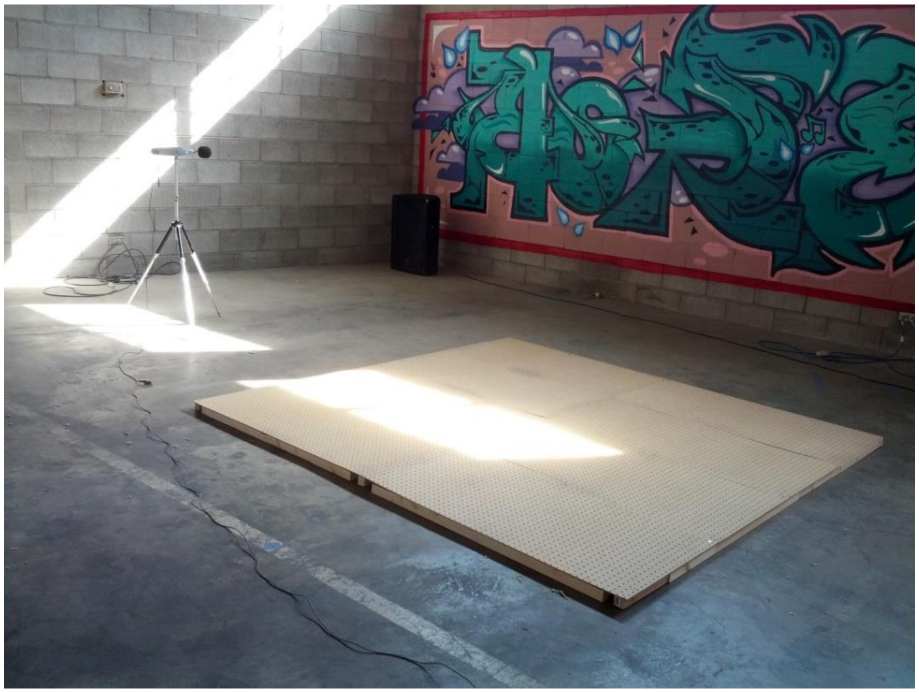


Figure 30. Reverberation-Chamber Experimental Setup.

4.4 Transfer-Matrix Method Validation

As seen in **Section 3.3**, a typical single-stage HR-array system can be divided into several different independent layers: a perforated panel which serves as a collection of multiple HR necks or openings, a damping material (typically a porous material) of a certain thickness, a fluid medium of a certain volume, and a solid backing. Each of these layers contributes differently to the acoustic impedance of the whole system; the first layer determines the acoustic inertance (mass) of the system, the second layer determines the damping factor, while the third and fourth layers determine the acoustic compliance (stiffness). To validate the Transfer-Matrix Method (TMM) for HR arrays, it is of interest to investigate the performance characteristics of HR-array systems with multiple different configurations of each layer, both theoretically and experimentally.

To test the contributions of each layer on the overall system performance, only the impedance-tube measurement technique was used to characterize the performance, based on its relatively small sample size and time investment compared to the other two methods. However, for the final prototype, all three experimental methods were used and compared.

4.4.1 Effects of Changing Parameters of an HR Array

There are several parameters of interest which affect the performance of an HR-array system: cavity volume, perforation size and shape, and system damping. To observe the effects of these parameters individually, and validate the TMM method for various configurations, several preliminary prototypes were created and tested in an impedance tube.

4.4.1.1 Cavity Volume

In classical HR theory, the air cavity typically behaves similarly to a spring in a mechanical mass-spring-damper system, in which the resonance frequency is inversely related to the cavity volume. In this first preliminary prototype phase, an HR-array system was developed such that it consisted of: a perforated panel with hole diameter of 7 mm, depth of 6 mm, and perforation ratio of 0.071 (refer to **Figure 31**); an air layer with depths of 2 cm, 4 cm, 6 cm, or 8 cm; and a rigid stainless-steel backing. Based on the Transfer-Matrix Model and classical HR theory, the resulting resonance frequency of the system should occur between 500 and 1000 Hz, increasing with air-layer depth, for the specified cavity volume. By using the impedance-tube method described in **Section 4.1**, the sound-absorption coefficients of the system configurations were measured, as seen in **Figure 32** below.

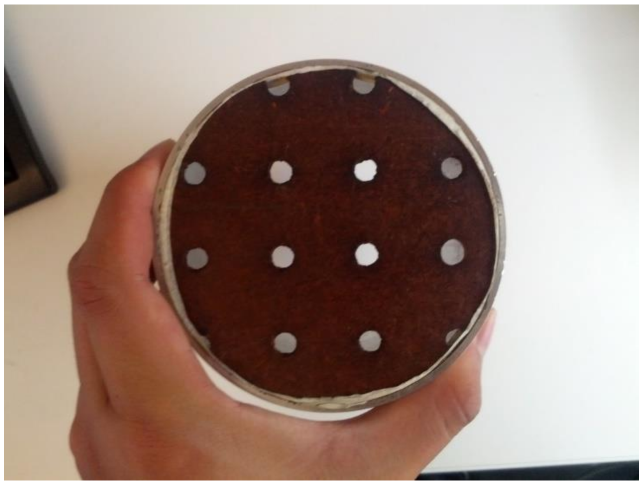
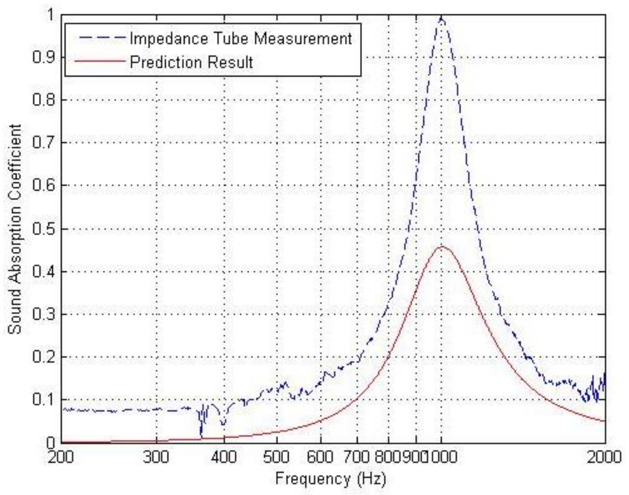


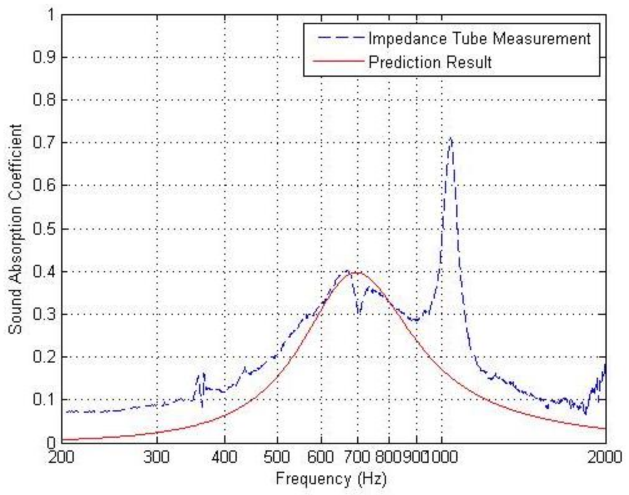
Figure 31. Perforated Panel Used in the First Preliminary Prototype.

As seen in **Figure 32**, the transfer-matrix model accurately predicted the resonance frequencies of the system configurations; however, it slightly underestimated the absorption coefficient for all four cases. Furthermore, there was a noticeable second resonance at 1000 Hz which was missing in the TMM prediction result in all configurations. Due to the behavior of the second resonance, which was independent of the cavity volume, it was hypothesized to be most likely caused by the unperforated-panel mode of the system. Unlike the infinitely-large perforated-panel cases, in which the acoustic mode mostly dominates the system response, the panel mode can’t be neglected in the case of a small and finite perforated panels such as the one tested in this experiment. While this second resonance may yield a beneficial result at a certain frequency, depending on the particular sample size and construction material, the added sound absorption was significantly narrower compared to the acoustic mode. The panel-vibration mode will be discussed in detail later in this chapter.

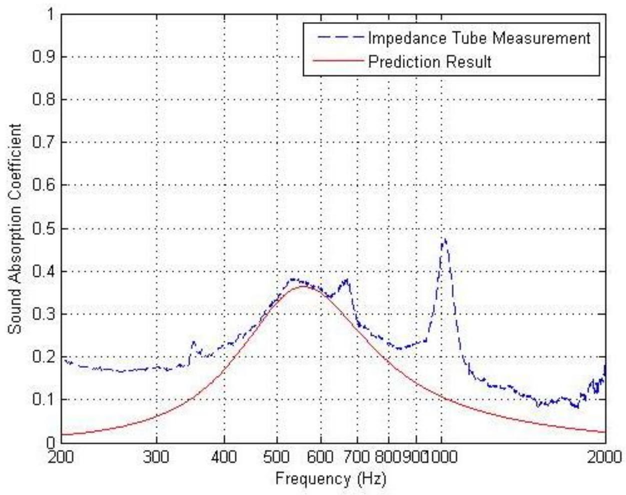
First Preliminary Prototype Sound Absorption Coefficient



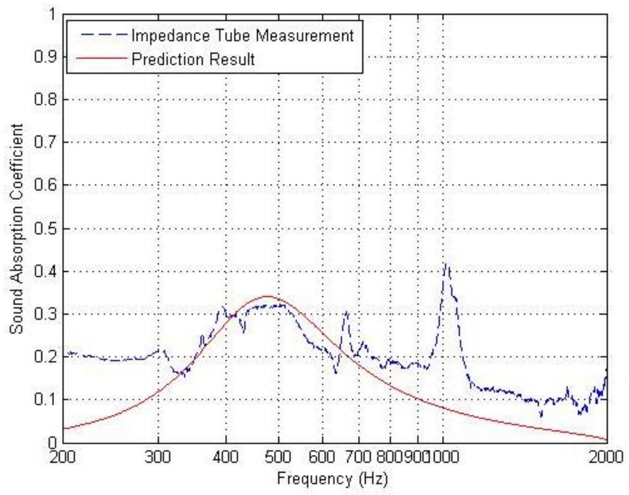
2 cm cavity depth



4 cm cavity depth



6 cm cavity depth



8 cm cavity depth

Figure 32. Low-Frequency Impedance Tube Measurement Results for First Preliminary Prototype Configurations.

4.4.1.2 Perforation Size and Shape

While the cavity volume determines the stiffness of an HR system, the neck or opening of the system behaves similarly to the mass in a mechanical mass-spring-damper resonator. In a single HR system, the acoustic mass or inertance is simply a function of the neck diameter. However, in an HR array, both neck diameter and perforation ratio should be considered when calculating the impedance and resulting sound-absorption coefficient of the system. To investigate these two parameters and the relation between them, and compare the predicted and experimental results, a second prototype was developed using a pair of perforated aluminum plates with thickness of 6.55 mm. These plates have similar total perforation areas and ratios of 0.045 and 0.05, respectively, but have vastly different neck diameters (6.25 mm for system A and 20.5 mm for system B) as seen in **Figure 33**. Keeping the cavity depth constant at 2 cm, the effect of perforation size on the sound-absorption coefficient of this HR system was measured in the impedance tube.

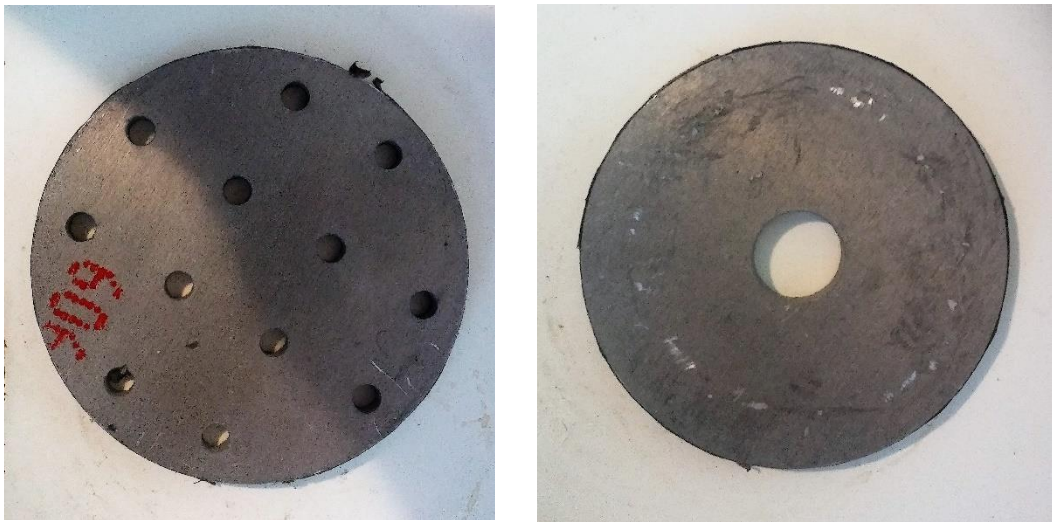


Figure 33. Pair of Perforated Aluminum Plates Used in the Second Preliminary Prototype with A) 12, 6.25 mm Diameter Holes and B) a single 20.5 mm hole.

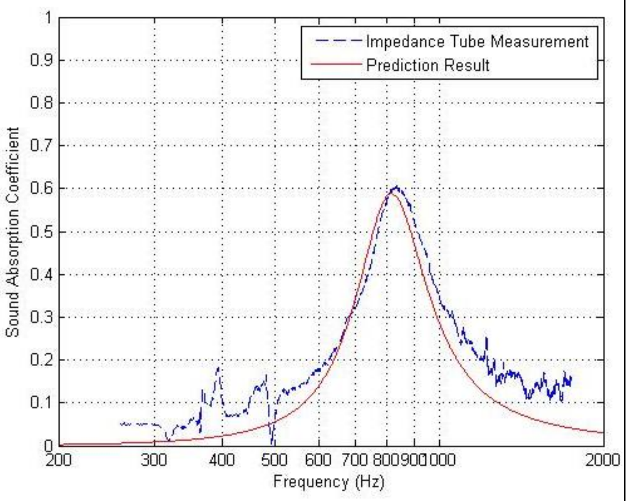
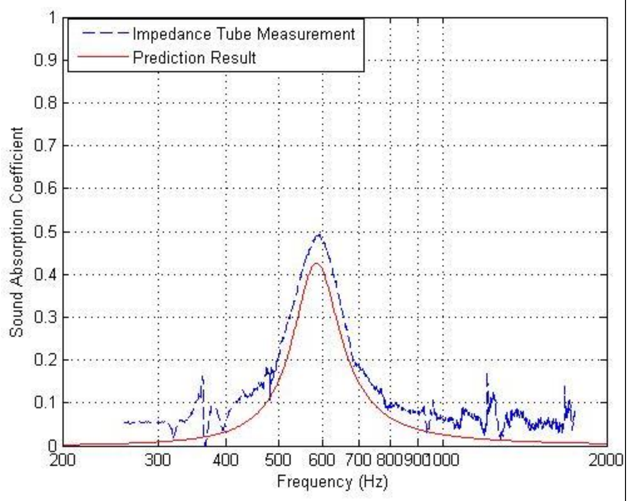
Second Preliminary Prototype Sound Absorption Coefficient	
	
6.25 mm Perforation Diameter	20.5 mm Perforation Diameter

Figure 34. Low-Frequency Impedance Tube Measurement Results for Second Preliminary Prototype.

As seen in **Figure 34**, the TMM accurately predicted the resonance frequencies; however, it slightly underestimated the sound-absorption coefficients. Furthermore, there was also a shift in resonance frequency, which was expected due to the changes in the perforation diameter. By using the classical theory for an HR array, as seen in **Section 3.2**, the resonance frequency of the system can be estimated to be a function of speed of sound (c), perforation ratio (ε), cavity depth (d), neck length (l), neck radius (a), and correction factor (δ), such that:

$$\omega_n = c\sqrt{\frac{\varepsilon}{dl'}} = c\sqrt{\frac{\varepsilon}{d(l + \delta a)}} \tag{147}$$

In addition to the shift in resonance frequency, the maximum sound-absorption coefficient of system A was also measured to be higher than for system B. This was mainly caused by the difference in the thermoviscous losses in the system, which were more prominent in system A.

4.4.1.3 Damping Material

In addition to changing the geometry of the HR system, as seen in **Section 4.4.1.1** and **4.4.1.2**, the sound-absorption performance of an HR system can also be manipulated by adding damping material to the system. For a purely resistive material, the addition of a damping material in an HR system will mainly determine the quality factor of the response curve. However, in practice, there is no purely resistive acoustic material. Because of this, by adding a damping material inside of an HR system, the resonance frequency may be shifted slightly.

For the third preliminary prototype, a 12.5-mm-thick sample of Sonoflex (refer to **Figure 35**) was added to an HR system consisting of the same perforated panel and cavity depth as seen in **Figure 31** and **Figure 32.** In this prototyping stage, it was of interest to see the effect of the location of the damping material inside of the HR cavity on the sound-absorption performance. First. the damping material was inserted such that it was located directly behind the perforations and the sound absorption of the system was measured by using the impedance-tube method. Then, the damping material was moved to the back of the cavity, directly in front of the rigid backing, for the second measurement.

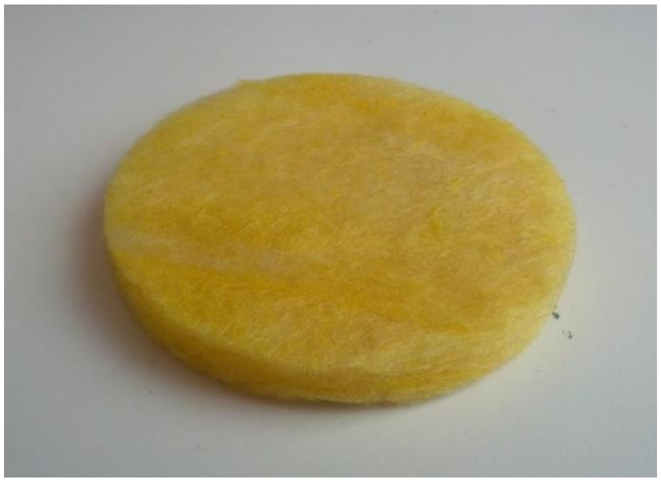
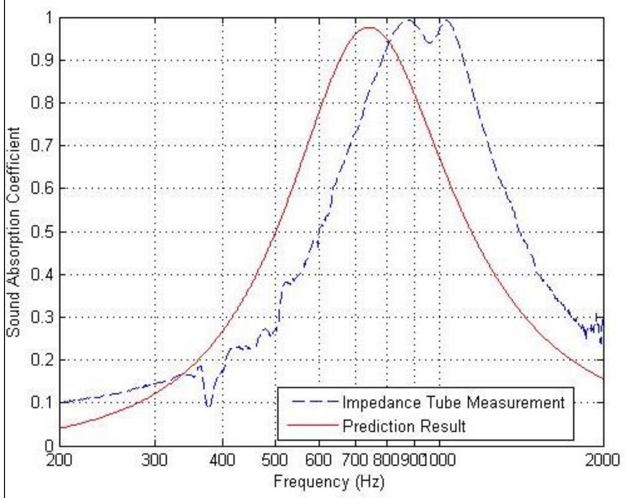
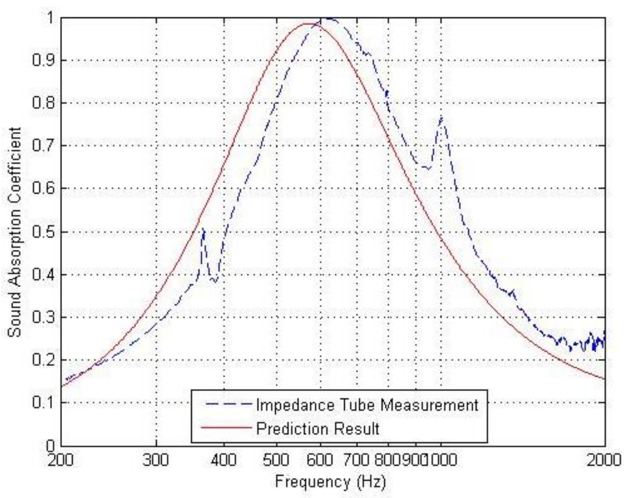


Figure 35. Sonoflex with 12.5 mm Thickness.

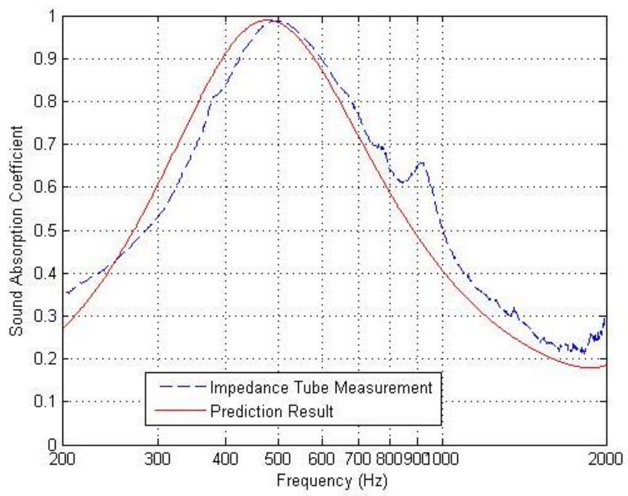
Damping Material Directly Behind the Perforation



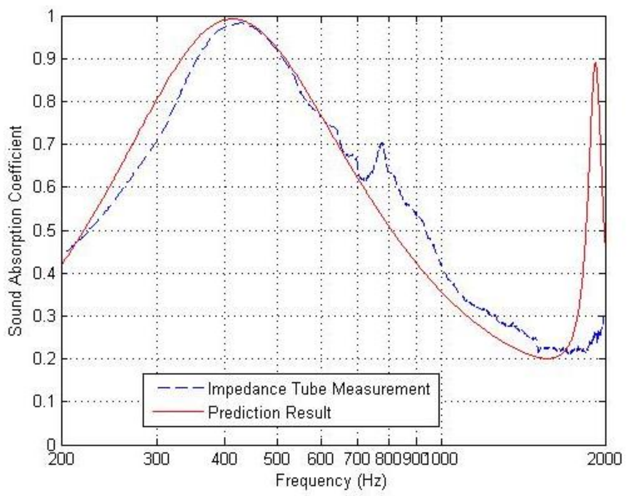
2 cm cavity depth



4 cm cavity depth



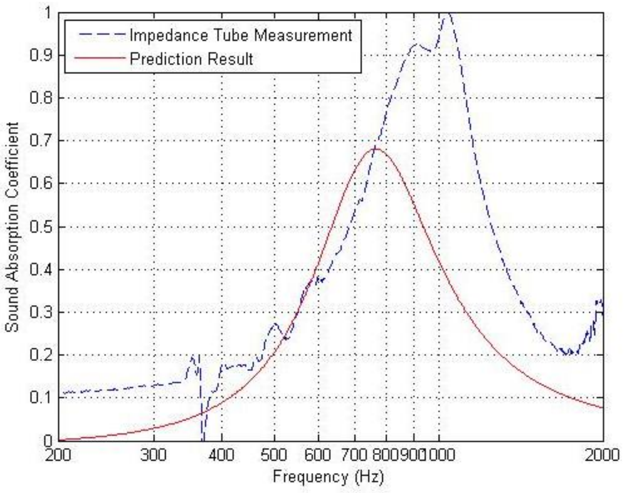
6 cm cavity depth



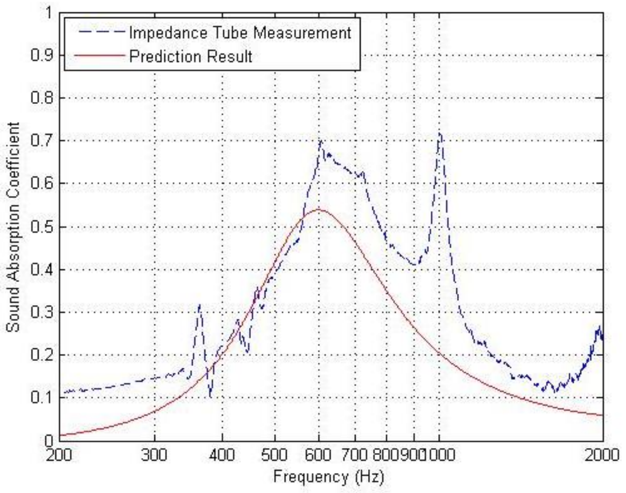
8 cm cavity depth

Figure 36. Low-Frequency Impedance-Tube Measurement Results for Third Preliminary Prototype with Damping Material behind Perforation

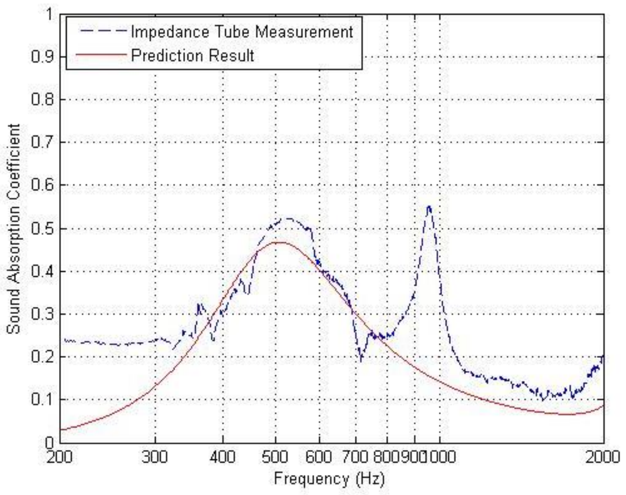
Damping Material in front of Rigid Backing



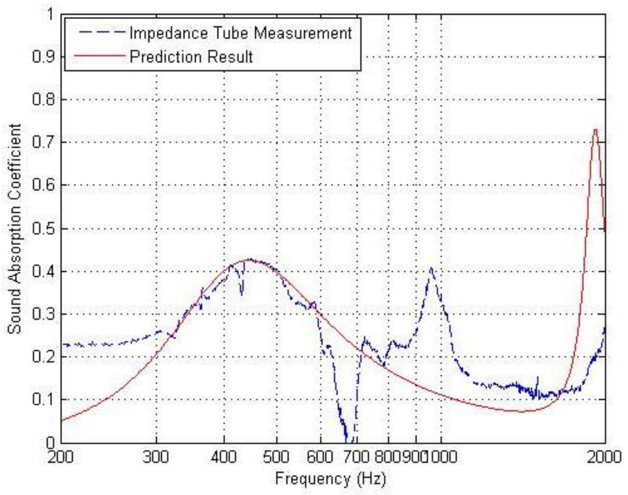
2 cm cavity depth



4 cm cavity depth



6 cm cavity depth



8 cm cavity depth

Figure 37. Low-Frequency Impedance-Tube Measurement Results for Third Preliminary Prototype with Damping Material in front of Rigid Backing.

As seen in **Figure 36** and **Figure 37**, resonance frequencies of the system were indeed shifted slightly to lower values compared to the ones in **Figure 32**. This is expected considering that the damping material (Sonoflex) isn't 100% porous, given that some of the air volume inside of the cavity is replaced by the fibers in the damping material. In addition to the slight change in resonance frequency, the sound-absorption coefficients of the system configurations were also noticeably higher compared to the results without any damping material.

Comparing the results in **Figure 36** and **Figure 37**, the sound-absorption-coefficient performance was significantly better if the damping material was located directly behind the perforations. This is somewhat expected, especially if we consider the mechanical mass-spring-damper analogy of the system. Assuming that the mass, represented by the air columns inside of the perforations, oscillated back and forth with a relatively small displacement compared to the cavity depth, and that the damping material behaves similarly to a viscous damper, maximum damping occurs at the center of the perforation where particle velocity is at its maximum value. Therefore, higher sound-absorption results are achieved at shorter distances between the damping material and the perforations.

In addition to the resonance caused by the acoustic mode, there were also several other resonances observed in **Figure 36** and **Figure 37**. Similar to the previous results shown in **Section 4.4.1.1**, a panel-vibration mode can be observed at 1000 Hz. While this narrow-band absorption was somewhat prominent in the second case, where the damping material was located at the back end of the HR cavity, this mode was not significant when the damping material was located at a more optimized location (close to the perforations). As seen in **Figure 36**, the acoustic mode completely dominates the sound-absorption mode for this particular case.

Interestingly, both the TMM prediction model and the measurement result indicated that there might be an additional resonance sound-absorption mode above the current impedance-tube frequency range (> 1750 Hz). This high-frequency mode will be discussed later in this chapter.

4.4.1.4 Damping inside the Perforation of an HR Array

In addition to using a porous material, such as Sonoflex, inside of the cavity, it might also be possible to add the damping inside the perforations themselves. Theoretically, the damping of the system reaches a maximum value at the center of the perforation. To test this theory, another pair of perforated aluminum plates similar to the one in **Figure 33-A** was created. However, for these two plates, the holes were threaded with M6 or M7 thread such that the minor diameter (for M7) and major diameter (for M6) of each respective perforation were close to the 6.25 mm diameter of the second prototype. Comparison of the sound-absorption characteristics of the two threaded prototype and the unthreaded one from **Figure 33-A** can be seen in **Figure 38**.

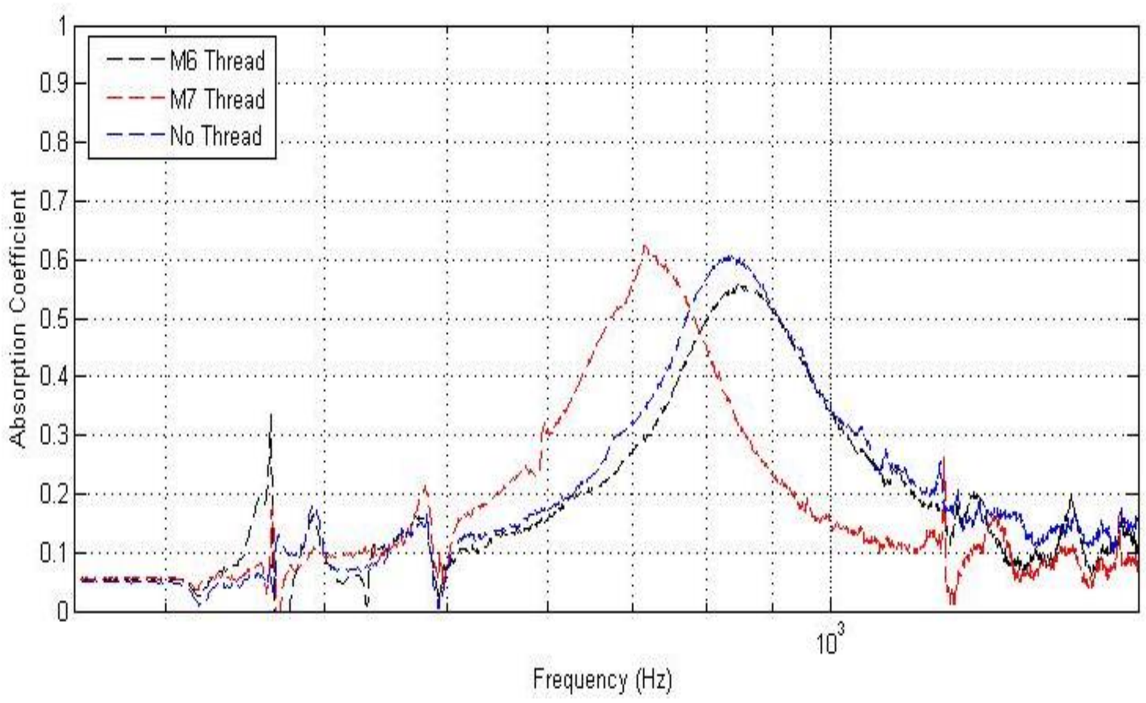


Figure 38. Low-Frequency Sound-Absorption Result Comparison between Threaded and Unthreaded Prototypes for a Cavity Depth of 20 mm.

As seen in **Figure 38**, the thread didn't provide any significant damping —thus only a negligible change in sound absorption— as compared to the unthreaded prototype. In fact, the threaded system behaved very similarly to an unthreaded one with the perforation diameter equal to its minimum diameter. While the idea of increasing the damping of an HR-array system by introducing a thread inside of the perforation itself was an interesting idea, it was found that the effect of threading on the sound absorption of the HR arrays was relatively negligible. Further testing should be considered to see how this effect varies with various types of threads or corrugations inside of the perforations.

4.4.2 Additional Modes of Resonance in a Helmholtz-Resonator Array

4.4.2.1 Panel-Vibration Mode

As seen in **Section 4.4.1**, there are noticeable resonance modes observed in the sample which stay at the same frequencies independent of the changes in the HR-array parameters. This can only be explained by the panel-vibration mode of the system. To observe this mode individually, a repeat measurement was conducted on the same sample as in **Section 4.4.1.1**. However, for this new measurement, all of the air channels in the perforated panel were blocked with a petroleum jelly as seen in **Figure 39,** such that there was no acoustic mode in the system.

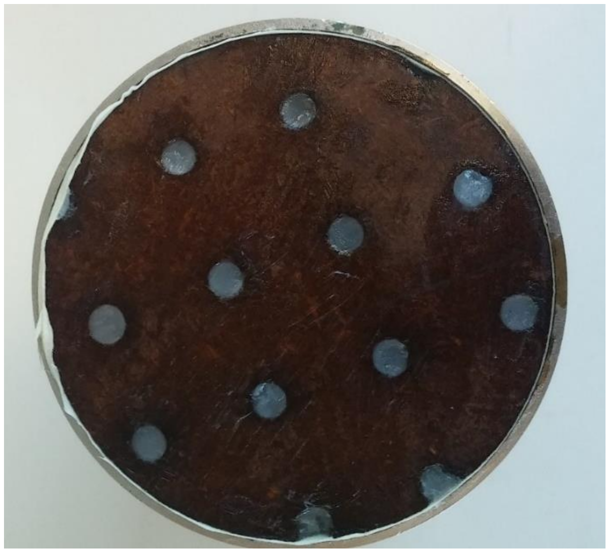


Figure 39. Perforated Panel with Blocked Air Channels.

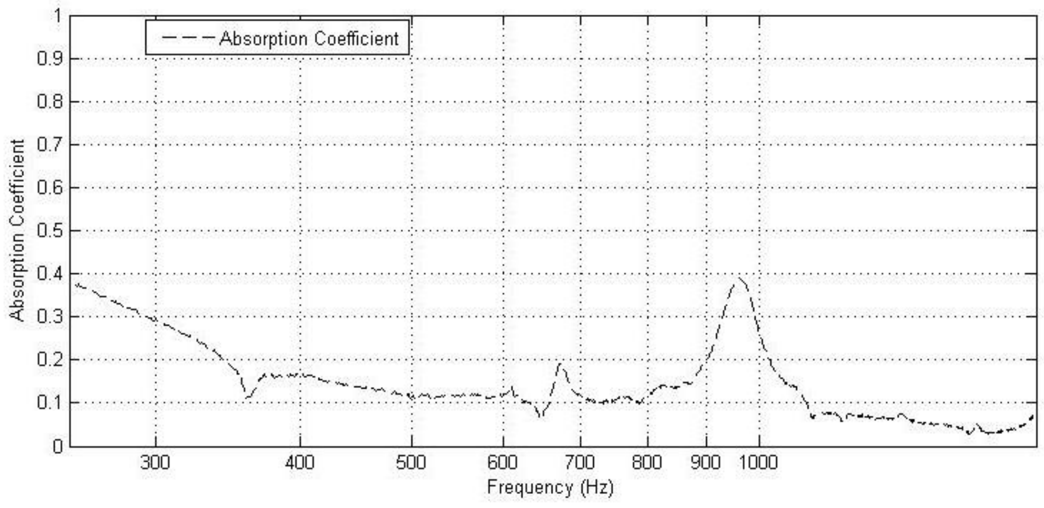


Figure 40. Impedance-Tube Measurement Result for Perforated Plate with Blocked Air Channels.

As seen in **Figure 40**, the panel-vibration mode of the perforated panel can be observed in isolation. As hypothesized previously, this resulting narrow-band sound absorption at slightly below 1000 Hz occurs at the same frequency as observed in **Figure 32**, **Figure 36** and **Figure** 37. While this plate-vibration mode is definitely interesting, and may provide additional sound absorption to an HR array, it is relatively low, and occurs only in a very narrow frequency band, such that it is not really applicable in solving the low absorption of CLT which occurs over a relatively wide frequency range. Furthermore, this mode is completely dominated by the acoustic mode of an HR system with optimized damping, as seen in **Figure 36**. Further research into this vibration mode should be considered if narrow-band sound absorption is required.

4.4.2.2 Air-Vibration Mode inside the Cavity

In **Section 4.4.1.1** to **4.4.1.3**, all of the experiments were conducted in a low-frequency impedance tube, since the resonance frequency caused by the acoustic and panel-vibration modes of the tested systems occurred in the frequency range of that tube (200 Hz to 1750 Hz). However, as seen in both the TMM prediction model and measurement results in **Figure 36** and **Figure 37**, there is a strong indication that an additional mode other than those of the acoustic and panel-vibration modes may have occurred at a frequency above the impedance-tube frequency range.

The additional mode at high frequency may correspond to the air vibration inside of the cavity itself. Looking at the cavity inside of an HR-array system (refer to **Figure 41**), there are two types of boundary conditions — closed-closed and closed-open. Because of these two different boundary conditions, standing waves may occur inside of the system, which results in resonance frequencies corresponding to the multiples of half-wavelength modes for the closed-closed boundary condition (**Figure 42**) and quarter-wavelength mode for the closed-open boundary condition (**Figure 43**).

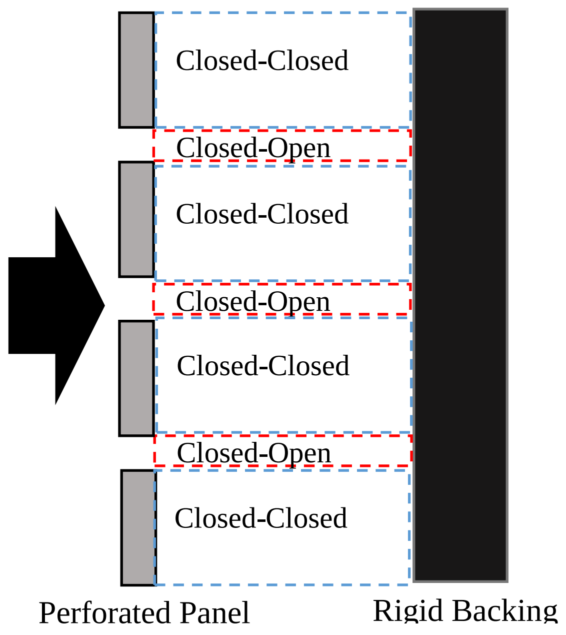


Figure 41. Simplified Diagram of the Closed-Open and Closed-Closed Boundary Conditions.

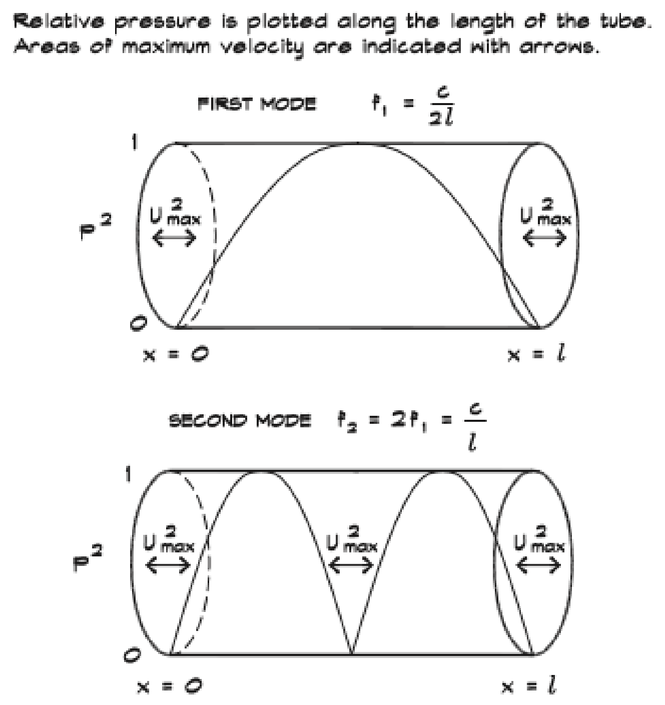

Relative pressure is plotted along the length of the tube.
Areas of maximum velocity are indicated with arrows.
FIRST MODE $f_1 = \frac{c}{2l}$
1
P^2
U^2_{max}
U^2_{max}
0
$x = 0$
$x = l$
SECOND MODE $f_2 = 2f_1 = \frac{c}{l}$
1
P^2
U^2_{max}
U^2_{max}
U^2_{max}
0
$x = 0$
$x = l$

Figure 42. Standing Wave inside of an Air Channel with Closed-Closed Boundary Conditions, as described in [45].

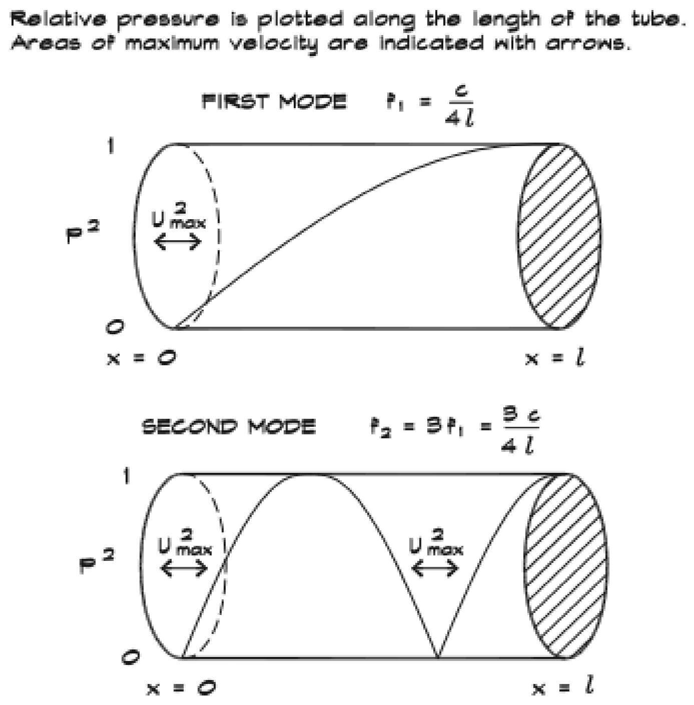

Relative pressure is plotted along the length of the tube.
Areas of maximum velocity are indicated with arrows.
FIRST MODE $f_1 = \frac{c}{4l}$
1
P^2
U^2_{max}
0
$x = 0$
$x = l$
SECOND MODE $f_2 = 3f_1 = \frac{3c}{4l}$
1
P^2
U^2_{max}
U^2_{max}
0
$x = 0$
$x = l$

Figure 43. Standing Wave inside of an Air Channel with Closed-Open Boundary Conditions, as described in [48].

It was of interest to check the quarter-wavelength and half-wavelength modes of an HR array and see their significances to the overall sound-absorption performance. To investigate these two additional modes, further tests should be conducted using the high-frequency impedance tube, which has a frequency range of between 900 to 6000 Hz. A new prototype panel was developed in a pair as seen in **Figure 44** below.

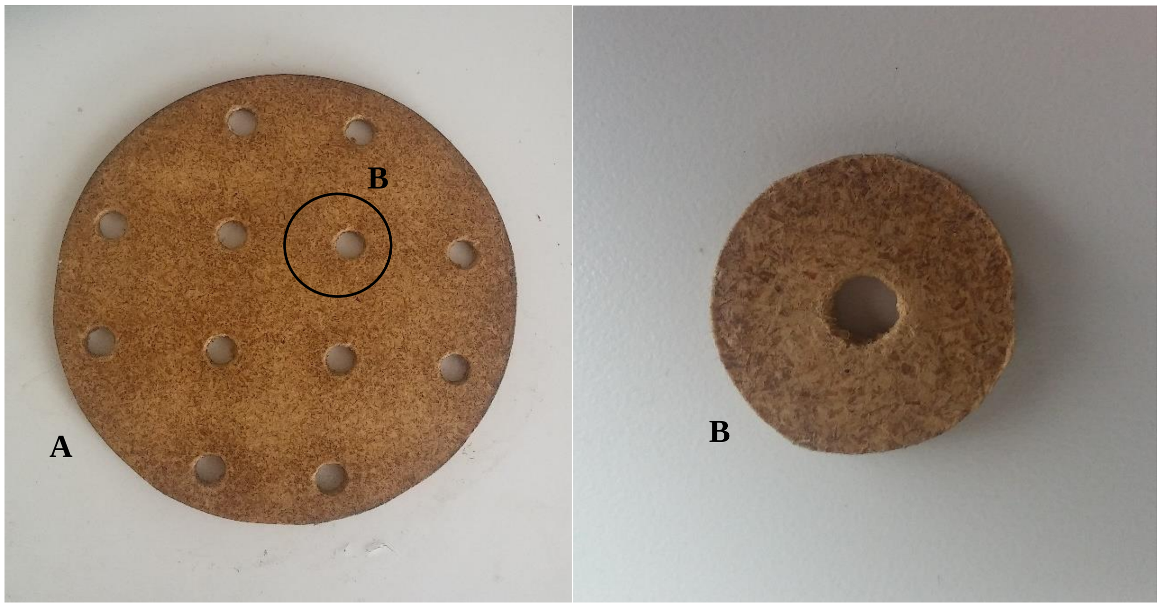


Figure 44. Fourth Preliminary Prototype Pair for Testing A) Low-Frequency Response and B) High-Frequency Response.

For consistency's sake, the cavity depth behind the perforated panels was set to be 2, 4, 6, or 8 cm. These distances corresponded to quarter-wavelength frequencies of 4250, 2125, 1416 and 1062.5 Hz, respectively, and their respective multiples. Furthermore, the half-wavelength frequencies were calculated to be 8500, 4250, 2833, 2125 Hz, respectively, and their respective multiples.

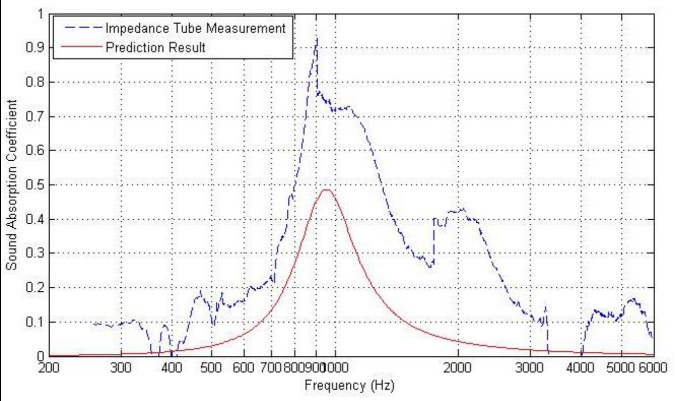
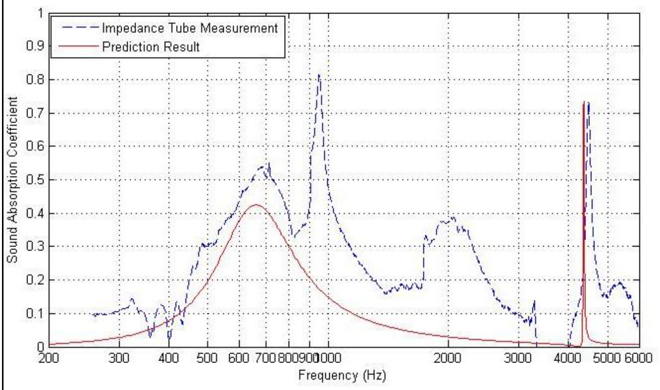
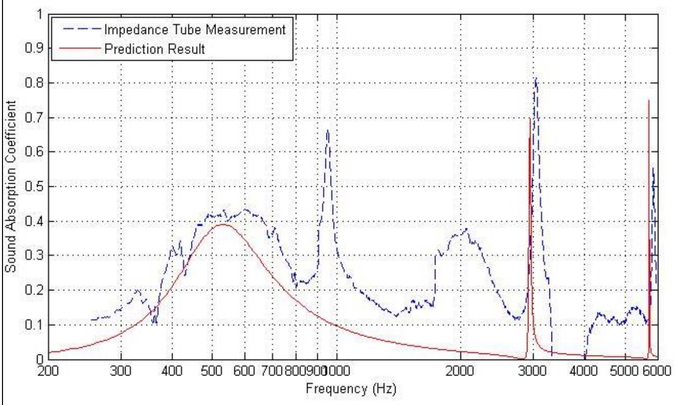
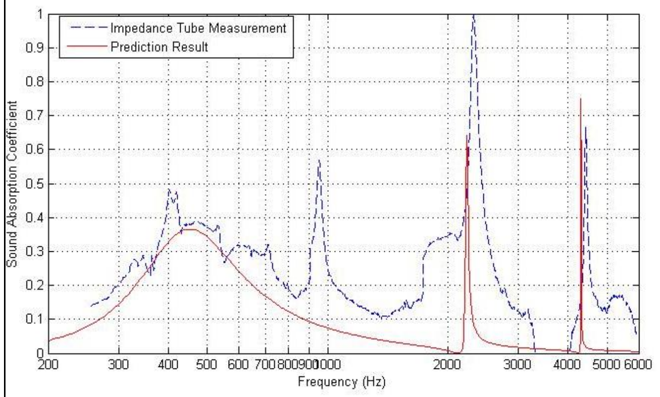
Damping Material in front of Rigid Backing	
 2 cm cavity depth	 4 cm cavity depth
 6 cm cavity depth	 8 cm cavity depth

Figure 45. Low- and High-Frequency Impedance-Tube Measurement Results for Fourth Preliminary Prototype.

As seen in the **Figure 45** above, the impedance-tube measurement results were created by combining the low-frequency result (250-1750 Hz) obtained from prototype A in **Figure 44** and the high-frequency result (1750-6000 Hz) from prototype B in **Figure 44**. There were five different modes expected for each configuration— the acoustic mode, the panel-vibration mode for prototype A, the panel-vibration mode for prototype B, the quarter-wavelength mode, and the half-wavelength mode.

Based on the data shown in **Figure 45**, it can clearly be seen that the panel-vibration mode caused resonances at around 950 and 2000 Hz. Furthermore, this specific mode also caused an anti-resonance at around 3600 Hz. Interestingly, both the prediction and measurement results showed the resonance caused by the half-wavelength mode which corresponded to the air cavity with closed-closed end; however, no resonances caused by the quarter-wavelength mode were observed in either result. This was expected, considering that the perforation ratio of this prototype was very small.

In addition to all of these expected acoustic and vibration modes, there is one unexplained increase in sound absorption occurring at 2000 Hz as seen in **Figure 45.** This slight jump in sound absorption comes from the potential measurement error at the very end of the low-frequency tube's valid frequency range and the very beginning of the valid frequency range of the high-frequency tube.

5 Prototype Development

5.1 Final Prototype Design Considerations

While Cross-Laminated Timber has been proven to have very good sound-insulation and mechanical-strength properties [11], its sound-absorption coefficient is very low, as seen in the in-situ measurement result shown in **Figure 15**. Based on this result, Cross-Laminated Timber was measured to have a diffuse-field sound-absorption coefficient of about 0.08 to 0.13 at low frequency (125 Hz) and about 0.02 to 0.06 at high frequency (8000 Hz), with a significant dip in sound-absorption performance at the middle frequency around 1000 Hz.

In a typical acoustical environment, multiple solutions for increasing the sound-absorption coefficient can generally be based on three main categories of sound absorbers, depending on the frequency of interest: panel sound absorber for low frequencies, resonant sound absorbers for middle frequencies and porous sound absorbers for high frequencies. Based on the in-situ measurement results, a combination of resonant and porous absorber may be required to improve the sound absorption of CLT, depending on the room application. While there is a large selection of commercially-available porous absorbers to choose from, it is slightly more difficult to reduce the sound absorption at middle frequency. This is mostly due to the complexity of designing this type of sound absorber, and the customizability of its sound- absorption characteristics.

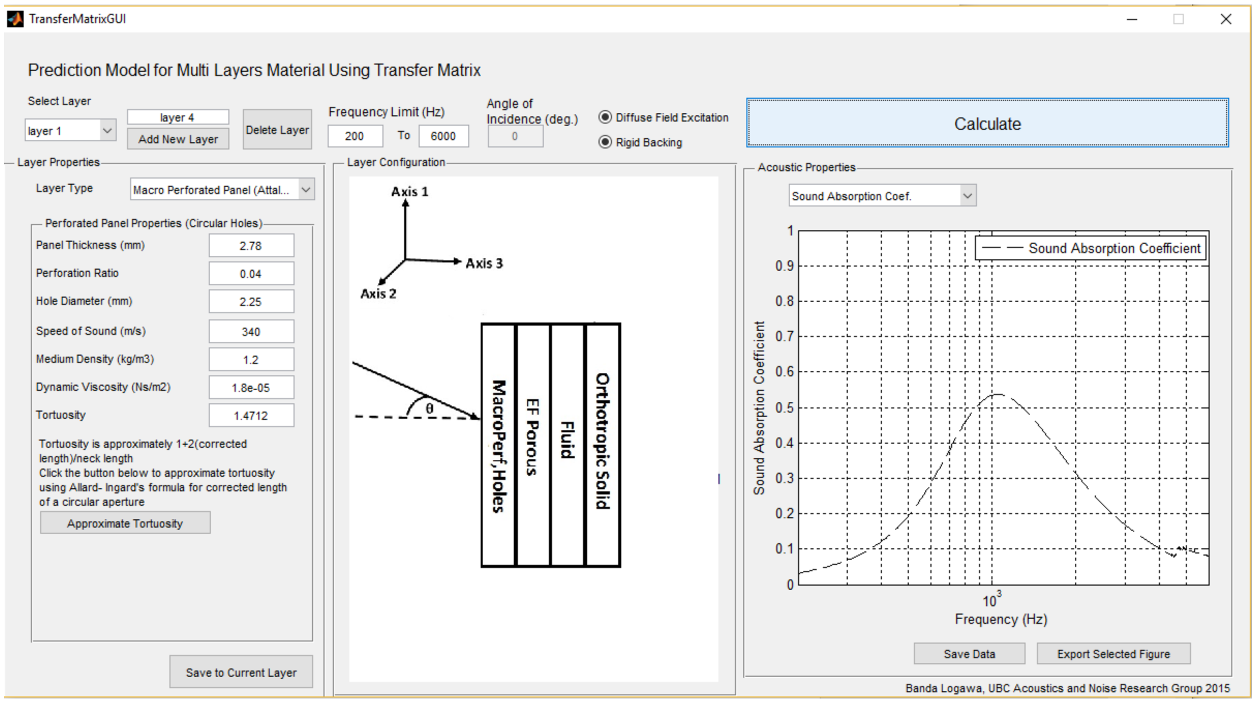
In this study, there are several requirements for the final prototype, which is aimed at improving the sound absorption of CLT for architectural applications, as follows:

- To preserve the mechanical strength and sound-transmission performance of CLT, a sound-absorber layer is added onto the bare CLT panel;

- Diffuse-field sound-absorption-coefficient peaks at around 1000 Hz, with sufficient damping that the sound absorption is broad enough to cover the dip in sound absorption measured in multiple CLT buildings;
- Prototype can be manufactured easily using existing materials and machineries which are utilized for creating CLT;
- Prototype test samples are large enough such that they can be tested, by using all three experimental methods discussed in the previous section: impedance tube, spherical decoupling, reverberation chamber.

5.2 Final Prototype Specifications

Based on the previous study seen in **Section 4.4**, it is known that the sound-absorption characteristics of an HR-array system mainly depend on the perforation size, cavity volume, and damping material. While the resonance frequency of the system mainly depends on the perforation size and cavity volume, the sound absorption itself comes from the thermoviscous losses at the perforations and the damping material inside of the system. Combining all of these factors and the design specification outlined in **Section 5.1**, a final prototype was developed that satisfies all the design requirements, as shown in the predicted diffuse-field sound-absorption performance in **Figure 46**. The diffuse-field sound-absorption condition was chosen, as opposed to just the normal-incidence performance, because it reflected the real-life scenario where sound waves typically hit a surface over a wide range of angles of incidence. The specifications of the final prototype can be seen in **Table 8.**


TransferMatrixGUI

Prediction Model for Multi Layers Material Using Transfer Matrix

Select Layer: layer 1 | layer 4 | Add New Layer | Delete Layer

Frequency Limit (Hz): 200 To 6000

Angle of Incidence (deg.): 0

● Diffuse Field Excitation

● Rigid Backing

Calculate

Layer Properties

Layer Type: Macro Perforated Panel (Attal...

Perforated Panel Properties (Circular Holes)

Panel Thickness (mm)	2.78
Perforation Ratio	0.04
Hole Diameter (mm)	2.25
Speed of Sound (m/s)	340
Medium Density (kg/m3)	1.2
Dynamic Viscosity (Ns/m2)	1.8e-05
Tortuosity	1.4712

Tortuosity is approximately 1+2(corrected length)/neck length
Click the button below to approximate tortuosity using Allard- Ingard's formula for corrected length of a circular aperture

Approximate Tortuosity

Save to Current Layer

Layer Configuration

Acoustic Properties

Sound Absorption Coef.

Save Data | Export Selected Figure

Banda Logawa, UBC Acoustics and Noise Research Group 2015

Figure 46. Predicted Diffuse-Field Sound-Absorption Performance of the Final Prototype using an In-House Transfer Matrix Model (TMM) Prediction Tool.

Table 8. Final Prototype Configurations.

Specifications	**Final Prototype**
Perforated Panel	Perforation Diameter: 2.25 mm Perforation Spacing: 10 mm Panel Thickness: 2.80 mm
Cavity Depth	Spacing Height: 1.50” or 3.81 mm
Damping Material	SoundTex Fabric
Specimen Backing	3-layer KLH Cross-Laminated Timber
Resonance Frequency	1000 Hz

5.2.1 Perforated Panel

According to Maa [4], the ratio between the perforation radius and the viscous boundary-layer thickness is very important in determining the sound-absorption characteristics of a perforated panel. Because of this, a perforated panel with submillimetric perforations, typically called as Microperforated Panel (MPP), can achieve wide-band sound absorption due to its acoustic resistance and low acoustic mass reactance. The perforated panel for the final prototype was created at UBC Centre for Advanced Wood Processing (CAWP) with this aspect in mind. While it wasn’t possible to create submillimetric perforations with the currently available equipment, the final perforation diameter could be minimized to about 2.25 mm by using a Hurricane laser jet cutter. To achieve the desired resonance frequency, the spacing between perforations was 10 mm such that the perforation ratio was about 0.04.

5.2.2 Air Cavity and Damping Material

The cavity depth for the final prototype was chosen based on the width of 2 by 4-dimensional lumber, which was commonly used in the CLT manufacturing processes. In the final prototype (1.5” spacing), a SoundTex Acoustic Nonwoven fabric was chosen as the damping material. This material is commonly used in the industry to provide additional damping to both perforated and micro-perforated panels, and was chosen due to its acoustic performance and size.

5.2.2.1 Specimen Backing

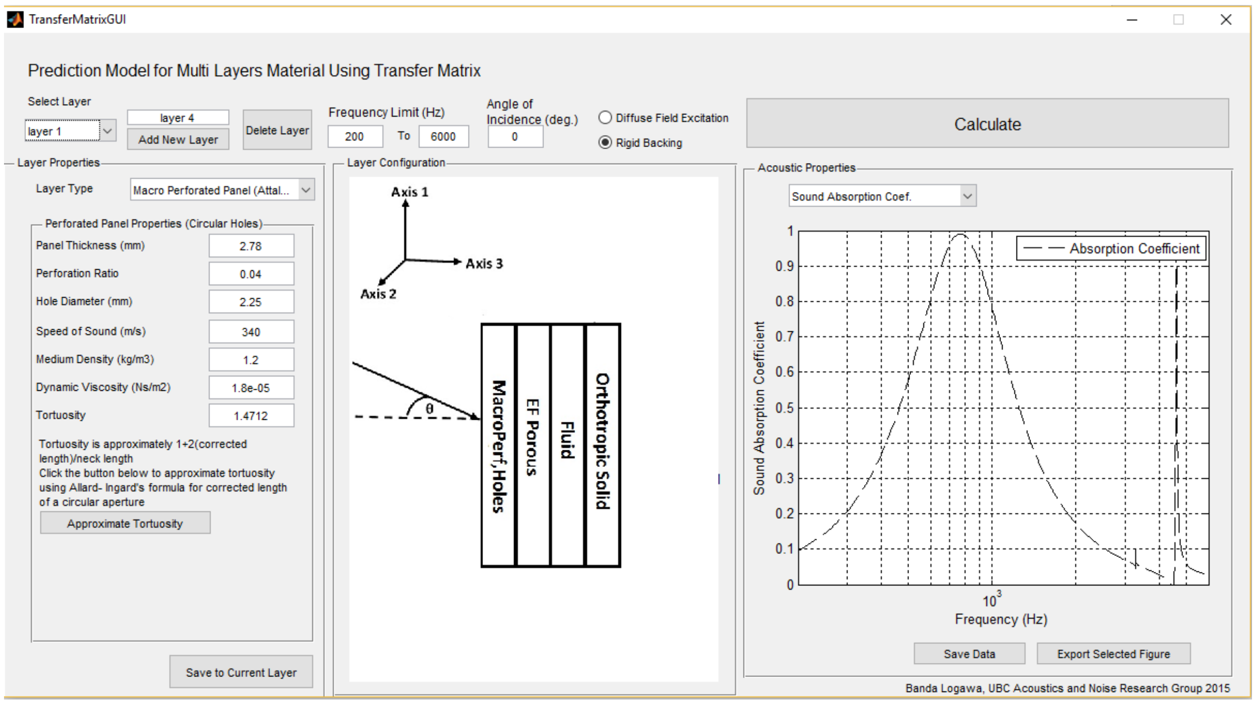
To simulate an actual condition in a CLT building, a 95-mm-thickness 3-layer CLT, manufactured by Kreuss Lagen Holz (KLH), was used as the backing of the HR system in the final prototype. Material properties of the CLT panel used in the TMM model were derived based on the datasheet provided by KLH [52] and previous findings on properties of CLT by Gsell [53].

5.3 Experimental Results

5.3.1 Impedance Tube

Similar to the experimental method outlined in **Section 4.1**, the low- and high-frequency sound-absorption coefficients for the normal-incidence case were measured by using two different impedance tubes which had frequency ranges of 250 to 1750 Hz and 900 to 6000 Hz, respectively. The two results were then combined and compared to the prediction result from TMM.

According to the TMM prediction model (refer to **Figure 47**), for the prototype to possess the desired sound-absorption characteristics, which peak at 1000 Hz frequency, the normal-incidence sound absorption of the system will have a resonance at 800 Hz.


TransferMatrixGUI

Prediction Model for Multi Layers Material Using Transfer Matrix

Select Layer: layer 1

layer 4

Frequency Limit (Hz): 200 To 6000

Angle of Incidence (deg.): 0

Diffuse Field Excitation [✓ Rigid Backing]

Layer Properties

Layer Type: Macro Perforated Panel (Attal...

Perforated Panel Properties (Circular Holes)

Panel Thickness (mm)	2.78
Perforation Ratio	0.04
Hole Diameter (mm)	2.25
Speed of Sound (m/s)	340
Medium Density (kg/m3)	1.2
Dynamic Viscosity (Ns/m2)	1.8e-05
Tortuosity	1.4712

Tortuosity is approximately 1+2(corrected length)/neck length
Click the button below to approximate tortuosity using Allard- Ingard's formula for corrected length of a circular aperture

Acoustic Properties: Sound Absorption Coef.

Banda Logawa, UBC Acoustics and Noise Research Group 2015

Figure 47. Predicted Normal Incidence Sound Absorption Performance of the Final Prototype using an In-House Transfer Matrix Model (TMM) Prediction Tool.

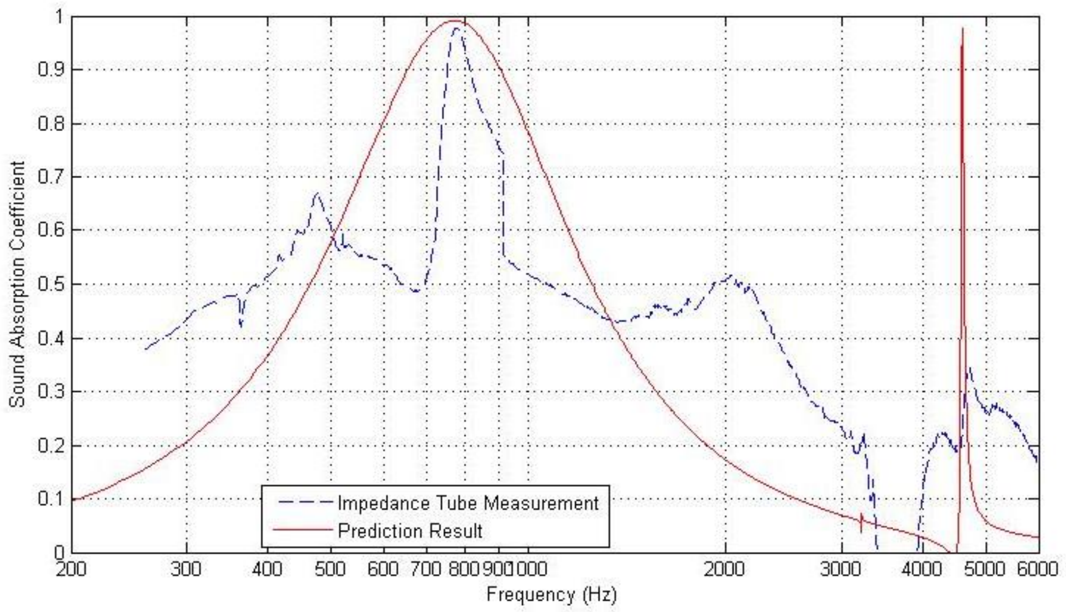


Figure 48. Impedance-Tube Measurement Result.

As seen in **Figure 48**, both the prediction and measurement results show the same resonant frequency for the normal-incidence case. However, the impedance-tube measurement shows a very interesting result. Instead of a smooth curve with steadily decreasing slope towards the peak at the resonance frequency, typically found in HR arrays in the perforated-panel case, there is a sharp jump in sound absorption in a narrow frequency band at around 800 Hz. This seems to be unique to the impedance tube result as it doesn't show up during the later experiments.

5.3.2 Spherical Decoupling

In addition to the normal-incidence sound-absorption cases, the spherical decoupling measurement method is also capable of measuring the sound absorption at various angles of incidence. For the purpose of this research, five different angles of incidence were considered (15°, 30°, 45°, 60°, 75°, and 90°/ normal incidence). By combining all measured angles, the diffuse-field sound absorption can be calculated from **Equation 125.**

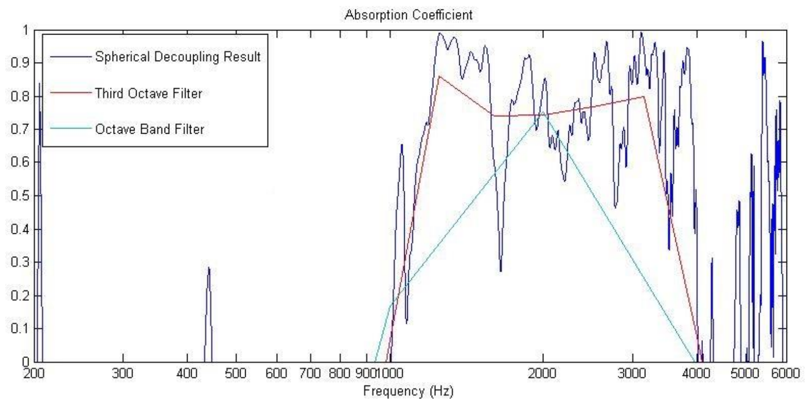


Figure 49. Spherical Decoupling 15 degree from plate surface.

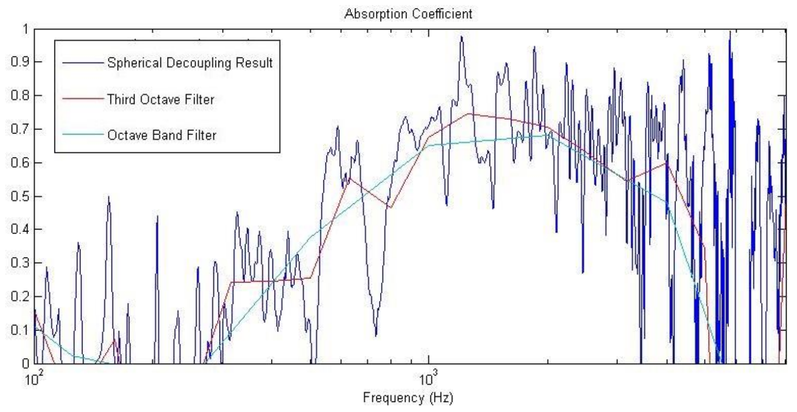


Figure 50. Spherical Decoupling 30 degree from plate surface.

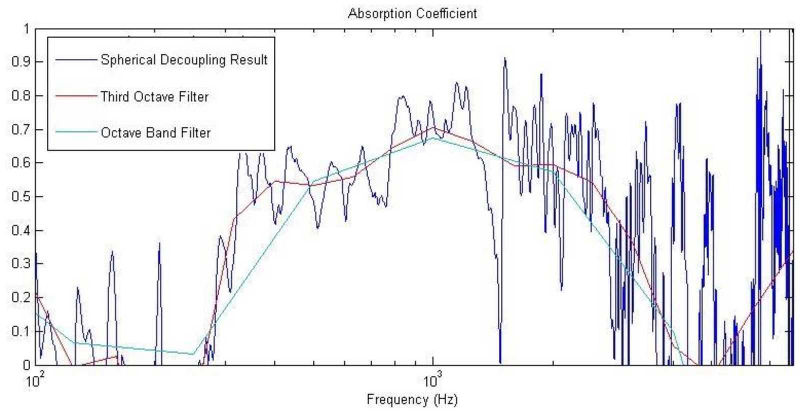


Figure 51. Spherical Decoupling 45 degree from plate surface.

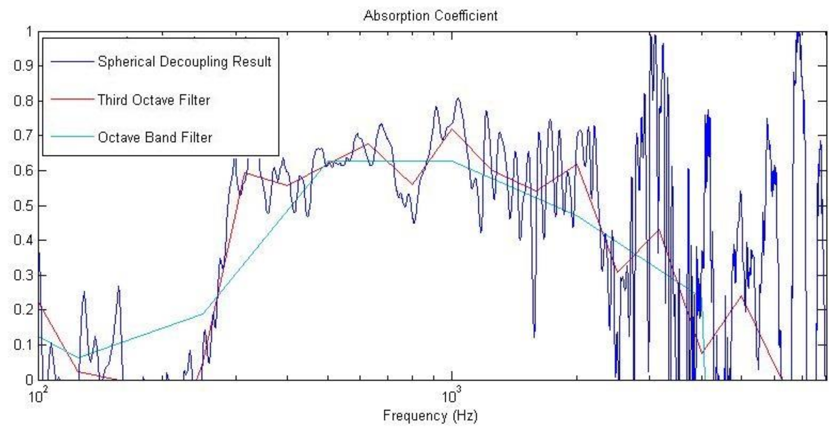


Figure 52. Spherical Decoupling 60 degree from plate surface.

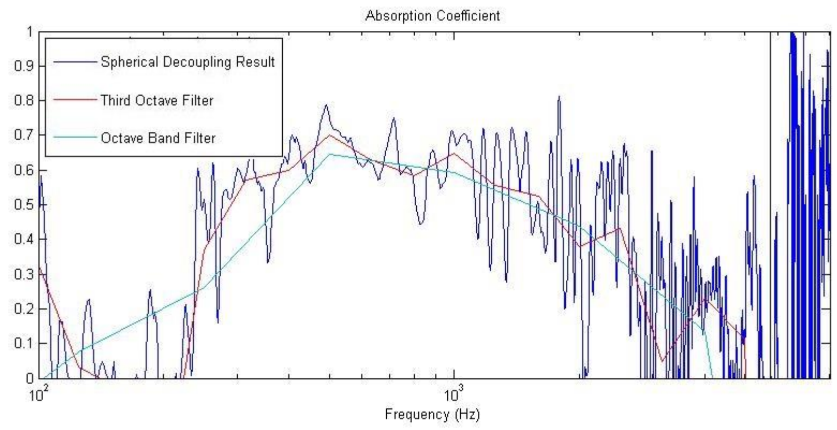


Figure 53. Spherical Decoupling 75 degree from plate surface.

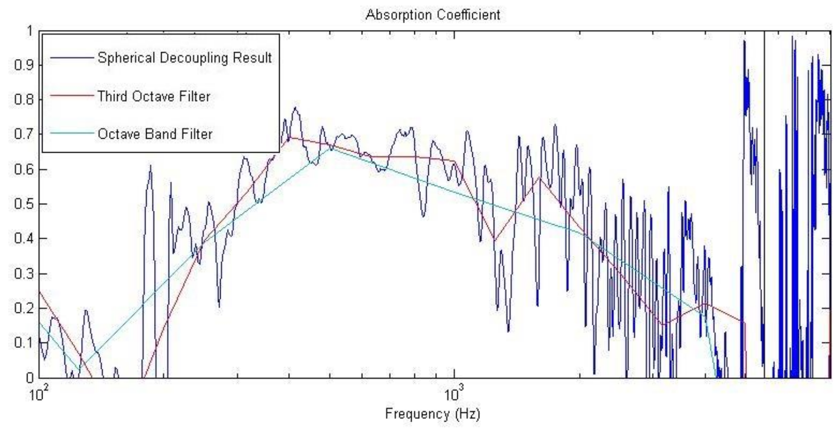


Figure 54. Spherical Decoupling 90 degree from plate surface.

As seen in **Figure 49** to **Figure 54**, while the spherical-decoupling method is capable of measuring the sound-absorption coefficient of the sample at various angle of incidence, there is a considerable amount of noise observed in the measurement results, which is most likely caused by the non-uniform characteristic of a Helmholtz-Resonator array system. In the high-frequency region, in which the wavelength of the sound wave is similar to the spacing between the perforations in the panel, the accuracy of the measurement decreases considerably.

As the angle of incidence of the incoming sound wave approaches the normal-incidence (90 degree to the plate surface), the resonance of the system shifts to lower frequency until it is approximately 700 Hz. This resonance location is considerably lower compared to the one predicted by the Transfer-Matrix Model and measured in the impedance tube. Furthermore, normal-incidence result from spherical decoupling (**Figure 54**) is also considerably different compared to the one in the impedance tube (**Figure 48**). Instead of a sharp peak at resonance frequency, the spherical decoupling result shows a more gradual curve.

5.3.3 Reverberation Chamber

To get the diffuse-field sound-absorption-coefficient, a reverberation-chamber measurement was conducted in VanAir Design sound-transmission facility (refer to Figure **29**). By fully separating the receiver and source rooms in the facility with a rigid partition, the two rooms could be considered as two separate reverberation chambers. Reverberation-chamber measurements, as outlined in **Section 4.3**, were then conducted in both rooms.

As seen in **Figure 55**, the reverberation-chamber measurement results between the two rooms show considerably large differences in magnitude. The peak sound-absorption coefficient of the prototype test sample far exceeded 1 for the measurement in the receiver room. This is somewhat expected, considering the significant difference in the surface area of the two rooms (**Section 4.3.2)**. The large difference in the test-sample surface area to the surface area of the receiver room most likely causes this overestimation in the sound-absorption coefficient.

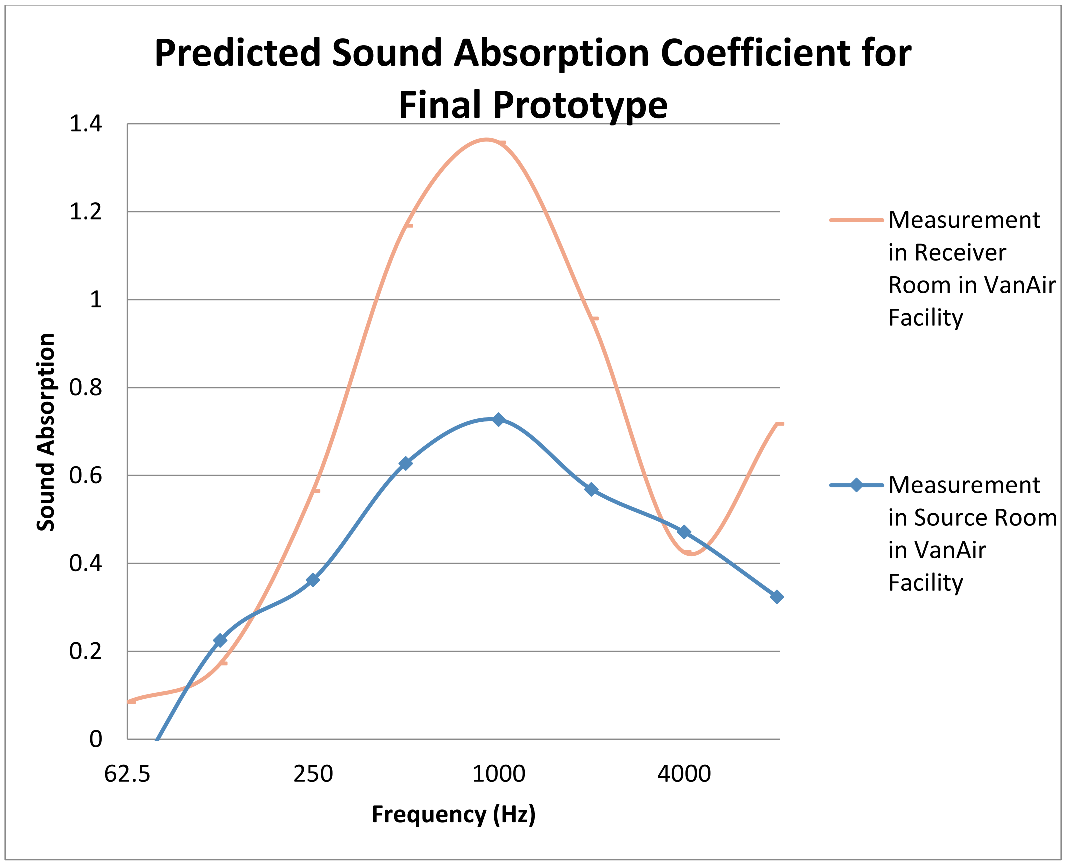


Figure 55. Measured Diffuse Field Sound Absorption Coefficient in VanAir Design Sound Transmission Facility.

As seen in the reverberation-chamber measurement in the source room, the measured diffuse-field sound-absorption coefficient of the final prototype panel has confirmed that the proposed prototype design has the desired diffuse-field characteristics outlined in **Section 5.1**.

6 Conclusion

Introduction of Cross Laminated Timber (CLT) in North America has re-popularized the mid-to-high-rise wood-building construction industry. Due to its cross-lamination process, CLT has relatively high in-plane and out-of-plane strengths and stiffnesses, which make it a great renewable alternative to concrete. Furthermore, it has also been proven to have good structural, vibration, fire, and sound-insulation performance.

Despite the numerous studies on the material properties of CLT, there is a noticeable lack of available information on its sound absorption, which indicates how much sound energy a material can absorb when it's exposed to an incident sound wave. This affects the acoustical conditions in a room— in particular, the reverberation and how effectively speech communication can be conducted. To study these effects in actual buildings, in-situ measurements were conducted in 5 different CLT buildings in British Columbia (8 total rooms) to measure the average CLT surfaces' sound-absorption, and room reverberation time and speech intelligibility. The sound-absorption coefficients of the CLT surfaces were measured to be about 0.02 to 0.13, corresponding to approximately 98.7% to 99.8% of sound being reflected by the room surfaces. The reverberation times of the buildings varied from 0.98 to 2.58 s with the peak in the 1000 Hz frequency band. Speech intelligibility was mostly in the 'fair' category for 1 m and 2 m talker-listener distances, NC-30 background-noise level, and a casual or normal speaker vocal effort.

Based on the performance measured in multiple CLT buildings, it was evident that the sound absorption of CLT panels needed to be improved such that excessive noise build-up and low speech intelligibility inside of rooms with CLT internal surfaces can be alleviated. Due to the characteristics of the measured sound absorption and reverberation time, with an obvious trough or peak respectively at the middle frequencies, a resonant or Helmholtz Resonator (HR) sound-absorber design was considered.

The performance of an HR system depends mainly on several parameters: cavity depth, perforation size and shape, and damping material in the system. By using the transfer-matrix method and separating an HR array system into several inter-connected layers with different properties, the effects of each parameter on the overall sound-absorption performance were identified.

After identifying the important parameters of an HR system and their effects on performance, a final prototype was created with the goal of improving the relatively low sound-absorption of CLT, as well as responding to input from the CLT manufacturers and experts. To verify the resulting sound absorption of the proposed solution, laboratory measurements were conducted. The result confirm that the proposed prototype solution has the required sound-absorption performance, and that the objectives of the research project have been achieved.

Bibliography

[1] K. Nore, J. Aarstad, A. Øvrum, G. Glasø and J. A. Austnes, Massivtreelement Med Akustisk Dempning [Solid Wood Element with Acoustic Attenuation], 2012.

[2] J. Allard and N. Atalla, Propagation of Sound in Porous Media: Modelling Sound Absorbing Materials, Second Edition, Chichester: John Wiley & Sons, 2009.

[3] A. Wareing, "Acoustical Modeling of Rooms with Extended Reaction Surfaces," *MASc. Thesis*, Department of Mechanical Engineering, University of British Columbia1995.

[4] D.-Y. Maa, "Potential of Microperforated Panel Absorber," *The Journal of Acoustical Society of America*, vol. 104, no. 5, pp. 2861-2866, 1998.

[5] N. Attala and F. Sgard, "Modelling of Perforated Plates and Screens Using Rigid Frame Porous Models," *Journal of Sound and Vibration*, vol. 303, pp. 195-208, 2007.

[6] V. Bucur, Acoustics of Wood, Berlin: Springer-Verlag, 2006.

[7] ISO/CD 10534-2: Determination of Sound Absorption Coefficient and Impedance in Impedance Tubes- Part 2: Transfer Function Method, 1998.

[8] L. De Geetere, *Analysis and Improvement of the Experimental Techniques to assess the Acoustical Reflection Properties of Boundary Surfaces (Doctoral Dissertation)*, Katholieke Universiteit Leuven, 2004.

[9] *ANSI S1.18 American National Standard Method for Determining the Acoustic Impedance of Ground Surfaces*, 1999.

[10] ASTM C423-09a: Standard Test Method for Sound Absorption and Sound Absorption Coefficients by the Reverberation Room Method.

[11] S. Gagnon and C. Pirvu, CLT Handbook: Cross-laminated Timber, Québec: FPInnovations, 2011.

[12] E. Karacabeyli and B. Douglas, CLT Handbook: Cross Laminated Timber (US Edition), Pointe-Claire: FPInnovations, 2013.

[13] L. Pagnoncelli, A. Gadott, A. Frattari, E. Bazzini and L. Moran, "Acoustic Performance of Cross- Laminated Timber System (CLT): In Situ Measurements of Airborne and Impact Sound Insulation for Different Configurations," in *40th IAHS World Congress on Housing: Sustainable Housing Construction*, Funchal- Portugal, 2014.

[14] E. Karacabeyli and C. Lum, Technical Guide for the Design and Construction of Tall Wood Buildings in Canada, Pointe-Claire: FPInnovations, 2014.

[15] GHL Consultants Ltd., "A Look at the 18 Storey Tall Wood Building," 2015.

[16] ANSI/APA PRG 320-2012: Standard for Peformance-Rated Cross-Laminated Timber, APA- The Engineered Wood Association.

[17] "ETA-06/0138: European Technical Approval on KLH Solid Wood Slabs," [Online]. Available: http://www.klhuk.com/media/9379/en%20eta-06%200138%20klh%20electronic%20copy.pdf.

[18] M. Pérez and M. Fuente, "Acoustic Design through Predictive Methods in Cross Laminated Timber (CLT) Panel Structures for Buildings," in *Proceedings of Internoise 2013,* Innsbruck- Austria, 2013.

[19] S. Schoenwald, B. Zeitler, F. King and I. Sabourin, "Acoustics- Sound Insulation in Mid-Rise Wood Buildings (Report to Research Consortium for Wood and Wood-Hybrid Mid-Rise Buildings)," National Research Council Canada, Ottawa, 2014.

[20] M. Hodgson, "Acoustical Evaluation of Six ‘Green’ Office Buildings," *Journal of Green Building,* vol. 3, no. 4, pp. 108-118, 2008.

[21] R. J. Orlowski, "The Arrangement of Sound Absorbers for Noise Reduction- Results of Model Experiments at 1:16 Scale," *Noise Control Engineering Journal,* vol. 22, no. 2, pp. 54-60, 1984.

[22] J. S. Bradley, "Reverberation Chamber Measurement of Theatre Chair Absorption," *Journal of Canadian Acoustical Association,* vol. 19, no. 4, 1991.

[23] M. Long, Architectural Acoustics, London: Elsevier Academic Press, 2006.

[24] F. A. Everest and K. Pohlmann, Master Handbook of Acoustics, 5th ed., McGraw-Hill, 2009.

[25] T. Houtgast, H. Steeneken and R. Plomp, "Predicting Speech Intelligibility on Rooms from the Modulation Transfer Function," *Acoustica,* vol. 16, pp. 60-72, 1980.

[26] ANSi S3.5: American National Standard Method for Calculation of the Speech Intelligibility Index, 1997.

[27] "ISO 3382: Acoustics- Measurement of the Reverberation Time of Rooms with Reference to Other Acoustical Parameters," 1997.

[28] "ASTM E2235: Standard Test Method for Determination of Decay Rates for Use in Sound Insulation Test Methods," 2012.

[29] W. C. Sabine, Collected Papers on Acoustics, Cambridge: Harvard University Press, 1922.

[30] American Society of Heating, Refrigerating and Air-Conditioning Engineers, ASHRAE Handbook, 2014.

[31] ANSI S3.5: American National Standard Method for Calculation of the Speech Intelligibility Index, 1997.

[32] L. E. Kinsler, A. R. Frey, A. B. Coppens and J. V. Sanders, Fundamentals of Acoustics, 4th ed., New York: John Wiley & Sons, 2000.

[33] J. W. S. Rayleigh, The Theory of Sound, vol. 2, London: Macmillan, 1894.

[34] U. Ingard, "On the Theory and Design of Acoustic Resonators," *the Journal of the Acoustical Society of America*, vol. 25, no. 6, pp. 1037-1061, 1953.

[35] A. W. Guess, "Calculation of Perforated Plate Liner Parameters from Specified Acoustic Resistance and Reactance," *Journal of Sound and Vibration*, vol. 40, no. 1, pp. 119-137, 1975.

[36] L. Cremer and H. A. Muller, Principles and Applications of Room Acoustics, Applied Science Publishers, 1978, p. 187.

[37] R. T. Randeberg, "Perforated Panel Absorbers with Viscous Energy Dissipation Enhanced by Orifice Design," 2000.

[38] D. Folds and C. Loggins, "Transmission and Reflection of Ultrasonic Waves in Layered Media," *the Journal of the Acoustical Society of America*, vol. 62, pp. 1102-1109, 1977.

[39] M. Delaney and E. Bazley, "Acoustical Properties of Fibrous Absorbent Materials," *Applied Acoustics*, vol. 3, no. 2, pp. 105-116, April 1970.

[40] Y. Miki, "Acoustical Properties of Porous Materials – Modifications of Delany-Bazley Models," *Journal of the Acoustical Society of Japan*, vol. 11, pp. 19-28, 1990.

[41] T. Komatsu, "Improvement of the Delaney-Bazley and Miki Models for Fibrous Sound-Absorbing Materials," *Acosutical Science and Technology* , vol. 29, no. 2, pp. 121-129, 2008.

[42] K. Attenborough, "Acoustical Characteristics of Rigid Fibrous Absorbents and Granular Materials," *Journal of Acoustical Society of America,* vol. 73, no. 3, pp. 785-799, 1983.

[43] D. L. Johnson and R. D. Joel Koplik, "Theory of Dynamic Permeability and Turtuosity in Fluid- Saturated Porous Media," *Journal of Fluid Mechanics,* vol. 176, no. 1, pp. 379-402, 1987.

[44] J.-F. Allard and Y. Champoux, "Dynamic Turtuosity and Bulk Modulus in Air-Saturated Porous Media," *Journal of Applied Physics,* vol. 70, no. 4, pp. 1975-1979, 1991.

[45] T. J. Cox and D. Peter, Acoustic Absorbers and Diffusers, 2nd ed., New York: Taylor & Francis, 2009.

[46] ASTM C384-04: Standard Test Method for Impedance and Absorption of Acoustical Materials by Impedance Tube Method, 2004.

[47] ASTM E1050-12: Standard Test Method for Impedance and Absorption of Acoustical Materials Using a Tube, Two Microphones and a Digital Frequency Analysis System, 2012.

[48] L. L. Beranek, Acoustic Measurement, New York: J. Wiley, 1949, pp. 72-73.

[49] P. M. Morse, Vibration and Sound, Acoustical Society of America, 1936.

[50] ASTM C423-17: Standard Test Method for Sound Absorption and Sound Absorption Coefficients by the Reverberation Room Method, 2017.

[51] E. Moosavimehr, "Characterization of the VanAir Design Sound Transmission Testing Facility," 2014.

[52] KLH Massivholz GmbH, "Engineering," 2008.

[53] D. Gsell, G. Feltrin, S. Schubert, R. Steiger and M. Motavalli, "Cross-Laminated Timber Plates: Evaluation and Verification of Homogenized Elastic Properties," *Journal of Structural Engineering,* vol. 133, no. 1, pp. 132-138, 2007.

[54] O. Doutres, Y. Salissou, N. Atalla and R. Panneton, "Evaluation of the Acoustic and Non-acoustic Properties of Sound Absorbing Materials Using a Three-Microphone Impedance Tube," *Applied Acoustics,* vol. 71, no. 6, pp. 506-509, 2010.

[55] M. Delaney and E. Bazley, "Acoustical properties of fibrous absorbent materials," *Applied Acoustics,* vol. 24, no. 1, pp. 63-70, 1988.

[56] M. D. Egan, Architectural Acoustics, New York: MCGraw-Hill, 1988.

[57] A. Warnock, "Specifying Acoustical Citeria for Building," *Construction Technology Update No. 50,* June 2001.

Appendices

Appendix A: Transfer Matrix Model GUI— Matlab Code

```
function varargout = TransferMatrixGUI(varargin)
% TRANSFERMATRIXGUI MATLAB code for TransferMatrixGUI.fig
%      TRANSFERMATRIXGUI, by itself, creates a new TRANSFERMATRIXGUI or raises 
the existing
%      singleton*.
%
%      H = TRANSFERMATRIXGUI returns the handle to a new TRANSFERMATRIXGUI or 
the handle to
%      the existing singleton*.
%
%      TRANSFERMATRIXGUI('CALLBACK',hObject,eventData,handles,...) calls the 
local
%      function named CALLBACK in TRANSFERMATRIXGUI.M with the given input 
arguments.
%
%      TRANSFERMATRIXGUI('Property','Value',...) creates a new 
TRANSFERMATRIXGUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before TransferMatrixGUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to TransferMatrixGUI_OpeningFcn via 
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help TransferMatrixGUI

% Last Modified by GUIDE v2.5 06-Feb-2017 11:23:11

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @TransferMatrixGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @TransferMatrixGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before TransferMatrixGUI is made visible.
function TransferMatrixGUI_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to TransferMatrixGUI (see VARARGIN)

% Choose default command line output for TransferMatrixGUI
handles.output = hObject;
axes(handles.configfigure);
imshow('LayerBackground.jpg');
% Update handles structure
guidata(hObject, handles);
clc;
clear all;

global layerData;
layerData= NaN(10,15);
global propertyData;
propertyData= NaN(10,5);
global globalCounter
globalCounter=1;
global tmData;
tmData=cell(1,1);
global intData;
intData=cell(1,1);
global layerNumber;
layerNumber=0;


% UIWAIT makes TransferMatrixGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = TransferMatrixGUI_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



% --- Executes on button press in calculatebutton.
function calculatebutton_Callback(hObject, eventdata, handles)
% hObject    handle to calculatebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global layerData;
global tmData;
global intData;
global layerNumber;
global propertyData;

h = waitbar(0,'Setting Up Layer Data');

layerProp=layerData;
minFreq= str2num(get(handles.minFreqEdit,'String'));
maxFreq= str2num(get(handles.maxFreqEdit,'String'));
f=linspace(minFreq,maxFreq,1000);

Zc=413; %assuming air at 20degC

%Initializing Cell Data
tmFreq=cell(length(f),length(layerProp(:,1)));
inter=cell(layerNumber,2);

%Creating interface matrices
numofInter=layerNumber;
for counter=1:numofInter
    if counter==1
        leftProp=1;
        rightProp=layerProp(1,:);
    else
        leftProp=layerProp(counter-1,:);
        rightProp=layerProp(counter,:);
    end
    [I,J] = intmatrix(leftProp,rightProp);
    inter{counter,1}=I;
    inter{counter,2}=J;
end
intData=inter;

%Specifying termination condition
if (get(handles.RBbutton,'Value') == get(handles.RBbutton,'Max'))
    rb=1; %rigid backing termination
else
    rb=0; %semi-infinite fluid backing
end
waitbar(0.25,h,'Creating Transfer Matrix for Each Layer (May Take a While)');
%Creating transfer matrix and Dmatrix of each individual layer for each
frequency
if(get(handles.DFbutton,'Value')==get(handles.DFbutton,'Max'))
    angles=linspace(0,pi()/2,100);

    %Initilizing Property Data
    R=zeros(length(f),length(angles));
    Z=zeros(length(f),length(angles));
    T=zeros(length(f),length(angles));
    TL=zeros(length(f),length(angles));
    for anglesCounter=1:length(angles)
        aofi=angles(anglesCounter);
        for freqCounter=1:length(f)
            omega=2*pi()*f(freqCounter);

            for layerCounter = 1:length(layerProp(:,1))
                %Checking if there is maximum layer number is exceeded
                if isnan((layerProp(layerCounter,1)))
                    break
                end
                matProp=layerProp(layerCounter,:);
                Tm=tmatrix(matProp,aofi,omega);
                tmFreq{freqCounter,layerCounter}=Tm;
            end
            temp=tmFreq(freqCounter,1:layerCounter);
            D=dmatrix(layerProp,temp,inter,rb,layerNumber,Zc,aofi);
            [Ref,Imp,Trans,TransLoss]=calcprop(D,Zc,aofi,rb);
            R(freqCounter,anglesCounter)=Ref;
            Z(freqCounter,anglesCounter)=Imp;
            T(freqCounter,anglesCounter)=Trans;
            TL(freqCounter,anglesCounter)=TransLoss;
        end
        tmData=tmFreq;
    end
    %calculating diffuse field values
    waitbar(0.5,h,'Calculating Diffuse Field Acoustic Properties');
    denumerator=((cos(angles)).*(sin(angles))).';
    alpha=1-(abs(R).^2);
    Rcoef=abs(R).^2;
    Tcoef=abs(T).^2;
    alphad=sum(alpha*denumerator,2)./sum(denumerator);
    Rd=sum(Rcoef*denumerator,2)./sum(denumerator);
    Zd=sum(Z*denumerator,2)./sum(denumerator);
    Td=sum(Tcoef*denumerator,2)./sum(denumerator);
    TLd=sum(TL*denumerator,2)/sum(denumerator);
    propertyData=[f.',alphad,Rd,Zd,Td,TLd];

else
    %Initilizing Property Data
    R=zeros(length(f),1);
    Z=zeros(length(f),1);
    T=zeros(length(f),1);
    TL=zeros(length(f),1);

    aofi= degtorad(str2num(get(handles.angleEdit,'String')));
    for freqCounter=1:length(f)
        omega=2*pi()*f(freqCounter);

        for layerCounter = 1:length(layerProp(:,1))
            %Checking if there is maximum layer number is exceeded
            if isnan((layerProp(layerCounter,1)))
                break
            end
            matProp=layerProp(layerCounter,:);
            Tm=tmatrix(matProp,aofi,omega);
            tmFreq{freqCounter,layerCounter}=Tm;
        end
        temp=tmFreq(freqCounter,1:layerCounter);
        D=dmatrix(layerProp,temp,inter,rb,layerNumber,Zc,aofi);
        [Ref,Imp,Trans,TransLoss]=calcprop(D,Zc,aofi,rb);
        R(freqCounter)=Ref;
        Z(freqCounter)=Imp;
        T(freqCounter)=Trans;
        TL(freqCounter)=TransLoss;
    end
    waitbar(0.5,h,'Calculating Acoustic Properties');
    tmData=tmFreq;
    alpha=1-(abs(R)).^2;
    Rcoef=(abs(R)).^2;
    Tcoef=(abs(T)).^2;
    propertyData=[f.',alpha,Rcoef,Z,Tcoef,TL];
end

%Create corresponding workspace variables
assignin('base','layerData',layerData);
assignin('base','intData',intData);
assignin('base','tmData',tmData);
assignin('base','propertyData',propertyData);

%Plotting the graph in propertiesfigure
waitbar(0.75,h,'Plotting Figures');
axes(handles.propertiesfigure);
figureNumber=get(handles.propertiespopup,'value');
fmin=min(f);
fmax=max(f);
switch figureNumber
   case 1
        semilogx(propertyData(:,1),real(propertyData(:,4)),'k--');
        legend('Surface Impedance (Rayl, Real)')
        xlabel('Frequency (Hz)');
        ylabel('Surface Impedance (Rayl, Real)');
        grid on
    case 2
        semilogx(propertyData(:,1),imag(propertyData(:,4)),'k--');
        legend('Surface Impedance (Rayl, Imag)')
        xlabel('Frequency (Hz)');
        ylabel('Surface Impedance (Rayl, Imag)');
        grid on
    case 3
        semilogx(propertyData(:,1),(propertyData(:,3)),'k--');
        legend('Reflection Coefficient')
        axis([fmin fmax -0 1]);
        xlabel('Frequency (Hz)');
        ylabel('Sound Reflection Coefficient');
        grid on
    case 4
        semilogx(propertyData(:,1),propertyData(:,2),'k--');
        legend('Sound Absorption Coefficient')
        axis([fmin fmax -0 1]);
        xlabel('Frequency (Hz)');
        ylabel('Sound Absorption Coefficient');
        grid on
    case 5
        semilogx(propertyData(:,1),(propertyData(:,5)),'k--');
        legend('Sound Transmission Coefficient')
        axis([fmin fmax -0 1]);
        xlabel('Frequency (Hz)');
        ylabel('Sound Transmission Coefficient');
        grid on
    case 6
        semilogx(propertyData(:,1),propertyData(:,6),'k--');
        legend('Transmission Loss')
        xlabel('Frequency (Hz)');
        ylabel('Transmission Loss (dB)');
        grid on
end
waitbar(1,h,'Done');
close(h);
% --- Executes on button press in addbutton.
function addbutton_Callback(hObject, eventdata, handles)
% hObject    handle to addbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global layerNumber;

%For Popup Menu Only
%old_str=get(handles.popupmenu2,'String');
%temp=cellstr(get(handles.popupmenu2,'String'));
%new_str=char(old_str, strcat('Layer',num2str(length(temp)+1)));
%set(handles.popupmenu2,'String',new_str)

%For listbox; code taken from listboxexample http://www.mathworks.com/
...matlabcentral/answers/94191-how-can-i-manipulate-the-entries-of-a-
    ...listbox-or-popup-menu-in-matlab

entries = get(handles.layerpopup,'String');
value   = get(handles.layerpopup,'Value');
new_str = get(handles.newedit,'String');

% Add the new entry
% Notice that we have to put it into a cell to concatenate it with the
existing entries
if isempty(entries)
    entries=char(new_str);
else
    entries = char(entries, new_str);
end

% Update the listbox
set(handles.layerpopup,'Value',value,'String',entries)
layerNumber=layerNumber+1; %updating layer number


% --- Executes on button press in deletebutton.
function deletebutton_Callback(hObject, eventdata, handles)
% hObject    handle to deletebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Codes below were modified from listboxexample http://www.mathworks.com/
...matlabcentral/answers/94191-how-can-i-manipulate-the-entries-of-a-listbox
    ...-or-popup-menu-in-matlab

global layerNumber;
entries = cellstr(get(handles.layerpopup,'String'));
value   = get(handles.layerpopup,'Value');
nentries = length(entries);
if nentries>1
    % Setting the cell to [] will remove it from the cell array
    entries(value) = [];
    nentries = length(entries);

    % If we removed the last one, decrement the value
    if value > nentries
        value = value-1;
    end

    % Update the listbox
    set(handles.layerpopup,'Value',value,'String',char(entries))
    layerNumber=layerNumber-1; %updating layer number
end

% --- Executes on button press in exportfigurebutton.
function exportfigurebutton_Callback(hObject, eventdata, handles)
% hObject    handle to exportfigurebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%figure(1);
Fig2 = figure;
%set(handles.figure1, 'CurrentAxes', handles.propertiesfigure);
copyobj(handles.propertiesfigure, Fig2);


% --- Executes on button press in savebutton.
function savebutton_Callback(hObject, eventdata, handles)
% hObject    handle to savebutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global layerData;
global layerNumber;
layerType=get(handles.typepopup,'Value');
layerNum=get(handles.layerpopup,'Value');
prop=NaN(1,15);
switch layerType
    case 1
        prop(1,1)=layerType;
        prop(1,2)=str2num(get(handles.fluid1,'String'));
        prop(1,3)=str2num(get(handles.fluid2,'String'));
        prop(1,4)=str2num(get(handles.fluid3,'String'));
```

```
case 2
    prop(1,1)=layerType;
    prop(1,2)=str2num(get(handles.solid1,'String'));
    prop(1,3)=str2num(get(handles.solid2,'String'));
    prop(1,4)=str2num(get(handles.solid3,'String'));
    prop(1,5)=str2num(get(handles.solid4,'String'));
case 3
    prop(1,1)=layerType;
    prop(1,2)=str2num(get(handles.ortho1,'String'));
    prop(1,3)=str2num(get(handles.ortho2,'String'));
    prop(1,4)=str2num(get(handles.ortho3,'String'));
    prop(1,5)=str2num(get(handles.ortho4,'String'));
    prop(1,6)=str2num(get(handles.ortho5,'String'));
    prop(1,7)=str2num(get(handles.ortho6,'String'));
    prop(1,8)=str2num(get(handles.ortho7,'String'));
    prop(1,9)=str2num(get(handles.ortho8,'String'));
    prop(1,10)=str2num(get(handles.ortho9,'String'));
    prop(1,11)=str2num(get(handles.ortho10,'String'));
    prop(1,12)=str2num(get(handles.ortho11,'String'));
case 4
    prop(1,1)=layerType;
    prop(1,2)=str2num(get(handles.efporous1,'String'));
    prop(1,3)=str2num(get(handles.efporous2,'String'));
    prop(1,4)=str2num(get(handles.efporous3,'String'));
    prop(1,5)=str2num(get(handles.efporous4,'String'));
    prop(1,6)=str2num(get(handles.efporous5,'String'));
    prop(1,7)=str2num(get(handles.efporous6,'String'));
    prop(1,8)=str2num(get(handles.efporous7,'String'));
    prop(1,9)=str2num(get(handles.efporous8,'String'));
    prop(1,10)=str2num(get(handles.efporous9,'String'));
    prop(1,11)=str2num(get(handles.efporous10,'String'));
    prop(1,12)=str2num(get(handles.efporous11,'String'));
    prop(1,13)=str2num(get(handles.efporous12,'String'));
    prop(1,14)=str2num(get(handles.efporous13,'String'));
    prop(1,15)=str2num(get(handles.efporous14,'String'));
case 5
    prop(1,1)=layerType;
    prop(1,2)=str2num(get(handles.perf1,'String'));
    prop(1,3)=str2num(get(handles.perf2,'String'));
    prop(1,4)=str2num(get(handles.perf3,'String'));
    prop(1,5)=str2num(get(handles.perf4,'String'));
    prop(1,6)=str2num(get(handles.perf5,'String'));
    prop(1,7)=str2num(get(handles.perf6,'String'));
    prop(1,8)=str2num(get(handles.perf7,'String'));

case 6
    prop(1,1)=layerType;
    prop(1,2)=str2num(get(handles.microperf1,'String'));
    prop(1,3)=str2num(get(handles.microperf2,'String'));
    prop(1,4)=str2num(get(handles.microperf3,'String'));
    prop(1,5)=str2num(get(handles.microperf4,'String'));
    prop(1,6)=str2num(get(handles.microperf5,'String'));
    prop(1,7)=str2num(get(handles.microperf6,'String'));
    prop(1,8)=str2num(get(handles.microperf7,'String'));

case 7
```

```
        prop(1,1)=layerType;
        prop(1,2)=str2num(get(handles.slots1,'String'));
        prop(1,3)=str2num(get(handles.slots2,'String'));
        prop(1,4)=str2num(get(handles.slots3,'String'));
        prop(1,5)=str2num(get(handles.slots4,'String'));
        prop(1,6)=str2num(get(handles.slots5,'String'));
        prop(1,7)=str2num(get(handles.slots6,'String'));
        prop(1,8)=str2num(get(handles.slots7,'String'));
        prop(1,9)=str2num(get(handles.slots8,'String'));

    case 8
        prop(1,1)=layerType;
        prop(1,2)=str2num(get(handles.simpPorous1,'String'));
        prop(1,3)=str2num(get(handles.simpPorous2,'String'));

end

layerData(layerNum,:)=prop(1,:);

%Updating layer configuration figure
switch layerNum
    case 1
        axes(handles.axes3);
        switch layerType
            case 1
                imshow('Fluid.jpg');
            case 2
                imshow('IsoSolid.jpg');
            case 3
                imshow('OrthoSolid.jpg');
            case 4
                imshow('EFPorous.jpg');
            case 5
                imshow('MacroPerf,Holes.jpg');
            case 6
                imshow('MicroPerf,Holes.jpg');
            case 7
                imshow('MacroPerf,Slots.jpg');
            case 8
                imshow('EFPorous.jpg');
        end
    case 2
        axes(handles.axes4);
        switch layerType
            case 1
                imshow('Fluid.jpg');
            case 2
                imshow('IsoSolid.jpg');
            case 3
                imshow('OrthoSolid.jpg');
            case 4
                imshow('EFPorous.jpg');
            case 5
                imshow('MacroPerf,Holes.jpg');
            case 6
                imshow('MicroPerf,Holes.jpg');
            case 7
                imshow('MacroPerf,Slots.jpg');
            case 8
                imshow('EFPorous.jpg');
        end
    case 3
        axes(handles.axes5);
        switch layerType
            case 1
                imshow('Fluid.jpg');
            case 2
                imshow('IsoSolid.jpg');
            case 3
                imshow('OrthoSolid.jpg');
            case 4
                imshow('EFPorous.jpg');
            case 5
                imshow('MacroPerf,Holes.jpg');
            case 6
                imshow('MicroPerf,Holes.jpg');
            case 7
                imshow('MacroPerf,Slots.jpg');
            case 8
                imshow('EFPorous.jpg');
        end
    case 4
        axes(handles.axes6);
        switch layerType
            case 1
                imshow('Fluid.jpg');
            case 2
                imshow('IsoSolid.jpg');
            case 3
                imshow('OrthoSolid.jpg');
            case 4
                imshow('EFPorous.jpg');
            case 5
                imshow('MacroPerf,Holes.jpg');
            case 6
                imshow('MicroPerf,Holes.jpg');
            case 7
                imshow('MacroPerf,Slots.jpg');
            case 8
                imshow('EFPorous.jpg');
        end
    case 5
        axes(handles.axes7);
        switch layerType
            case 1
                imshow('Fluid.jpg');
            case 2
                imshow('IsoSolid.jpg');
            case 3
                imshow('OrthoSolid.jpg');
            case 4
                imshow('EFPorous.jpg');
            case 5
```

```
                imshow('MacroPerf,Holes.jpg');
            case 6
                imshow('MicroPerf,Holes.jpg');
            case 7
                imshow('MacroPerf,Slots.jpg');
            case 8
                imshow('EFPorous.jpg');
        end
    case 6
         axes(handles.axes8);
         imshow('Extra.jpg');
end







% --- Executes on button press in savedatabutton.
function savedatabutton_Callback(hObject, eventdata, handles)
% hObject    handle to savedatabutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global propertyData;
global layerData;
uisave({'layerData','propertyData'});

%informationUnit={'°C';'m';'';'Hz';'Hz'};
%[FileName,PathName] = uiputfile('*.xls','Save File Name');
%dataLocation=fullfile(PathName,FileName);
%colHeader1={'Frequency','Absorption Coefficient','Surface
Impedance','Transmission Coefficient','Transmission Loss'}; %Column headers
%colHeader2={'Hz','','','','dB'}; %Column headers
%xlswrite(dataLocation,colHeader1,'Sheet1','A1'); %Write the column header to
spreadsheet
%xlswrite(dataLocation,colHeader2,'Sheet1','A2'); %Write the column header to
spreadsheet
%xlswrite(dataLocation,propertyData,'Sheet1','A3'); %Write the column header
to spreadsheet





function fluid1_Callback(hObject, eventdata, handles)
% hObject    handle to fluid1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fluid1 as text
%        str2double(get(hObject,'String')) returns contents of fluid1 as a
double



% --- Executes during object creation, after setting all properties.
function fluid1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to fluid1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function fluid2_Callback(hObject, eventdata, handles)
% hObject    handle to fluid2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fluid2 as text
%        str2double(get(hObject,'String')) returns contents of fluid2 as a
double


% --- Executes during object creation, after setting all properties.
function fluid2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fluid2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function fluid3_Callback(hObject, eventdata, handles)
% hObject    handle to fluid3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of fluid3 as text
%        str2double(get(hObject,'String')) returns contents of fluid3 as a
double


% --- Executes during object creation, after setting all properties.
function fluid3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to fluid3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes on selection change in typepopup.
function typepopup_Callback(hObject, eventdata, handles)
% hObject    handle to typepopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns typepopup contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from typepopup
contents=get(hObject,'Value');
switch contents
    case 1
        set(handles.fluidpanel,'Visible','on');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','off');
    case 2
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','on');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','off');
    case 3
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','on');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','off');
    case 4
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','on');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','off');
    case 5
```

```
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','on');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','off');
    case 6
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','on');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','off');
    case 7
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','on');
        set(handles.simpporouspanel,'Visible','off');
    case 8
        set(handles.fluidpanel,'Visible','off');
        set(handles.solidpanel,'Visible','off');
        set(handles.efporouspanel,'Visible','off');
        set(handles.perfpanel,'Visible','off');
        set(handles.microperfpanel,'Visible','off');
        set(handles.orthopanel,'Visible','off');
        set(handles.slotsperfpanel,'Visible','off');
        set(handles.simpporouspanel,'Visible','on');
end


% --- Executes during object creation, after setting all properties.
function typepopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to typepopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in propertiespopup.
function propertiespopup_Callback(hObject, eventdata, handles)
% hObject    handle to propertiespopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns propertiespopup
contents as cell array
%        contents{get(hObject,'Value')} returns selected item from
propertiespopup
global propertyData;
axes(handles.propertiesfigure);
figureNumber=get(handles.propertiespopup,'value');
f=propertyData(:,1);
fmin=min(f);
fmax=max(f);
switch figureNumber
    case 1
        semilogx(propertyData(:,1),real(propertyData(:,4)),'k--');
        legend('Surface Impedance (Rayl, Real')
        xlabel('Frequency (Hz)');
        ylabel('Surface Impedance (Rayl, Real)');
        grid on
    case 2
        semilogx(propertyData(:,1),imag(propertyData(:,4)),'k--');
        legend('Surface Impedance (Rayl, Imag)')
        xlabel('Frequency (Hz)');
        ylabel('Surface Impedance (Rayl, Imag)');
        grid on
    case 3
        semilogx(propertyData(:,1),(propertyData(:,3)),'k--');
        legend('Reflection Coefficient')
        axis([fmin fmax -0 1]);
        xlabel('Frequency (Hz)');
        ylabel('Sound Reflection Coefficient');
        grid on
    case 4
        semilogx(propertyData(:,1),propertyData(:,2),'k--');
        legend('Absorption Coefficient')
        axis([fmin fmax -0 1]);
        xlabel('Frequency (Hz)');
        ylabel('Sound Absorption Coefficient');
        grid on
    case 5
        semilogx(propertyData(:,1),(propertyData(:,5)),'k--');
        legend('Transmission Coefficient')
        axis([fmin fmax -0 1]);
        xlabel('Frequency (Hz)');
        ylabel('Sound Transmission Coefficient');
        grid on
    case 6
        semilogx(propertyData(:,1),propertyData(:,6),'k--');
        legend('Sound Transmission Loss')
        xlabel('Frequency (Hz)');
        ylabel('Transmission Loss (dB)');
        grid on
end


% --- Executes during object creation, after setting all properties.
```

```
function propertiespopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to propertiespopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function newedit_Callback(hObject, eventdata, handles)
% hObject    handle to newedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of newedit as text
%        str2double(get(hObject,'String')) returns contents of newedit as a
double


% --- Executes during object creation, after setting all properties.
function newedit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to newedit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in layerpopup.
function layerpopup_Callback(hObject, eventdata, handles)
% hObject    handle to layerpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns layerpopup contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from layerpopup
global layerData;

value=get(handles.layerpopup,'Value');
prop=layerData;
layerType=prop(value,1);
switch layerType
    case 1
        set(handles.typepopup,'Value',1)
        set(handles.fluid1,'String',num2str(prop(value,2)))
        set(handles.fluid2,'String',num2str(prop(value,3)))
        set(handles.fluid3,'String',num2str(prop(value,4)))
    case 2
        set(handles.typepopup,'Value',2)
        set(handles.solid1,'String',num2str(prop(value,2)))
        set(handles.solid2,'String',num2str(prop(value,3)))
        set(handles.solid3,'String',num2str(prop(value,4)))
        set(handles.solid4,'String',num2str(prop(value,5)))
    case 3
        set(handles.typepopup,'Value',3)
        set(handles.ortho1,'String',num2str(prop(value,2)))
        set(handles.ortho2,'String',num2str(prop(value,3)))
        set(handles.ortho3,'String',num2str(prop(value,4)))
        set(handles.ortho4,'String',num2str(prop(value,5)))
        set(handles.ortho5,'String',num2str(prop(value,6)))
        set(handles.ortho6,'String',num2str(prop(value,7)))
        set(handles.ortho7,'String',num2str(prop(value,8)))
        set(handles.ortho8,'String',num2str(prop(value,9)))
        set(handles.ortho9,'String',num2str(prop(value,10)))
        set(handles.ortho10,'String',num2str(prop(value,11)))
        set(handles.ortho11,'String',num2str(prop(value,12)))
    case 4
        set(handles.typepopup,'Value',4)
        set(handles.efporous1,'String',num2str(prop(value,2)))
        set(handles.efporous2,'String',num2str(prop(value,3)))
        set(handles.efporous3,'String',num2str(prop(value,4)))
        set(handles.efporous4,'String',num2str(prop(value,5)))
        set(handles.efporous5,'String',num2str(prop(value,6)))
        set(handles.efporous6,'String',num2str(prop(value,7)))
        set(handles.efporous7,'String',num2str(prop(value,8)))
        set(handles.efporous8,'String',num2str(prop(value,9)))
        set(handles.efporous9,'String',num2str(prop(value,10)))
        set(handles.efporous10,'String',num2str(prop(value,11)))
        set(handles.efporous11,'String',num2str(prop(value,12)))
        set(handles.efporous12,'String',num2str(prop(value,13)))
        set(handles.efporous13,'String',num2str(prop(value,14)))
        set(handles.efporous14,'String',num2str(prop(value,15)))
    case 5
        set(handles.typepopup,'Value',5)
        set(handles.perf1,'String',num2str(prop(value,2)))
        set(handles.perf2,'String',num2str(prop(value,3)))
        set(handles.perf3,'String',num2str(prop(value,4)))
        set(handles.perf4,'String',num2str(prop(value,5)))
        set(handles.perf5,'String',num2str(prop(value,6)))
        set(handles.perf6,'String',num2str(prop(value,7)))
        set(handles.perf7,'String',num2str(prop(value,8)))

    case 6
        set(handles.typepopup,'Value',6)
        set(handles.microperf1,'String',num2str(prop(value,2)))
        set(handles.microperf2,'String',num2str(prop(value,3)))
        set(handles.microperf3,'String',num2str(prop(value,4)))
        set(handles.microperf4,'String',num2str(prop(value,5)))
        set(handles.microperf5,'String',num2str(prop(value,6)))
        set(handles.microperf6,'String',num2str(prop(value,7)))
        set(handles.microperf7,'String',num2str(prop(value,8)))
    case 7
        set(handles.typepopup,'Value',7)
        set(handles.slots1,'String',num2str(prop(value,2)))
        set(handles.slots2,'String',num2str(prop(value,3)))
        set(handles.slots3,'String',num2str(prop(value,4)))
        set(handles.slots4,'String',num2str(prop(value,5)))
        set(handles.slots5,'String',num2str(prop(value,6)))
        set(handles.slots6,'String',num2str(prop(value,7)))
        set(handles.slots7,'String',num2str(prop(value,8)))
        set(handles.slots8,'String',num2str(prop(value,9)))

    case 8
        set(handles.typepopup,'Value',8)
        set(handles.simpPorous1,'String',num2str(prop(value,2)))
        set(handles.simpPorous2,'String',num2str(prop(value,3)))

end
typepopup_Callback(handles.typepopup, eventdata, handles);





% --- Executes during object creation, after setting all properties.
function layerpopup_CreateFcn(hObject, eventdata, handles)
% hObject    handle to layerpopup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function solid1_Callback(hObject, eventdata, handles)
% hObject    handle to solid1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of solid1 as text
%        str2double(get(hObject,'String')) returns contents of solid1 as a
double


% --- Executes during object creation, after setting all properties.
function solid1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to solid1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function solid2_Callback(hObject, eventdata, handles)
% hObject    handle to solid2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of solid2 as text
%        str2double(get(hObject,'String')) returns contents of solid2 as a
double


% --- Executes during object creation, after setting all properties.
function solid2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to solid2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function solid3_Callback(hObject, eventdata, handles)
% hObject    handle to solid3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of solid3 as text
%        str2double(get(hObject,'String')) returns contents of solid3 as a
double


% --- Executes during object creation, after setting all properties.
function solid3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to solid3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function solid4_Callback(hObject, eventdata, handles)
% hObject    handle to solid4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of solid4 as text
%        str2double(get(hObject,'String')) returns contents of solid4 as a
double


% --- Executes during object creation, after setting all properties.
function solid4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to solid4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous5_Callback(hObject, eventdata, handles)
% hObject    handle to efporous5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous5 as text
%        str2double(get(hObject,'String')) returns contents of efporous5 as a
double


% --- Executes during object creation, after setting all properties.
function efporous5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function efporous6_Callback(hObject, eventdata, handles)
% hObject    handle to efporous6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous6 as text
%        str2double(get(hObject,'String')) returns contents of efporous6 as a
double


% --- Executes during object creation, after setting all properties.
function efporous6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous7_Callback(hObject, eventdata, handles)
% hObject    handle to efporous7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous7 as text
%        str2double(get(hObject,'String')) returns contents of efporous7 as a
double


% --- Executes during object creation, after setting all properties.
function efporous7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous8_Callback(hObject, eventdata, handles)
% hObject    handle to efporous8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of efporous8 as text
%        str2double(get(hObject,'String')) returns contents of efporous8 as a
double


% --- Executes during object creation, after setting all properties.
function efporous8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous9_Callback(hObject, eventdata, handles)
% hObject    handle to efporous9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous9 as text
%        str2double(get(hObject,'String')) returns contents of efporous9 as a
double


% --- Executes during object creation, after setting all properties.
function efporous9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous10_Callback(hObject, eventdata, handles)
% hObject    handle to efporous10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous10 as text
%        str2double(get(hObject,'String')) returns contents of efporous10 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function efporous10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous11_Callback(hObject, eventdata, handles)
% hObject    handle to efporous11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous11 as text
%        str2double(get(hObject,'String')) returns contents of efporous11 as a
double


% --- Executes during object creation, after setting all properties.
function efporous11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous12_Callback(hObject, eventdata, handles)
% hObject    handle to efporous12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous12 as text
%        str2double(get(hObject,'String')) returns contents of efporous12 as a
double


% --- Executes during object creation, after setting all properties.
function efporous12_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to efporous12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous13_Callback(hObject, eventdata, handles)
% hObject    handle to efporous13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous13 as text
%        str2double(get(hObject,'String')) returns contents of efporous13 as a
double


% --- Executes during object creation, after setting all properties.
function efporous13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous14_Callback(hObject, eventdata, handles)
% hObject    handle to efporous14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous14 as text
%        str2double(get(hObject,'String')) returns contents of efporous14 as a
double


% --- Executes during object creation, after setting all properties.
function efporous14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous1_Callback(hObject, eventdata, handles)
% hObject    handle to efporous1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous1 as text
%        str2double(get(hObject,'String')) returns contents of efporous1 as a
double


% --- Executes during object creation, after setting all properties.
function efporous1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous2_Callback(hObject, eventdata, handles)
% hObject    handle to efporous2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous2 as text
%        str2double(get(hObject,'String')) returns contents of efporous2 as a
double


% --- Executes during object creation, after setting all properties.
function efporous2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function efporous3_Callback(hObject, eventdata, handles)
% hObject    handle to efporous3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous3 as text
%        str2double(get(hObject,'String')) returns contents of efporous3 as a
double


% --- Executes during object creation, after setting all properties.
function efporous3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function efporous4_Callback(hObject, eventdata, handles)
% hObject    handle to efporous4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of efporous4 as text
%        str2double(get(hObject,'String')) returns contents of efporous4 as a
double


% --- Executes during object creation, after setting all properties.
function efporous4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to efporous4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function minFreqEdit_Callback(hObject, eventdata, handles)
% hObject    handle to minFreqEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of minFreqEdit as text
%        str2double(get(hObject,'String')) returns contents of minFreqEdit as
a double


% --- Executes during object creation, after setting all properties.
function minFreqEdit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to minFreqEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function maxFreqEdit_Callback(hObject, eventdata, handles)
% hObject    handle to maxFreqEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of maxFreqEdit as text
%        str2double(get(hObject,'String')) returns contents of maxFreqEdit as
a double



% --- Executes during object creation, after setting all properties.
function maxFreqEdit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to maxFreqEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function angleEdit_Callback(hObject, eventdata, handles)
% hObject    handle to angleEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of angleEdit as text
%        str2double(get(hObject,'String')) returns contents of angleEdit as a
double


% --- Executes during object creation, after setting all properties.
function angleEdit_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angleEdit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

%Gamma matrix function to calculate transfer matrix adapted from Andrew
%Wareing's Beam Tracing Code
function gm = gmf(omega, z, wn, dr, C, lt)

%omega = angular frequency
%z = length in the 3 direction (along thickness of layer)
%wn = [k1 kp3 ks3] for a solid layer
%wn = [k1 kQL kQT kT] for isotropic and orthotropic solid
%wn = [k1 k13 k33 k23]for a porous layer
%dr = [] for a isotropic and orthotropic solid layer
%dr = [disp_ratio1 disp_ratio2 disp_ratio3]for a porous layer
%C = [D1  E1 E2 ] for an isotropic solid layer
%C = [D1  D2 D3 ] for an orthotropic Solid
%C = [D1  E1 E2 D2 F1 F2] for a porous layer
%lt = layer type

gm = zeros(6, 6);

%elements of gm that are common for both isotropic solid and porous layers:

gm(1, 1) = omega*wn(1)*cos(wn(2)*z);
gm(2, 1) = -1j*omega*wn(2)*sin(wn(2)*z);
gm(4, 1) = -C(1)*cos(wn(2)*z);
gm(5, 1) = 1j*C(2)*wn(2)*sin(wn(2)*z);

gm(1, 2) = -1j*omega*wn(1)*sin(wn(2)*z);
gm(2, 2) = omega*wn(2)*cos(wn(2)*z);
gm(4, 2) = 1j*C(1)*sin(wn(2)*z);
gm(5, 2) = -C(2)*wn(2)*cos(wn(2)*z);

gm(1, 5) = 1j*omega*wn(3)*sin(wn(3)*z);
gm(2, 5) = omega*wn(1)*cos(wn(3)*z);
gm(4, 5) = 1j*C(2)*wn(3)*sin(wn(3)*z);
gm(5, 5) = C(3)*cos(wn(3)*z);
gm(1, 6) = -omega*wn(3)*cos(wn(3)*z);
gm(2, 6) = -1j*omega*wn(1)*sin(wn(3)*z);
gm(4, 6) = -C(2)*wn(3)*cos(wn(3)*z);
gm(5, 6) = -1j*C(3)*sin(wn(3)*z);

if lt == 'p'
   %additional elements for a porous layer:

   gm(3, 1) = -1j*omega*wn(2)*dr(1)*sin(wn(2)*z);
   gm(6, 1) = -C(5)*cos(wn(2)*z);

   gm(3, 2) = omega*dr(1)*wn(2)*cos(wn(2)*z);
   gm(6, 2) = 1j*C(5)*sin(wn(2)*z);

   gm(1, 3) = omega*wn(1)*cos(wn(4)*z);
   gm(2, 3) = -1j*omega*wn(4)*sin(wn(4)*z);
   gm(3, 3) = -1j*omega*wn(4)*dr(2)*sin(wn(4)*z);
   gm(4, 3) = -C(4)*cos(wn(4)*z);
   gm(5, 3) = 1j*C(2)*wn(4)*sin(wn(4)*z);
   gm(6, 3) = -C(6)*cos(wn(4)*z);

   gm(1, 4) = -1j*omega*wn(1)*sin(wn(4)*z);
   gm(2, 4) = omega*wn(4)*cos(wn(4)*z);
   gm(3, 4) = omega*dr(2)*wn(4)*cos(wn(4)*z);
   gm(4, 4) = 1j*C(4)*sin(wn(4)*z);
   gm(5, 4) = -C(2)*wn(4)*cos(wn(4)*z);
   gm(6, 4) = 1j*C(6)*sin(wn(4)*z);

   gm(3, 5) = omega*wn(1)*dr(3)*cos(wn(3)*z);

   gm(3, 6) = -1j*omega*wn(1)*dr(3)*sin(wn(3)*z);
elseif lt=='s'
   %remove rows and cols for solid matrix
   gm(:, 3) = [];
   gm(:, 3) = [];
   gm(3, :) = [];
   gm(5, :) = [];
else
   gm = zeros(4, 6); %reinitialize matrix for orthotropic solid
   gm(1,1)=omega*wn(1)*cos(wn(2)*z);
   gm(1,2)=-1j*omega*wn(1)*sin(wn(2)*z);
   gm(1,3)=1j*omega*wn(3)*sin(wn(3)*z);
   gm(1,4)=-omega*wn(3)*cos(wn(3)*z);
   gm(1,5)=1j*omega*wn(4)*sin(wn(4)*z);
   gm(1,6)=omega*wn(4)*cos(wn(4)*z);

   gm(2,1)=-1j*omega*wn(2)*sin(wn(2)*z);
   gm(2,2)=omega*wn(2)*cos(wn(2)*z);
   gm(2,3)=omega*wn(1)*cos(wn(3)*z);
   gm(2,4)=-1j*omega*wn(1)*sin(wn(3)*z);
   gm(2,5)=omega*wn(1)*cos(wn(4)*z);
   gm(2,6)=-1j*omega*wn(1)*sin(wn(4)*z);

   gm(3,1)=C(1)*cos(wn(2)*z);
   gm(3,2)=-1j*C(1)*sin(wn(2)*z);
   gm(3,3)=-1j*C(2)*sin(wn(3)*z);
   gm(3,4)=C(2)*cos(wn(3)*z);
   gm(3,5)=-1j*C(3)*sin(wn(4)*z);
   gm(3,6)=C(3)*cos(wn(4)*z);

   gm(4,1)=-1j*C(4)*sin(wn(2)*z);
   gm(4,2)=C(4)*cos(wn(2)*z);
   gm(4,3)=C(5)*cos(wn(3)*z);
   gm(4,4)=-1j*C(5)*sin(wn(3)*z);
   gm(4,5)=C(6)*cos(wn(4)*z);
   gm(4,6)=-1j*C(6)*sin(wn(4)*z);
end

function Tm = tmatrix(matProp,aofi,omega)
    layerType=matProp(1);
    c=344; %Assuming air temperature at 20degree Celsius
    k1 = omega/c*sin(aofi);    %component of wave number parallel to surface
    switch layerType
            case 1
                %Codes adapted from Andrew Wareing Beam Tracer Code
                %get material parameters:
                h=matProp(2)*0.001;%length, convert to m
                density = matProp(3);
                c = matProp(4);

                %calculate wave number:
                k = omega/c;

                %calculate component of wave number in the direction normal to
surface:
                k3 = sqrt(k^2 - k1^2);

                %calculate the transfer matrix:
                Tm = zeros(2, 2);

                if k3 == 0      %grazing incidence
                    Tm(1, 1) = 1;
                    Tm(2, 2) = 1;
                else
                    tm1 = omega*density/k3;

                    Tm(1, 1) = cos(k3*h);
                    Tm(1, 2) = tm1*1j*sin(k3*h);
                    Tm(2, 1) = (1/tm1)*1j*sin(k3*h);
                    Tm(2, 2) = cos(k3*h);
                end
            case 2
                %Codes adapted from Andrew Wareing Beam Tracer Code
                %get user input parameters:
                h = matProp(2)*1.0e-3;      %convert to m
                density = matProp(3);
                E = matProp(4)*1.0e6;      %convert to Pa
                pr = matProp(5); %Poisson's ratio
                %Calculate Lame coefficients:
                lambda = (pr*E)/((1 + pr)*(1-2*pr));
                mu = E/(2*(1 + pr));

                %Set displacement ratio vector to empty (for porous layers
only):
                dr = [0 0 0];

                %calculate longitudnal and shear wave speeds (no attenuation):
                cp = sqrt((lambda + 2*mu)/density);
                cs = sqrt(mu/density);

                %Calculate wave numbers:
                kp = omega/cp;
                ks = omega/cs;

                %Calculate component of wave numbers in the 3 direction:
                kp3 = sqrt(kp^2 - k1^2);
                ks3 = sqrt(ks^2 - k1^2);

                wn = [k1 kp3 ks3 0];

                %calculate additional parameters:
                D1 = lambda*(kp3^2 + k1^2) + 2*mu*kp3^2;
                E1 = 2*mu*k1;
                E2 = mu*(ks3^2 - k1^2);

                C = [D1 E1 E2 0 0 0];

                %Intermediate matrices (tau) to determine transfer matrix
                %(tp) for the solid layer
                ts1 = gmf(omega, -h, wn, dr, C, 's');
                ts2 = gmf(omega, 0, wn, dr, C, 's');

                %Calculate the transfer matrix for the porous layer:
                Tm = ts1*pinv(ts2);
            case 3
                %Wave number derived by using method described in
                %"Acoustics of Woods (2nd ed) page 54
                %get user input parameters:
                h = matProp(2)*1.0e-3;      %convert to m
                density = matProp(3);
                E11 = matProp(4)*1.0e6;      %convert to Pa
                E22 = matProp(5)*1.0e6;      %convert to Pa
                E33 = matProp(6)*1.0e6;      %convert to Pa
                v12 = matProp(7);
                v13 = matProp(8);
                v23 = matProp(9);
                G12 = matProp(10)*1.0e6;
                G13 = matProp(10)*1.0e6;
                G23 = matProp(10)*1.0e6;

                %Symetry in the Compliance Matrix
                v31=v13*E33/E11;
```

```
v32=v23*E33/E22;
v21=v12*E33/E11;

%Creating Stiffness matrix
S=(1-v12*v21-v23*v32-v13*v31-2*v21*v32*v31)/(E11*E22*E33);
C11=(1-v23*v32)/(E22*E33*S);
C22=(1-v13*v31)/(E11*E33*S);
C33=(1-v12*v21)/(E11*E22*S);
C12=(v21+v23*v31)/(E22*E33*S);
C13=(v13+v12*v23)/(E22*E11*S);
C23=(v32+v31*v12)/(E11*E33*S);
C44=G23;
C55=G13;
C66=G12;

%Deriving Velocity and Wave Number for each Wave
n1=sin(aofi);
n3=cos(aofi);
R11=C11*n1^2+C55*n3^2; %Solving Cristoffel's Eqution
R33=C33*n3^2+C55*n1^2;
R13=(C13+C55)*n1*n3;

VQL=sqrt(((R11+R33)+sqrt((R11-R33)^2+4*R13^2))/(2*density));
VQT=sqrt(((R11+R33)-sqrt((R11-R33)^2+4*R13^2))/(2*density));
VT=sqrt((C66*n1^2+C44*n3^2)/density);

deltaQL=omega/VQL; %Quasi-Longitudinal wave number
deltaQT=omega/VQT; %Quasi-Transverse wave number
deltaT=omega/VT;   %Transverse wave number;

%wave number vector in x3 axis
kQL=sqrt(deltaQL^2-k1^2);
kQT=sqrt(deltaQT^2-k1^2);
kT=sqrt(deltaT^2-k1^2);

%Calculating intermediate transfer matrix
dr = [0 0 0];   %only for porous media
wn = [k1 kQL kQT kT];

%calculate additional parameters:
D1 = (-C13*k1^2-C33*kQL^2);
D2 = (C13*k1*kQT-C33*k1*kQT);
D3 = (C13*k1*kT-C33*k1*kT);
D4 = (-2*C55*k1*kQL);
D5 = (C55*kQT^2-C55*k1^2);
D6 = (C55*kT^2-C55*k1^2);

C = [D1 D2 D3 D4 D5 D6];

%Intermediate matrices (tau) to determine transfer matrix
%(tp) for the solid layer
tos1 = gmf(omega, -h, wn, dr, C, 'os');
tos2 = gmf(omega, 0, wn, dr, C, 'os');

Tm = tos1*pinv(tos2);
```

```
        case 4
            %Codes adapted from Andrew Wareing's Beam Tracer Code
            %get fluid parameters for the porous layer:

            StdPress = matProp(11)*1000;
            PFDen = matProp(12);
            PFVis = matProp(13);
            SpHeatRatio = matProp(14);
            Pr = matProp(15);

            %get solid parameters for the porous layer:

            h = matProp(2)*0.001;      %convert to m
            frame_den = matProp(3);
            frame_sm = matProp(4)*1.0e6;     %convert to Pa
            frame_pr = matProp(5);
            flow_res = matProp(6);
            por = matProp(7);
            tort = matProp(8);
            vis_dim = matProp(9);
            therm_dim = matProp(10);

            %Calculation of  intermediate parameters:

            %   Elasticity coefficients:
            %     Kf = Bulk modulas of air inside pores
            %     Kb = Bulk modulas of frame
            %     P, Q, R = elasticity coefficients required in Biot
theory


            %   the next 4 lines are intermediate calculations for Kf
            Kfa = SpHeatRatio*StdPress;
            Kfb = omega*Pr*therm_dim^2;
            Kfc = sqrt(1 + 1j*PFDen*Kfb/(16*PFVis));
            Kfd = 1j*therm_dim^2*Pr*omega*PFDen;

            Kf = Kfa/(SpHeatRatio - (SpHeatRatio - 1)*(1 +
(8*PFVis)/Kfd*Kfc)^(-1));
            Kb = (2*frame_sm*(frame_pr + 1))/(3*(1 - 2*frame_pr));
            P = 4/3*frame_sm + Kb + (1 - por)^2/por*Kf;
            Q = Kf*(1 - por);
            R = por*Kf;

            %   Inertial coupling terms:
            %     rho_11, rho_12, and rho_22
            %     intermediate parameters g, r, and rho_a

            %   the next 2 lines are intermediate calculations for g
            g1 = 4j*tort^2*PFVis*PFDen*omega;
            g2 = flow_res^2*vis_dim^2*por^2;
            g = sqrt(1 + g1/g2);
            r = 1j*flow_res*por^2*g/omega;
            rho_a = PFDen*por*(tort - 1);
```

```
                rho_11 = frame_den + rho_a-r;
                rho_12 = -rho_a + r;
                rho_22 = por*PFDen + rho_a - r;

                %   Wave numbers for all Biot waves:
                %   kp1 and kp2 are wave numbers for the two compressional
waves
                %   ks is the wave number for the shear wave
                %   the next 4 lines are intermediate calculations for kp1,
kp2, and ks
                ka = 2*(P*R - Q^2);
                kb = P*rho_22 + R*rho_11 - 2*Q*rho_12;
                kc = rho_11*rho_22 - rho_12^2;
                delta = kb^2 - 2*ka*kc;

                kp1 = omega^2/ka*(kb - sqrt(delta));
                kp2 = omega^2/ka*(kb + sqrt(delta));
                ks = omega^2/frame_sm*(kc/rho_22);

                %   Displacement ratios for the frame and air:
                %      disp_ratio1 and disp_ratio2 are for the 2 compressional
waves
                %      disp_ratio3 is for the shear wave
                %   the next 2 lines are intermediate calculations for the
displacement ratios
                d1 = omega^2*rho_11;
                d2 = omega^2*rho_12;

                disp_ratio1 = (P*kp1 - d1)/(d2 - Q*kp1);
                disp_ratio2 = (P*kp2 - d1)/(d2 - Q*kp2);
                disp_ratio3 = -rho_12/rho_22;

                dr = [disp_ratio1 disp_ratio2 disp_ratio3];

                %   Wave number components in the x3 direction (direction of
propagation):
                %      k13 and k23 are the x3 components for the two
compressional wave numbers
                %       k33 is the x3 component of the shear wave number

                k13 = sqrt(kp1 - k1^2);
                k23 = sqrt(kp2 - k1^2);
                k33 = sqrt(ks - k1^2);

                wn = [k1 k13 k33 k23];

                %   Coefficients required to calculate intermediate matrices
for the transfer matrix

                D1 = (P + Q*disp_ratio1)*(k1^2 + k13^2)- 2*frame_sm*k1^2;
                D2 = (P + Q*disp_ratio2)*(k1^2 + k23^2)- 2*frame_sm*k1^2;
                E1 = 2*frame_sm*k1;
                E2 = frame_sm*(k33^2 - k1^2);
                F1 = (R*disp_ratio1 + Q)*(k1^2 + k13^2);
                F2 = (R*disp_ratio2 + Q)*(k1^2 + k23^2);

                C = [D1  E1 E2 D2 F1 F2];

                %   Intermediate matrices (tau) to determine transfer matrix
(tp) for the porous layer

                tp1 = gmf(omega, -h, wn, dr, C, 'p');
                tp2 = gmf(omega, 0, wn, dr, C, 'p');

                %Calculate the transfer matrix for the porous layer:

                Tm = tp1*pinv(tp2);
            case 5

                %Codes adapted from Attala and Sgard(2007)
                t=matProp(2)*0.001;%length, convert to m
                perfRatio = matProp(3);
                d = matProp(4)*0.001; %hole diameter, convert to m
                c= matProp(5); %speed of sound
                rho= matProp(6); %medium density
                eta=matProp(7); %dynamic viscosity
                tort=matProp(8); %tortuosity


                %Using Sgard, Attala equivalent fluid model
                Rs=3*(0.5*sqrt(2*eta*omega*rho)); %According to Ingard, using
values twice as large give a much better fit to the curve
                %Rs=rho/perfRatio*sqrt(8*omega*15e-6)*(1+t/(2*d/2));

Za=tort*2*t/(d/2)*Rs/perfRatio+1j*rho*omega/perfRatio*tort*t+1j*tort*(2*t/(d/2
))*Rs/perfRatio;
                Tm=zeros(2,2);
                Tm(1, 1) = 1;
                Tm(1, 2) = Za*cos(aofi);
                Tm(2, 1) =  0;
                Tm(2, 2) = 1;
            case 6
                %Codes adapted from Maa
                t=matProp(2)*0.001;%length, convert to m
                perfRatio = matProp(3);
                d = matProp(4)*0.001; %hole diameter, convert to m
                c= matProp(5); %speed of sound
                rho= matProp(6); %medium density
                eta=matProp(7); %dynamic viscosity
                viscousDim=matProp(8); %Viscous Characteristic Length


                %calculate the transfer matrix (Maa, 1987):
                k4=d*sqrt(omega*rho/(4*eta)); %microperf behave as a locally
reacting material
                kr=sqrt((1+(k4^2)/32))+sqrt(2)*k4*d/(32*t);
                r=32*eta*t*kr/(perfRatio*d^2);
                km=1+(9+(0.5)*k4^2)^(-0.5)+0.85*d/t;
                wm=omega*t*km*rho/(perfRatio);
                Tm=zeros(2,2);
                Tm(1, 1) = 1;
                Tm(1, 2) = (r+1j*wm)*cos(aofi);
                Tm(2, 1) =  0;
                Tm(2, 2) = 1;

        case 7

                %Codes adapted from Attala and Sgard(2007)
                t=matProp(2)*0.001;%length, convert to m
                perfRatio = matProp(3);
                a = matProp(4)*0.001; %slots length, convert to m
                b = matProp(5)*0.001; %slots width, convert to m
                d = 2*a*b/(a+b); %hydraulic radius
                c= matProp(6); %speed of sound
                rho= matProp(7); %medium density
                eta=matProp(8); %dynamic viscosity
                tort=matProp(9); %tortuosity


                %Using Sgard, Attala equivalent fluid model
                Rs=2*0.5*sqrt(2*eta*omega*rho);

Za=tort*2*t/(d/2)*Rs/perfRatio+1j*rho*omega/perfRatio*tort*t+1j*tort*(2*t/(d/2
))*Rs/perfRatio;
                Tm=zeros(2,2);
                Tm(1, 1) = 1;
                Tm(1, 2) = Za*cos(aofi);
                Tm(2, 1) =  0;
                Tm(2, 2) = 1;

        case 8
                %Codes adapted from Delaney Bazley, "Acosutical Properties of
Fibrous Absorbent Material
                %get material parameters:
                h=matProp(2)*0.001;%length, convert to m
                sigma = matProp(3);
                rho = 1.21;

                X= rho*(omega/(2*pi()))/sigma; %Delaney Bazley
                %X= (omega/(2*pi()))/sigma; %Miki

                %calculate characteristic impedance of media
                zc=rho*c*(1+0.0571*(X.^-0.754)-1j*0.087*(X.^-0.732)); %Delaney
Bazley
                %zc=rho*c*(1+0.070*(X.^-0.632)-1j*0.107*(X.^-0.632)); %Miki

                %calculate wave number:
                k =(omega/c).*(1+0.0978*(X.^-0.7)-1j*0.189*(X.^-0.595));
%Delaney Bazley
                %k =(omega/c).*(1+0.109*(X.^-0.618)-1j*0.160*(X.^-0.618));
%Miki
```

```
                %calculate component of wave number in the direction normal to
surface:
                k3 = sqrt(k^2 - (k^2)*(sin(aofi))^2);

                %calculate the transfer matrix:
                Tm = zeros(2, 2);

                if k3 == 0      %grazing incidence
                    Tm(1, 1) = 1;
                    Tm(2, 2) = 1;
                else
                    Tm(1, 1) = cos(k3*h);
                    Tm(1, 2) = zc.*1j*sin(k3*h);
                    Tm(2, 1) = (1/zc).*1j*sin(k3*h);
                    Tm(2, 2) = cos(k3*h);
                end

    end

function [I,J]=intmatrix(leftProp,rightProp)
leftLayer=num2str(leftProp(1));
rightLayer=num2str(rightProp(1));
interface=strcat(leftLayer, rightLayer);

switch interface
case '11'
    I = eye(2);
    J = -I;
case '12'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '13'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '14'
    por = rightProp(7);
    %I = [0, -1; por, 0; (1-por), 0; 0, 0];
    I = [0, -1; (1-por), 0; 0, 0; por, 0];
    %J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
0, 0, 1, 0];
    J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
0, 0, 0, 1];
case '15'
    I = eye(2);
    J = -I;
case '16'
    I = eye(2);
    J = -I;
case '17'
    I = eye(2);
    J = -I;
case '18'
    I = eye(2);
    J = -I;
case '22'
    I = eye(4);
```

```
        J = -I;
    case '21'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '23'
        I = eye(4);
        J = -I;
    case '24'
        I = [1 0 0 0; 0 1 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = -[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 1; 0 0 0 0 1 0];
    case '25'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '26'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '27'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '28'
        I = eye(2);
        J = -I;
    case '31'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '32'
        I = eye(4);
        J = -I;
    case '33'
        I = eye(4);
        J = -I;
    case '34'
        I = [1 0 0 0; 0 1 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = -[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 1; 0 0 0 0 1 0];
    case '35'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '36'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '37'
        I = [0 1 0 0; 0 0 1 0; 0 0 0 1];
        J = [0 -1; 1 0; 0 0];
    case '38'
        I = eye(2);
        J = -I;
    case '44'
        p1 = leftProp(7);
        p2 = rightProp(7);
        I = eye(6);
        J = -[1 0 0 0 0 0; 0 1 0 0 0 0; 0 (1-p2/p1) p2/p1 0 0 0; 0 0 0 1 0 (1-
    p1/p2);...
            0 0 0 0 1 0; 0 0 0 0 0 p1/p2];
    case '41'
        por = leftProp(7);
        %I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
    0, 0, 1, 0];
        I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
    0, 0, 0, 1];
        %J = [0, -1; por, 0; (1-por), 0; 0, 0];
        J = [0, -1; (1-por), 0; 0, 0; por, 0];
    case '45'
        por = leftProp(7);
        %I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
    0, 0, 1, 0];
        I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
    0, 0, 0, 1];
        %J = [0, -1; por, 0; (1-por), 0; 0, 0];
        J = [0, -1; (1-por), 0; 0, 0; por, 0];
    case '46'
        por = leftProp(7);
        %I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
    0, 0, 1, 0];
        I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
    0, 0, 0, 1];
        %J = [0, -1; por, 0; (1-por), 0; 0, 0];
        J = [0, -1; (1-por), 0; 0, 0; por, 0];
    case '47'
        por = leftProp(7);
        %I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
    0, 0, 1, 0];
        I = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
    0, 0, 0, 1];
        %J = [0, -1; por, 0; (1-por), 0; 0, 0];
        J = [0, -1; (1-por), 0; 0, 0; por, 0];
    case '42'
       %I = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 1; 0 0 0 0 1 0];
        %J = -[1 0 0 0; 0 1 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
        I = -[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 1; 0 0 0 0 1 0];
        J = [1 0 0 0; 0 1 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
    case '43'
       %I = [1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 1; 0 0 0 0 1 0];
        %J = -[1 0 0 0; 0 1 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
        I = -[1 0 0 0 0 0; 0 1 0 0 0 0; 0 0 1 0 0 0; 0 0 0 1 0 1; 0 0 0 0 1 0];
        J = [1 0 0 0; 0 1 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1];
    case '48'
        I = eye(2);
        J = -I;
    case '51'
        I = eye(2);
        J = -I;
    case '52'
        I = [0 -1; 1 0; 0 0];
        J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
    case '53'
        I = [0 -1; 1 0; 0 0];
        J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
    case '54'
        por = rightProp(7);
        %I = [0, -1; por, 0; (1-por), 0; 0, 0];
        I = [0, -1; (1-por), 0; 0, 0; por, 0];
        %J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
    0, 0, 1, 0];
```

```
    J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
0, 0, 0, 1];
case '55'
    I = eye(2);
    J = -I;
case '56'
    I = eye(2);
    J = -I;
case '57'
    I = eye(2);
    J = -I;
case '58'
    I = eye(2);
    J = -I;
case '61'
    I = eye(2);
    J = -I;
case '62'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '63'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '64'
    por = rightProp(7);
    %I = [0, -1; por, 0; (1-por), 0; 0, 0];
    I = [0, -1; (1-por), 0; 0, 0; por, 0];
    %J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
0, 0, 1, 0];
    J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
0, 0, 0, 1];
case '65'
    I = eye(2);
    J = -I;
case '66'
    I = eye(2);
    J = -I;
case '67'
    I = eye(2);
    J = -I;
case '68'
    I = eye(2);
    J = -I;
case '71'
    I = eye(2);
    J = -I;
case '72'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '73'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '74'
    por = rightProp(7);
    %I = [0, -1; por, 0; (1-por), 0; 0, 0];
    I = [0, -1; (1-por), 0; 0, 0; por, 0];
    %J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
0, 0, 1, 0];
    J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
0, 0, 0, 1];
case '75'
    I = eye(2);
    J = -I;
case '76'
    I = eye(2);
    J = -I;
case '77'
    I = eye(2);
    J = -I;
case '78'
    I = eye(2);
    J = -I;
case '81'
    I = eye(2);
    J = -I;
case '82'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '83'
    I = [0 -1; 1 0; 0 0];
    J = [0 1 0 0; 0 0 1 0; 0 0 0 1];
case '84'
    por = rightProp(7);
    %I = [0, -1; por, 0; (1-por), 0; 0, 0];
    I = [0, -1; (1-por), 0; 0, 0; por, 0];
    %J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 0, 0, 1; 0, 0, 0, 1, 0, 0; 0, 0,
0, 0, 1, 0];
    J = [0, (1-por), por, 0, 0, 0; 0, 0, 0, 1, 0, 0; 0, 0, 0, 0, 1, 0; 0, 0,
0, 0, 0, 1];
case '85'
    I = eye(2);
    J = -I;
case '86'
    I = eye(2);
    J = -I;
case '87'
    I = eye(2);
    J = -I;
case '88'
    I = eye(2);
    J = -I;
end

function D=dmatrix(layerProp,Tm,inter,rb,numofLayers,Zc,aofi)
%Generating D matrix
for i=1:numofLayers+1
    switch i
        case 1
            D=[inter{i,1}, inter{i,2}*Tm{1,i}];
        case numofLayers+1
            if rb ==1 %rigid solid wall termination
                lastlayer=layerProp(numofLayers,1);
                switch lastlayer
```

```
                    case 1
                    Y=[0,1];
                    D=[D;zeros(1,size(D,2)-2),Y];
                    case 2
                    Y=[1,0,0,0;0,1,0,0];
                    D=[D;zeros(2,size(D,2)-4),Y];
                    case 3
                    Y=[1,0,0,0;0,1,0,0];
                    D=[D;zeros(2,size(D,2)-4),Y];
                    case 4
                    Y=[1,0,0,0,0,0;0,1,0,0,0,0;0,0,1,0,0,0];
                    D=[D;zeros(3,size(D,2)-6),Y];
                    case 8
                    Y=[0,1];
                    D=[D;zeros(1,size(D,2)-2),Y];
                end
            else
                lastlayer=layerProp(numofLayers,:);
                lastInter=cell(2);

[lastInter{1,1},lastInter{1,2}]=intmatrix(lastlayer,[1,0,0]);%Assuming last
layer is semi infinite fluid

D=[D,zeros(size(D,1),2);zeros(size(lastInter{1,1},1),(size(D,2)-
size(lastInter{1,1},2))),...
                    lastInter{1,1},lastInter{1,2};zeros(1,size(D,2)),-
1,Zc/cos(aofi)];
            end
        otherwise
            D1=zeros(size(D,1),size(inter{i,2},2));
            D2=zeros(size(inter{i,1},1),(size(D,2)-size(inter{i,1},2)));
            D=[D,D1;D2,inter{i,1}, inter{i,2}*Tm{1,i}];
    end
end

function [R, Z, T, TL] = calcprop(D, Zc, aofi,rb)
%This is adapted from original function made by Andrew Wareing

Z1 = Zc/cos(aofi);

%remove the first column of the D' matrix and get its determinent
D1 = D;
D1(:, 1) = [];
D1 = det(D1);

%remove the second column of the D' matrix and get its determinent
D2 = D;
D2(:, 2) = [];
D2 = det(D2);

Z = -D1/D2;     %Surface Impedance

R = (Z - Z1)/(Z + Z1);  %Reflection Coefficient
```

```
%The following lines of code calculate the pressure transmission coefficient
and
%sound transmission loss
%slc represents the second-last column in the D' matrix
slc = size(D, 2) - 1;

%remove the second-last column of the D' matrix and get its determinent
Dn = D;
Dn(:, slc) = [];
Dn = det(Dn);

%Calculate absorption coefficient only if there is a rigid backing, otherwise
%calculate both absorption and transmission coefficient
if rb == 0
   T = (1 + R)*Dn/D1;                  %pressure transmission coefficient
   tau = (abs(T)).^2;
   TL = -10*log10(abs(tau));           %Transmission Loss
else
   T = NaN;
   TL = NaN;
end



% --- Executes on button press in RBbutton.
function RBbutton_Callback(hObject, eventdata, handles)
% hObject    handle to RBbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of RBbutton




function microperf5_Callback(hObject, eventdata, handles)
% hObject    handle to microperf5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf5 as text
%        str2double(get(hObject,'String')) returns contents of microperf5 as a
double


% --- Executes during object creation, after setting all properties.
function microperf5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function microperf6_Callback(hObject, eventdata, handles)
% hObject    handle to microperf6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf6 as text
%        str2double(get(hObject,'String')) returns contents of microperf6 as a
double


% --- Executes during object creation, after setting all properties.
function microperf6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function microperf1_Callback(hObject, eventdata, handles)
% hObject    handle to microperf1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf1 as text
%        str2double(get(hObject,'String')) returns contents of microperf1 as a
double


% --- Executes during object creation, after setting all properties.
function microperf1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function microperf2_Callback(hObject, eventdata, handles)
% hObject    handle to microperf2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf2 as text
%        str2double(get(hObject,'String')) returns contents of microperf2 as a
double


% --- Executes during object creation, after setting all properties.
function microperf2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function microperf3_Callback(hObject, eventdata, handles)
% hObject    handle to microperf3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf3 as text
%        str2double(get(hObject,'String')) returns contents of microperf3 as a
double


% --- Executes during object creation, after setting all properties.
function microperf3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function microperf4_Callback(hObject, eventdata, handles)
% hObject    handle to microperf4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf4 as text
%        str2double(get(hObject,'String')) returns contents of microperf4 as a
double


% --- Executes during object creation, after setting all properties.
function microperf4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function microperf7_Callback(hObject, eventdata, handles)
% hObject    handle to microperf7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of microperf7 as text
%        str2double(get(hObject,'String')) returns contents of microperf7 as a
double


% --- Executes during object creation, after setting all properties.
function microperf7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to microperf7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf5_Callback(hObject, eventdata, handles)
% hObject    handle to perf5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf5 as text
%        str2double(get(hObject,'String')) returns contents of perf5 as a
double


% --- Executes during object creation, after setting all properties.
function perf5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function perf6_Callback(hObject, eventdata, handles)
% hObject    handle to perf6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf6 as text
%        str2double(get(hObject,'String')) returns contents of perf6 as a
double


% --- Executes during object creation, after setting all properties.
function perf6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function perf1_Callback(hObject, eventdata, handles)
% hObject    handle to perf1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf1 as text
%        str2double(get(hObject,'String')) returns contents of perf1 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function perf1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf2_Callback(hObject, eventdata, handles)
% hObject    handle to perf2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf2 as text
%        str2double(get(hObject,'String')) returns contents of perf2 as a
double


% --- Executes during object creation, after setting all properties.
function perf2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf3_Callback(hObject, eventdata, handles)
% hObject    handle to perf3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf3 as text
%        str2double(get(hObject,'String')) returns contents of perf3 as a
double


% --- Executes during object creation, after setting all properties.
function perf3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf4_Callback(hObject, eventdata, handles)
% hObject    handle to perf4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf4 as text
%        str2double(get(hObject,'String')) returns contents of perf4 as a
double


% --- Executes during object creation, after setting all properties.
function perf4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf7_Callback(hObject, eventdata, handles)
% hObject    handle to perf7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf7 as text
%        str2double(get(hObject,'String')) returns contents of perf7 as a
double


% --- Executes during object creation, after setting all properties.
function perf7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf8_Callback(hObject, eventdata, handles)
% hObject    handle to perf8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf8 as text
%        str2double(get(hObject,'String')) returns contents of perf8 as a
double


% --- Executes during object creation, after setting all properties.
function perf8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function perf9_Callback(hObject, eventdata, handles)
% hObject    handle to perf9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf9 as text
%        str2double(get(hObject,'String')) returns contents of perf9 as a
double


% --- Executes during object creation, after setting all properties.
function perf9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function perf11_Callback(hObject, eventdata, handles)
% hObject    handle to perf11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of perf11 as text
%        str2double(get(hObject,'String')) returns contents of perf11 as a
double


% --- Executes during object creation, after setting all properties.
function perf11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to perf11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho1_Callback(hObject, eventdata, handles)
% hObject    handle to ortho1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho1 as text
%        str2double(get(hObject,'String')) returns contents of ortho1 as a
double


% --- Executes during object creation, after setting all properties.
function ortho1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function ortho2_Callback(hObject, eventdata, handles)
% hObject    handle to ortho2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho2 as text
%        str2double(get(hObject,'String')) returns contents of ortho2 as a
double


% --- Executes during object creation, after setting all properties.
function ortho2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho3_Callback(hObject, eventdata, handles)
% hObject    handle to ortho3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho3 as text
%        str2double(get(hObject,'String')) returns contents of ortho3 as a
double


% --- Executes during object creation, after setting all properties.
function ortho3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho4_Callback(hObject, eventdata, handles)
% hObject    handle to ortho4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of ortho4 as text
%        str2double(get(hObject,'String')) returns contents of ortho4 as a
double


% --- Executes during object creation, after setting all properties.
function ortho4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho5_Callback(hObject, eventdata, handles)
% hObject    handle to ortho5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho5 as text
%        str2double(get(hObject,'String')) returns contents of ortho5 as a
double


% --- Executes during object creation, after setting all properties.
function ortho5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho6_Callback(hObject, eventdata, handles)
% hObject    handle to ortho6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho6 as text
%        str2double(get(hObject,'String')) returns contents of ortho6 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function ortho6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho7_Callback(hObject, eventdata, handles)
% hObject    handle to ortho7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho7 as text
%        str2double(get(hObject,'String')) returns contents of ortho7 as a
double


% --- Executes during object creation, after setting all properties.
function ortho7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho8_Callback(hObject, eventdata, handles)
% hObject    handle to ortho8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho8 as text
%        str2double(get(hObject,'String')) returns contents of ortho8 as a
double


% --- Executes during object creation, after setting all properties.
function ortho8_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to ortho8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho9_Callback(hObject, eventdata, handles)
% hObject    handle to ortho9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho9 as text
%        str2double(get(hObject,'String')) returns contents of ortho9 as a
double


% --- Executes during object creation, after setting all properties.
function ortho9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho10_Callback(hObject, eventdata, handles)
% hObject    handle to ortho10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho10 as text
%        str2double(get(hObject,'String')) returns contents of ortho10 as a
double


% --- Executes during object creation, after setting all properties.
function ortho10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function ortho11_Callback(hObject, eventdata, handles)
% hObject    handle to ortho11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of ortho11 as text
%        str2double(get(hObject,'String')) returns contents of ortho11 as a
double


% --- Executes during object creation, after setting all properties.
function ortho11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ortho11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in approximateButton.
function approximateButton_Callback(hObject, eventdata, handles)
% hObject    handle to approximateButton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t=str2num(get(handles.perf1,'String'));
d=str2num(get(handles.perf3,'String'));
perfRatio=str2num(get(handles.perf2,'String'));
D=sqrt((pi()*(d/2)^2)/perfRatio);
syms m n
Ee=symsum(symsum(heaviside(m)*heaviside(n)*2*D*(besselj(1,2*pi()*(d/2/D)*sqrt(
m^2+n^2)))^2/((pi()^2)*((m^2+n^2)^(3/2))),n,1,20),m,0,20)...
+heaviside(0)*heaviside(1)*2*D*(besselj(1,2*pi()*(d/2/D)*sqrt(1^2+0^2)))^2/((p
i()^2)*((1^2+0^2)^(3/2)));
tort=eval(1+2*Ee/t);
%Ee=0.48*sqrt(pi()*(d/2)^2)*(1-1.47*sqrt(perfRatio)+0.47*sqrt(perfRatio^3));
%tort=1+2*Ee/t


set(handles.perf7,'String',num2str(tort));
```

```
% --- Executes on button press in DFbutton.
function DFbutton_Callback(hObject, eventdata, handles)
% hObject    handle to DFbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of DFbutton
if (get(hObject,'Value') == get(hObject,'Max'))
    set(handles.angleEdit, 'enable', 'off')
else
    set(handles.angleEdit, 'enable', 'on')
end




function slots1_Callback(hObject, eventdata, handles)
% hObject    handle to slots1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots1 as text
%        str2double(get(hObject,'String')) returns contents of slots1 as a
double


% --- Executes during object creation, after setting all properties.
function slots1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function slots2_Callback(hObject, eventdata, handles)
% hObject    handle to slots2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots2 as text
%        str2double(get(hObject,'String')) returns contents of slots2 as a
double


% --- Executes during object creation, after setting all properties.
function slots2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function slots3_Callback(hObject, eventdata, handles)
% hObject    handle to slots3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots3 as text
%        str2double(get(hObject,'String')) returns contents of slots3 as a
double


% --- Executes during object creation, after setting all properties.
function slots3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function slots4_Callback(hObject, eventdata, handles)
% hObject    handle to slots4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots4 as text
%        str2double(get(hObject,'String')) returns contents of slots4 as a
double


% --- Executes during object creation, after setting all properties.
function slots4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function slots5_Callback(hObject, eventdata, handles)
% hObject    handle to slots5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots5 as text
%        str2double(get(hObject,'String')) returns contents of slots5 as a
double


% --- Executes during object creation, after setting all properties.
function slots5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function slots6_Callback(hObject, eventdata, handles)
% hObject    handle to slots6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots6 as text
%        str2double(get(hObject,'String')) returns contents of slots6 as a
double


% --- Executes during object creation, after setting all properties.
function slots6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
```

```
end



function slots7_Callback(hObject, eventdata, handles)
% hObject    handle to slots7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots7 as text
%        str2double(get(hObject,'String')) returns contents of slots7 as a
double


% --- Executes during object creation, after setting all properties.
function slots7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in approximateSlots.
function approximateSlots_Callback(hObject, eventdata, handles)
% hObject    handle to approximateSlots (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
t=str2num(get(handles.slots1,'String'));
a=str2num(get(handles.slots3,'String'));
b=str2num(get(handles.slots4,'String'));
perfRatio=str2num(get(handles.slots2,'String'));
A=sqrt(a*b/perfRatio); %assuming square cell for cavity with dimension A

%---Ingard's full formula---%

% syms m n
% xi=a/A;
% eta=b/A;
%Gmn=xi*eta*(sin(pi()*m*xi)*sin(pi()*m*eta)/((pi()^2)*m*xi*n*eta))^2;
%Ee=sqrt(a*b)*(2/pi())*(symsum(symsum(heaviside(m)*heaviside(n)*Gmn/sqrt((eta*
m^2)+(xi*n^2)),n,1,10),m,1,10))
%eval(Ee)
%tort=eval(1+2*Ee/t)

%---Ingards's approximation for perfRatio<0.16
%Ee=0.48*sqrt(a*b)*(1-1.25*sqrt(perfRatio));
%tort=1+2*Ee/t;
```

```
%---Allard's formula pp 193.---%
syms m n
Ee=2*A*symsum(heaviside(0)*heaviside(n)*a*(sin(pi()*n*b/A))^2/((pi()^3)*(n^3)*
b),n,1,20)...

+2*A*symsum(heaviside(m)*heaviside(0)*b*(sin(pi()*m*a/A))^2/((pi()^3)*(m^3)*a)
,m,1,20)...

+2*(A^3)/((a*b)*(pi()^5))*symsum(symsum(heaviside(m)*heaviside(n)*((sin(pi()*m
*a/A))^2)*((sin(pi()*n*b/A))^2)...
    /((m^2)*(n^2)*sqrt(m^2+n^2)),n,1,20),m,1,20);
tort=eval(1+2*Ee/t);




set(handles.slots8,'String',num2str(tort));



function slots8_Callback(hObject, eventdata, handles)
% hObject    handle to slots8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of slots8 as text
%        str2double(get(hObject,'String')) returns contents of slots8 as a
double


% --- Executes during object creation, after setting all properties.
function slots8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slots8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function simpPorous1_Callback(hObject, eventdata, handles)
% hObject    handle to simpPorous1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of simpPorous1 as text
%        str2double(get(hObject,'String')) returns contents of simpPorous1 as
a double
```

```
% --- Executes during object creation, after setting all properties.
function simpPorous1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to simpPorous1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function simpPorous2_Callback(hObject, eventdata, handles)
% hObject    handle to simpPorous2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of simpPorous2 as text
%        str2double(get(hObject,'String')) returns contents of simpPorous2 as
a double


% --- Executes during object creation, after setting all properties.
function simpPorous2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to simpPorous2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit189_Callback(hObject, eventdata, handles)
% hObject    handle to edit189 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit189 as text
%        str2double(get(hObject,'String')) returns contents of edit189 as a
double


% --- Executes during object creation, after setting all properties.
function edit189_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit189 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit190_Callback(hObject, eventdata, handles)
% hObject    handle to edit190 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit190 as text
%        str2double(get(hObject,'String')) returns contents of edit190 as a
double


% --- Executes during object creation, after setting all properties.
function edit190_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit190 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit191_Callback(hObject, eventdata, handles)
% hObject    handle to edit191 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit191 as text
%        str2double(get(hObject,'String')) returns contents of edit191 as a
double


% --- Executes during object creation, after setting all properties.
function edit191_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit191 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit192_Callback(hObject, eventdata, handles)
% hObject    handle to edit192 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit192 as text
%        str2double(get(hObject,'String')) returns contents of edit192 as a
double


% --- Executes during object creation, after setting all properties.
function edit192_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit192 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit193_Callback(hObject, eventdata, handles)
% hObject    handle to edit193 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit193 as text
%        str2double(get(hObject,'String')) returns contents of edit193 as a
double


% --- Executes during object creation, after setting all properties.
function edit193_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit193 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit194_Callback(hObject, eventdata, handles)
% hObject    handle to edit194 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit194 as text
%        str2double(get(hObject,'String')) returns contents of edit194 as a
double


% --- Executes during object creation, after setting all properties.
function edit194_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit194 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit195_Callback(hObject, eventdata, handles)
% hObject    handle to edit195 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit195 as text
%        str2double(get(hObject,'String')) returns contents of edit195 as a
double


% --- Executes during object creation, after setting all properties.
function edit195_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit195 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit196_Callback(hObject, eventdata, handles)
% hObject    handle to edit196 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit196 as text
%        str2double(get(hObject,'String')) returns contents of edit196 as a
double


% --- Executes during object creation, after setting all properties.
function edit196_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit196 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit197_Callback(hObject, eventdata, handles)
% hObject    handle to edit197 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit197 as text
%        str2double(get(hObject,'String')) returns contents of edit197 as a
double


% --- Executes during object creation, after setting all properties.
function edit197_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit197 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit198_Callback(hObject, eventdata, handles)
% hObject    handle to edit198 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit198 as text
%        str2double(get(hObject,'String')) returns contents of edit198 as a
double


% --- Executes during object creation, after setting all properties.
function edit198_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit198 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit199_Callback(hObject, eventdata, handles)
% hObject    handle to edit199 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit199 as text
%        str2double(get(hObject,'String')) returns contents of edit199 as a
double


% --- Executes during object creation, after setting all properties.
function edit199_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit199 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit200_Callback(hObject, eventdata, handles)
% hObject    handle to edit200 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit200 as text
%        str2double(get(hObject,'String')) returns contents of edit200 as a
double


% --- Executes during object creation, after setting all properties.
function edit200_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit200 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```