

DESIGN AND ANALYSIS OF TUNABLE ACOUSTIC LOUVERS USING AN OPEN
SOURCE FINITE ELEMENT PLATFORM



by
Mete Ögüç

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy in
Mechanical Engineering

Yeditepe University

2023

DESIGN AND ANALYSIS OF TUNABLE ACOUSTIC LOUVERS USING AN OPEN
SOURCE FINITE ELEMENT PLATFORM

APPROVED BY:

Assist. Prof. Dr. Ali Fethi Okyar
(Thesis Supervisor)
(Yeditepe University)

Assoc. Prof. Dr. Nezih Topaloğlu
(Yeditepe University)

Assist. Prof. Dr. Namık Çıplak
(Yeditepe University)

Prof. Dr. Kerim Mehmet Murat Tunç
(Bilgi University)

Assist. Prof. Dr. Tahsin Khajah
(The University of Texas at Tyler)

DATE OF APPROVAL: / / 20.....

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. If any material used in the thesis requires copyright, the necessary permissions have been obtained. No material from this thesis has been used for the award of another degree.

I accept all kinds of legal liability that may arise in case contrary to these situations.

Name, Last Name

Mete Öğüç

Signature

.....

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Dr. Ali Fethi Okyar for his role as an exceptional mentor, role model, and friend. His expertise and high standards as a lecturer have inspired me to reach for excellence and have had a profound impact on my development as a researcher.

I would also like to thank the members of my thesis committee: Dr. Tahsin Khajah, and Dr. Nezih Topaloğlu for their valuable insights, guidance, and support throughout the process. Dr. Tahsin Khajah has been an invaluable resource, helping me to deepen my understanding of the topic and guiding me in a very productive way. Dr. Nezih Topaloğlu has provided critical support and valuable suggestions that have helped me to navigate the process effectively.

I am also grateful to the lecturers in our department who have supported me throughout my studies, and to our head of department, Prof. Dr. Mehmet Akgün, who encouraged me to continue my studies and inspired me with his passion for our profession.

I would also like to extend my deepest gratitude to Dr. İlkin Okyar, who has welcomed me into her home and treated me like family, and I am forever grateful for her guidance and friendship.

Finally, I would like to express my deepest gratitude to my parents, who have always been my biggest supporters and sources of motivation. My father, M. Metin Ögüç has been particularly excited about this accomplishment, and I am grateful for the opportunity to give him this satisfaction. My mother, N. Nilgün Ögüç has always been patient and understanding, and has provided me with unending moral support throughout my studies.

This achievement would not have been possible without the support and guidance of all of these individuals, and I am deeply grateful for their contributions to my success.

ABSTRACT

DESIGN AND ANALYSIS OF TUNABLE ACOUSTIC LOUVERS USING AN OPEN SOURCE FINITE ELEMENT PLATFORM

In this study, the acoustic scattering problem was solved using the open-source finite element analysis platform, FEniCS. Modular scripts were written in the web-based interactive command-line Jupyter Notebook, which can also be integrated with various libraries to create a solution package for related engineering problems. The GMSH library, which provides high control and precision for creating meshes and defining boundary conditions for two-dimensional geometries, was used. The infinite solution domain of the external Helmholtz problem describing open space was truncated using the absorbing boundary condition and a finite solution domain was created. Error analyses were carried out for the Sommerfeld, BGT-1, and BGT-2 absorbing boundary conditions. It became possible to optimize the meshes created based on design variables. An optimization algorithm package was developed based on parameter-sweep and the Nelder-Mead methods. A crescent-shaped section with low sensitivity to small geometric changes and high sound attenuation performance is proposed as a result of the optimization analyzes adapted for single and multiple scattering problems. When evaluated with the error analyses, it was found that the use of the Sommerfeld ABC was sufficient for the purpose of the study. Thus, an additional software package was prepared to calculate the insertion losses of acoustic louvers used in noise control studies. With this software package, incident and total wave are separated into standard octaves using weighting and spectrum density filters. The results showed that the critical frequency is sensitive to the thickness of the acoustic louver. It is thought that this open-source software package can be used in the future to design acoustic metamaterials.

ÖZET

AÇIK KAYNAKLI SONLU ELEMAN PLATFORMU KULLANARAK AKORTLANABİLİR AKUSTİK MENFEZLERİN TASARIMI VE ANALİZİ

Bu çalışmada, akustik saçılma problemi açık kaynaklı bir sonlu elemanlar analiz platformu olan FEniCS kullanılarak çözüldü. İlgili mühendislik problemlerine bir çözüm paketi oluşturmak amacıyla FEniCS'in yanı sıra çeşitli kitaplıklarla da entegre edilebilen web-tabanlı interaktif komut satırı Jupyter Notebook'ta modüler betikler yazıldı. İki boyutlu geometriler için eleman ağları oluşturmak ve sınır koşullarını tanımlamak için yüksek kontrol ve hassasiyet sağlayan GMSH kütüphanesi kullanıldı. Açık alan tarifleyen dış Helmholtz problemi sonsuz çözüm alanı yutma sınır koşulu kullanılarak kırıldı ve sonlu bir çözüm alanı oluşturuldu. Sommerfeld, BGT-1 ve BGT-2 yutma sınır koşulları için hata analizleri yapıldı. Tasarım değişkenlerine bağlı oluşturulan eleman ağları ile optimizasyon yapmak mümkün oldu. Parametre taraması ve Nelder-Mead yöntemleri üzerine bir optimizasyon algoritma paketi geliştirildi. Tekli ve çoklu saçınım problemlerine uyarlanan bu en iyileme yöntemleri ile yapılan analizler sonucunda küçük geometrik değişikliklere az hassasiyet gösteren ve yüksek ses azaltım performansına sahip hilal şekilli bir kesit önerildi. Yapılan hata analizleri ile birlikte değerlendirildiğinde Sommerfeld ABC kullanımı çalışmanın amacı için yeterli olduğu anlaşıldı. Böylece gürültü kontrol çalışmalarında kullanılan akustik menfezlerin eklenti kayıplarını hesaplamak üzere bir ek yazılım paketi hazırlandı. Bu yazılım paketi ile gelen ve toplam dalga ağırlıklandırma ve spektrum yoğunluk filtreleri ile standart oktavlara ayrıldı. Sonuçlar, kritik frekansın akustik menfezin kalınlığına duyarlı olduğunu gösterdi. Bu açık kaynaklı yazılım paketi ile ileride akustik metamateryallerin tasarlanabileceği düşünülmektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iv
ABSTRACT.....	v
ÖZET.....	vi
TABLE OF CONTENTS.....	vii
LIST OF FIGURES.....	x
LIST OF TABLES.....	xiv
LIST OF ALGORITHMS.....	xv
LIST OF SYMBOLS/ABBREVIATIONS.....	xviii
1. INTRODUCTION.....	1
2. BACKGROUND.....	4
2.1. ACOUSTIC LOUVERS.....	5
2.2. DIFFRACTION AND SCATTERING FROM PERIODIC STRUCTURES.....	8
2.3. THE LINEAR WAVE EQUATION.....	10
2.4. HELMHOLTZ EQUATION.....	12
2.5. THE FINITE ELEMENT METHOD.....	13
3. MODELING THE ACOUSTIC SCATTERING PROBLEM.....	15
3.1. ABSORBING BOUNDARY CONDITIONS.....	16
3.2. ERROR ANALYSIS.....	18
3.2.1. Exact solution.....	20
3.2.2. The Modified Exact Solution.....	22
3.3. THE FINITE ELEMENT MODEL.....	23
3.3.1. Meshing Strategy.....	24
3.3.2. Scattered Wave Solution.....	27
3.3.3. Total Wave Solution.....	27
3.3.4. Incident Wave Generation.....	27
3.4. FINITE ELEMENT ANALYSIS.....	29
3.4.1. Sound-Hard Circular Cylinder in a Cylindrical Domain.....	30
3.4.2. Sound-Soft Circular Cylinder in a Cylindrical Domain.....	33
3.4.3. Non Structured Mesh Generation.....	34
3.4.4. Cylinder in a Rectangular Domain.....	35

4. SHAPE OPTIMIZATION	37
4.1. OPTIMIZATION FOR A SINGLE OBSTACLE.....	38
4.1.1. Optimization with Parameter-Sweep Method.....	40
4.1.2. Optimization with Nelder-Mead Method.....	43
4.2. OPTIMIZATION FOR MULTIPLE OBSTACLES	46
4.2.1. Homogeneous Neumann Boundary Condition.....	48
4.2.1.1. Half-Cylinder in a Rectangular Domain.....	48
4.2.1.2. Vertically Aligned Finite Number of Cylinders.....	50
4.2.1.3. Vertically Aligned Infinite Number of Cylinders.....	51
4.2.2. Periodic Boundary Condition.....	52
4.2.2.1. Vertically Aligned Infinite Cylinders.....	53
4.2.3. Optimization for Vertically Aligned Crescents.....	54
5. ANALYSIS OF CONVENTIONAL ACOUSTIC LOUVERS.....	57
5.1. BENCHMARK STUDY ON CONVENTIONAL ACOUSTIC LOUVERS.....	57
5.2. FINITE ELEMENT ANALYSIS.....	60
6. CONCLUSIONS.....	68
REFERENCES	70
APPENDIX A.....	83
A.1. EXACT SOLUTION FOR SOUND-HARD BODIES.....	83
A.2. EXACT SOLUTION FOR SOUND-SOFT BODIES	84
A.3. EXACT SOLUTION MODIFIED TO BGT-2 ABC.....	85
APPENDIX B.....	86
B.1. WEAK FORM: SCATTERING WAVE SOLUTION.....	88
B.1.1. Absorbing Boundary Condition.....	89
B.1.1.1. Sommerfeld ABC.....	89
B.1.1.2. BGT-1 ABC.....	90
B.1.1.3. BGT-2 ABC.....	91
B.1.2. Scattering From a Sound-Hard Body.....	94
B.1.3. Scattering From a Sound-Soft Body.....	95
B.2. WEAK FORM: TOTAL WAVE SOLUTION.....	96
APPENDIX C.....	100
C.1. CODES: EXACT SOLUTION.....	100

C.2. CODES: MODIFIED EXACT SOLUTION.....	101
C.3. CODES: MESH GENERATION.....	102
C.4. CODES: FINITE ELEMENT MODELS.....	122
C.4.1. Cylindrical Domain, Sound Hard, Cylinder Body.....	122
C.4.2. Cylindrical Domain, Sound Soft, Cylinder Body.....	130
C.4.3. Cylindrical Domain, Sound Hard, Arbitrary-Shapes.....	132
C.4.4. Rectangular Domain, Sound-Hard, Arbitrary-Shapes, Single Body	136
C.4.5. Rectangular Domain, Sound-Hard, Arbitrary-Shapes, Symmetric Bodies.....	138
C.4.6. Rectangular Domain, Sound-Hard, Arbitrary-Shapes, Periodic Bodies.....	140
C.5. CODES: NELDER-MEAD OPTIMIZATION.....	142
C.6. CODES: INSERTION LOSS.....	144
APPENDIX D.....	146
D.1. OPTIMIZATION WITH PARAMETER-SWEEP ALGORITHM.....	146
D.1.1. Cylindrical Domain, Sound-Hard, Single Body	146
D.1.2. Cylindrical Domain, Sound-Soft, Single Body.....	147
D.1.3. Rectangular Domain, Sound-Hard, Single Body	148
D.1.4. Rectangular Domain, Sound-Soft, Single Body	149
D.1.5. Rectangular Domain, Sound-Hard, Periodic Bodies	150
D.1.6. Rectangular Domain, Sound-Soft, Periodic Bodies.....	151
D.1.7. Rectangular Domain, Sound-Hard, Periodic Bodies	151
D.2. OPTIMIZATION WITH NELDER-MEAD ALGORITHM.....	152
D.2.1. Cylindrical Domain, Sound-Hard, Single Body	152
D.2.2. Cylindrical Domain, Sound-Soft, Single Body.....	152
D.2.3. Rectangular Domain, Sound-Hard, Single Body	153
D.2.4. Rectangular Domain, Sound-Soft, Single Body	153
D.2.5. Rectangular Domain, Sound-Hard, Periodic Bodies	154
D.2.6. Rectangular Domain, Sound-Soft, Periodic Bodies.....	154
D.3. ACOUSTIC LOUVER BENCHMARK DATA.....	155
D.4. ACOUSTIC LOUVER FEA RESULTS.....	156

LIST OF FIGURES

Figure 2.1. Acoustic louvres for acoustic treatment in a generator room.....	5
Figure 2.2. Render of a sectioned conventional acoustic louver.....	7
Figure 2.3. Illustration of Snell's law of refraction for a ray of light.....	8
Figure 3.1. Acoustic scattering problem.....	15
Figure 3.2. The unbound solution domain to a two-dimensional scattering problem.....	16
Figure 3.3. Scattering from an arbitrarily shaped object.....	17
Figure 3.4. Scattering of a plane wave from a circular cylinder.....	20
Figure 3.5. 3D plot of exact solution	21
Figure 3.6. Exact solutions for (a) sound-hard and (b) sound-soft 2D circular cylinder acoustic scattering obstacles.....	22
Figure 3.7. Scattering cylinder FEM truncated by ABC.....	24
Figure 3.8. (a) A structured mesh with $R_{out} = R_{in} + \lambda$. (b) Boundary normals.....	26
Figure 3.9. Real part and imaginary part contour plots of an incident wave plane wave with the phase shift observable.....	28
Figure 3.10. Polar plots of real part, imaginary part and magnitude of the scattered wave on the cylinder surface with (a) B_0 : Sommerfeld, (b) B_1 : BGT-1, and (c) B_2 : BGT-2 ABCs.	31
Figure 3.11. Absolute error comparison on solution domain for ABCs.....	32
Figure 3.12. Error analysis showing the performance of (a) Sommerfeld, BGT-1 and BGT-2 ABCs, (b) BGT-2 ABC with higher function-space order.....	32
Figure 3.13. Finite element analysis and exact solution results compared with polar plots of real part, imaginary part and magnitude of the scattered wave on a sound-soft circular cylinder.....	33

Figure 3.14. Comparison for (a) number of cells n generated to discretization density n_λ , and (b) error performance with increasing function-space order, p , depending on mesh structure.	34
Figure 3.15. FEA results and error analysis for a sound-hard cylinder in a rectangular solution domain.....	36
Figure 4.1. Crescent-shaped scattering obstacle design parameters for (a) cylindrical, and (b) rectangular solution domains.	38
Figure 4.2. Parameter-sweep optimization result for a crescent-shaped single sound-hard body in a cylindrical solution domain.....	41
Figure 4.3. Parameter-sweep optimization algorithm flowchart.....	42
Figure 4.4. Nelder-Mead optimization algorithm flowchart.....	44
Figure 4.5. Nelder-Mead optimization result for a crescent-shaped single sound-hard body in a cylindrical solution domain.....	45
Figure 4.6. Examples of spatial lattices for various geometrical units.....	47
Figure 4.7. Cylinder solution obtained by utilizing the symmetry of the half-cylinder problem using a HNBC on the top boundary.....	49
Figure 4.8. Analysis of finite number of cylinders using combinations of homogeneous Neumann and Robin BCs on outer boundaries.....	50
Figure 4.9. Results for a rectangular solution domain.....	52
Figure 4.10. PBC application on horizontally symmetrical and periodic infinite cylinders problem.....	53
Figure 4.11. Vertically aligned infinite crescents modelled with PBCs.....	54
Figure 4.12. PS result for 1-D periodic crescent.....	55
Figure 4.13. NM optimization result for 1-D periodic crescent.....	56
Figure 5.1. Geometrical features of a conventional acoustic louver.....	59

Figure 5.2. Solution domain depending on the blade center point.....	60
Figure 5.3. Finite element model designed for acoustic louver analysis.....	62
Figure 5.4. Sound pressures at octave-band center frequencies for an acoustic louver.....	64
Figure 5.5. FEA results for the acoustic louver sections investigated, showing (a) the worst and (b) the best insertion loss performance.....	67
Figure D.1. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the parameter sweep algorithm.....	146
Figure D.2. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the parameter sweep algorithm.....	147
Figure D.3. Optimization results for a single crescent-shaped obstacle in a rectangular solution domain utilizing Sommerfeld ABC with the parameter sweep algorithm.....	148
Figure D.4. Optimization results for a single sound-soft crescent-shaped obstacle in a rectangular solution domain utilizing Sommerfeld ABC with the parameter sweep algorithm.....	149
Figure D.5. Optimization results for a periodic sound-hard crescent-shaped obstacle in a rectangular solution domain utilizing PBC and Robin BC with the parameter sweep algorithm.....	150
Figure D.6. Optimization results for a periodic sound-soft crescent-shaped obstacle in a rectangular solution domain utilizing PBC and Robin BC with the parameter sweep algorithm.....	151
Figure D.7. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the Nelder-Mead algorithm.....	152
Figure D.8. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the Nelder-Mead algorithm.....	152
Figure D.9. Optimization results for a single crescent-shaped obstacle in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.....	153

Figure D.10. Optimization results for periodic crescent-shaped obstacles in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.....	153
Figure D.11. Optimization results for periodic crescent-shaped obstacles in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.....	154
Figure D.12. Optimization results for periodic crescent-shaped obstacles in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.....	154
Figure D.13. Insertion loss for a conventional acoustic louver with $w = 100\text{mm}$ thickness (a) 1/3 octave band and (b) 1/1 octave band results.	156
Figure D.14. Insertion loss for a conventional acoustic louver with $w = 150\text{mm}$ thickness (a) 1/3 octave band and (b) 1/1 octave band results.	157
Figure D.15. Insertion loss for a conventional acoustic louver with $w = 200\text{mm}$ thickness (a) 1/3 octave band and (b) 1/1 octave band results.	158
Figure D.16. Insertion loss for a conventional acoustic louver with $w = 300\text{mm}$ thickness (a) 1/3 octave band and (b) 1/1 octave band results.	159

LIST OF TABLES

Table D.1. Octave Band Insertion Loss Data from Manufacturers' Brochures.....	155
---	-----



LIST OF ALGORITHMS

Algorithm 3.1. Function to define incident wave.	29
Algorithm 4.1. Geometrical constraints of a left-facing crescent-shaped body.	39
Algorithm C.1. Exact solution for a sound-hard cylinder.	100
Algorithm C.2. Exact solution for a sound-soft cylinder.	100
Algorithm C.3. Modified exact solution for a sound-hard cylinder adapted to BGT-2 ABC.	101
Algorithm C.4. Circular cylinder structured mesh.	102
Algorithm C.5. Circular cylinder in a cylindrical domain with a non-structured mesh.	104
Algorithm C.6. Circular cylinder in a rectangular domain.	105
Algorithm C.7. Rectangular domain with half cylinder on top.	106
Algorithm C.8. Crescent in circular domain.	107
Algorithm C.9. Crescent in rectangular domain.	108
Algorithm C.10. Rectangular domain with half cylinders on top and bottom.	109
Algorithm C.11. Rectangular domain with half cylinder on top and full cylinder in the middle of the domain.	111
Algorithm C.12. Rectangular domain with two cylinders in a vertical alignment.	113
Algorithm C.13. Louver section: Case 1.	114
Algorithm C.14. Louver section: Case 2.	116
Algorithm C.15. Louver section: Case 3.	118
Algorithm C.16. Mesh file conversion routine.	120
Algorithm C.17. Mesh data extraction routine.	121

Algorithm C.18. Scattering wave solution for a sound-hard cylinder in a cylindrical solution domain, utilizing Sommerfeld ABC.....	122
Algorithm C.19. Scattering wave solution for a sound-hard cylinder in a cylindrical solution domain, utilizing BGT-1 ABC.....	124
Algorithm C.20. Scattering wave solution for a sound-hard cylinder in a cylindrical solution domain, utilizing BGT-2 ABC.....	126
Algorithm C.21. Scattering wave solution for a sound-soft cylinder in a cylindrical solution domain utilizing Sommerfeld ABC.....	130
Algorithm C.22. Scattering wave solution for a sound-hard arbitrarily-shaped body in a cylindrical solution domain utilizing BGT-2 ABC.....	132
Algorithm C.23. Total wave solution for a sound-hard arbitrarily-shaped body in a cylindrical solution domain utilizing Sommerfeld ABC.....	135
Algorithm C.24. Scattering wave solution for a sound-hard arbitrarily-shaped body in a rectangular solution domain utilizing Sommerfeld ABC.....	136
Algorithm C.25. Scattering wave solution for a sound-hard symmetric body in a rectangular solution domain utilizing homogeneous Neumann boundary conditions on the horizontal, Robin boundary conditions on the vertical outer domain boundaries.....	138
Algorithm C.26. Scattering wave solution for a sound-hard symmetric body in a rectangular solution domain utilizing homogeneous Neumann boundary condition on the top, Robin boundary conditions on the left, bottom and right outer domain boundaries.....	139
Algorithm C.27. Scattering wave solution for sound-hard vertically periodic bodies in a rectangular solution domain utilizing periodic boundary condition on the top and bottom, Robin boundary condition on the left and right outer domain boundaries.....	140
Algorithm C.28. Objective function for the Nelder-Mead optimization.....	142
Algorithm C.29. Nelder-Mead optimization.....	143
Algorithm C.30. Decomposing sound pressure data to sound pressure spectrum levels...	144

Algorithm C.31. Calculation of 1/3 and 1/1 octave band insertion loss levels	145
--	-----



LIST OF SYMBOLS/ABBREVIATIONS

a	Louver blade vertical air space distance
b	Louver blade thickness
B	Adiabatic bulk modulus, Dirichlet-to-Neumann mapping term
c	Speed of sound
d_x	Offset distance between cropped and cropping cylinders
e	Change in volume
f	Source function, frequency
$f_c, f_{\min}, f_{\max}$	Octave band center, lower, and upper frequencies
i	Imaginary number
I	Integration area where the total wave or pressure is calculated
IL	Insertion loss
k	Angular wavenumber
L_p	Sound pressure level
L_b	Sound pressure level for a band of frequencies
n_r	Discretization in radial direction
n_t	Discretization in tangential direction
n_R	Normal on domain outer boundary
n_S	Normal on scattering body boundary
n_λ	Discretization density
O	Geometric center of the solution domain
O_1, O_2	Geometric center of the cropped and cropping cylinders
p	Pressure
p_b	Pressure for a band of frequencies
p_{ref}	Reference pressure level
R	Radius
R_1	Radius for the cropped cylinder
R_2	Radius for the cropping cylinder
R_{in}	Radius for a cylindrical body in a cylindrical solution domain
R_{out}	Radius for a cylindrical solution domain outer boundary

t	Time
TL	Transmission loss
$u, u_{\text{inc}}, u_{\text{s}}$	Total, incident, and scattered waves
$u_{\text{s}}^{\text{ex}}, u_{\text{s,hard}}^{\text{ex}}$	Exact solution for sound-hard body
$u_{\text{s,hard}}^{\text{ex,m}}$	Modified exact solution for sound-hard body
$u_{\text{s,soft}}^{\text{ex}}$	Exact solution for sound-soft body
v	Test function
V	Volume, particle velocity
w	Louver cross-section width
Z	Acoustic impedance
α	Louver blade angle
Γ_{P}	Periodic boundary condition
Γ_{R}	Robin boundary condition, absorbing boundary condition
Γ_{S}	Boundary condition on the scattering body
λ	Wavelength
ω	Angular frequency
Ω^+	Infinite solution domain
Ω	Finite (truncated) solution domain
ϕ	Scalar function, velocity potential
ρ, ρ_0	Density, static density of air
BC	Boundary condition
ABC	Absorbing boundary condition
AMCA	Air Movement and Control Association International, Inc.
ANSI	American National Standards Institute
BEM	Boundary element method
BGT	Bayliss-Gunzburger-Turkel absorbing boundary condition
CFD	Computational fluid dynamics
DtN	Dirichlet-to-Neumann absorbing boundary condition
EN	European standard

FEM	Finite element method, finite element model
HNBC	Homogeneous Neumann boundary condition
HVAC	Heating, ventilation, and air conditioning
ISO	International Organization for Standardization
OAR	Free area ratio, open area ratio
PBC	Periodic boundary condition
SPL	Sound pressure level
SPSL	Sound pressure spectrum level
WHO	World Health Organization



1. INTRODUCTION

There is a growing interest in acoustics as the production and simulation technologies improve, regulations and standards become more strict, and the impact of noise on human health is better understood. These factors have led to a greater demand for tools that can help professionals predict and control the acoustic properties of products and incorporate acoustic considerations into processes. However, not all engineers and architects have access to commercial software, and even if they do, it can take time to become proficient in using it. In such cases, tools that are easily accessible and able to provide in-depth analysis of complex technical information become increasingly important.

Open-source finite element software packages are typically available at no cost, which can save significant amounts of money compared to commercial software. These packages are typically highly flexible, allowing users to modify and extend them to meet their specific needs. This can be especially useful for solving complex or specialized problems that require specialized tools or capabilities. They are developed and maintained by a community of users and developers, who are often willing to provide support and assistance to other users. This can be a valuable resource for users who need help with the software or have questions about its capabilities. Because open-source software is distributed with the source code, users have access to the underlying code and can modify it as needed. This can be useful for understanding how the software works and for making customizations or improvements. Therefore, it is possible to create a solution package on noise control using open source finite element software.

This study aimed to improve the effectiveness of acoustic louvers by utilizing an open-source finite element platform, FEniCS [1], to model and analyze sound attenuation. Noise reduction has long been a concern for architects and engineers, but constructing nature-friendly buildings with more open space and natural ventilation presents a challenge because the openings in the facade increase noise exposure. The construction of high-density housing and other densely populated areas can lead to increased noise levels, particularly between different living spaces. To address this issue, it is important to take measures to reduce noise pollution, especially noise from mechanical rooms and shafts. To control noise in these areas, acoustic louvers are often employed.

In Chapter 2, a comprehensive literature review on louvers is included, covering various subjects such as their design and applications. However, there is relatively little published research specifically on acoustic louvers, with most of the available literature focusing on standard louvers.

A few studies have focused on investigating the acoustic performance of louvers and shutters used in natural ventilation systems, such as the work by Butera et al. [2], Cui et al. [3], Koffi et al. [4], and Oldham et al. [5]. In addition, Watts et al. [6] used a modified boundary element method (BEM) program and scaled models to study vertically louvred noise barriers, while Yadav et al. [7] used statistical energy analysis and experimental data to predict the noise reduction performance of a louvered enclosure. Laboratory and field measurements of the acoustical and airflow performance of interior natural-ventilation openings and silencers have been investigated by Bibby and Hodgson [8,9] who used open area ratio (OAR) as a single-number combined acoustical and airflow performance metric in their research.

Research on mufflers and silencers can also be found on the literature, which are structurally similar to acoustic louvers but typically have a larger cross-sectional area and higher sound reduction capacity to improve sound attenuation. Kirby et al. [10] studied the acoustic performance of a dissipative splitter silencer using both experimental and theoretical techniques, and presented IL predictions. Shinde et al. [11] explored the performance characteristics of different types of mufflers and proposed the use of computational fluid dynamics (CFD) methods for numerical analysis. There have also been several approaches using finite element modeling to analyze sound attenuation in lined and parallel-baffle rectangular ducts, including the work by Borelli et al. [12] and Antebas et al. [13], which both account for various physical properties of sound-absorbing materials and geometries of baffles.

Chapter 2 continues with a discussion on how to derive the wave equation and model it using numerical methods. It introduces two main approaches to defining sound fields: methods derived from solutions of the wave equation and geometric methods based on beam optics analogies. It compares these methods and explains why wave-based methods are preferred in this study due to their ability to handle scattering and diffraction. Detailed information about the software infrastructure and libraries used is also provided in this chapter.

In Chapter 3, a fundamental study on the solution of the acoustic scattering problem was conducted prior to noise control studies. This study involved examining sound absorbing boundary conditions which are used to solve the propagation of sound waves in outdoor spaces by truncating the infinite domain to a finite one. Review papers by Dan Givoli [14] and Thomas Hagstrom [15] about the working principles of these boundary conditions provide valuable information. These boundary conditions are based on advanced mathematical equations and can be complex to apply. In some cases, they may not be suitable for use depending on the type of problem being addressed. Therefore, an error analysis study was conducted to compare the performances of Sommerfeld, BGT-1 and BGT-2 absorbing boundary conditions (ABCs). The results of this study were used to inform the development of a finite element model (FEM) scheme used in studies with shape optimization as well as insertion loss calculations.

In Chapter 4, shape optimization studies are conducted in two-dimensional single and multiple scattering problems to minimize the total downstream wave for a geometry that meets the incident plane wave. The application of shape optimization to acoustic scattering problems has been frequently studied in recent years, and is used to optimize wave-structure interaction [16], reduce noise [17], and design high performance metamaterial geometries [18] for both single and multiple scattering obstacles. Here, first, the general behavior of the response function of the problem is observed by parameter scanning for a range of single and multiple scattering obstacles. In line with this information, it is aimed to determine the most efficient geometry with the solution scheme designed using the Nelder-Mead optimization method for both types of obstacles.

In Chapter 5, the performance of acoustic louvers is evaluated, including an analysis of the geometric and material properties of the most commonly used models in the industry. The software has been updated to calculate insertion loss and decompose results into standard octaves. The chapter presents results for a variety of commonly encountered acoustic louver models.

In Chapter 6, a summary of the main conclusions and proposed directions for future research are presented.

2. BACKGROUND

Noise has always been a significant environmental problem. Dense urbanization due to rapid population growth in the last few decades has increased this problem. According to the surveys conducted in 2016-2017, 32% reported problems with noise. In the European Union, 20% of the population is exposed to levels exceeding 55dB(A) which are disturbing to sleep [19].

Noise has negative impacts on human health and well-being. Significant evidence of environmental noise on health has been accumulated within the last two decades. World Health Organization (WHO) published community noise guidelines in 1999 [20], night noise guidelines for Europe in 2009 [21], and environmental noise guidelines for the European region in 2018 [19]. According to these publications, noise exposure may have a large temporary, as well as permanent impact on physiological functions. People may develop permanent effects, such as hypertension and ischaemic heart disease associated with exposure to high sound levels [20].

One of the major sources of the increasing environmental noise problem is the inevitable use of high-capacity equipment and installation elements in multi-storey buildings and mass housings in densely populated cities.

Specifying parallel baffle silencers, sound traps, sound-absorbing duct liners, vibration isolators for HVAC equipment, designing building elements with high sound insulation performances, or even changing the location of the noise source can be considered as noise control measures [22]. However, due to the monetary value of the spaces open to residents in a building, priority is usually given to reducing the volume allocated to mechanical rooms as much as possible. This creates the need for high performance noise control in a confined space, often adjacent to a noise sensitive area.

Noise source equipment to be considered in mechanical equipment rooms are typically air handling units, ventilation and exhaust fans, boilers, chillers, pumps, electrical transformers, compressors, and generators. Equipment as such need a significant amount of air to cool their engine. Air intake and exhaust openings that are left for cooling inevitably require the use of sound attenuators [23]. Using acoustic louvers is a common option in this case.

2.1. ACOUSTIC LOUVERS

The insertion of an acoustic louvre is a common acoustic treatment for a ventilation aperture. Unlike ordinary louver, they are building elements that also provide sound attenuation while regulating air flow and daylight entry. They can be used either as part of a building facade or an acoustic enclosure. They are mostly used in ventilation openings of equipment rooms and air shafts. An acoustic louver aims to provide the least interruption in air flow while screening the sound path to achieve the desired sound attenuation with its angled blades [22,24,25].

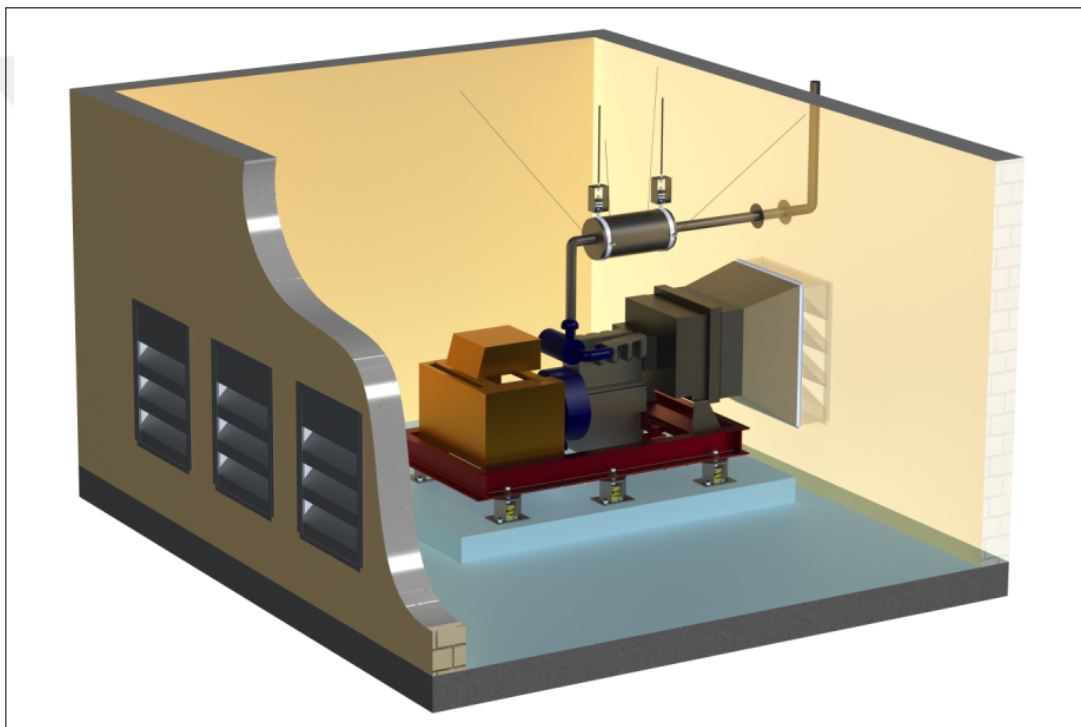


Figure 2.1. Acoustic louvres for acoustic treatment in a generator room.

A depiction of the uses of acoustic louvers in a mechanical room is given in Figure 2.1. As shown in the figure, acoustic louvers installed on the façade can reduce the transmission of noise from the mechanical room through the façade, while also providing critical benefits in the overall cooling of the mechanical room. An acoustic louver installed on the exhaust duct of equipment can also help to reduce noise transmission, cool the equipment, and direct transmitted sound. An acoustic louver may be relocated to a different façade in order to optimize its ability to reduce the transmission of noise based on the positions of the sensitive receivers and the impact of the current façade on noise reduction.

Most of the standards and specifications available are on standard louvers since they are more commonly used in building construction and are more integral to the overall performance of a building's envelope. Standard louvers, hereinafter simply referred to as *louvers*, are typically used to control the flow of air into and out of a building, which can have a significant impact on the energy efficiency and indoor air quality of the building. As a result, there are a number of industry standards and codes that apply to louvers to ensure that they are designed, manufactured, and installed in a way that meets certain minimum performance requirements and safety standards. For this reason, it will be necessary to make use of these resources first to understand some basic concepts and principles.

Louvers consists of a group of sloping blades that are designed to regulate airflow or light penetration. When installed in an opening it allows air flow but inhibits the entry of rain and sun light in the reverse direction [26]. Louvers are typically used in applications where ventilation is the primary concern. They may be installed in walls, ceilings, or other openings to allow for the flow of air into or out of a space.

In the industrial practice, EN 13030 and ANSI/AMCA 500-L standards are accepted for weather testing of louvers. Water penetration rate and pressure drop are the parameters indicated in these standards as performance characteristics of a louver. The water penetration rate provides a measure for the louvers' ability to prevent rain from entering through the louver, while the pressure drop provides information on how freely the louver allows air to pass through [27,28].

On the other hand, an acoustic louver is expected to provide sound attenuation in addition to the functions of a standard louver. It is either used at the end of a machinery for sound attenuation within a limited space where it is not possible to use a silencer or to support the fresh air ventilation where required. An acoustic louver is suitable for moderate noise control requirements as specified in ISO 14163 [29].

Acoustic louvers are typically rectangular in shape and are mounted in a frame that is attached to the opening through which the sound is being transmitted. An acoustic louver typically consists of a series of slats termed blades. The blades are spaced apart from one another to allow sound to pass through the acoustic louver, but the angle of the blades and the spacing between them helps to reduce the transmission of sound energy.

The blades of an acoustic louver are typically filled with an absorbing material such as fiberglass, and are oriented at an angle to the direction of the incoming sound. The blades are typically perforated from underneath in order to increase the surface area of the blades that is exposed to the incoming sound waves. This helps to improve the acoustic louver's ability to absorb sound energy and reduce the transmission of sound through the louver. A three-dimensional cross-section of a typical acoustic louver is given in Figure 2.2.

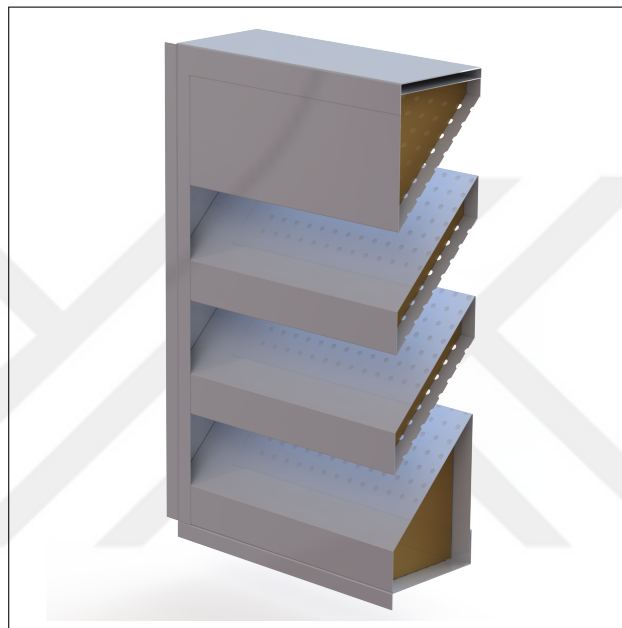


Figure 2.2. Render of a sectioned conventional acoustic louver.

Ensuring that they are long-lasting and low-maintenance is another important factor in their design. Galvanized steel and aluminum are therefore commonly used materials in the construction of acoustic louvers due to their durability, strength and corrosion resistance. Galvanized steel is a strong and durable material that is well-suited for use in acoustic louvers due to its ability to withstand harsh environmental conditions. Aluminum, on the other hand, is preferred because of its strength and durability, as well as its lightweight and corrosion resistance.

Using resistive or reactive sound attenuators are also highly preferred noise control measures at the ventilation inlet or outlet of a duct-work, since acoustic louvers obstruct airflow more than a similarly sized opening due to the reduced airway and resistance of the blade surfaces to airflow. The main design criterion is to balance these factors before making the decision to install an acoustic louver [25].

2.2. DIFFRACTION AND SCATTERING FROM PERIODIC STRUCTURES

Acoustic louvers show poor sound attenuation performance at low frequencies and better at high frequencies due to the sound diffraction around their blades [25]. Diffraction can be defined as the spreading of waves with no change in the average propagation direction. Another term that is often confused with diffraction or sometimes used interchangeably is scattering. It can be said that the main difference of scattering is the clear change of propagation direction of a deflecting wave [30]. Diffraction mainly happens for two reasons: either the wave confronts a boundary that acts like an obstacle or the wave passes through a boundary between two different materials with different impedances [31]. Diffraction concerns perturbations whose dimensions are large compared to a wavelength. Unlike diffraction, acoustic wavelengths are of the order of obstacles for scattering.

Research on acoustics simulation focuses on accurately modeling wave propagation effects such as diffraction and/or scattering. The accuracy of the modelling of these phenomenon depends on the modelling approach preferred. The two main approaches in acoustic modeling are the geometrical acoustics and the wave-based [32].

In geometrical acoustics, sound is assumed to behave like rays, hence the wavelength of sound is neglected. This means that all wave-based phenomena, such as diffraction and scattering, are missing from this numerical approach.

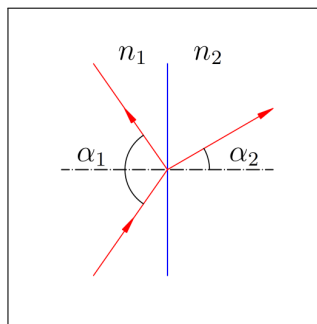


Figure 2.3. Illustration of Snell's law of refraction for a ray of light.

Ray tracing is based on the idea of discrete sound rays propagating in straight lines along which the acoustic energy is transported. The rays are reflected from boundaries according to Snell's law similar to light propagation studies using optical rays. Sound energy is associated with the number of rays striking a receiver volume typically shaped as a sphere [33].

According to the ray tracing algorithm, when a sound wave hits an object, some of it is reflected, some of it is absorbed, and some of it is transmitted, as shown in Figure 2.3. Considering a single wavelength of sound, the incident ray reflects with the same angle it hits the surface of the object; the absorbed sound energy converts into heat energy; the transmitted sound wave refracts as it passes through materials with different refractive indices. The path of the ray and the refractive indices of the materials are related to Snell's law,

$$\frac{\sin \alpha_1}{n_1} = \frac{\sin \alpha_2}{n_2}, \quad (2.1)$$

where n_1 and n_2 represent the indices of refraction for the two substances, α_1 is the angle of incidence, and α_2 is the angle of refraction that the sound ray makes with the perpendicular line at the boundary [34].

Since scattering and refraction are typical wave phenomena, they are not accounted for by basic simulation algorithms such as ray tracing. Such wave phenomena can be introduced into the ray tracing algorithms as statistical properties, but a serious increase in computation times also arises [33].

In wave-based modelling, wave equation is solved numerically. They are computationally intensive and the workload increase proportionally with the increasing frequency. That is why these techniques have so far been found to be more suitable for low frequency analysis. Though, with the developments in recent years, good results can also be obtained at high frequencies. Regardless, reliably solving the wave phenomena such as scattering and diffraction is possible with wave-based models [33]. For these reasons, it is necessary to understand the wave equation first.

2.3. THE LINEAR WAVE EQUATION

The discovery of the wave equation has a rather exciting history to follow up, including key figures of scientific revolution such as Isaac Newton (1643-1727), Daniel Bernoulli (1700-1782), Joseph-Louis Lagrange (1736-1813), Jean le Rond d'Alembert (1717-1783), and their works on vibration of strings, bars and membranes, hydrodynamics, and other mathematical applied sciences. However, it would be more appropriate to follow up the history directly from the part that concerns this study.

In December 1759, Leonhard Euler (1707-1783), wrote three remarkable papers on mathematical acoustics in which he derived the linear three dimensional wave equation from his previous works on hydrodynamics [35]. To reach this equation, Euler used three equations and relevant assumptions that are briefly explained below.

In an ideal fluid, which is in a stable equilibrium, small perturbations in pressure, $p = p_0 + p_1(x, t)$, and density, $\rho = \rho_0 + \rho_1(x, t)$ propagate as waves.

First, The linearized continuity equation connects the relationship between the particle velocity $\mathbf{u}$ and the change in volume with $e = \Delta\rho/\rho = -\Delta V/V$.

$$\frac{\partial e}{\partial t} + \nabla \cdot \mathbf{u} = 0, \quad (2.2)$$

Second, adiabatic bulk modulus, B , that comes from thermodynamics constructs the linear relationship between the pressure fluctuation p and the change in density e ,

$$B = p/e, \quad (2.3)$$

Third, the linear Euler's equation which is basically a force equation derived from Newton's second law applied to a fluid element.

$$\rho_0 \frac{\partial \mathbf{u}}{\partial t} = -\nabla p. \quad (2.4)$$

Combining these three linearized equations yield to the lossless wave equation which is a

single differential equation with one dependent variable,

$$\nabla^2 p = \frac{\rho_0}{B} \frac{\partial^2 p}{\partial t^2}. \quad (2.5)$$

Defining the thermodynamic speed of sound as $c^2 = B/\rho_0$, a more familiar format such as, $\nabla^2 p = 1/c^2 (\partial^2 p / \partial t^2)$, can be obtained for Equation (2.5).

Derivation of Equation (2.5) never required a restriction with respect to spatial variables on B or ρ_0 . Therefore, Equation (2.5) is valid for wave propagation in any media with the speed of sound as a function of spatial variables. A similar looking wave equation but for the vector field particle velocity can be written as:

$$\nabla^2 \mathbf{u} = \frac{1}{c^2} \frac{\partial^2 \mathbf{u}}{\partial t^2}. \quad (2.6)$$

Particle velocity is also irrotational. Hence, $\nabla \times \nabla p = 0$ requires $\nabla \times \mathbf{u} = 0$. This means that particle velocity can be expressed as the gradient of a scalar function. This scalar function is called the velocity potential, ϕ ,

$$\mathbf{u} = \nabla \phi, \quad p = -\rho_0 \frac{\partial \phi}{\partial t}. \quad (2.7)$$

Regardless of coordinate system or number of dimensions, ϕ , may also represent gravitational potential in free space, the electrostatic potential in a uniform dielectric, the magnetic potential in free space, the electric potential in the steady flow of currents in solid conductors [36]. Thereby, a general scalar wave equation for waves propagating through a homogeneous medium can be written as,

$$\nabla^2 \phi = \frac{1}{c^2} \frac{\partial^2 \phi}{\partial t^2} \quad (2.8)$$

2.4. HELMHOLTZ EQUATION

The linear wave equation which is useful for modelling acoustic waves propagating through air can be written with a term $F(t, \mathbf{x})$ with $\mathbf{x} := x, y, z$ to define noise source as

$$\frac{\partial^2 \phi}{\partial t^2} = c(\mathbf{x})^2 \nabla^2 \phi + F(t, \mathbf{x}). \quad (2.9)$$

Considering only time-harmonic motions,

$$\phi(t, \mathbf{x}) = u(\mathbf{x})e^{-i\omega t}, \quad F(t, \mathbf{x}) = f(\mathbf{x})e^{i\omega t} \quad (2.10)$$

then the scalar wave equation with the source becomes

$$-\omega^2 u(\mathbf{x})e^{-i\omega t} = c(\mathbf{x})^2 \nabla^2 u(\mathbf{x})e^{-i\omega t} + f(\mathbf{x})e^{-i\omega t}. \quad (2.11)$$

Thus, the harmonic time dependence turns the wave equation into Helmholtz equation,

$$\nabla^2 u + k^2 u = f, \quad (2.12)$$

where f represents the source term, and the *wavenumber* k is defined by

$$k = \omega/c. \quad (2.13)$$

Expressing the wave function as a time-harmonic function can be viewed as a special case of choosing a set of solutions that are eigenfunctions of the time translation operator. This form of the wave function has several advantages, including its periodic nature, which allows for the straightforward analysis of the wave's behavior over time, and the fact that it can be broken down into a series of sinusoidal components, which allows for the analysis of the wave's frequency content.

2.5. THE FINITE ELEMENT METHOD

The finite element method (FEM) is the most commonly used numerical prediction technique for solving differential equations. It has been proven to be versatile in terms of finding the distribution of field variables in a continuum domain, governed by an appropriate set of partial differential equations and boundary conditions. They have been long used for solving acoustics of enclosed spaces such as cavities or rooms. Solving time-harmonic acoustic problems with FEM is not an exception. They are also used in exterior acoustic problems that require a computational domain unbounded in space [37].

The FEM relies on first transforming the original problem into an equivalent integral formulation, then approximating the field variable distributions and the geometry of the continuum in terms of a set of shape functions that are locally defined within the subdomains of the continuum field. A central aspect of the finite element method is to construct such subspaces by patching together local function spaces defined by a set of finite elements.

Ciarlet [38] defines a finite element as:

- (i) $K \subseteq \mathbb{R}$ be a bounded closed set with nonempty interior and piecewise smooth boundary (the **element domain**)
- (ii) $V = V(K)$ be a finite-dimensional space of functions on K (the space of **shape functions**), e.g. a polynomial of order p in K , and
- (iii) $N = \{N_1, N_2, \dots, N_k\}$ be a basis for V' (the set of **nodal variables**).

Then (K, V, N) is called a **finite element**. The basis functions $\{\psi_1, \psi_2, \dots, \psi_k\}$ of the space V is called the **nodal basis** of V and are defined by $N_i(\psi_j) = \delta_{ij}$ where δ_{ij} is the Kronecker delta. The basis functions ψ_i is often called the trial functions. For a finite-dimensional space the solution u can be written as:

$$u = \sum_{i=0}^{N-1} u_i \psi_i(x, y), \quad (2.14)$$

where u_i are the coefficients to be determined and are often called the **degrees of freedom**.

This results in $N + 1$ unknowns to be determined in u . A test function v is introduced to the differential equation to obtain the necessary $N + 1$ equations for solving this system of equations [39].

Solution to these complex problems require efficient software tools and numerical solution strategies. The practical application of FEM is known as finite element analysis (FEA). To do FEA, a software package must be used. Today, there are many commercial software packages as such are available for finite element analysis. However, these softwares have disadvantages such as having expensive license fees and not giving their users full control over the solution. The methods used in solution to these problems required non-standard routines and access to a deeper level of problem solving schemes. In this study, the FEniCS Project, which consists of free and open source software components that provides automatic solution of differential equations, was used. It automates significant portion of the finite-element assembly through the unified form language (UFL) it adopts, which is a domain-specific language for declaration of variational forms with a syntax that closely follows mathematical notation. FEniCS/DOLFIN libraries not only provide their users with full control over their formulations, but also provide a unique learning and development opportunity with platforms that bring users together over the internet and provide an environment for discussion.

Pre-configured images of the FEniCS project are available to user through a virtualisation technology, namely a container, which is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Docker is an open-source platform that provides such virtual environments. Docker virtual machine installation on Windows offers nearly the same experience and transparency as its users on Linux [40].

Once the FEniCS image has been successfully installed, another platform is needed to write code, and display results. Jupyter Notebooks, a module of Project Jupyter which was spun off from IPython, provides the perfect environment for this work. IPython (Interactive Python) is a command shell for interactive computing in multiple programming languages, originally developed for the Python programming language. Jupyter Notebooks has continued this feature by extending it with other programming languages. Since it is an application that runs on any browser, it offers a user-friendly and simple experience for these tasks.

3. MODELING THE ACOUSTIC SCATTERING PROBLEM

The acoustic scattering problem, which occurs when a time-independent planar sound wave hits an irregularly shaped object will be discussed in this chapter. This problem has applications in many fields such as radar, sonar, geophysical exploration, medical imaging and non-destructive testing. The objective of the scattering problem is to find the scattered wave u_s , from an object Ω_S , caused by an incident plane wave $u_{\text{inc}}(x) = e^{ikx}$ coming from infinity, as shown in Fig.3.1.

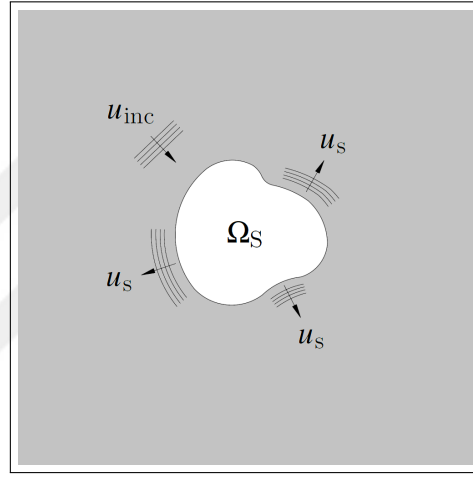


Figure 3.1. Acoustic scattering problem.

The total wave, u , is the summation of incident and scattered waves:

$$u = u_{\text{inc}} + u_s. \quad (3.1)$$

Scattering problem can be modeled as shown in Fig.3.2 and solved with utilizing the harmonic acoustic wave equation. Let Γ_S represent the surface of an arbitrarily shaped scattering object, for which $\Gamma_S := \partial\Omega_S$ where $\Omega_S \subset \mathbb{R}^2$ denote the domain representing the scattering object. The wave satisfies the Helmholtz equation,

$$\nabla^2 u_s + k^2 u_s = 0 \quad \text{in } \Omega^+, \quad (3.2)$$

where u_s is the acoustic pressure or velocity potential spatial component, k is the wavenumber, and Ω^+ is the unbounded solution domain.

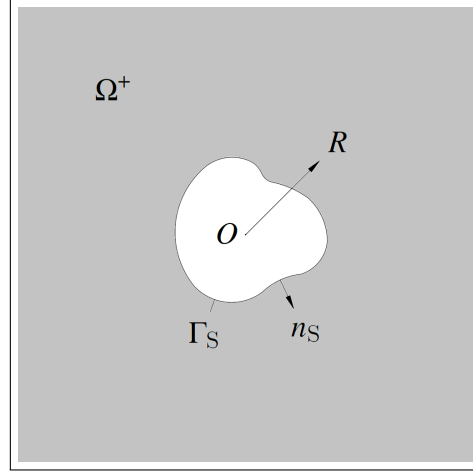


Figure 3.2. The unbound solution domain to a two-dimensional scattering problem.

For such an unbounded exterior Helmholtz problem, solutions must satisfy the Sommerfeld radiation condition,

$$\lim_{R \rightarrow \infty} \sqrt{R} \left(\frac{\partial u_s}{\partial R} - i k u_s \right) = 0, \quad (3.3)$$

where $i = \sqrt{-1}$ is the imaginary number and $R = |x|$ is the radial distance from the center. With this condition, only waves that diverge towards infinity are allowed.

If a sound-hard assumption is made for the scattering object, then the boundary condition will be:

$$\frac{\partial u_s}{\partial n_S} = -\frac{\partial u_{\text{inc}}}{\partial n_S} \quad \text{on } \Gamma_S, \quad (3.4)$$

where n_S is the normal to the surface of the scattering object.

Many solution strategies have been developed for this problem utilizing the finite element method, such as perfectly matched layer [41,42], absorbing boundary conditions [43,44,45], isogeometric analysis [46,47].

3.1. ABSORBING BOUNDARY CONDITIONS

Exterior problems such as acoustic scattering can be solved using the finite element method with ABCs. The unbounded domain, Ω^+ , is truncated by an artificial boundary, Γ_R , from a distance R to the origin, O , for an Eulerian coordinate system, yielding a bounded computational domain, Ω , as shown in Figure 3.3.

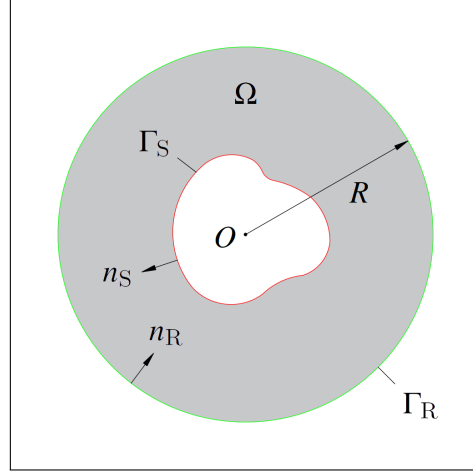


Figure 3.3. Scattering from an arbitrarily shaped object.

With the use of ABCs, the problem is solved only for a finite Ω solution space. The applied ABC on the boundary, $\Gamma_R = \{\mathbf{x} \in \mathbb{R}^2 : |\mathbf{x}| = R\}$, by artificially truncating the infinite domain, approximates the Sommerfeld's radiation condition at infinity. Correct far-field behaviour is then enforced by specifying proper boundary conditions on this boundary. The terms "radiation", "non-reflecting", "absorbing" boundary conditions are used in the context of such exterior wave problems.

The first complete finite-element approach for modeling infinite-domain acoustic radiation and scattering appears in the impedance matching technique presented by Hunt *et al.* [48,49]. Other examples to ABCs designed to adequately absorb incoming waves on the fictional diameter include, but not limited to Enquist and Majda [43], Feng [44] and Bayliss-Gunzburger-Turkel (BGT) [45].

Instead of Equation 3.3, in FEMs utilizing ABCs, the general form for this artificial boundary condition is defined as:

$$\frac{\partial u_s}{\partial n_R} - Bu_s = 0, \quad \text{on } \Gamma_R, \quad (3.5)$$

where operator B is the Dirichlet-to-Neumann (DtN) mapping term. Thus, the weak form of the Helmholtz equation,

$$\int_{\Omega} (\nabla u_s \cdot \nabla v - k^2 u_s v) \, d\Omega - \int_{\Gamma_R} Bu_s v \, d\Gamma_R = \int_{\Gamma_S} v \frac{\partial u_{\text{inc}}}{\partial n_s} \, d\Gamma_S, \quad (3.6)$$

can be written by multiplying the weighting function v with all terms [37,50]. Detailed

derivation of the weak form is given in Appendix B.1.

The simplest ABC application on the fictional outer boundary is to define the Sommerfeld radiation condition,

$$\frac{\partial u_s}{\partial n_R} - iku_s = 0, \quad \text{on } \Gamma_R, \quad (3.7)$$

as a Robin boundary condition, on radius R instead of infinity.

For the two-dimensional Helmholtz equation, an n^{th} order BGT ABC can be written in its general form as:

$$B_n = \prod_{j=1}^n \left(\frac{\partial}{\partial r} + \frac{2j - 3/2}{r} - ik \right) \quad \text{for } n \geq 1. \quad (3.8)$$

Hence, weak form equations can be formed by substituting the B operator in Equation 3.6 with the ABC preferred. Detailed weak form derivations utilizing Sommerfeld, BGT-1 and BGT-2 ABCs are given in Appendix B.1.1.1, Appendix B.1.1.2 and Appendix B.1.1.3 respectively.

3.2. ERROR ANALYSIS

All computational results obtained with FEMs involve numerical errors to some extent. They can be divided into three categories: domain approximation errors which is caused by spatial and temporal discretization of the physical geometry of the system, approximation errors caused by approximating the exact solution by a series of nodal values of the solution, quadrature and finite precision errors coming from evaluating the integrals numerically in the solution process with a finite number of individual deterministic solutions [51,52].

Error estimation procedures can be grouped into two different types as *a priori* and *a posteriori* estimations. *A priori* (before the solution of the problem) error estimates provide qualitative information about the asymptotic rate of convergence of a particular FEM solution, but they do not give quantitative information about the local or global errors of the solution. In contrast, *a posteriori* (after the solution of the problem) error estimation aims at developing quantitative methods in which the error is estimated in post-processing procedures using the numerical solution as data for the error estimates, providing confirmation of accuracy [51,52]. *A priori* error estimation procedure was not integrated into this study. For this reason, *a posteriori* local and global error estimations were taken into account.

In acoustic scattering problem the major error mechanisms are domain truncation and pollution errors [47]. Domain truncation error is caused by truncating the infinite solution domain of wave propagation with a fictional boundary. The mathematical model used in the approximate representation of the boundary geometry directly affects the amount of error. Using absorbing boundary condition enrichment techniques is an option to reduce this error, but numerical analyzes to date have revealed that the domain truncation error is higher than the discretization error [53]. As a consequence, the approximate solution is affected to some degree by spurious reflection from the artificial boundary. It is necessary to use higher order ABCs to eliminate these errors, but that also requires correct calculation of high-order derivatives. Pollution error on the other hand, is type of a numerical dispersion that causes a phase shift in the numerical solution. Dispersion occurs when the ABCs do not accurately model the dispersion of the wave as it propagates through the solution domain. This can cause the wave to become distorted and exhibit different properties than it would in the real case, leading to errors in the simulation results. To alleviate the pollution error, the mesh density, n_λ should be increased faster than the wavenumber, k , which leads to an increase in overall computational cost. This requirement practically limits the FEM to a frequency upper bound [46]. As a result, acoustic scattering problem suffers from both the first and the second of the above-mentioned categories of numerical errors.

The performance of a finite element model can be evaluated by comparing numerical results obtained for a simple geometry with its exact solution available in literature. Numerical solution with the FEM, f^h , on the finite-dimensional subspace, $V_h \in H^1(\Omega_R)$, is never equal to the exact solution, f^{ex} .

The relative L_2 error can be estimated by [53]:

$$\epsilon_2 = \frac{\left\{ \int_{\Omega} |f^h - f^{\text{ex}}|^2 \, dx \right\}^{1/2}}{\left\{ \int_{\Omega} |f^{\text{ex}}|^2 \, dx \right\}^{1/2}}. \quad (3.9)$$

Using this metric, the minimum error that can be expected for each refinement made in FEMs can be evaluated.

3.2.1. Exact solution

The problem of acoustic scattering for simple shapes has been solved by several different mathematical techniques for rigid or elastic spherical solids and shells [54,55], infinite cylinders [54,56], and spheroids [57,58]. Exact solution for the scattering from a circular cylinder is going to be used to validate the finite element model here.

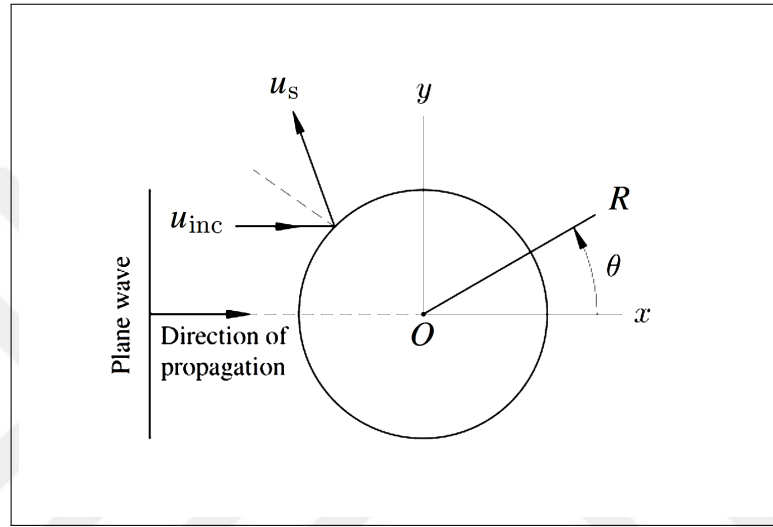


Figure 3.4. Scattering of a plane wave from a circular cylinder.

Consider the scattering of a plane, harmonic sound wave, propagating normally to the axis of a circular cylinder of radius R_{in} . The pressure field is independent of the axial direction. Therefore, the problem can be defined two-dimensional with Cartesian coordinates (x, y) , the origin O at the center of the cylinders cross-section, and plane coordinates (R, θ) as shown in Fig. 3.4.

Exact solution for this problem can be obtained by using the separation of variables method on acoustic wave equation [59,60] for both sound-hard and sound-soft bodies. A sound-hard body is an object that is solid and does not deform or vibrate when a sound wave strikes it. In other words, a sound-hard body is one that does not absorb or dissipate energy from a sound wave, but rather reflects the sound wave back in the direction it came from. A sound-soft body is the opposite of a sound-hard body. It is a term used to describe an object that is able to absorb or dissipate energy from a sound wave, rather than reflecting it back in the direction it came from.

The exact solution for sound-hard bodies is:

$$u_{s,\text{hard}}^{\text{ex}} = - \sum_{m=0}^{\infty} \epsilon_m i^m \frac{J'_m(kR_{in})}{H'_m(kR_{in})} H_m(kR) \cos [m(\theta - \alpha)], \quad (3.10)$$

where the cylindrical Bessel function is J_m , the cylindrical Hankel function is H_m , mode number is m , the wave incidence angle is α , Neumann factor ϵ_m is $\epsilon_0 = 1$ for $m = 0$, and $\epsilon_m = 2$ for $m \geq 1$. The detailed derivation of the exact solution for acoustic scattering of sound-hard bodies can be found in Appendix A.1. As an example, for $k = 25$, $kR_{in} = 1$, $\theta = 0$, and $m = 110$ the scattered wave is plotted for the sound-hard case as shown in Figure 3.5.

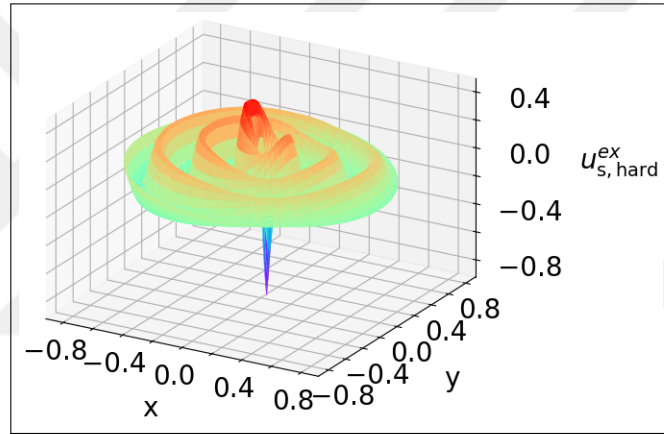


Figure 3.5. 3D plot of exact solution .

For sound-soft bodies, the solution is:

$$u_{s,\text{soft}}^{\text{ex}} = - \sum_{m=0}^{\infty} \epsilon_m i^m \frac{J_m(kR_{in})}{H_m(kR_{in})} H_m(kR) \cos [m(\theta - \alpha)]. \quad (3.11)$$

The detailed derivation of the exact solution for acoustic scattering of sound-soft bodies can be found in Appendix A.2.

Exact solution for both sound-hard and sound-soft cases can be calculated in the FEniCS environment with the Python codes given in Algorithm C.1 and Algorithm C.2 respectively. The solution is obtained by projecting onto a suitable mesh that can also be used for FEA and error analysis.

For a 2D circular cylinder problem with cylinder radius $R = 1$, wavenumber $k = 2\pi$, $m = 36$,

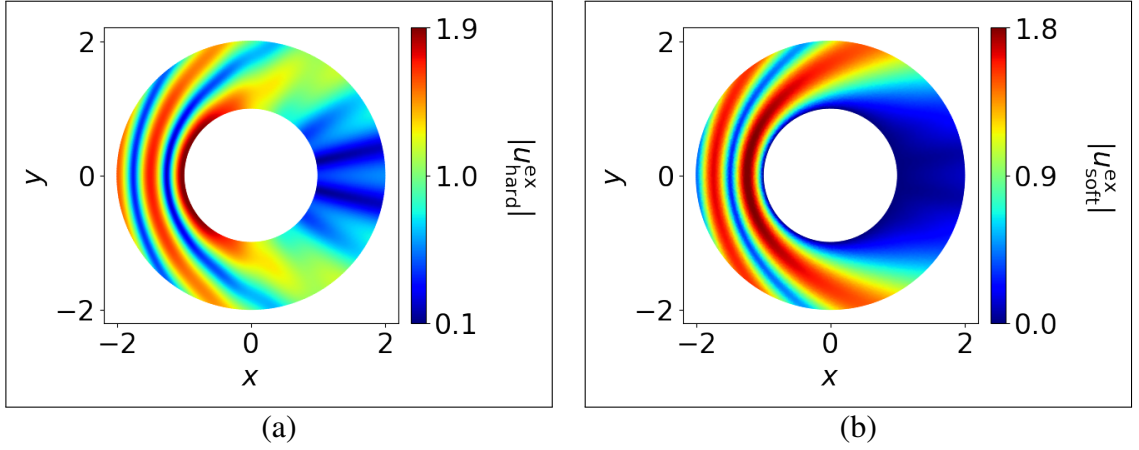


Figure 3.6. Exact solutions for (a) sound-hard and (b) sound-soft 2D circular cylinder acoustic scattering obstacles.

and discretization density $n_\lambda = 15$, sound-hard and sound-soft results are given in Figure 3.6.

3.2.2. The Modified Exact Solution

It is possible to obtain a modified exact solution and eliminate the domain truncation error for BGT-2 ABC as described in Khajah *et al.*[46]. The modified exact solution can be obtained as a summation of modal solutions

$$u_{s,\text{hard}}^{\text{ex},m} = \sum_{m \in \mathbb{Z}} i^m u_{s,m}^{\text{ex}}, \quad (3.12)$$

for which the series expansion is truncated by summing up m from $-m^{\max}$ to $m^{\max} := [kR_0] + 30$. A detailed expansion of Equation 3.12 can be found in Appendix A.3.

3.3. THE FINITE ELEMENT MODEL

By using different finite element models, performance of the absorbing boundary conditions in a variety of different scenarios, and how well each method performs under different conditions has been evaluated. This way, it was possible to identify the strengths and limitations of different absorbing boundary conditions, and to determine which methods are most appropriate for different types of acoustic scattering problems.

Results of the simulations are compared via relative error analysis while changing the type of ABC utilized, wavenumber, k , discretization density, n_λ , order of the basis functions, p , and the radius of the absorbing boundary, R_{out} .

The angular wavenumber, k is a measure of the rate at which a sound wave oscillates as it propagates through a medium. It has units of radians per meter (rad/m). One of the most challenging problems in scientific computing is achieving simultaneous accuracy and efficiency at high wavenumbers [61]. Employing enriched FEMs [62,63], using Padé-type artificial boundary conditions [64], making an Isogeometric Finite Element Analysis (IGA) [65,66,67,47], and utilizing Partition of Unity Isogeometric Analysis (PUIGA) [68] are some methods that are performing well in high-frequency FEA.

In the context of finite element modeling, discretization density refers to the number of control points per wavelength and is denoted with the symbol n_λ . It is a measure of the level of detail or resolution of the model, and is typically used to indicate the accuracy and computational efficiency of the model. This term, together with the basis function order, has a direct effect on the degrees of freedom. As a consequence, higher discretization densities generally result in more accurate models, but require more computational resources [69,46]. There are reported studies on achieving highly accurate results with low discretization density using methods such as Scaled Boundary Finite Element Method (SBFEM) [70], and Nystrom discretization [71].

The basis order, p , refers to the degree of the polynomial functions that are used to approximate the model. It is expected that higher-order basis functions to be more accurate than lower-order ones, but they also require more computational resources to evaluate. Therefore, the choice of basis order will be a trade-off between accuracy and computational efficiency [72,73].

For relative error analyzes to be made with a two-dimensional circular cylinder, let $\Gamma_S = \{x \in \mathbb{R}^2 : |x| = R_{in}\}$ be the scattering circle of radius R_{in} , and let $\Gamma_R = \{x \in \mathbb{R}^2 : |x| = R_{out}\}$ be the artificial boundary, yielding to the bounded finite computational domain $\Omega_R = \{x \in \mathbb{R}^2 : R_{in} < |x| < R_{out}\}$. Figure 3.7 depicts the mathematically described model.

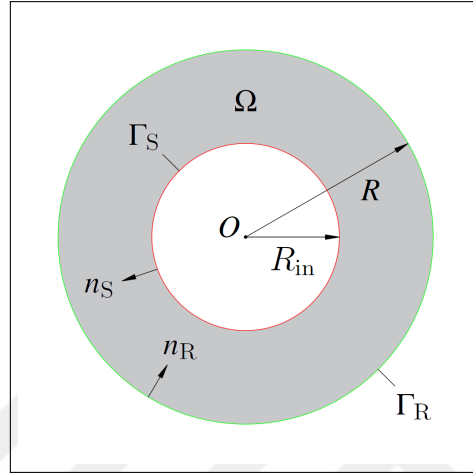


Figure 3.7. Scattering cylinder FEM truncated by ABC.

The results of FEAs have been compared with the exact solution which gave valuable information about both the accuracy of the models and the CPU requirements that were necessary for such analysis.

3.3.1. Meshing Strategy

Using a meshing strategy in finite element analysis allows for the analysis to be performed more efficiently, as the meshing process can be automated and the elements can be arranged in a way that reduces the computational burden of the analysis. This is also a requirement since we need to control element size for varying cylinder diameter, basis order, discretization density and wavenumber when generating meshes.

In order to control dispersion and resolve the waves accurately, the element size must be adapted to the wavenumber such that the discretization density, n_λ , is held below a resolution limit. An over-refined mesh may require excess computational power whereas a coarse mesh would not be able to resolve the waves. Stewart *et. al.*[74] suggest the use of a mesh with roughly three to six elements per wave. Lagrouche and Bettess [62] and Langdon *et. al* [75]

recommends using ten nodal points per wavelength. In general, the smaller the elements and the more closely they are spaced, the more accurately the wave will be resolved.

In order to model the acoustic waves scattered from a unit circular cylinder, first, a rectangular solution domain can be created to be able to generate a structured mesh, then polar transformation can be used to map the points of the rectangular domain onto a polar coordinate system with the inner radius as the origin.

To generate such a structured mesh, as a first step, the solution domain can be represented with a rectangle which is generated in $(\bar{x}, \bar{y})$ space such that $a \leq \bar{x} \leq b$ and $0 \leq \bar{y} \leq 1$. Any point $(\bar{x}, \bar{y})$ in this rectangular geometry can be mapped to a point $(\hat{x}, \hat{y})$ using a standard mapping from Cartesian polar coordinates. A rotation angle of $\Theta = 2\pi$ degrees would results with a full-circle hollow cylinder using the mapping formulas

$$\hat{x} = \bar{x} \cos (\Theta \bar{y}) \quad \text{and} \quad \hat{y} = \bar{x} \sin (\Theta \bar{y}) . \quad (3.13)$$

For a mesh with uniformly spaced coordinates $x_0, \dots, x_{M-1}$ in $[a, b]$, to map x onto $\xi \in [0, 1]$, the coordinate transformation $\xi = (x - a)/(b - a)$ can be used, so that

$$\bar{x} = a + (b - a) \left(\frac{x - a}{b - a} \right) \quad (3.14)$$

A circular cylinder with diameter R_{in} is used to represent the scattering object. The medium in which the waves travel is truncated by a boundary of radius R_{out} . The degrees of freedom can be characterized with $n_r = \lceil n_\lambda (R_{out} - R_{in}) k / 2\pi \rceil$ and $n_t = \lceil 2\pi n_r \rceil$ in the radial and tangential directions respectively. This approach enables a total control over the parameters relevant to the mesh domain. In Figure 3.8 (a), mesh for a wavenumber $k = 2\pi$ and $R_{out} = R_{in} + \lambda$ is shown. For such a case, the number of cells in radial direction matches the discretization density, n_λ , since wavelength is defined as $\lambda = 2\pi/k$. For the error analysis, structured meshes are generated with the outer radius at a unit distance to the scattering cylinder.

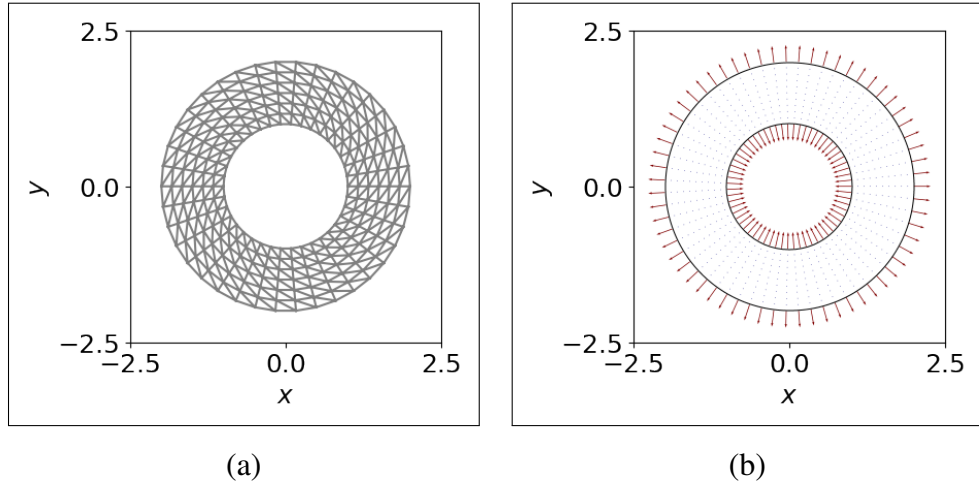


Figure 3.8. (a) A structured mesh with $R_{out} = R_{in} + \lambda$. (b) Boundary normals.

Throughout the entire study, an open-source meshing software, GMSH, has been used to generate meshes, except for structured mesh described above, with the Algorithm C.4, for which the Mesher library was used.

GMSH provides full control over all geometric parameters. Its scripting capability provides the ability to automate mesh generation processes and create complex mesh generation algorithms. Meshing parameters, such as the element size, element type, and other relevant details are defined through GMSH scripts.

GMSH's meshing algorithms handle the mesh generation process and FEniCS has the ability to read the meshes that have been generated by GMSH. In FEniCS, it is possible to mark the boundaries, and define boundary normals. As a default property of FEniCS boundary normals always point outward from the domain as demonstrated in Figure 3.8 (b). The GMSH codes written for all geometries throughout this study are given in Appendix C.3.

Installing GMSH libraries through Docker allows to the use of GMSH within the Jupyter environment, which provides a convenient and interactive way to create, edit, and visualize meshes in FEniCS.

3.3.2. Scattered Wave Solution

Solving for the scattered wave involves calculating the wave that is generated by the scattering objects. Scattered wave solution is useful for understanding the specific mechanisms by which the scattering objects interact with the incident wave. Scattered wave solution for weak form is given in Equation 3.6 and it's derivation can be found in Appendix B.1.

3.3.3. Total Wave Solution

The total wave solution gives a more real-life reconcilable result about acoustic scattering from a body. Total wave can be found by superposing the incident wave to the scattered wave solution. However, as another approach, Helmholtz equation can be modified by adding an inhomogeneity term that describes the incident wave on the solution domain, Ω , as shown in Equation 3.15 [76]. Detailed derivation for this equation is given in Appendix B.2

$$\int_{\Gamma_R + \Gamma_S} \mathbf{n} \cdot \nabla u \, d\Gamma - \int_{\Omega} \nabla v \cdot \nabla u \, d\Omega + k^2 \int_{\Omega} v u \, d\Omega = \int_{\Omega} v f \, d\Omega. \quad (3.15)$$

Total wave and scattered wave solutions can be used to validate the acoustic scattering model. Total wave solution provides the complete solution to the acoustic scattering problem, including both the incident and scattered waves. Scattered wave solutions, on the other hand, provide only the scattered wave component of the solution. Knowing the incident wave, comparing the predictions of both acoustic scattering models, it is possible to assess the accuracy of the model and identify any discrepancies or errors.

3.3.4. Incident Wave Generation

Plane waves are often used as the incident wave in acoustic scattering problems because they provide a convenient and well-defined starting point for analyzing the scattering of sound waves by an object. A plane wave is a type of sound wave that propagates through a medium in a straight line, without spreading out or diffracting. It is characterized by having a constant amplitude and phase across its cross-section, and it maintains its shape as it travels through

the medium. This makes it a useful idealization for studying the behavior of sound waves. Plane waves are commonly used in computational models of acoustic scattering, and are a fundamental concept in the study of wave phenomena in general.

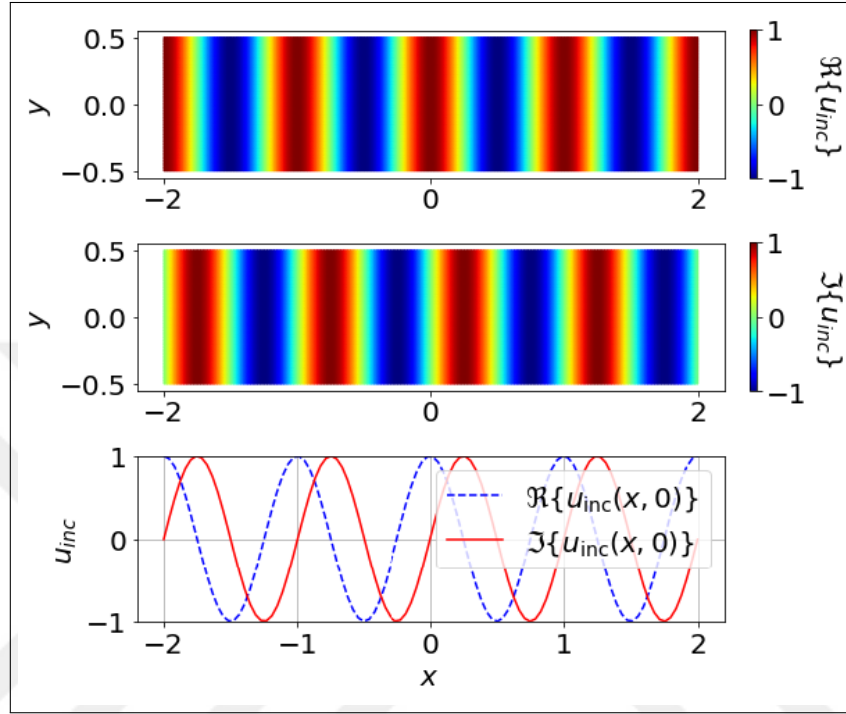


Figure 3.9. Real part and imaginary part contour plots of an incident wave plane wave with the phase shift observable.

Monochromatic waves are often used in finite element analysis. A monochromatic wave is a type of wave that has a single, well-defined frequency. This means that the wave oscillates at a single, specific frequency, without any other frequencies present [77]. The real and imaginary parts of a monochromatic plane acoustic wave have a phase because they are mathematically represented as sinusoidal functions. The real and imaginary parts of the wave, which are obtained by separating the complex-valued function can be observed in Figure 3.9.

3.4. FINITE ELEMENT ANALYSIS

The prepared finite element model was run with FEniCS v.2019.1.0, deployed by Docker Toolbox using Python 2.7.17, on a computer with 4GB of available RAM and an Intel® Core™ i5 M520 2.4GHz processor.

Jupyter Notebooks provided a user-friendly, interactive environment for writing and executing code. Therefore, all analyzes were performed using Jupyter Notebooks integrated with FEniCS. Open-source Python libraries `scipy` and `numpy` have been imported for scientific computing and data analysis. `Scipy` provides functions for optimization, interpolation, and numerical integration, while `numpy` provides support for multi-dimensional arrays and fast mathematical operations on these arrays [78,79].

Jupyter Notebooks also support the popular data visualization library, `matplotlib`, which has been used to create all the plots and other visualizations of the solutions.

Once the necessary FEniCS libraries are imported, Jupyter Notebooks have been used to write and execute the codes that perform the finite element analysis, including defining the mesh for the analysis, setting up the finite element space and boundary conditions, solving the weak form, and visualizing the results.

Unfortunately, UFL does not support complex numbers. UFL is a core component of the FEniCS software ecosystem. UFL is used to define the equations that are being solved in a FEM problem, and it provides a way to write these equations in a standardized, machine-readable format. FEniCS, in turn, uses these equations to generate the finite element matrices and vectors that are used to solve the problem [80]. Therefore, it was necessary to separate all the equations into real and imaginary parts that make up the linear system of coupled equations.

Algorithm 3.1. Function to define incident wave.

```

1 def Plane_Wave(k, x):
2
3     u_inc = np.cos(k*x) + 1j*np.sin(k*x)
4
5     return u_inc

```

Since the solution can either be obtained for the scattered wave or the total wave, it made sense not to introduce any errors by solving the incident plane wave with the FEM as well. The exact solution for the incident wave $u_{\text{inc}} = e^{ikx}$ has been solved on each mesh node with the code given in Algorithm 3.1.

3.4.1. Sound-Hard Circular Cylinder in a Cylindrical Domain

A comparison of FEMs utilizing Sommerfeld, BGT-1, and BGT-2 ABCs have been made via absolute and relative error analysis. The scattering of a time-harmonic incident wave from a single cylinder body in a two dimensional cylindrical domain is studied using the scattered wave solution. Radius of the scattering cylinder used is $R_{\text{in}} = 1$, radius of the outer domain where the ABCs defined is $R_{\text{out}} = 2$. Angular wavenumber has been taken as $k = 2\pi$.

Continuous Galerkin, or in other words, the standard Lagrange family of elements are used as both are synonyms in FEniCS [81]. For all the analysis made, the discretization density is $n_\lambda = 15$, polynomial degree of the basis functions on the elements is $p = 3$, and the expression degree is $q = 4$. An expression degree must be provided since the solution is represented in terms of a polynomial expression on the given general element and expressions are also used as coefficients in variational forms.[39]

Meshes used in these analysis are generated with the script given in Algorithm C.4. Exact solution for the acoustic scattering problem is solved using Algorithm C.1. The modified exact solution, which is valid only for BGT-2 ABC, was solved using Algorithm C.3 and projected on the mesh.

The finite element models require the solution of weak forms derived for each ABC, given in Appendix B, which are adapted to the FEniCS environment using the scripts given in Algorithm C.18, Algorithm C.19, and Algorithm C.20.

Polar plots of the scattered wave generated on the surface of the scattering cylinders obtained from the FEA results indicate that the numerical and exact solutions match; the error can be observed more clearly on the right side of the cylinder, that is, in the shadow area behind it; in Figure 3.10.

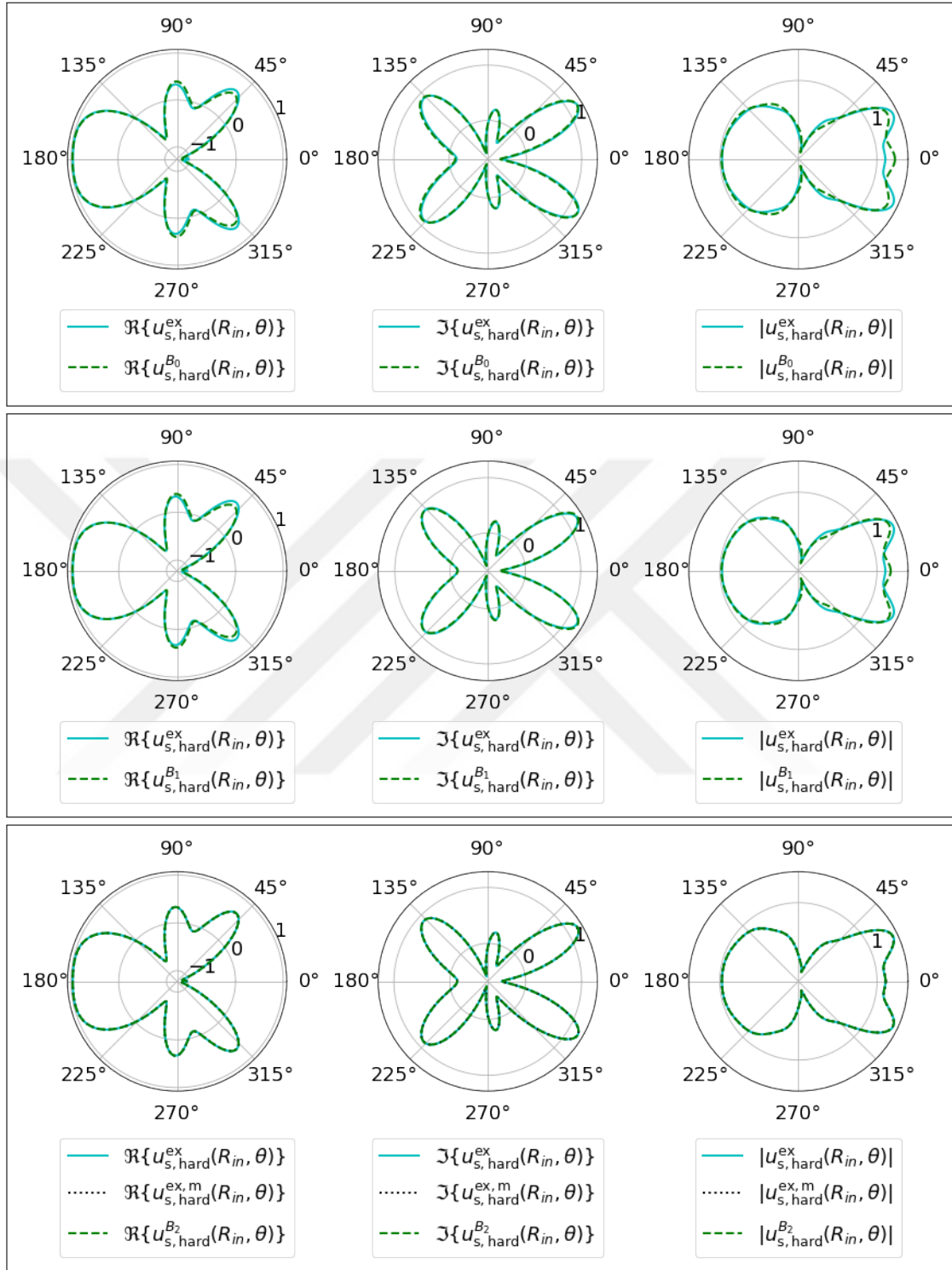


Figure 3.10. Polar plots of real part, imaginary part and magnitude of the scattered wave on the cylinder surface with (a) B_0 : Sommerfeld, (b) B_1 : BGT-1, and (c) B_2 : BGT-2 ABCs.

With the contour plots showing the absolute error, the difference between the exact and numerical solution in the shadow area can be seen more clearly. The amount of error obtained with Sommerfeld and BGT-1 is close to each other, but BGT-2 performed much better than both as can be seen in Figure 3.11.

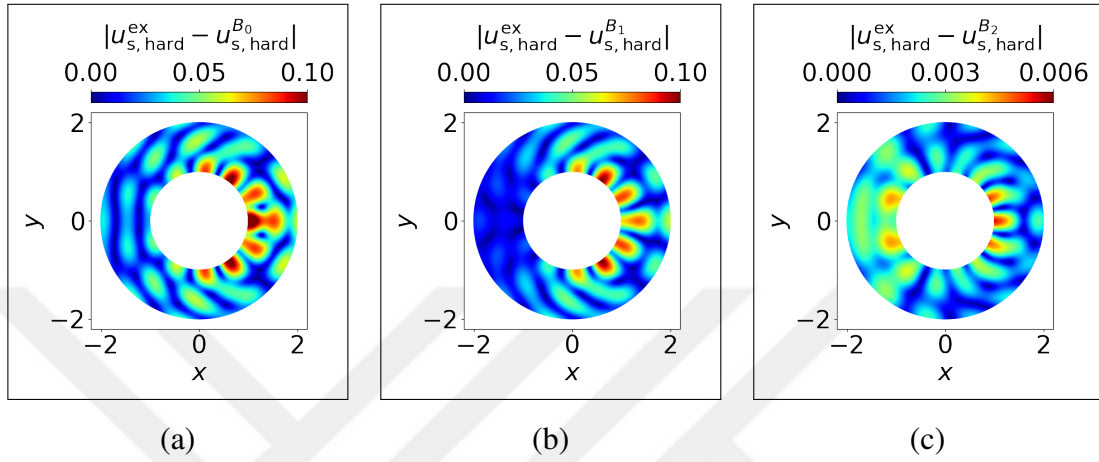


Figure 3.11. Absolute error comparison on solution domain for ABCs.

A more detailed study for error analyzes was done by varying the outer radius and the function-space order. Under the same numerical experiment conditions, the performance difference between ABCs become remarkable, especially with the increasing function-space order, p . BGT-2 ABC outperforms Sommerfeld and BGT-1 ABCs as expected.

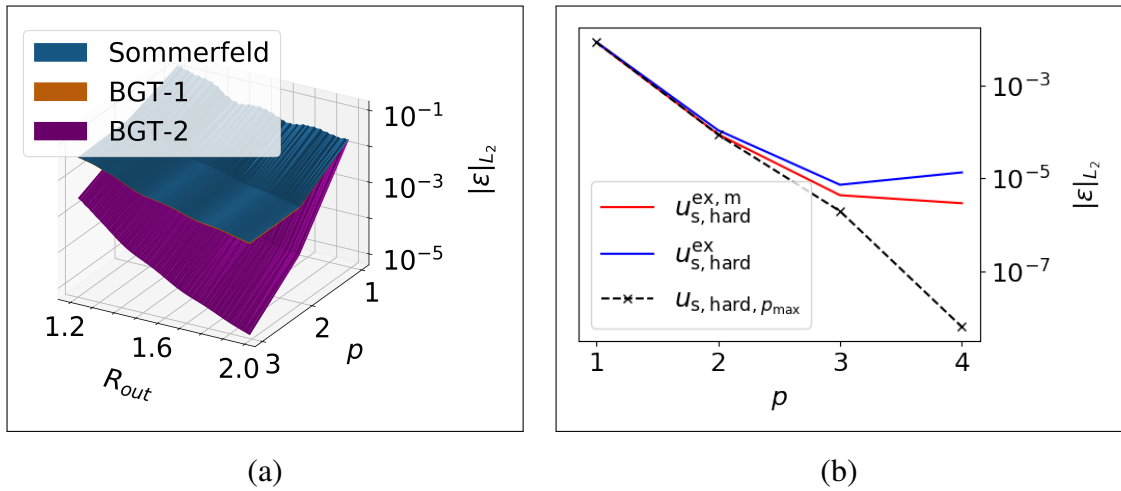


Figure 3.12. Error analysis showing the performance of (a) Sommerfeld, BGT-1 and BGT-2 ABCs, (b) BGT-2 ABC with higher function-space order.

3.4.2. Sound-Soft Circular Cylinder in a Cylindrical Domain

To obtain the weak form for a sound-soft scattering obstacle, the boundary conditions must be adapted accordingly. In the case of a sound-soft object, the Dirichlet boundary condition, $u = 0$ or $u_s = -u_{\text{inc}}$ must be used. Then the weak form of the Helmholtz equation becomes:

$$\int_{\Omega_R} (\nabla u_s \cdot \nabla v - k^2 u_s v) d\Omega_R - \int_{\Gamma_R} B u_s v d\Gamma_R - \int_{\Gamma_S} \mathbf{n} \cdot \nabla u_s v d\Gamma_S = \int_{\Gamma_\Omega} u_{\text{inc}} v d\Gamma_\Omega, \quad (3.16)$$

Notice that the incident wave, u_{inc} is defined on the solution domain, Ω instead of on the scattering obstacle, Γ_S , in Equation 3.16. Detailed derivation of the weak form is given in Appendix B.1.3.

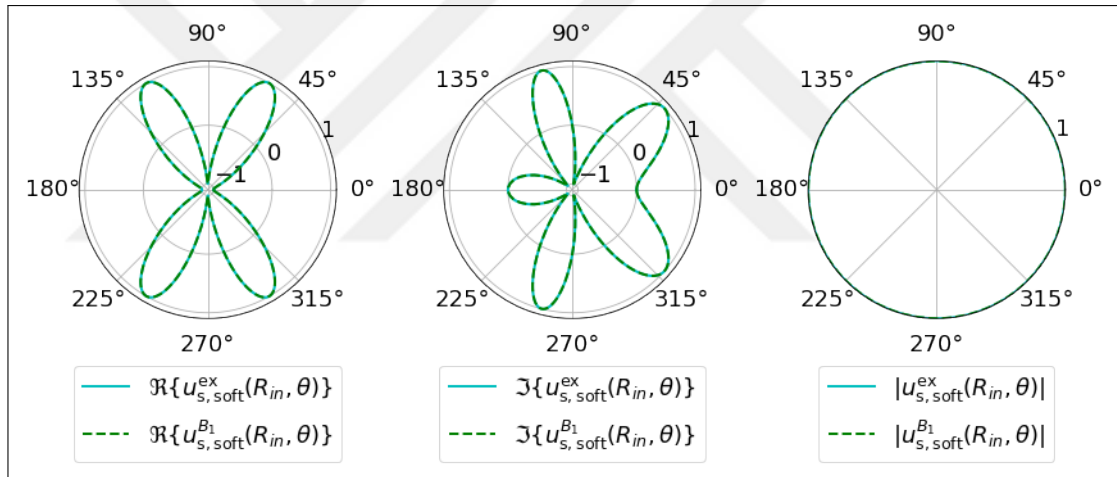


Figure 3.13. Finite element analysis and exact solution results compared with polar plots of real part, imaginary part and magnitude of the scattered wave on a sound-soft circular cylinder.

The exact solution result was obtained using Algorithm C.2, and the finite-element solution was obtained using Algorithm C.21. Both are solved on the same mesh generated using Algorithm C.4. The model is validated by comparing their polar plots of the scattered waves on the scattering cylinder boundary of both solutions as shown in Figure 3.13.

3.4.3. Non Structured Mesh Generation

It is not always possible to make FEA using structured meshes. Especially when the scattering occurs from an arbitrarily shaped obstacle. Non-structured meshes are generally easier and faster to generate, and they can often provide results that are just as accurate as those obtained using structured meshes. The choice between using a structured or non-structured mesh ultimately depends on the specific requirements of the analysis, the complexity of the geometry being analyzed, and the desired level of accuracy. Changing the shape of the outer boundaries of the solution domain also requires a careful construction of the mesh if a structured mesh is especially desired. For these reasons, a meshing strategy is developed to adjust discretization density, n_λ to a given problem geometry.

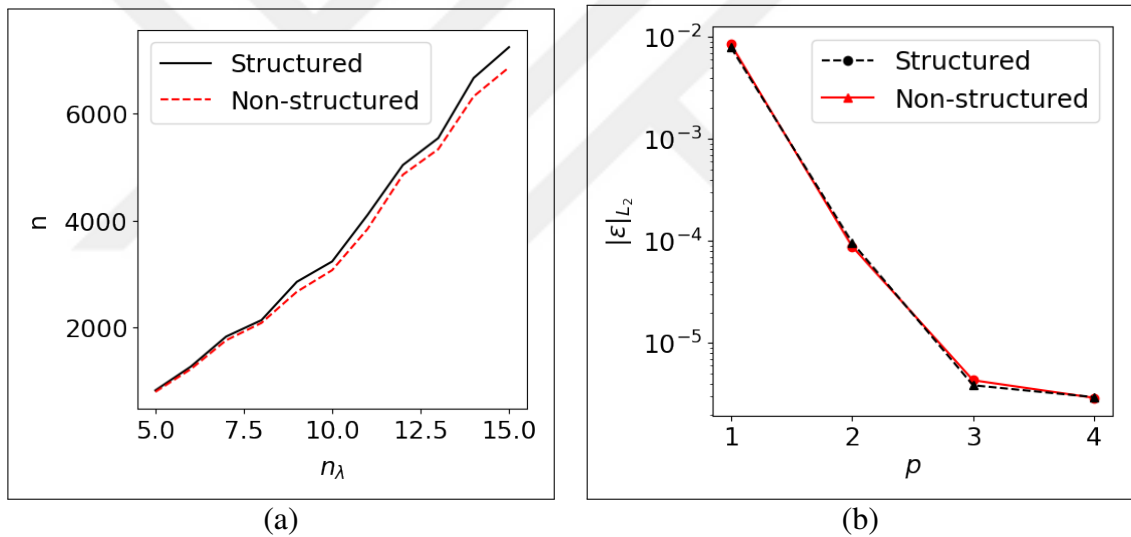


Figure 3.14. Comparison for (a) number of cells n generated to discretization density n_λ , and (b) error performance with increasing function-space order, p , depending on mesh structure.

An analysis made with the intention to show that the non-structured mesh algorithm is successful in matching the number of mesh elements with the structured mesh algorithm and the error is in acceptable levels as shown in Figure 3.14.

3.4.4. Cylinder in a Rectangular Domain

Comparing cylindrical and rectangular domain solutions for the same 2D acoustic scattering circular cylinder problem can provide insight into the differences between the two approaches. Throughout this study, what is expressed as a **cylindrical domain** or a **rectangular domain** actually refers to the shape of the truncating absorbing boundary on the outer perimeter of the solution domain. In general, using a "*cylindrical domain*" can be more efficient computationally because a symmetrical geometry can reduce the degrees of freedom (DOF) of the problem. However, a "*rectangular domain*" can be more flexible and allow for more complex geometries to be considered. Therefore, a rectangular solution domain with a cylinder body in its centroid is generated using the script given in Algorithm C.6.

The outer boundaries of the domain are defined as homogeneous Robin boundary conditions as:

$$V_n + \frac{1}{Z}u_s = 0 \quad \text{for } \Gamma_R, \quad (3.17)$$

where V_n denotes the particle velocity inward normal direction to the boundary Γ_R and Z denotes the acoustic impedance of the relevant boundary. The particle velocity can be linked to the pressure using Euler equation,

$$-i\omega\rho_0 V_n = \frac{\partial u_s}{\partial n}, \quad (3.18)$$

where ρ_0 denotes the static density of air. Introducing this into the Equation 3.17 yields:

$$\frac{\partial u_s}{\partial n} = -i\frac{\omega\rho_0 V_n}{Z}. \quad (3.19)$$

Free-field conditions can be simulated by matching the impedance of the boundary to the medium impedance of air, $Z_0 = \rho_0 c$. Since the wavenumber, k and the angular frequency, ω are directly proportional to each other, with the constant of proportionality being the speed of sound c , this boundary condition reduces to the Sommerfeld ABC derived before.

A scattering cylinder with a unit radius in a rectangular solution domain has been generated in GMSH using Algorithm C.6. The wavenumber, $k = 2\pi$, discretization density, $n_\lambda = 15$, and Lagrange elements with a polynomial degree of $p = 3$ has been chosen for the analysis. The FEM used for the analysis is given in Algorithm C.24.

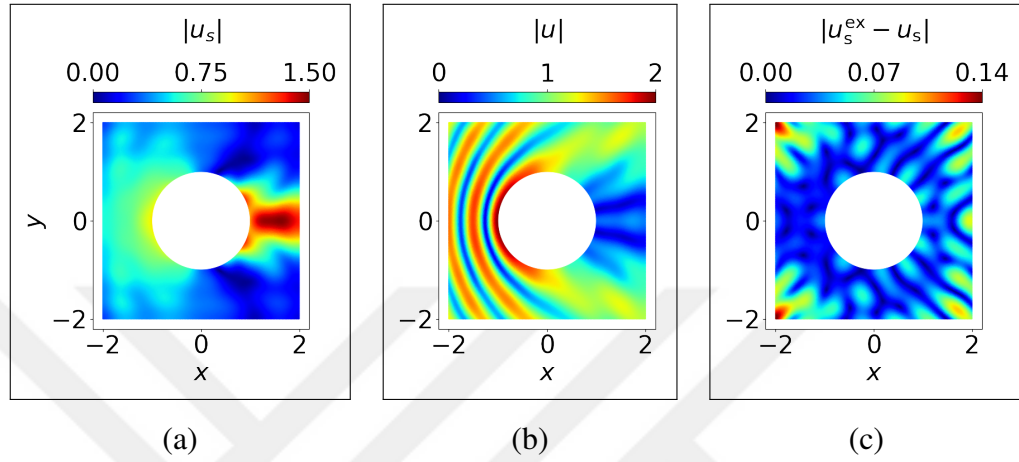


Figure 3.15. FEA results and error analysis for a sound-hard cylinder in a rectangular solution domain.

The solutions obtained for the scattered wave and the total wave in the rectangular solution domain that can be seen in Figure 3.15 (a) and (b) respectively match the visual results obtained with the exact solution given in Figure 3.6 (a). The absolute error contour plot given in Figure 3.15 (c) also proves this. The increase in the amount of error towards the corners is expected due to the problem of resolving the waves as they approach the corners of the solution domain.

4. SHAPE OPTIMIZATION

Shape optimization is often used in acoustic scattering studies to determine the optimal shape of a structure that will minimize the scattering of sound waves. This is important in many applications, such as in the design of noise barriers or in the study of how underwater structures can impact the transmission of sound. By minimizing the scattering of sound waves, the structure can be made more effective at reducing noise or allowing sound to pass through. Additionally, shape optimization can be used to minimize other effects of sound scattering, such as the absorption or reflection of sound waves.

Before making a shape optimization analysis for minimizing acoustic scattering, a number of decisions need to be made which can be summarized as: specifying the design variables that will be used to describe the geometry of the object, defining the constraints on the design variables, defining the objective function to be minimized, identifying the type of geometry optimization method to use, and selecting the numerical method to be used for solving the optimization problem [82].

The objective of this study is to minimize the acoustic pressure for a specified frequency at the downstream side of a scattering object. It was decided that a geometry that would serve as a Helmholtz resonator would be a good choice. A Helmholtz resonator consists of a cavity with a small opening or neck, and it works by resonating at a specific frequency determined by the volume and neck size of the cavity. When a sound wave with the same frequency as the resonant frequency of the resonator is incident on the resonator, it will be absorbed and dissipated as heat. This makes Helmholtz resonators effective at reducing the amplitude of specific frequencies of sound, which can be useful in noise control applications [83].

Helmholtz resonators are already commonly used in exhaust systems to reduce the noise produced by an engine [84], in industrial ventilation systems to reduce the noise generated by air moving through ducts [85], and in buildings to improve the acoustics and reduce the level of unwanted noise [86]. In contrast, an acoustic louver consists of a series of parallel baffles placed in the path of the incident sound. While this type of an attenuator can be effective at attenuating a broad range of frequencies, it is not specifically tuned to any particular frequency and is not as effective at attenuating a pure tone.

4.1. OPTIMIZATION FOR A SINGLE OBSTACLE

Interest in Helmholtz resonators has also increased with the development of acoustic metamaterials in recent years [87,88,89]. Their acoustic performances are being examined by taking into account the changes made in their placement and geometry in multiple array forms [90,91].

A novel geometry that acts like a Helmholtz resonator can be designed by defining a unit cylinder with a radius R_1 on the origin O_1 , and a second cylinder with a radius R_2 that crops this unit cylinder from a certain offset distance, d_x from its origin O_2 , which is also defined on the x-axis, as shown in Figure 4.1. This geometry resembles a Helmholtz resonator and all of its geometrical parameters can be controlled through these design variables.

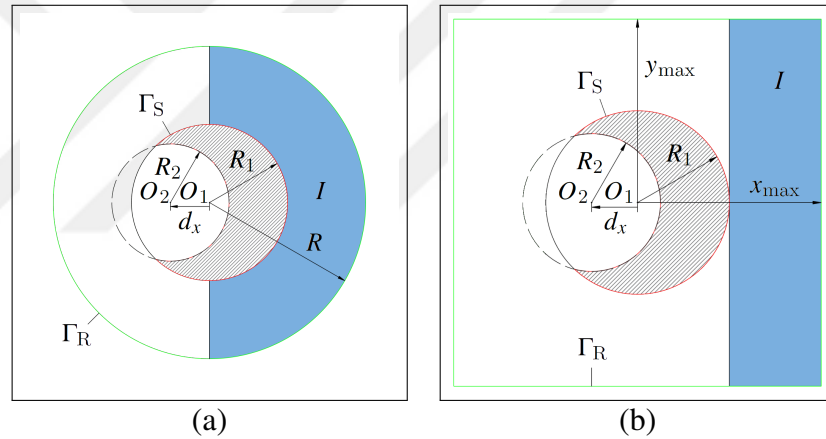


Figure 4.1. Crescent-shaped scattering obstacle design parameters for (a) cylindrical, and (b) rectangular solution domains.

Analyses were performed in both cylindrical and rectangular domains. Comparing the results obtained for both types of domains provided significant insight for the evaluation of the work done using the rectangular domain within this study. Meshes for the crescent in circular and rectangular domains have been created using Algorithm C.8, and Algorithm C.9, respectively.

Total wave solution at the downstream side of the incident wave, on the shaded area I , shown in Figure 4.1(a) and Figure 4.1(b) have been integrated. This area does not have a geometric interaction with the scattering body. Likewise, the outer boundaries of the solution domain were not changed during the optimization analyses. Therefore, this shaded area is a constant. Thus, the total wave solution calculated within this area was used as a performance metric,

and it is considered as the objective function in the optimization problem.

The absorbing boundary condition utilized on Γ_R is BGT-2 for the cylinder domain solutions and Sommerfeld for the rectangular domain solutions. Finite element analyses were performed using Algorithm C.22 for the cylinder and Algorithm C.24 for the rectangle domains. With these analyses, the absorbing boundary conditions is compared in terms of their performance when they are used for optimization.

Algorithm 4.1. Geometrical constraints of a left-facing crescent-shaped body.

```

1 def constraints(r_1, r_2, x_offset):
2
3     if (x_offset >= round(r_1 + r_2,3)):
4         # no intersection
5         geo_type = 1
6     elif (r_2 < r_1):
7         if (round(x_offset + r_2,3) <= r_1):
8             # internal cut
9             geo_type = 2
10        else:
11            geo_type = 0
12    elif (r_2 > r_1):
13        if (round(r_2 - x_offset,3) >= r_1):
14            # no scattering object
15            geo_type = 3
16        else:
17            geo_type = 0
18    else:
19        geo_type = 0
20
21    return geo_type

```

In order to construct an optimization algorithm, it was first necessary to determine the constraints that depend on the geometry of this particular scattering body. The position of the cropping cylinder relative to the cropped cylinder plays a significant role in defining these constraints. First, if the cropping cylinder completely covers the cropped cylinder, there won't be any scattering body to analyze. Second, if the cropping cylinder stays inside the cropped cylinder, the outer surface of the scattering body will simply still be a cylinder. Last, if the cropping cylinder does not have an intersection with the cropped cylinder, the scattering body will again be a regular cylinder. These constraints are applied with a short script given in Algorithm 4.1

In general, it is best to use multiple optimization methods and algorithms, and to carefully

validate the results of the optimization to ensure that the solutions obtained are correct and optimal. Since no analysis with a similar geometry could be found in the literature, first, brute force search was performed by sweeping all design variables. Later, Nelder-Mead (NM) optimization method, which is based on a more sophisticated and efficient algorithm, was modified and adapted to this problem, speeding up the optimization procedure.

4.1.1. Optimization with Parameter-Sweep Method

Parameter-sweep (PS), also known as brute-force search, is a method for optimizing a function by systematically exploring a range of possible values for the function's parameters. This is done by evaluating the function at a number of different parameter combinations, and selecting the combination that results in the best function value [92].

The parameter-sweep method is a simple and intuitive approach to optimization, in which the algorithm systematically evaluates all possible solutions to a problem and selects the one that provides the best result. While this method can be effective for small, simple problems, it becomes impractical for larger, more complex problems, as the number of possible solutions grows exponentially. Despite its limitations, it can be a useful approach when trying to optimize the performance of a model, especially when there are multiple parameters that can affect the model's performance and it is not clear which parameter settings will produce the best result.

In constrained optimization, the term "feasible region" is used to refer to the solution domain of the problem. The feasible region is the set of all possible solutions to the optimization problem that satisfy the constraints. It is the region of the solution space where the objective function is defined and can be evaluated. The feasible region is often represented graphically, with the constraints defining the boundaries of the region, and the objective function being plotted within the region [93]. A map of geometries that can be obtained on the feasible region is projected on the solution in Figure 4.2. This map makes it easier to understand the shape of the geometries investigated.

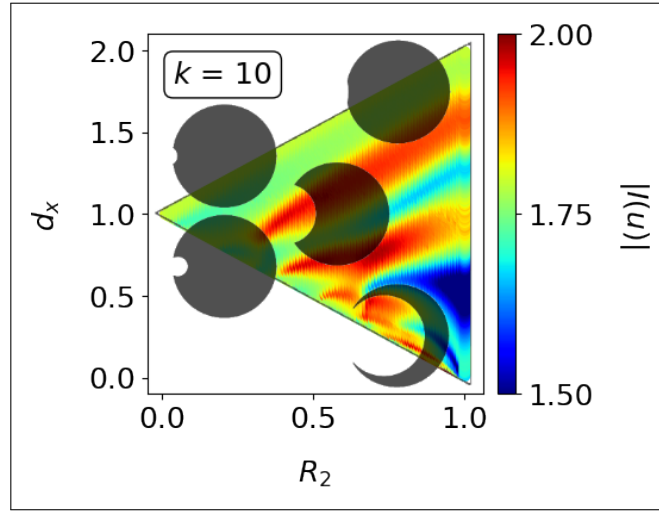


Figure 4.2. Parameter-sweep optimization result for a crescent-shaped single sound-hard body in a cylindrical solution domain.

The goal of the parameter-sweep optimization was to find the point in the feasible region that minimizes the objective function, subject to the constraints, for all possible solutions. Thus, the maximum sound attenuating geometry for a given wavenumber can be found.

The input variables of the optimization problem were determined as radius of the cropping cylinder, R_2 and x -axis offset distance of this cylinder to the center of the first cylinder, d_x . Input constants were determined as the diameter of the cropped cylinder, R_1 , domain dimensions, x_{Γ_R} , wavenumber, k , discretization density, n_λ , base function degree, p , and geometrical tolerance, tol . Algorithm flowchart is given in Figure 4.3.

All analyzes were repeated for wavenumbers, $k = 5$, $k = 10$, and $k = 15$. It was observed that bodies with similar geometry gave the minimum total wave result for the wavenumber range examined. As expected, the total wave has a much lower amplitude for the sound-soft objects, but the optimum geometry on the feasible region was obtained in similar regions. Complementary results are shared in Appendix D.1.

The results of PS has been used to validate the results obtained with the NM algorithm as well. The NM algorithm is designed to find good solutions to complex optimization problems more efficiently. It was possible to assess the accuracy and effectiveness of both algorithms by simply comparing their results.

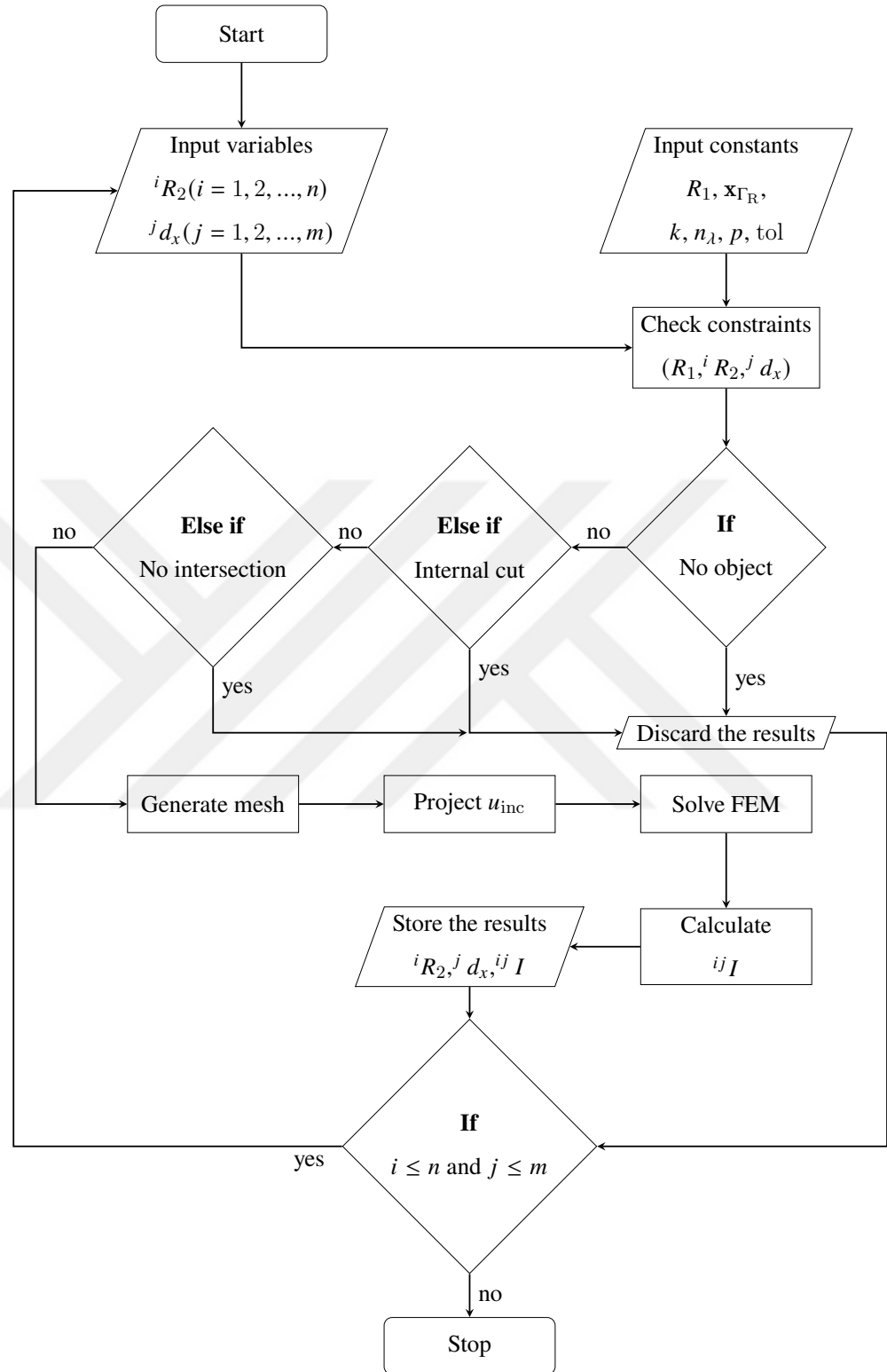


Figure 4.3. Parameter-sweep optimization algorithm flowchart.

4.1.2. Optimization with Nelder-Mead Method

The scipy optimization library contains a number of algorithms for finding the minimum of an objective function under its minimize function [78]. The NM algorithm is one of them that can be used and adapted to the problem in hand.

The NM optimization method is a mathematical optimization technique that uses the simplex algorithm to find the minimum or maximum of a function. It is an iterative method that starts with a set of points in n -dimensional space and uses a series of simple mathematical operations to move the points towards the minimum or maximum of the function [94,93]. The method is named after the two researchers who developed it, John Nelder and Roger Mead. It is commonly used in a wide range of fields, including engineering, finance, and science, to solve complex optimization problems.

NM algorithm has been used in several studies in the literature on acoustic scattering problem and shape optimization. Zhou and Fang used the NM algorithm to optimize coating structures and material parameters to the sound absorption coefficient [95]. Baulac *et al.* [96] optimized the acoustic efficiency of barriers using NM algorithm.

The NM method is a good choice for many optimization problems, especially when the objective function is complex and the initial guess is uncertain. However, it is important to note that the NM method is a local optimization method, which means it only guarantees convergence to a local minimum or maximum. As a result, it may not be able to find the global minimum or maximum of a constrained problem, especially if the objective function has multiple local minima or maxima [97]. For this reason, the optimization algorithm was enriched with the flowchart given in Figure 4.4. It was ensured that it performs repetitive analyzes through the feasible region to find a solution with the step sizes given by the user until a satisfactory result has been found comparatively. The NM optimization script can be found in Algorithm C.28.

The variables and constants used in parameter sweep optimization were also used in this analysis. With this algorithm, the optimization was accelerated by about three times for the same problem.

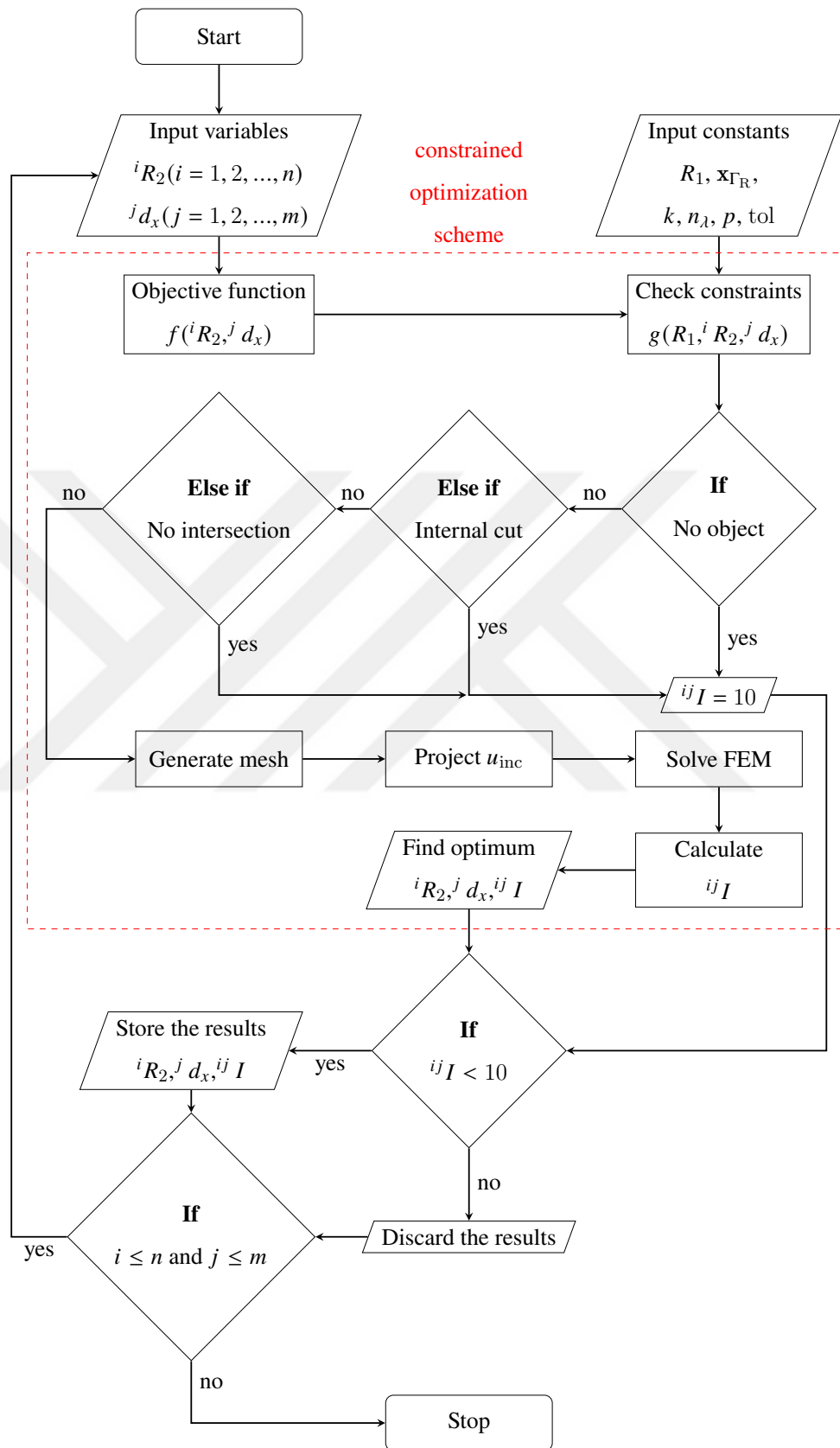


Figure 4.4. Nelder-Mead optimization algorithm flowchart.

The cluster of optimum values in an area within the feasible region that can be seen in Figure 4.5 means that the analysis has identified multiple values of the optimization variable that yield similar, good results. These values represent multiple local optimal solutions to the optimization problem. The optimum values found match with the parameter-sweep optimization results in Figure 4.2.

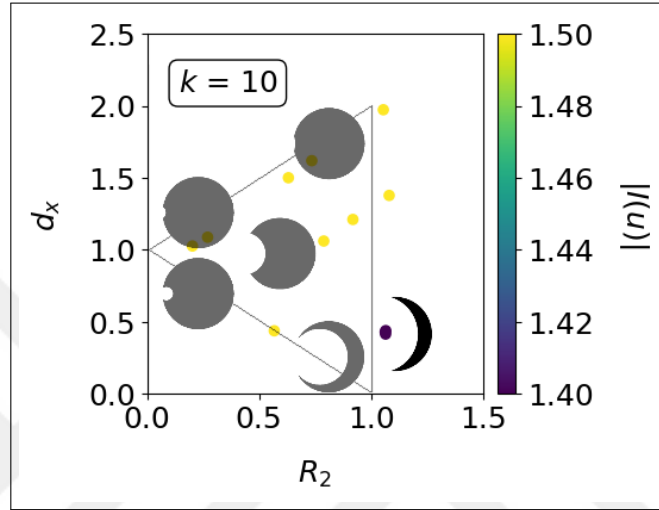


Figure 4.5. Nelder-Mead optimization result for a crescent-shaped single sound-hard body in a cylindrical solution domain.

Sound-hard scattering body analyzes repeated for different wavenumbers again. The results for different wavenumbers are also close to each other within the feasible region. It is almost possible to pin-point the optimum geometry in plots given for sound-hard cases, in Appendix D.2.1, and in Appendix D.2.3.

For sound-soft analyses, however, the results are a bit more spread out relative to each other as can be seen in Appendix D.2.2 and in Appendix D.2.4. On closer inspection, it can be seen that the results for both the cylindrical and the rectangular domains match well with each other. This gives confidence on making further analysis using rectangular domains.

4.2. OPTIMIZATION FOR MULTIPLE OBSTACLES

Multiple scattering problems are studied in different fields of science such as acoustics, optics, electromagnetism and particle physics. The specific meaning and application of the term "multiple scattering" can vary depending on the field in which it is used. In the context of waves, multiple scattering refers to the scattering of a wave by multiple objects or obstacles, resulting in a complex pattern of reflections and transmissions. In the context of particles, multiple scattering refers to the scattering of a particle by multiple atoms or molecules, resulting in a random change in the particle's direction of motion.

Multiple scattering problem was first studied by Lord Rayleigh [98] in 1892. It can be classified as deterministic or non-deterministic based on the context of the problem. For particle physics, it is usually a non-deterministic problem since the behavior of each particle cannot be predicted precisely, given the initial conditions of the system. It is often described using statistical methods, such as multiple scattering theory (MST), which can provide an average description of the behavior of the particles [99]. MST starts with a wave equation, obtains solutions for a single particle, introduces the effects of many particles, and then considers statistical averages [100]. Applying MST in computational methods is very complicated and challenging.

In deterministic scattering, on the other hand, the trajectory of each particle is determined by the initial conditions and the interactions with the scattering centers, and the same initial conditions will always produce the same outcome. Deterministic multiple scattering formulations have become an intensively studied subject in the literature [101,102], especially with the development of metamaterials.

These artificial materials are engineered and designed to exhibit non-natural properties such as enhancing sound attenuation, focusing waves, cloaking objects [103]. They can be formed from spatial lattice of rods, beams, plates or other slender units as shown in Figure 4.6, or from optimized geometries and arrangements. It was realized that by shape-optimization scattered waves could be manipulated. Thus, it inspired many studies such as noise control, vibration isolation, sensor designing, creating art works to get extraordinary outputs.

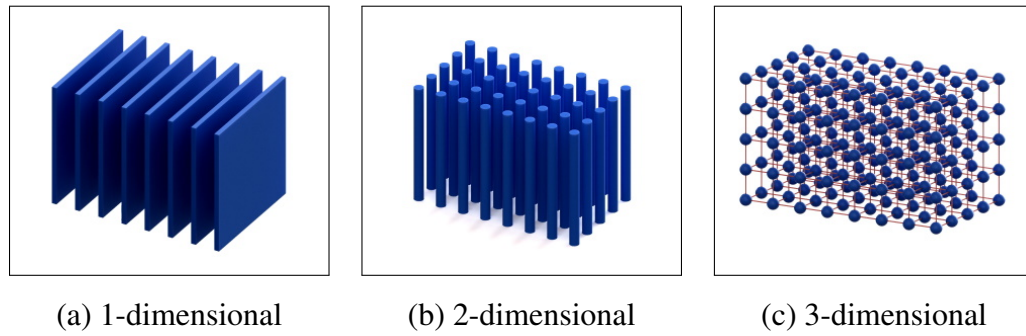


Figure 4.6. Examples of spatial lattices for various geometrical units.

In acoustics, the superposition of elementary scattered fields, which are resulting from the interaction between the scattering bodies on one hand, and the incident wave and scattering bodies on the other hand needs to be considered. Several different solution methods are suitable for modeling multiple scattering in exterior domains such as perfectly matched layer [41] and infinite elements [104]. However, the most studied approach in this regard is to adapt absorbing boundary conditions to multiple scattering problems, simply by enclosing all scattering obstacles in a single solution domain.

Grote and Kirsch derived a multiple-Dirichlet-to-Neumann (mDtN) map technique which is based on a decomposition of the scattered field into several purely outgoing wave fields [105,106], greatly reducing the computational cost of the problem. Acosta *et al.* [107] coupled the mDtN technique with a finite difference method. Geuzaine *et al.* [108] presented a formulation that replaces the original multiple scattering problem by an iterative series of single scattering problems. Villamizar *et al.* [109] worked on expanding high order local absorbing boundary conditions to far-fields, which was coupled with isogeometric analysis (IGA) by Khajah and Villamizar [47] which is eventually applied to acoustic multiple scattering [110].

There are several benefits to using the finite element method for solving multiple scattering problems in acoustics. The finite element method allows for the efficient and accurate solution of complex scattering problems, even when the scattering objects have arbitrary shapes and are located in complex geometries. The finite element method can be easily adapted to incorporate various types of boundary conditions, including periodic and homogeneous Neumann boundary conditions.

4.2.1. Homogeneous Neumann Boundary Condition

Application of homogeneous Neumann boundary conditions (HNBCs) can be used to validate the use of periodic boundary conditions (PBCs) for a perfectly symmetric problem. In general, a HNBC requires that the normal derivative of the solution to the problem be zero at the boundary. This means that the solution must be constant along the boundary, and it cannot change in a normal direction [61]. If the problem is perfectly symmetric and the other boundary conditions are also symmetric, then the solution to the problem will also be symmetrical relative to the boundary where the HNBC is used.

If we consider two opposing boundaries for a rectangular domain, FEA should yield the same result if these boundaries are defined as either PBCs or HNBCs. Such a symmetry can occur if the scattering obstacle is perfectly symmetric as well, such as a cylinder. For this reason, first of all, the symmetrical behavior of the scattered and total wave using the HNBC was examined, taking into account the results found in the literature.

4.2.1.1. Half-Cylinder in a Rectangular Domain

In order to analyze symmetrical or periodic structures, it is necessary to study rectangular domains because a rectangle can be treated as a unit cell which may reproduce itself along one or both of its primitive axes [111]. Therefore, the single body scattering problem solved in Chapter 3 is repeated for a half-cylinder geometry.

A rectangular solution domain with a unit cylinder body in its centroid is generated using the script given in Algorithm C.6. The same problem for a shifted cylinder is also solved to validate the finite element model. To do so, a half-cylinder on top is generated within the rectangular domain using Algorithm C.7.

FEA for the scattered wave solution was performed using Algorithm C.26. HNBC was defined at the upper boundary and Sommerfeld BC was defined at the remaining boundaries. Wavenumber $k = 2\pi$, polynomial degree $p = 3$, and discretization density $n_\lambda = 15$ are used.

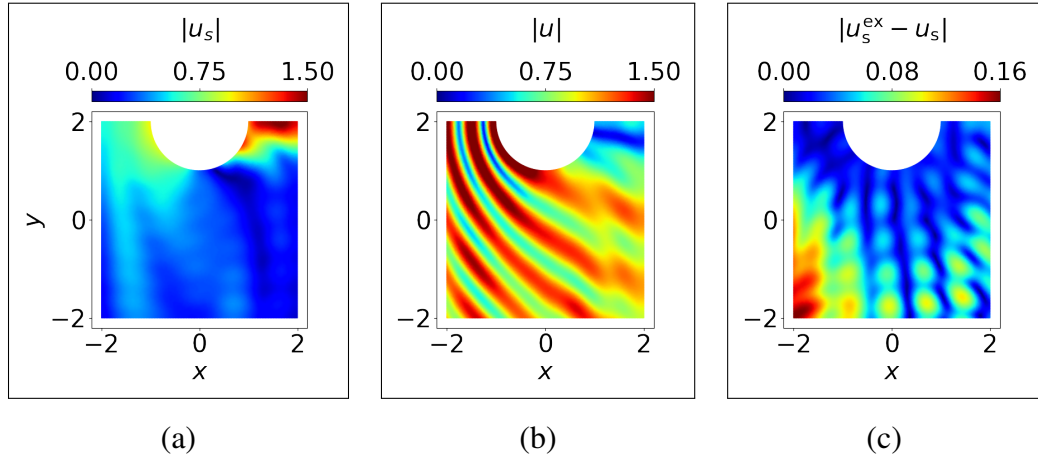


Figure 4.7. Cylinder solution obtained by utilizing the symmetry of the half-cylinder problem using a HNBC on the top boundary.

The numerical solution for the half-circle problem with a symmetrical upper boundary is obtained as shown in Figure 4.7 where (a) scattered wave, $|u_s|$, (b) total wave, $|u|$, and (c) absolute error, $|u_s^{\text{ex}} - u_s|$ results are given.

The contours of the scattered wave and total wave solutions given in Figure 4.7 match the solution in which the cylinder is located at the center of a rectangular domain where the scattering cylinder is completely surrounded by ABCs given in Figure 3.15.

As has been done before, the exact solution, u_s^{ex} , is calculated using Algorithm C.1 by simply projecting it onto the mesh. Absolute error is found from the difference of the magnitudes of the exact solution, $|u_s^{\text{ex}}|$ and the numerical solution, $|u_s|$. The absolute error for this analysis was higher at the far corners from the scattering cylinder. This can be explained by the fact that the incident and therefore the scattered wave are first calculated on the cylinder boundary, Γ_S .

With the changes to be made in the geometry of this problem, it is possible to make analyzes for multiple cylinders as well. Solving the same problem without any change in the boundary conditions will give valuable information about the scattered wave and total wave behavior for multiple scattering cylinders.

4.2.1.2. Vertically Aligned Finite Number of Cylinders

A single cylinder placed in the middle for a rectangular domain would actually be equivalent to the incident plane wave problem striking two cylinders, provided that the upper boundary satisfies the symmetry of the problem by having HNBC. It is possible to find the conceptual twin of the first cylinder and the waves surrounding them by simply mirroring the contour result around the symmetry axis.

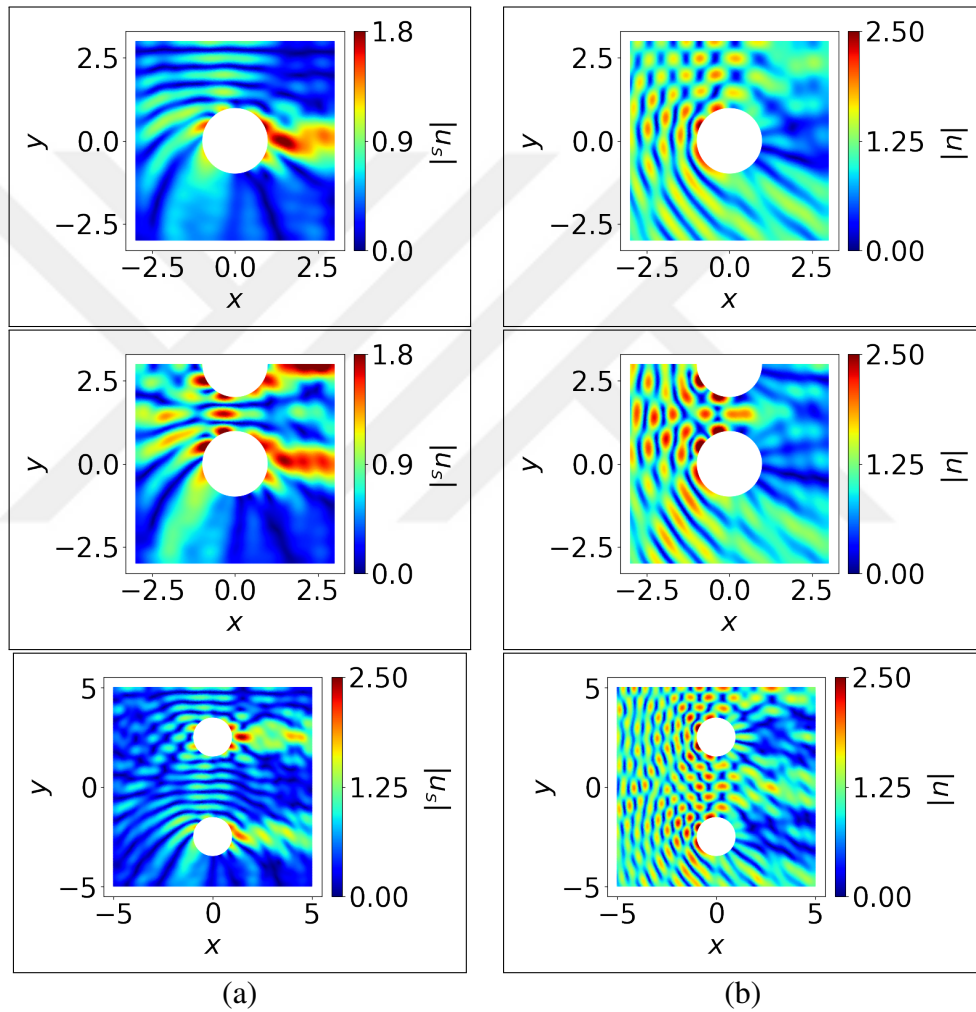


Figure 4.8. Analysis of finite number of cylinders using combinations of homogeneous Neumann and Robin BCs on outer boundaries.

Using correct geometrical layouts, it is possible to obtain solutions for three cylinders and four cylinders with the same method, similar to the conceptual example of obtaining two cylinders using only one cylinder.

Geometries are obtained using Algorithm C.7 for two cylinders, Algorithm C.11 for three

cylinders, and C.12 for four cylinders. FEA for the scattered wave solution was performed using Algorithm C.26 with wavenumber $k = 2\pi$, polynomial degree $p = 3$, and discretization density $n_\lambda = 15$, again.

The results shown in Figure 4.8 have been compared with [112], and it was seen that the total wave contour is compatible with the solution obtained by the highly accurate and high order numerical method given there.

If the results are briefly interpreted: sound-hard cylinders act as impenetrable barriers to the incident wave, and the wave completely reflects off the cylinders. This results in a sharp change in the direction of the wave, and the interference between the reflected and the incident waves produce standing wave patterns as can be seen in Figure 4.8 for both (a) scattered wave and (b) total wave solutions. As the number of cylinders increases, the complexity of the wave behavior produced by the scattering increase.

4.2.1.3. Vertically Aligned Infinite Number of Cylinders

Performing FEA for a 1-D periodic structure using HNBCs alone can also provide an important reference result for testing PBC application with the same analysis. This can be achieved by either using two half-cylinders on the bottom and top boundaries of the rectangular domain or placing a single cylinder at the center of the domain. Both simulations represents an array of infinitely many vertically arranged cylinders, and the results of the simulations can be compared with each other for validation.

However, the result to be obtained using only HNBC needs to be verified beforehand. Since vertically repeating cylinders are considered, comparing the same half cylinder and full cylinder results will confirm this application in itself.

For single cylinder, Algorithm C.6, and for half-cylinders, Algorithm C.10 used to generate the necessary meshes. Algorithm C.25 used as the FEM for both problems, in which HNBCs are defined for top and bottom boundaries. Parameters are kept the same for both analysis with $k = 2\pi$, $R = 1$, $n_\lambda = 25$, $p = 2$.

Contour plots for (a) scattering wave and (b) total wave solutions are given in Figure 4.9. Clearly, the solutions obtained for infinite cylinders with horizontal symmetry match well for

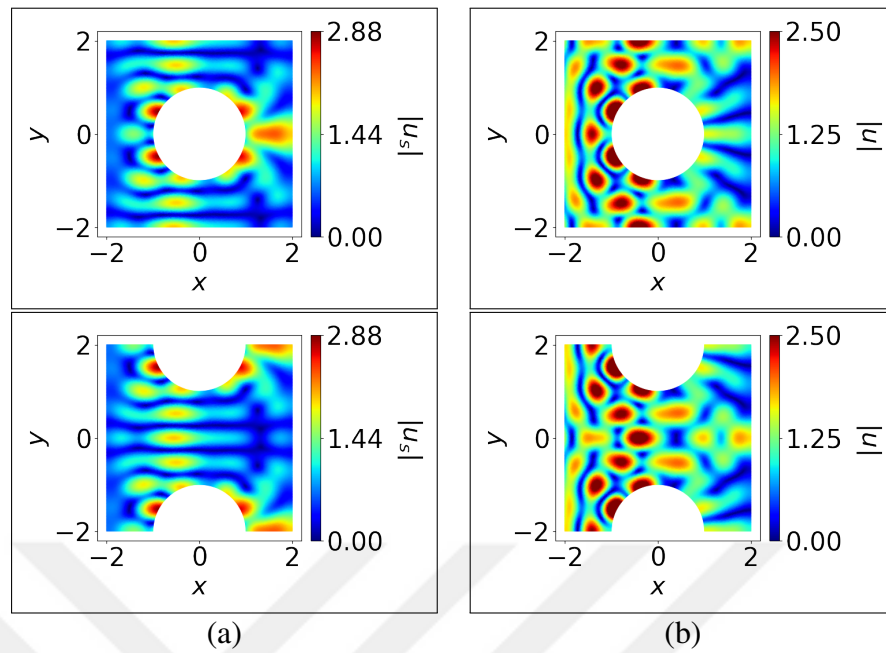


Figure 4.9. Results for a rectangular solution domain.

both the scattered wave and the total wave. This result shows that the same problem can also be solved using PBCs.

4.2.2. Periodic Boundary Condition

Periodic boundary conditions specify that the values of the variables being modeled must remain the same at the beginning and end of each period, while symmetric boundary conditions require that the solution to the problem be symmetrical with respect to a given axis or plane. In other words, periodic boundary conditions are used to model systems that exhibit periodic behavior, while symmetric boundary conditions are used to model systems that exhibit symmetry.

The fundamental ability that PBC provides is that it becomes possible to perform FEA for objects on an array with different orientations that can break the symmetry of the problem. That is, the angle of incident wave can also be changed if desired.

4.2.2.1. Vertically Aligned Infinite Cylinders

It was primarily aimed to obtain a working FEM using PBC. Using the same infinite cylinders problem, which is horizontally symmetrical and periodic in nature, it is expected to obtain matching results with both HNBC and PBC applications. Having the same results provides the evidence that both models are valid and the PBC is working as it is intended to.

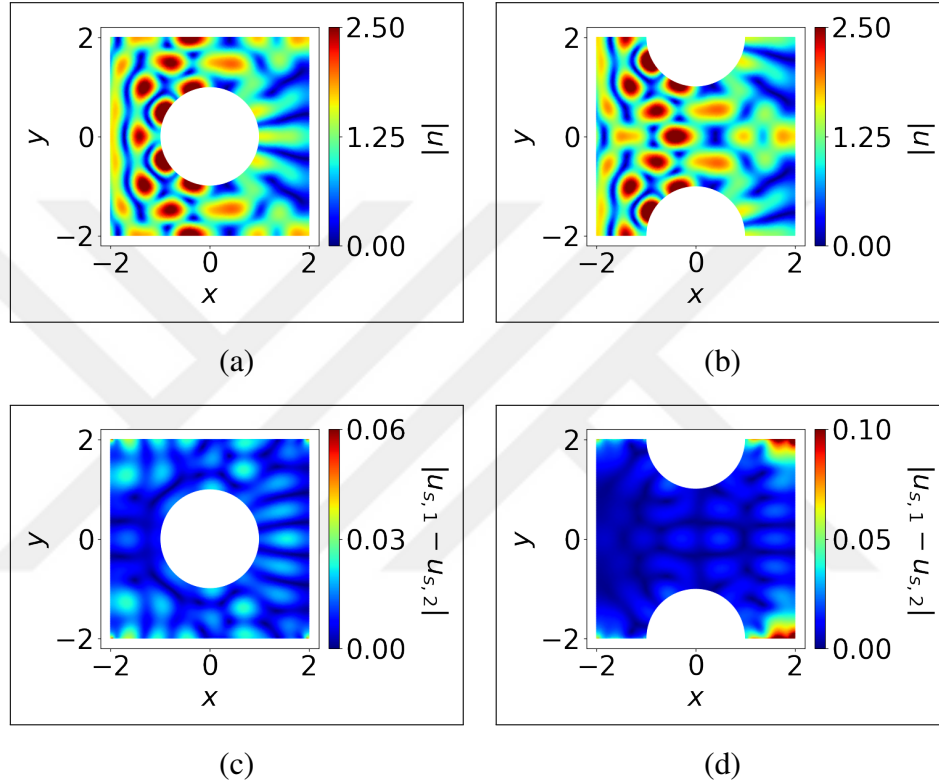


Figure 4.10. PBC application on horizontally symmetrical and periodic infinite cylinders problem.

This problem was solved using Algorithm C.27. Thanks to the modular structure provided by Jupyter Notebook and the ease of defining functions in FEniCS, the mesh was created using Algorithm C.6 for single cylinder and Algorithm C.10 for half cylinders, which were also used in previous problems.

Results obtained with PBCs application for this problem are given in Figure 4.10 for (a) single cylinder and (b) half-cylinders. Since both are actually modeling the same problem of infinite number of cylinders with horizontal symmetry, contour plots of the total waves match as well.

In addition, a relative absolute error comparison was made between PBC and HNBC applications by substituting the magnitudes of the scattered waves in-between HNBC and PBC solutions, as shown in Figure 4.10 (c) and (d). The small absolute error indicates that the solutions produced by the two models are similar, and that the models are likely to be valid.

4.2.3. Optimization for Vertically Aligned Crescents

The acoustic scattering problem for geometries with horizontal alignment symmetry and arbitrary shape can also be solved with the FEM given in Algorithm C.27. As shown in Figure 4.11, PBC, Γ_P , was defined at the lower and upper boundaries, and Sommerfeld ABC, Γ_R was defined at the right and left boundaries. The scattering body is assumed sound-hard on Γ_S . Sum of the total wave within the colored area, I , is minimized using PS and NM optimization with design variables being the cropping cylinder, R_2 , and the cropping offset, d_x , as defined previously. Mesh for the problem is generated using Algorithm C.9.

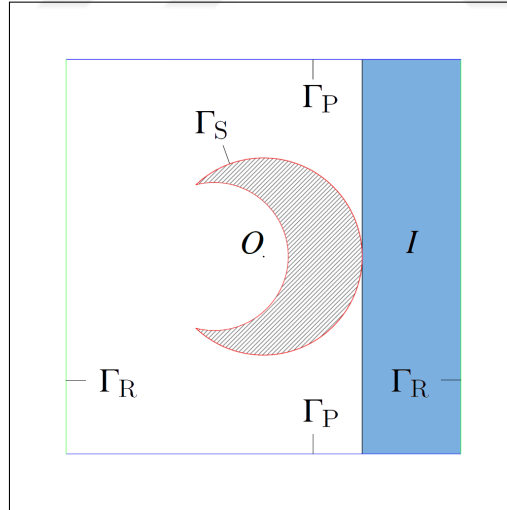


Figure 4.11. Vertically aligned infinite crescents modelled with PBCs.

Optimization analyzes were repeated for $k = 5$, $k = 10$ and $k = 15$ wave numbers with both PS and NM optimization algorithms introduced in Chapter 4. The flowcharts in Figure 4.3 and Figure 4.4 are valid for PS and NM optimizations, respectively.

The sound attenuation performance of a vertically aligned system of crescent-shaped

resonators depends on the the size and shape of the resonators, the frequency of the incident wave, and the spacing between the resonators. In general, however, from such a system of tuned resonators is expected to produce significant sound attenuation over a narrow range of frequencies, based on the operating principles of Helmholtz resonators. However, since the distance between scatterers is kept constant, it will not be possible to draw a general conclusion that will be valid under all circumstances. The primary goal of these analyzes was to draw conclusions from the difference of using circular and rectangular solution domains, the effect of selected ABC on the results, the similarities of single scatter and multi scatter analyzes, the comparison of the selected areas for total scattered wave calculation.

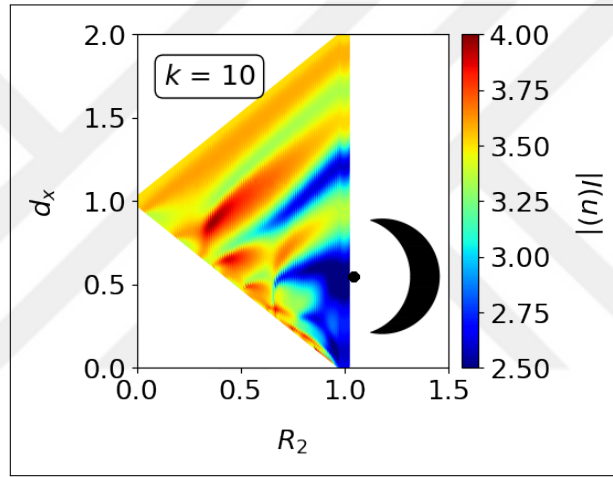


Figure 4.12. PS result for 1-D periodic crescent.

In Figure 4.12, the result of the PS analysis for $k = 10$ is shared. A feasible-region contour view similar to the analyzes for single crescent in a circular domain that is given in 4.2 can be observed. In multiple scattering, however, the sensitivity of the total wave reduction to the design variables decreased. The optimum result obtained using the NM algorithm was slightly outside the region used for PS analyses. As expected, geometry with a concave structure was obtained. The location of this optimum geometry is marked with a black dot on both Figure 4.12 and Figure 4.13.

NM optimization also yielded results similar to the single scatter analysis from Chapter 4. Like the PS analysis, its search was limited by the constraints of the problem and the available computational capacity. Since the NM algorithm can scan in a wider area, it has found an optimum value a little outside the feasible-region, but within an expected area according to

the contour plots of the PS.

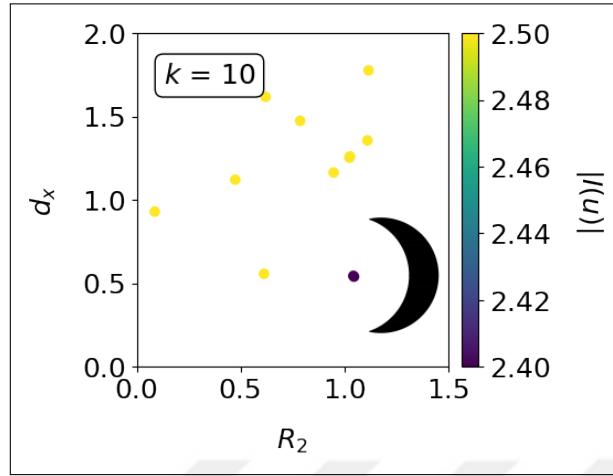


Figure 4.13. NM optimization result for 1-D periodic crescent.

It is understood that performing a single scattering analysis for a simple geometry provides a general prediction. Having such information at the beginning of a study will be valuable in terms of time savings. On the other hand, the benefits of using PBC and advanced optimization techniques for designing more complex systems are obvious. The change in wave behavior is highly related to the geometry and boundary conditions defined on it. It has been found that the sound-soft scatterer will perform optimally with a similar geometry but better at attenuating the total wave.

Complementary results are given in Appendix D.1.5 and Appendix D.1.7 for PS, and Appendix D.2.5 and Appendix D.2.6 for NM optimisation, for both sound-hard and sound-soft analysis.

It has been shown that the algorithms developed can be used as a tool under a systematic scheme to solve the acoustic scattering problem for a periodic geometry and optimize such a system at a certain pure tone frequency. This way, a group of tools that can be used to design noise control systems has been developed.

5. ANALYSIS OF CONVENTIONAL ACOUSTIC LOUVERS

By using the FEM introduced so far, it is possible to accurately model the geometry of an acoustic louver, as well as the behavior of acoustic waves propagating through it. Before performing a FEA, some of the most important considerations are: accurately representing the louver's geometry, including the size, shape, orientation and spacing of the blades and selection of appropriate boundary conditions. In order to do these, first of all, it is necessary to examine conventional acoustic louvers on the market.

5.1. BENCHMARK STUDY ON CONVENTIONAL ACOUSTIC LOUVERS

Conventional acoustic louvers on the market consist of blades that are placed in 45° angle. The in-fill material used within the blades is mineral wool. The bottom surface of each blade is perforated to benefit from the in-fill material's sound absorption. Typically a free area ratio of 50% is commonly adopted as the standard to follow. The free area ratio is the percentage of the free area to the core area of a louver. Front opening of a louver assembly without the louver blades is the core area, the minimum available opening area between the blades which the air can pass through is the free area [113].

There is no consensus on quantifying the sound attenuation performance of acoustic louvers yet. Manufacturers often share insertion loss, IL, or transmission loss, TL, performance data in their technical brochures. In some cases, this causes confusion and results in manufacturers pointing to incorrect or incompatible standards.

IL and TL are often used in specifying the sound attenuation required from a silencer. IL is the reduction in sound power level transmitted through a duct compared to that transmitted with no test subject in place. The insertion loss is equal to the noise reduction that can be expected at a reference point outside the duct outlet as a result of inserting the test subject, provided the duct outlet remains at a fixed point in space. This is why in some technical brochures it is called *noise reduction*, which causes further confusion. The transmission loss, on the other hand, is defined as the level difference between the sound power incident at the entry to the test subject to that transmitted by it [114].

Manufacturers refer to ISO 7235, ISO 10140-2 and ASTM E90 standards for their published data, as stated in their catalogues. The international standard ISO 7235 specifies laboratory measurement procedures for determining the IL and the TL of air-terminal units. Although there is no dedicated section for acoustic louvers, detailed measurement instructions are given for parallel-baffle silencers. Manufacturers rely on the methods described there. The sound pressure level of the transmitted wave can be measured in a reverberation room, in a test duct or in an essentially free field [115].

Although not included in any technical brochures reviewed within this study, it is worth mentioning ISO 11820, which specifies measurements on silencers in situ. It is again possible to measure insertion loss or transmission loss of an installed test subject on site following the procedures described within this standard [116].

The other two standards referenced by the manufacturers, ISO 10140-2 and ASTM E90, concern with the laboratory measurement of the sound insulation of building elements. The same principles are followed in both families of standards. The test specimen is built into an opening between two adjacent reverberation chambers, one source and one receiver. Assuming that the sound field in both rooms is sufficiently diffuse, the average level difference between source and receiving rooms is determined by sampling the sound field in both rooms. The transmission loss, TL , or the sound reduction index, R , in ASTM and ISO terminologies respectively is calculated from the difference between these sound pressure level measurements [117,118,119].

From a designer's point of view, the IL data is more valuable compared to the TL data, as it gives a better idea of how a louver performs in reducing the overall sound pressure level. In fact, there has been a consensus in the academy on using IL data since a long time [120,24], in contrast to the resistance that manufacturers show for years due to the ease of obtaining TL data.

Determining the IL performance of various noise control elements using FEM has been extensively studied in the literature. The devices studied include automobile mufflers [121], diesel engine exhaust mufflers [122], reactive silencers for power station exhaust stacks [123], railway noise barriers [124], louvered terminations positioned at the end of a rectangular duct [125], and sound barriers [126,127].

The terminology on louvers is well established in EN 12792 [113] and ANSI/AMCA Standard 500-L [28]. The area of the opening where the louver is mounted is called the face area. A typical geometrical feature of an acoustic louver is the series of horizontal blades arranged within this frame. Free area can be calculated by multiplying the sum of the minimum distances between intermediate blades, by the minimum distance between jambs. The free area of an acoustic louver is an important factor to consider when designing a noise control system, as it can significantly impact the effectiveness of the louver in reducing the transmission of sound.

In Figure 5.1, parameters for a conventional louver can be defined as the angle of the blades, α , the width of louver section, w , the spacing between successive blades, a , and the thickness of the blades, b . For an idealized geometry that is considered periodic, the free area ratio can be calculated using only these parameters.

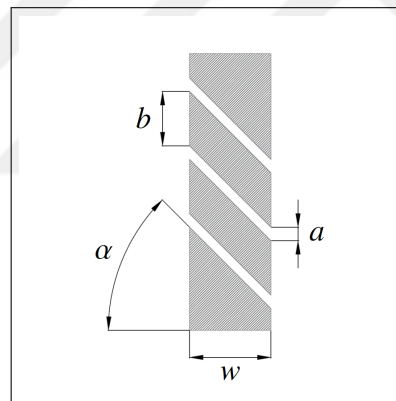


Figure 5.1. Geometrical features of a conventional acoustic louver.

The geometry of an acoustic louver can be optimized by varying the size and shape of the frame and blades, adjusting the spacing between the blades, and angle of the blades, to further improve the performance of the louver. It must be reminded that acoustic louvers are generally produced at a few fixed section widths, taking into account the standard facade wall thicknesses generally encountered.

Although the dimensions of an acoustic louver may vary according to the technical specifications it must provide, manufacturers produce them in a few fixed sizes. Consequently, louver continuity is usually ensured by using more than one louver within a face area it is installed. For this reason, a periodic geometry is assumed for the sections analyzed.

5.2. FINITE ELEMENT ANALYSIS

To be able to create a complex geometry in GMSH, if boolean operations cannot be used, it is necessary to know the coordinates of each point. This requires defining the shape and dimensions of the section to be examined. Therefore, as a first step, a detailed drawing of the section is created, including all relevant dimensions and angles.

In a rectangular domain with PBCs providing horizontal periodicity, there will be a blade in the middle and two half blades that complete each other at the upper and lower boundaries of an idealised acoustic louver section. The centers of the upper and lower blades must be on the PBC to ensure periodicity. Depending on the blade angles and their distance from each other, the shape of the blades on the PBCs change.

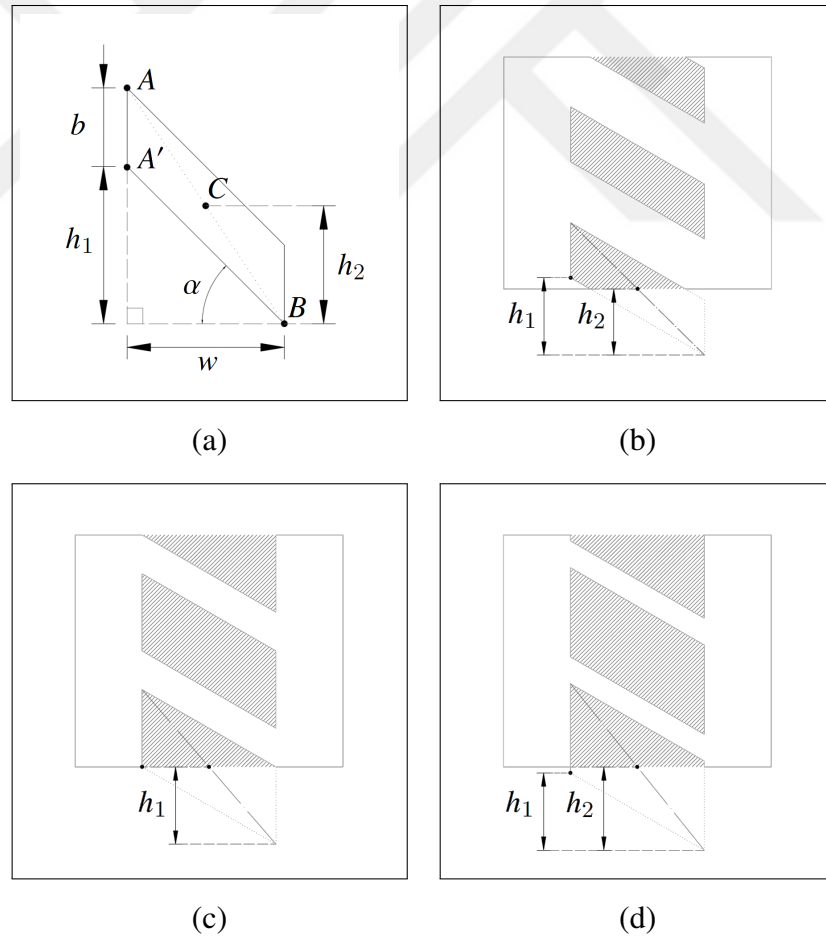


Figure 5.2. Solution domain depending on the blade center point.

In order to make comparisons between acoustic louvers with different sections, the domain height is fixed. Keeping the height of the rectangular domain constant, the overall solution of this periodic problem requires modelling three different cases of geometries. Detailed schematic illustrations are provided in Figure 5.2 to explain these cases.

Point A' , which plays an important role in determining these constraints, is shown in Figure 5.2 (a). The case where this point is above the lower PBC, $h_1 > h_2$, is shown in Figure 5.2 (b), when it is on the PBC, $h_1 = h_2$, is shown in Figure 5.2 (c), and when it is below the PBC, $h_1 < h_2$, is shown in Figure 5.2 (d). These three constraints are taken into account and the geometry for each of them is created separately. So that, it has been possible to make analyzes for all the different geometries using the chosen design variables.

The relationship between h_1 and h_2 can be easily deduced from the geometry given in Figure 5.2 (a). Since $h_1 = \tan(\alpha)w$, height of point A relative to B must be $h_1 + b = \tan(\alpha)w + b$. Considering that the shape of the blades is simply a parallelogram, the centroid, C , must coincide with the midpoint of the line segment that connects the two opposing corners of it. Hence, it also means $h_2 = (h_1 + b)/2$. Note that this point should also fall on the PBC for the lower and upper blades.

The coordinates of all the points required for generating the solution domains are found using the relations given. Meshes for these acoustic louver sections are created in GMSH using Algorithm C.13, Algorithm C.14 and Algorithm C.15.

In Figure 5.3, a schematic drawing showing the FEM is provided. To ensure horizontal symmetry, PBCs, marked with Γ_P , are defined on the upper and lower boundaries. Robin BC, marked with Γ_R , is applied on the upstream and downstream edges of the domain, located to the left and right ends, respectively. The blade sections of the acoustic louver marked with Γ_S are assumed sound-hard. FEM given in Algorithm C.27 was used to find the scattered waves.

In order to obtain a reliable and realistic response from the FEA results, an area is defined for which the mean acoustic pressure is calculated. This area is shown with a distinct color in Figure 5.3, marked with I . This approach is analogous to what is typically done in real-world measurements of sound pressure for insertion loss.

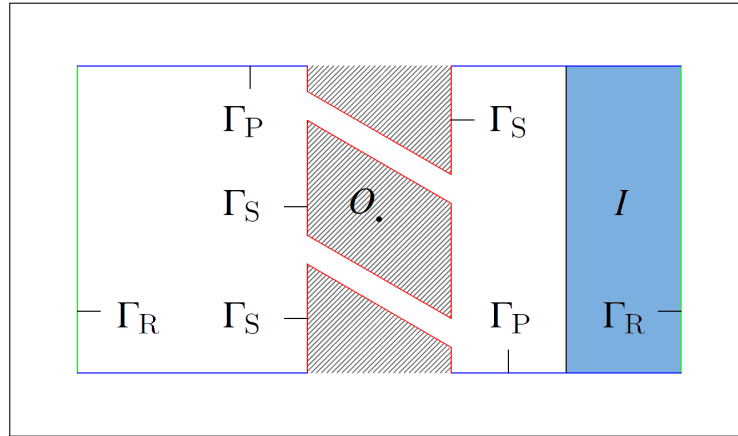


Figure 5.3. Finite element model designed for acoustic louver analysis.

The positioning of this area for which the average pressure is calculated is also important to achieve reliable results. Since the problem is of a periodic nature, it is plausible that this area covers the entire height of the solution domain from top to bottom.

On the other hand, when determining the distance of area *I* to the acoustic louver, the region where the acoustic waves show a predictable spatial distribution should be taken into account. Because of this, pure-tone analyzes have been performed to evaluate the total wave behavior at frequencies of interest. According to the results of these analyzes, it would be best to consider the region where the pressure fluctuation in the total wave is the least or enters a certain regime. In addition, it is necessary to decide for which frequencies these analyzes will be performed.

Insertion loss data for sound attenuation performance are often measured in 1/3 octave bands within the frequency range of 63 Hz to 8,000 Hz, as specified by the aforementioned standards. Measurement results are usually declared in octave band frequencies.

Octave bands and one-third octave bands are commonly used in noise control studies because they provide a convenient way to analyze the frequency content of a sound signal. The human ear has a nonlinear response to sound, which means that it does not perceive all frequencies with equal sensitivity. Octave bands and one-third octave bands are designed to approximate the response of the human ear and provide a way to analyze the frequency content of a sound signal in a way that is more relevant to human perception.

To determine the lower and upper frequencies of an octave band, the following formulas may be utilized:

$$f_{\min} = \frac{f_c}{\sqrt{2}} \quad \text{and} \quad f_{\max} = f_c 2^{1/2}, \quad (5.1)$$

where f_c is the octave band center frequency, $f_{\min}$ is the octave band lower frequency, and $f_{\max}$ is the octave band upper frequency [128]. It should be noted that octave bands are based on a logarithmic scale, meaning that the ratio of the upper frequency to the lower frequency is always 2:1. As such, if the center frequency is doubled, the upper and lower frequencies will also double, resulting in an octave band with the same width as the original band.

Standard octaves provide a useful framework for analyzing frequency-dependent data, particularly in noise control studies. They are straightforward to apply for evaluating data quickly and efficiently. Analyzing a wave data at high frequency resolution can be time-consuming and costly, especially if it involves conducting tests or simulations. Octave bands or one-third octave bands provide a good approximation without the need for such high resolution data [129].

Considering these, pure-tone analyzes showing the total wave contours at the octave band center frequencies have been performed. The results are given in Figure 5.4. It can be observed that the waves have a fairly uniform distribution up to a frequency of about 1 kHz. However, as the frequency of the incident wave increases, the scattered wave exhibits a more complex behavior. Which proves that it is beneficial to keep the area I as far as possible from the blades.

Visual inspection of the results at pure-tone frequency analyses helps to get a general idea. However, a FEA at an octave band center frequency performed as such gives the response of the system only at that specific frequency. It will not necessarily represent the response of the different frequencies within an entire octave band. To get a more complete picture of the response of the system over the entire octave band, it is necessary to perform FEA at multiple frequencies within the band. This can be done, by performing FEA at a series of frequencies spaced evenly within the band.

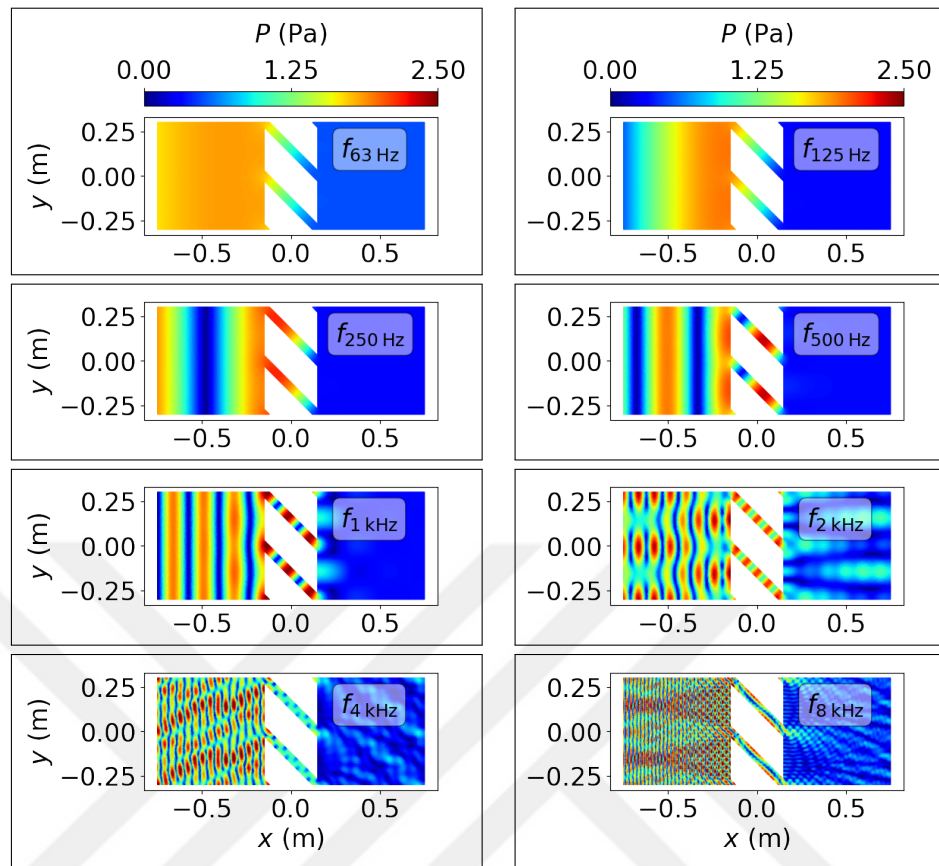


Figure 5.4. Sound pressures at octave-band center frequencies for an acoustic louver.

Performing finite element analysis (FEA) at a certain frequency resolution within an octave band can provide a more detailed understanding of the response of the system within that band. If the amplitude of the response at the additional frequencies is relatively uniform, the overall amplitude calculated for the band may not be significantly influenced by the additional data. However, if the amplitude of the response at the additional frequencies is significantly different from the amplitude at the original frequency, the overall amplitude calculated for the band may be influenced by the additional data.

For this reason, it is necessary to use a standardized measure when analyzing sound in different frequency bands. Sound pressure levels (SPLs) are commonly used in the octave band distribution of sound pressure data because they provide a standardized way to measure and compare the amplitude of sound. Because SPL is logarithmic, small changes in SPL correspond to large changes in the amplitude of sound, which makes it an ideal unit of measurement for detecting subtle changes in amplitude at different frequencies.

SPL, L_p , can be calculated from the measured or calculated pressure of a sound wave, p , relative to a reference pressure, p_{ref} , using:

$$L_p = 10 \log_{10} \left(\frac{p^2}{p_{\text{ref}}^2} \right) = 20 \log_{10} \left(\frac{p_{\text{rms}}}{p_{\text{ref}}} \right) \text{ dB}, \quad (5.2)$$

where the reference level is usually chosen as 20 micropascals (μPa) which is approximately the threshold of human hearing [130].

In general, the SPL of a signal will not be the same across all octave bands, even if the input signal has a constant SPL at all frequencies. This is because the power of a signal is distributed differently across different frequency ranges, and this can affect the SPL of the signal in each octave band. Therefore, sound pressure spectrum levels (SPSLs) are often used in the octave band distribution of a sound pressure data. The amplitude in frequency bands can be revealed by utilizing SPSLs.

If a constant sound pressure level signal is filtered into octave bands, the sound pressure level of the signal in each octave band may vary. This is because the signal may require more power at lower frequencies, which can affect the overall sound pressure level in each octave band. To ensure that the sound pressure level of the signal is consistent across all octave bands, a weighting filter can be used to adjust the power of the signal at different frequencies. Weighting filters are commonly used in acoustics and audio engineering to adjust the spectral density of a signal and ensure a consistent sound pressure level across different frequency ranges.

In the FEAs, an incident wave with a constant unit pressure is used. To accurately calculate and determine the contribution of each octave band up to 8000 Hz, solutions for frequencies between 1 Hz and 11360 Hz are obtained with a frequency resolution of 10 Hz. Standard octaves are preferred to obtain results comparable to the declared measurement data. A drop of 1dB of SPL in successive bands is applied using Algorithm C.30 according to the relation:

$$L_{b+1} - L_b = 10 \log_{10} \left(\frac{f_c(b+1)}{f_c(b)} \right) = \frac{1}{N} 10 \log_{10} 2 \approx \frac{3}{N}, \quad (5.3)$$

where $N = 3$ since 1/3 octave band is considered [131].

During post-processing of the results, the power of the signal at different frequency bands is adjusted by SPSLs to ensure consistency in the SPLs across all octave bands. The code that performs these functions is shared in Algorithm C.31.

First, the contribution of a band of frequencies between f_1 and f_2 has been calculated using the spectral density function, $(p_b^2)_{av}$, which is equal to:

$$(p_b^2)_{av} = \int_{f_1}^{f_2} p_f^2(f) df, \quad (5.4)$$

where p_f is the pressure for a pure-tone frequency, f . Then,

$$L_{ps} \approx 10 \log_{10} \left(\frac{(p_b^2)_{av} (\Delta f)_{ref} / (\Delta f)_b}{p_{ref}^2} \right), \quad (5.5)$$

where the reference bandwidth $(\Delta f)_{ref} = 1$ is used to calculate the total wave solution SPLs in 1/3 octave band frequencies [131].

To determine the insertion loss, IL, in 1/3 octave bands mean SPLs calculated on area I for both the incident wave, L_1 , and total wave, L_2 , are subtracted from each other:

$$IL = L_2 - L_1. \quad (5.6)$$

The SPSLs calculated for the incident wave idealizes the case without the acoustic louvers. The SPSLs calculated from the FEA results simulates the case with the acoustic louver installed.

Conversion of 1/3 octave band values to octave band values is carried out using the formula:

$$IL_{1/1} = -10 \log_{10} \left(\frac{1}{3} \sum_{k=1}^3 10^{-IL_{1/3,k}/10} \right). \quad (5.7)$$

Acoustic simulations were performed for 100 mm, 150 mm, 200 mm and 300 mm thicknesses, taking into account the frequently encountered acoustic louver dimensions. The open area ratio (OAR) is calculated by dividing the height of the domain, $h = 2(a + b)$, to the shortest distance between the blades, $a' = a \cos \alpha$.

Among the available acoustic louvers on the market, OARs are also found in a few values. See Table D.1. Although this is a parameter for which the manufacturers make the most customization, it is still possible to set a range to it. Considering the most common ones of these values, FEAs were performed for 20%, 30%, and 40% OARs.

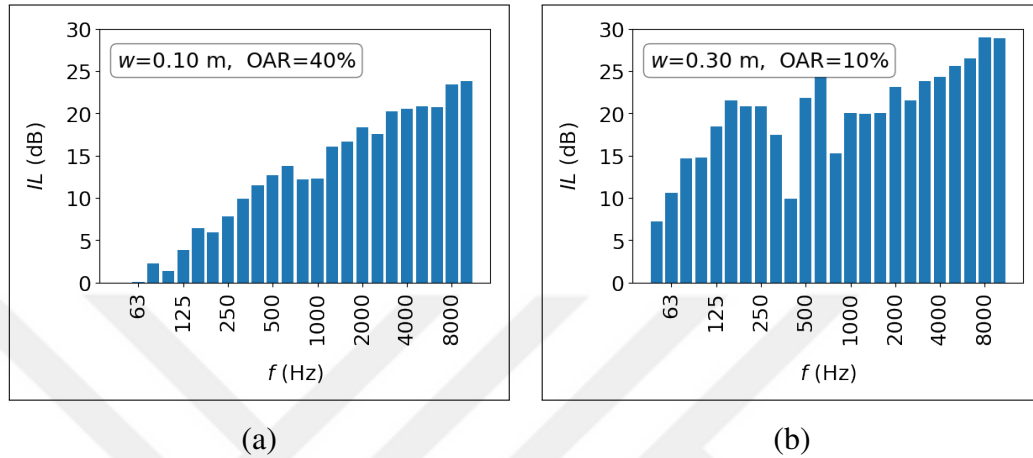


Figure 5.5. FEA results for the acoustic louver sections investigated, showing (a) the worst and (b) the best insertion loss performance.

Results for all models examined are presented in Appendix D.4. As anticipated, the thickest section with the smallest OAR performs the best, while the thinnest section with the largest OAR performs the worst. Bar plots showing the insertion loss in 1/3 octave bands for these acoustic louvers are shared in Figure 5.5.

Although the overall performance of the acoustic louver in Figure 5.5 (b) is the highest among all the analyzes made, it is observed that it has a critical frequency in the 400 Hz band. Moreover, it is observed that the critical frequency shifts towards lower frequencies as the acoustic louver thickness increases. This is a result that should be considered in relation to equipment noise, which is often more prominent at lower frequencies. This indicates that increasing the acoustic louver's cross-section may have negative impacts on the attenuation of certain frequencies.

Finally, despite the fact that it is common practice in the industry to report results in octave bands, it is important to note that noise control studies should ideally be conducted using 1/3 octave bands, if possible. This is because using 1/3 octave bands allows for a more detailed analysis and can provide a clearer understanding of the behavior of critical frequencies.

6. CONCLUSIONS

In this work, the acoustic scattering problem is investigated using an open-source finite element analysis platform, and new techniques are proposed for the related shape optimization. The developed solution scheme is adapted to the problem of calculating the insertion loss of acoustic louvers used in noise control studies.

Modular and readable scripts were created using Jupyter Notebook, which was integrated with the FEniCS image used in the virtual environment provided by Docker. The ability to utilize a variety of libraries and tools showed that non-commercial software has a unique adaptability advantage. Full control over the running algorithms enabled the use of all of the CPU power of the ordinary PC in use, maximizing the efficiency of the calculations. As a side output, it also shows that these platforms will play a crucial role in serving advanced engineering studies to wider audiences.

The development of a compatible computational platform and the verification of its accuracy through error analysis and comparison made between solutions throughout this work illustrated the potential applications of the software package presented, as well as its flexibility in solving a variety of problems in the future.

The Matplotlib library was utilized extensively throughout this study to create high-quality visualizations of the data and results, as shared throughout the study. Its integration with Jupyter Notebook allows for easy sharing of the algorithms and results online, making it an effective tool for communicating complex concepts and presenting findings in a clear and visually appealing way.

FEniCS's integration with GMSH has enabled the development of a system for creating two-dimensional geometries and meshes using written scripts. By having access to geometry and mesh element parameters, it is now possible to have full control over the design of shapes for wave manipulation. This enables the creation of "dynamic" meshes whose boundary conditions can be easily manipulated and communicate with FEM. The use of GMSH in conjunction with FEniCS allows for greater precision and control in the mesh generation process, which is crucial for accurate simulations and analyses.

In Chapter 3, ABC performances were compared using FEMs created for solution domains with circular or rectangular outer boundaries. Error analysis conducted on ABCs showed that simulations of simple shapes yield stable results that are suitable for industrial applications.

In Chapter 4, the development of optimization algorithms using the modular FEniCS platform over Jupyter Notebook and Python-based libraries such as Scipy and Numpy was presented. Research has demonstrated that incorporating Helmholtz resonator properties into a wavefront design can significantly improve sound attenuation at specific frequencies. Helmholtz resonators have a simple, straightforward design that is easy to manufacture and integrate into various applications, making them a particularly appealing choice for use in acoustic louvers, which could lead to cost savings and improved performance.

It has also been shown that horizontal symmetry in multiple acoustic scatterers can be achieved through the use of HNBCs (homogeneous Neumann boundary conditions) and PBCs (periodic boundary conditions). These methods can be used to design and optimize the properties of acoustic metamaterials by applying them to a mathematical model of a proposed metamaterial, which allows for the prediction of sound wave propagation.

Optimization studies have identified a crescent cross-section wave receiver that demonstrates high performance across a wide frequency range and is not sensitive to small geometric changes within the feasible region. This geometry has potential to be adapted and incorporated into various noise reduction systems.

In Chapter 5, a FEM was developed to calculate the insertion loss of acoustic louvers using the software scheme constructed in the previous sections. One of the key contributions of this work is the ability to separate incident and total wave results using appropriate weighting and spectrum density filters for octave band and 1/3 octave band frequencies. This allows for the analysis of acoustic louvers with variable blade angles, blade openings, and blade thicknesses. Results showed that the critical frequency is sensitive to the thickness of the acoustic louver.

As a future direction, the development of this model for use in air duct outlets is planned. In addition, the interaction between the presence of a flowing fluid and acoustic waves will be investigated.

REFERENCES

1. Alnæs M, Blechta J, Hake J, Johansson A, Kehlet B, Logg A, et al. The FEniCS project version 1.5. *Archive of Numerical Software*. 2015;3(100).
2. Butera F, Hewett K. Acoustic Performance of Louvred Facades for Brisbane Domestic Airport: An Integrated Approach. *Noise Control and Acoustics Division Conference*. vol. 45325. American Society of Mechanical Engineers. 2012:577-82.
3. Cui S, Perret-Gentil M, Gourdon E, Barthelmé AF, Mankibi ME, Wurtz E, et al. A global modelling approach of natural ventilation with acoustic and daylighting constraints. *International journal of ventilation*. 2016;15(3-4):233-52.
4. Koffi J, El Mankibi M, Gourdon E, Issoglio R. Assessment of single-sided ventilation with acoustic shutters on windows. *Building Simulation*. vol. 8. Springer. 2015:689-700.
5. Oldham D, De Salis M, Sharples S. Reducing the ingress of urban noise through natural ventilation openings. *Indoor Air*. 2004;14(8):118-26.
6. Watts GR, Hothersall D, Horoshenkov KV. Measured and predicted acoustic performance of vertically louvred noise barriers. *Applied Acoustics*. 2001;62(11):1287-311.
7. Yadav P, Bankar H, Karanth NV. Acoustic Enclosure Optimization for a Higher Capacity Diesel Generator Set Using Statistical Energy Analysis (SEA) based Approach. SAE Technical Paper; 2017.
8. Bibby C, Hodgson M. Field measurement of the acoustical and airflow performance of interior natural-ventilation openings and silencers. *Building and Environment*. 2013;67:265-73.
9. Bibby C, Hodgson M. Laboratory measurement of the acoustical and airflow performance of interior natural-ventilation openings and silencers. *Applied Acoustics*. 2014;82:15-22.
10. Kirby R, Amott K, Williams PT, Duan W. On the acoustic performance of rectangular

- splitter silencers in the presence of mean flow. *Journal of Sound and Vibration*. 2014;333(24):6295-311.
11. Shinde PV, Gavali P, Barawade R, Mohite Y, Shinde P. A review on muffler design for exhaust noise attenuation. *International Journal of Engineering and Technology*. 2017;9(3S):428-31.
 12. Borelli D, Schenone C. A finite element model to predict sound attenuation in lined and parallel-baffle rectangular ducts. *HVAC&R Research*. 2012;18(3):390-405.
 13. Antebas AG, Denia FD, Pedrosa AM, Fuenmayor FJ. A finite element approach for the acoustic modeling of perforated dissipative mufflers with non-homogeneous properties. *Mathematical and Computer Modelling*. 2013;57(7):1970-8.
 14. Givoli D. Non-reflecting boundary conditions. *Journal of computational physics*. 1991;94(1):1-29.
 15. Hagstrom T. Radiation boundary conditions for the numerical simulation of waves. *Acta numerica*. 1999;8:47-106.
 16. Chen L, Lian H, Liu Z, Chen H, Atroshchenko E, Bordas S. Structural shape optimization of three dimensional acoustic problems with isogeometric boundary element methods. *Computer Methods in Applied Mechanics and Engineering*. 2019;355:926-51.
 17. Barbieri R, Barbieri N. Finite element acoustic simulation based shape optimization of a muffler. *applied acoustics*. 2006;67(4):346-57.
 18. Khajah T, Liu L, Song C, Gravenkamp H. Shape optimization of acoustic devices using the scaled boundary finite element method. *Wave Motion*. 2021;104:102732.
 19. for Europe of the World Health Organization TRO. Environmental noise guidelines for the European Region. World Health Organization. 2018.
 20. Berglund B, Lindvall T, Schwela DH, Organization WH, et al. *Guidelines for community noise*. World Health Organization; 1999.
 21. Hurtley C, for Europe WHORO. *Night Noise Guidelines for Europe*. Euro Publication.

World Health Organization Europe; 2009.

22. Schaffer ME. *A practical guide to noise and vibration control for HVAC systems*. USA: American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Incorporated; 2019.
23. ASHRAE. *2019 ASHRAE Handbook: Heating, Ventilating and Air-Conditioning Applications - Chapter 48-Noise and Vibration Control*. USA: American Society of Heating, Refrigerating, and Air-Conditioning Engineers, Incorporated; 2019.
24. Viveiros EB, Gibbs BM, Gerges SNY. Measurement of sound insulation of acoustic louvres by an impulse method. *Applied Acoustics*. 2002;63:1301-13.
25. Salis MHFD, Oldham DJ, Sharples S. Noise control strategies for naturally ventilated buildings. *Building and Environment*. 2002;37:471-84.
26. American Society of Heating R, Engineers AC. *ASHRAE Terminology of Heating, Ventilation, Air Conditioning & Refrigeration*. American Society of Heating, Refrigerating, and Air-Conditioning Engineers; 1991.
27. BS EN 13030:2001. Ventilation for buildings – Terminals – Performance testing of louvres subjected to simulated rain. Geneva, CH: International Organization for Standardization; 2001.
28. ANSI/AMCA Standard 500-L-12. Laboratory Methods of Testing Louvers for Rating. Illinois, USA: Air Movement and Control Association International Inc.; 2015.
29. BS EN 14163:1998. Acoustics – Guidelines for noise control by silencers. Geneva, CH: International Organization for Standardization; 1998.
30. Berg MJ, Sorensen CM. A review and reassessment of diffraction, scattering, and shadows in electrodynamics. *Journal of Quantitative Spectroscopy and Radiative Transfer*. 2018;210:225-39.
31. Vorländer M. Recent progress in room acoustical computer simulations. *Building Acoustics*. 1997;4(4):229-46.
32. Siltanen S, Lokki T, Savioja L. Rays or waves? Understanding the strengths and

- weaknesses of computational room acoustics modeling techniques. *Proceedings of the International Symposium on Room Acoustics, ISRA*. 2010:29-31.
33. Vorländer M. Computer simulations in room acoustics: Concepts and uncertainties. *The Journal of the Acoustical Society of America*. 2013;133(3):1203-13.
 34. Simmons GF. *Differential Equations with Applications and Historical Notes, Third Edition*. 3rd ed. Textbooks in Mathematics. CRC Press;Chapman and Hall/CRC; 2017.
 35. Speiser D. *Discovering the principles of mechanics 1600-1800: essays by David Speiser*. Vol.1 ed. Springer Science and Bussiness Media; 2008.
 36. Andrews LC, Shivamoggi BS. *Integral Transforms for Engineers and Applied Mathematicians*. USA: MacMillan Publishing Company; 1988.
 37. Thompson LL. A review of finite-element methods for time-harmonic acoustics. *J Acoust Soc Am*. 2006;119(3):1315-30.
 38. Ciarlet PG. *The Finite Element Method for Elliptic Problems*. Carnegie-Rochester Conference Series on Public Policy. North-Holland Publishing Company; 1978.
 39. Langtangen HP, Mardal KA. *Introduction to Numerical Methods for Variational Problems*. Texts in Computational Science and Engineering. Springer International Publishing; 2019.
 40. Hale JS, Li L, Richardson CN, Wells GN. Containers for portable, productive, and performant scientific computing. *Computing in Science & Engineering*. 2017;19(6):40-50.
 41. Berenger JP. A perfectly matched layer for the absorption of electro-magnetic waves. *J Comput Phys*. 1994;114(1):185-200.
 42. Shirron JJ, Giddings TE. A finite element model for acoustic scattering from near a fluid-fluid interface. *Comput Methods Appl Mech Engrg*. 2006;196:279-88.
 43. Enquist B, Majda A. Absorbing boundary conditions for the numerical simulation of waves. *Math Comput*. 1977;31(139):629-51.

44. Kang F. Finite element method and natural boundary reduction. *Congress of Mathematicians, Warsaw, Poland*. 1983:1439-53.
45. A Bayliss ET M Gunzburger. Boundary Conditions for the Numerical Solution of Elliptic Equations in Exterior Regions. *SIAM J Appl Math*. 1982;42(2):430-51.
46. Khajah T, Antoine X, Bordas SPA. B-Spline for Time-Harmonic Acoustic Scattering and Propagation. *Journal of Theoretical and Computational Acoustics*. 2019;27(3):1850059.
47. Khajah T, Villamizar V. Highly accurate acoustic scattering: Isogeometric Analysis coupled with local high order Farfield Expansion ABC. *Comput Methods Appl Mech Engrg*. 2019;349:477-98.
48. Hunt JT, Knittel MR, Barach D. Finite element approach to acoustic radiation from elastic structures. *J Acoust Soc Am*. 1974;55(2):269-80.
49. Hunt JT, Knittel MR, Nichols CS, Barach D. Finite element approach to acoustic scattering from elastic structures. *J Acoust Soc Am*. 1975;57(2):287-99.
50. Shirron JJ, Babuska I. A comparison of approximate boundary conditions and infinite element methods for exterior Helmholtz problems. *Comput Methods Appl Mech Engrg*. 2006;164:121-39.
51. Braun S, Ewins D, Rao SS, Leissa A. Encyclopedia of vibration: Volumes 1, 2, and 3. *Appl Mech Rev*. 2002;55(3):B45-5.
52. Oberkampf WL, Roy CJ. *Verification and Validation in Scientific Computing*. Cambridge University Press; 2010.
53. Dsouza SM, Khajah T, Antoine X, Bordas S, Natarajan S. Non Uniform Rational B-Splines and Lagrange approximations for time-harmonic acoustic scattering: convergence, accuracy, absorbing boundary conditions. *Mathematical and Computer Modelling of Dynamical Systems*. 2021;27(1):290-321.
54. Faran JJ. Sound Scattering by Solid Cylinders and Spheres. *J Acoust Soc Am*. 1951;23:405-18.

55. Goodman RR, Stern R. Reflection and Transmission of Sound by Elastic Spherical Shells. *J Acoust Soc Am.* 1962;34:338-44.
56. Doolittle RD, Überall H. Sound Scattering by Elastic Cylindrical Shells. *J Acoust Soc Am.* 1965;39:272-5.
57. Spence RD, Granger S. The Scattering of Sound from a Prolate Spheroid. *J Acoust Soc Am.* 1951;23:701-6.
58. Silbriger A. Scattering of Sound by an Elastic Prolate Spheroid. *J Acoust Soc Am.* 1963;35:564-70.
59. Eringen AC. *Mechanics of Continua Second Edition*. USA: Robert E. Krieger Publishing Company, Inc.; 1980.
60. Martin PA. *Multiple scattering: interaction of time-harmonic waves with N-obstacles*. No. 107. Cambridge University Press; 2006.
61. Zienkiewicz OC. Achievements and some unsolved problems of the finite element method. *International Journal for Numerical Methods in Engineering*. 2000;47(1-3):9-28.
62. Laghrouche O, Bettess P. Short wave modelling using special finite elements. *Journal of Computational Acoustics*. 2000;8(01):189-210.
63. Laghrouche O, Bettess P, Astley R. Modelling of short wave diffraction problems using approximating systems of plane waves. *International Journal for Numerical Methods in Engineering*. 2002;54(10):1501-33.
64. Kechroud R, Soulaïmani A, Antoine X. A performance study of plane wave finite element methods with a Padé-type artificial boundary condition in acoustic scattering. *Advances in Engineering Software*. 2009;40(8):738-50.
65. Hughes TJ, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*. 2005;194(39-41):4135-95.
66. Cottrell JA, Hughes TJ, Bazilevs Y. *Isogeometric analysis: toward integration of CAD*

and FEA. John Wiley & Sons; 2009.

67. Venås JV, Kvamsdal T, Jensrud T. Isogeometric analysis of acoustic scattering using infinite elements. *Computer Methods in Applied Mechanics and Engineering*. 2018;335:152-93.
68. Dinachandra M, Raju S. Plane wave enriched partition of unity isogeometric analysis (PUIGA) for 2D-Helmholtz problems. *Computer Methods in Applied Mechanics and Engineering*. 2018;335:380-402.
69. Bruno OP, Kunyansky LA. A fast, high-order algorithm for the solution of surface scattering problems: basic implementation, tests, and applications. *Journal of Computational Physics*. 2001;169(1):80-110.
70. Khajah T, Liu L, Song C, Gravenkamp H. Shape optimization of acoustic devices using the scaled boundary finite element method. *Wave Motion*. 2021;104:102732.
71. Bruno OP, Kunyansky LA. Surface scattering in three dimensions: an accelerated high-order solver. *Proceedings of the Royal Society of London Series A: Mathematical, Physical and Engineering Sciences*. 2001;457(2016):2921-34.
72. Thompson LL, Pinsky PM. Complex wavenumber Fourier analysis of the p-version finite element method. *Computational Mechanics*. 1994;13(4):255-75.
73. Bouillard P, Ihlenburg F. Error estimation and adaptivity for the finite element method in acoustics: 2D and 3D applications. *Computer methods in applied mechanics and engineering*. 1999;176(1-4):147-63.
74. Steward JR, Hughes TJR. *h*-Adaptive finite element computation of time-harmonic exterior acoustics problems in two dimensions. *Comput Methods Appl Mech Engrg*. 1997;146:65-89.
75. Langdon S, Chandler-Wilde SN. Finite element methods for acoustic scattering. *Notes of Department of Mathematics, University of Reading, Whiteknights PO Box*. 2007;220.
76. Ern A, Guermond JL. *Finite Elements II*. Springer; 2021.
77. Mittal PK. *Oscillations, Waves and Acoustics*. I.K. International Publishing House Pvt.

Limited; 2010.

78. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*. 2020;17(3):261-72.
79. Harris CR, Millman KJ, Van Der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. *Nature*. 2020;585(7825):357-62.
80. Alnæs MS, Logg A, Ølgaard KB, Rognes ME, Wells GN. Unified form language: A domain-specific language for weak formulations of partial differential equations. *ACM Transactions on Mathematical Software (TOMS)*. 2014;40(2):1-37.
81. Logg A, Mardal KA, Wells G. *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg; 2012.
82. Nocedal J, Wright SJ. *Numerical optimization*. Springer; 1999.
83. Kinsler LE, Frey AR, Coppens AB, Sanders JV. *Fundamentals of Acoustics Fourth Edition*. USA: John Wiley & Sons, Inc.; 2000.
84. Vér IL, Beranek LL. *Noise and vibration control engineering: principles and applications*. John Wiley & Sons; 2005.
85. Munjal ML. *Acoustics of ducts and mufflers with application to exhaust and ventilation system design*. John Wiley & Sons; 1987.
86. Fuchs HV. *Applied acoustics: Concepts, absorbers, and silencers for acoustical comfort and noise control: Alternative solutions-Innovative tools-Practical examples*. Springer Science & Business Media; 2013.
87. Craster RV, Guenneau S. *Acoustic metamaterials: Negative refraction, imaging, lensing and cloaking*. vol. 166. Springer Science & Business Media; 2012.
88. Chen S, Fan Y, Fu Q, Wu H, Jin Y, Zheng J, et al. A review of tunable acoustic metamaterials. *Applied Sciences*. 2018;8(9):1480.

89. Mahesh K, Mini R. Helmholtz resonator based metamaterials for sound manipulation. *Journal of Physics: Conference Series*. vol. 1355-1. IOP Publishing, 2019:012031.
90. Herrero-Durá I, Cebrecos A, Picó R, Romero-García V, García-Raffi LM, Sánchez-Morcillo VJ. Sound absorption and diffusion by 2D arrays of Helmholtz resonators. *Applied Sciences*. 2020;10(5):1690.
91. Lan J, Li Y, Yu H, Li B, Liu X. Nonlinear effects in acoustic metamaterial based on a cylindrical pipe with ordered Helmholtz resonators. *Physics Letters A*. 2017;381(13):1111-7.
92. Öchsner A, Makvandi R. *Numerical Engineering Optimization*. Springer; 2020.
93. Corriou JP. *Numerical Methods of Optimization: Theory and Practice for Engineers*. Springer Optimization and Its Applications Vol. 187. Springer; 2021.
94. Nelder JA, Mead R. A simplex method for function minimization. *The computer journal*. 1965;7(4):308-13.
95. Zhou S, Fang Z. Optimization Design of Acoustic Performance of Underwater Anechoic Coatings. *Acoustics Australia*. 2022;50(3):297-313.
96. Baulac M, Defrance J, Jean P. Optimization of multiple edge barriers with genetic algorithms coupled with a Nelder–Mead local search. *Journal of sound and vibration*. 2007;300(1-2):71-87.
97. Gao F, Han L. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*. 2012;51(1):259-77.
98. Rayleigh L. LVI. On the influence of obstacles arranged in rectangular order upon the properties of a medium. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*. 1892;34(211):481-502.
99. Kafesaki M, Economou EN. Multiple-scattering theory for three-dimensional periodic acoustic composites. *Phys Rev B*. 1999 Nov;60:11993-2001.
100. Ishimaru A. *Wave Propagation and Scattering in Random Media*. Elsevier Science; 2013.

101. Gao TF, Shang EC, Li Q. *Theoretical and Computational Acoustics 2001: Beijing, China, 21-25 May 2001*. EBL-Schweitzer. World Scientific; 2002.
102. Romero-Garcia V, Hladky-Hennion AC. *Fundamentals and Applications of Acoustic Metamaterials: From Seismic to Radio Frequency*. Wiley; 2019.
103. Zangeneh-Nejad F, Fleury R. Active times for acoustic metamaterials. *Reviews in Physics*. 2019;4.
104. Bettess P, Zienkiewicz O. Diffraction and refraction of surface waves using finite and infinite elements. *International Journal for Numerical Methods in Engineering*. 1977;11(8):1271-90.
105. Grote MJ, Kirsch C. Dirichlet-to-Neumann boundary conditions for multiple scattering problems. *Journal of Computational Physics*. 2004;201(2):630-50.
106. Grote MJ, Sim I. Local nonreflecting boundary condition for time-dependent multiple scattering. *Journal of Computational Physics*. 2011;230(8):3135-54.
107. Acosta S, Villamizar V. Coupling of Dirichlet-to-Neumann boundary condition and finite difference methods in curvilinear coordinates for multiple scattering. *Journal of Computational Physics*. 2010;229(15):5498-517.
108. Geuzaine C, Vion A, Gaignaire R, Dular P, Sabariego RV. An amplitude finite element formulation for multiple-scattering by a collection of convex obstacles. *IEEE transactions on magnetics*. 2010;46(8):2963-6.
109. Villamizar V, Acosta S, Dastrup B. High order local absorbing boundary conditions for acoustic waves in terms of farfield expansions. *Journal of Computational Physics*. 2017;333:331-51.
110. Villamizar V, Badger J, Khajah T, Acosta S. High Order Farfield Expansion ABC coupled with IGA and finite differences applied to acoustic multiple scattering. *WAVES 2019, Vienna, Austria*. 2019.
111. Kittel C, McEuen P, McEuen P. *Introduction to solid state physics*. vol. 8. Wiley New York; 1996.

112. Villamizar V, Khajah T, Hale J, Jaiswal M. An Efficient High Order Numerical Method for Multiple Scattering. *WAVES 2022, Paris, France*. 2022.
113. BS EN 12792:2003. Ventilation for buildings – Symbols, terminology and graphical symbols. Geneva, CH: International Organization for Standardization; 2003.
114. Bies DA, Hansen CH, Howard CQ. *Engineering noise control*. CRC press; 2017.
115. ISO 7235:2003. Acoustics – Laboratory measurement procedures for ducted silencers and air-terminal units – Insertion loss, flow noise and total pressure loss. Geneva, CH: International Organization for Standardization; 2003.
116. ISO 11820:1996. Acoustics – Measurements on silencers in situ. Geneva, CH: International Organization for Standardization; 1996.
117. ISO 10140-2:2021. Acoustics – Laboratory measurement of sound insulation of building elements – Part 2: Measurement of airborne sound insulation. Geneva, CH: International Organization for Standardization; 2021.
118. ASTM E90-09. Standard Test Method for Laboratory Measurement of Airborne Sound Transmission Loss of Building Partitions and Elements. Pennsylvania, USA: American Society for Testing and Materials; 2016.
119. Höller C. Review and Comparison of ASTM and ISO Standards on Sound Transmission in Buildings. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*. vol. 258/5. Institute of Noise Control Engineering. 2018:2338-47.
120. Brittain F, Salter C. Insertion Losses of Acoustic Louvers. *The Journal of the Acoustical Society of America*. 1973;54(1):273-3.
121. Ma JH, Guo P. Analysis of performance of automotive exhaust muffler based on ANSYS finite element. *Applied Mechanics and Materials*. vol. 509. Trans Tech Publ. 2014:118-22.
122. Fu J, Xu M, Zhang Z, Kang W, He Y. Muffler structure improvement based on acoustic finite element analysis. *Journal of Low Frequency Noise, Vibration and Active Control*. 2019;38(2):415-26.

123. Cazzolato BS, Howard CQ, Hansen CH. Finite element analysis of an industrial reactive silencer. *The Fifth International Congress of Sound and Vibration, Adelaide, Australia (December 15–18, 1997)*. 1997:1659-67.
124. Bustos C, Jurdic V, Sharp C, Hiller D. Optimisation of railway noise barrier design using finite element and boundary element modelling methods. *Euronoise 2021*. 2021:390-9.
125. Chen GH, Herrin DW. A parametric investigation of louvered terminations for rectangular ducts. *SAE International Journal of Passenger Cars-Mechanical Systems*. 2015;8(2015-01-2356):1137-41.
126. Papadakis NM, Stavroulakis GE. Estimation Of Insertion Loss of Sound Barriers via Finite Element Method. *Proceedings of the 9th GRACM International Congress on Computational Mechanics, Chania, Greece*. 2018:4-6.
127. Papadakis NM, Stavroulakis GE. Finite element method for the estimation of insertion loss of noise barriers: comparison with various formulae (2D). *Urban Science*. 2020;4(4):77.
128. ISO 266:1997. Acoustics – Preferred frequencies. Geneva, CH: International Organization for Standardization; 1997.
129. Blevins RD. *Formulas for dynamics, acoustics and vibration*. John Wiley & Sons; 2015.
130. ANSI/ASA S1 8. Laboratory Methods of Testing Louvers for Rating. Illinois, USA: Air Movement and Control Association International Inc.; 2016.
131. Pierce AD. *Acoustics: An Introduction to Its Physical Principles and Applications*. Springer International Publishing; 2019.
132. Airwellcare. Acoustic Louvers. Technical Brochure. 2014.
133. Alara Lukagro Noise Control Solutions. Datasheet: acoustic louvers AL-R/150-300-300V-600V. Technical Brochure. 2017.
134. Barclay Engineering. Acoustic Louvers. Technical Brochure. 2009.

- 135. Caice. Continuous Line Acoustic Louvers. Technical Brochure. 2018.
- 136. Cestaro Bernardo. Light Acoustic Louvre: GAFI. Technical Brochure. 2017.
- 137. Ruskin. ACL445 Stationary Acoustical Louver - Formed Louver. Technical Brochure. 2022.
- 138. Stavoklima V O S . Acoustic Louvers: PHZE. Technical Brochure. 2011.
- 139. Swegon. ALD: Sound attenuating exterior wall grille. Technical Brochure. 2022.



APPENDIX A: SCATTERING FROM A CIRCULAR CYLINDER

A.1. EXACT SOLUTION FOR SOUND-HARD BODIES

The total field, u , is the summation of the incident, u_{inc} , and the scattered waves, u_s ,

$$u = u_{\text{inc}} + u_s. \quad (\text{A.1})$$

where the incident plane wave, u_{inc} is

$$u_{\text{inc}} = e^{ik(x \cos \alpha + y \sin \alpha)} = e^{ikR \cos(\theta - \alpha)}, \quad (\text{A.2})$$

In Eq.A.2, α , is angle of incidence. From Neumann's addition theorem, the Jacobi expansion is,

$$e^{i\omega \cos \psi} = \sum_{n=-\infty}^{\infty} i^n J_n(\omega) e^{in\psi} = \sum_{n=0}^{\infty} \epsilon_n i^n J_n(\omega) \cos n\psi, \quad (\text{A.3})$$

where ϵ_n is the Neumann factor. For $\epsilon_0 = 1$ and $\epsilon_n = 2$ for $n \geq 1$.

In order to satisfy Sommerfeld radiation condition, separated solutions of Helmholtz equation is combined

$$H_n^{(1)}(kR) e^{\pm in\theta} \equiv \{J_n(kR) + iY_n(kR)\} e^{\pm in\theta}. \quad (\text{A.4})$$

Hankel function $H_n^{(1)}$ satisfies Graf's addition theorem which gives rise to a cylindrical wave that propagates outwards. At this point, time dependence of the solution can be controlled by $e^{-i\omega t}$.

The scattered wave can be written as,

$$u_s = \sum_{n=0}^{\infty} \epsilon_n i^n B_n H_n(kR) \cos n(\theta - \alpha), \quad (\text{A.5})$$

to ensure that the symmetry with the incident wave is preserved by writing H_n instead of $H_n^{(1)}$.

The normal velocity at the surface of the cylinder, $R = R_{in}$, is equal to zero for a sound-hard

body,

$$\left. \frac{\partial u}{\partial R} \right|_{R=R_{in}} = 0. \quad (\text{A.6})$$

Differentiating Eq.A.2 and Eq.A.5, and setting $R = R_{in}$ gives

$$\sum_{n=0}^{\infty} \epsilon_n k i^n \{J'_n(kR_{in}) + B_n H'_n(kR_{in})\} \cos n(\theta - \alpha), \quad -\pi \leq \theta < \pi. \quad (\text{A.7})$$

It is possible to go a few steps more considering that $J'_m(\omega) = (d/d\omega)J_m(\omega)$ and $H'_m(\omega) = (d/d\omega)H_m(\omega)$. Then, orthogonality of the trigonometric functions implies that the expression inside the braces must be zero for each n , whence

$$B_n = -\frac{J'_n(kR_{in})}{H'_n(kR_{in})} \quad (\text{A.8})$$

and then the scattered field is given everywhere in $R \geq R_{in}$ by Eq.A.5.

A.2. EXACT SOLUTION FOR SOUND-SOFT BODIES

For sound-soft bodies, the boundary condition is:

$$u \Big|_{R=R_{in}} = 0. \quad (\text{A.9})$$

The corresponding solution for the scattered field is given by Equation A.5, but with:

$$B_n = -\frac{J_n(kR_{in})}{H_n(kR_{in})}. \quad (\text{A.10})$$

A.3. EXACT SOLUTION MODIFIED TO BGT-2 ABC

Considering an incident wave with a fixed mode m which is described as

$$u_m^{\text{inc}}(\mathbf{x}) = J_m(kR)e^{im\phi} \quad m \in \mathbb{Z}, \quad (\text{A.11})$$

where J_m is the m th order Bessel's function, the modified exact solution can be obtained as a summation of modal solutions

$$u_{\text{s,hard}}^{\text{ex,m}} = \sum_{m \in \mathbb{Z}} i^m u_{\text{s,m}}^{\text{ex}}, \quad (\text{A.12})$$

The exact exterior modal solution $u_{\text{s,m}}^{\text{ex}}$ to the truncated scattering problem of the inner sound-hard circular cylinder in polar coordinate (R, ϕ) is given as

$$u_{\text{s,m}}^{\text{ex}}(\mathbf{x}) = \left(a_m H_m^{(1)}(kR) + b_m H_m^{(2)}(kR) \right) e^{im\phi}, \quad R_{\text{out}} \geq r \geq R_{\text{in}}, \quad m \in \mathbb{Z} \quad (\text{A.13})$$

The functions $H_m^{(1)}$ and $H_m^{(2)}$ are the m^{th} order first- and second-kind Hankel functions, respectively. The Neumann boundary condition is applied on Γ_S and the ABC on Γ_R , resulting in the following linear system of equations to obtain the two unknown coefficients a_m and b_m :

$$a_m = \frac{A_{22}^m J'_m(kR_{\text{in}})}{D_m}, \quad b_m = \frac{A_{21}^m J'_m(kR_{\text{in}})}{D_m}, \quad (\text{A.14})$$

where

$$D_m = A_{11}^m A_{22}^m - A_{21}^m A_{12}^m,$$

$$A_{11}^m = H_m^{(1)}(kR_{\text{in}}),$$

$$A_{12}^m = H_m^{(2)}(kR_{\text{in}}),$$

$$A_{21}^m = kH_m^{(1)}(kR_{\text{out}}) - B_m H_m^{(1)}(kR_{\text{out}}),$$

$$A_{22}^m = kH_m^{(2)}(kR_{\text{out}}) - B_m H_m^{(2)}(kR_{\text{out}}),$$

$$B_m = -\left(\alpha_m \frac{m^2}{R_1^2} + \beta_m \right), \quad \alpha_m = -\frac{1}{2ik} \left(1 + \frac{i}{kR_{\text{out}}} \right)^{-1}, \quad \text{and}$$

$$\beta_m = -ik + \frac{1}{2R_{\text{out}}} + \frac{1}{8iR_{\text{out}}(i+kR_{\text{out}})}$$

APPENDIX B: WEAK FORM DERIVATIONS

FEniCS is not yet capable of making complex number calculations. Therefore, the weak forms must be represented in the format below:

$$a_r(u, v) + ia_i(u, v) = L_r(v) + iL_i(v). \quad (\text{B.1})$$

To do so, it is necessary to separate the equations to complex and real parts using:

$$u = u_r + iu_i \quad \text{and} \quad v = v_r + iv_i, \quad (\text{B.2})$$

where $u_r = \Re\{u\}$, $u_i = \Im\{u\}$, $v_r = \Re\{v\}$, and $v_i = \Im\{v\}$.

$$uv = u_r v_r + iu_r v_i + iu_i v_r - u_i v_i = (u_r v_r - u_i v_i) + i(u_r v_i + u_i v_r) \quad (\text{B.3})$$

$$\nabla u = \nabla u_r + i\nabla u_i \quad \nabla v = \nabla v_r + i\nabla v_i \quad (\text{B.4})$$

$$\nabla u \nabla v = (\nabla u_r \nabla v_r - \nabla u_i \nabla v_i) + i(\nabla u_r \nabla v_i + \nabla u_i \nabla v_r) \quad (\text{B.5})$$

If a complex conjugate test function is preferred to infer positive signs on the derived equations, then the real and imaginary parts can be decomposed with:

$$u = u_r + iu_i \quad \bar{v} = v_r - iv_i \quad (\text{B.6})$$

$$u\bar{v} = u_r v_r - iu_r v_i + iu_i v_r + u_i v_i = (u_r v_r + u_i v_i) + i(u_i v_r - u_r v_i) \quad (\text{B.7})$$

$$\nabla u = \nabla u_r + i\nabla u_i \quad \nabla \bar{v} = \nabla v_r - i\nabla v_i \quad (\text{B.8})$$

$$\nabla u \nabla \bar{v} = (\nabla u_r \nabla v_r + \nabla u_i \nabla v_i) + i(\nabla u_i \nabla v_r - \nabla u_r \nabla v_i) \quad (\text{B.9})$$

$$\bar{v}\nabla u = (v_r\nabla u_r + v_i\nabla u_i) + i(v_r\nabla u_i - v_i\nabla u_r) \quad (\text{B.10})$$

Remembering that $u_{\text{inc}} = e^{ikx}$ and using Euler's formula $e^{i\phi} = \cos \phi + i \sin \phi$, and taking $\phi = kx$, another useful transformation is:

$$\begin{aligned} u_{\text{inc}}\bar{v} &= (\cos(kx) + i \sin(kx))(v_r - iv_i) \\ &= \cos(kx)v_r + \sin(kx)v_i + i(\sin(kx)v_r - \cos(kx)v_i) \end{aligned} \quad (\text{B.11})$$



B.1. WEAK FORM: SCATTERING WAVE SOLUTION

The scattered wave u_s satisfies the Helmholtz equation:

$$\nabla^2 u_s + k^2 u_s = f \quad \text{in } \Omega^+. \quad (\text{B.12})$$

The boundary condition on the scattering obstacle for a sound-hard obstacle is:

$$\frac{\partial u_s}{\partial n} = -\frac{\partial u_{\text{inc}}}{\partial n}, \quad \text{on } \Gamma_S, \quad (\text{B.13})$$

and for a sound-soft obstacle is:

$$u = 0 \quad \implies \quad u_s = -u_{\text{inc}}. \quad (\text{B.14})$$

Problems with infinite exterior domains must also satisfy the Sommerfeld radiation condition which states that there are no incoming waves from infinity.

The time-harmonic Sommerfeld radiation condition is:

$$\lim_{r \rightarrow \infty} r^{(\delta-1)/2} \left(\frac{\partial u_s}{\partial r} - i k u_s \right) = 0, \quad (\text{B.15})$$

where $r = |\mathbf{x}|$, $i = \sqrt{-1}$, and $\delta = 2$ or 3 for two or three dimensions, respectively.

The Sommerfeld radiation condition is adapted to FEM by truncating the infinite domain, Ω^+ , from Γ_R where an ABC is used. The FEM is solved for the truncated solution domain, Ω .

By utilizing a test function v on the Helmholtz equation, the weak form required for the FEM can be obtained as:

$$\int_{\Omega} v \nabla^2 u_s \, d\Omega + k^2 \int_{\Omega} v u_s \, d\Omega = \int_{\Omega} v f \, d\Omega. \quad (\text{B.16})$$

Using the identity $v \nabla^2 u_s = \nabla \cdot (v \nabla u_s) - \nabla v \cdot \nabla u_s$, the weak form becomes:

$$\int_{\Omega} \nabla \cdot (v \nabla u_s) \, d\Omega - \int_{\Omega} \nabla v \cdot \nabla u_s \, d\Omega + k^2 \int_{\Omega} v u_s \, d\Omega = \int_{\Omega} v f \, d\Omega. \quad (\text{B.17})$$

Invoking the Green's theorem on Equation B.17, a general expression for the weak form is obtained as:

$$\int_{\Gamma_R} \mathbf{n} \cdot \nu \nabla u_s \, d\Gamma_R + \int_{\Gamma_S} \mathbf{n} \cdot \nu \nabla u_s \, d\Gamma_S - \int_{\Omega} \nabla \nu \cdot \nabla u_s \, d\Omega + k^2 \int_{\Omega} \nu u_s \, d\Omega = \int_{\Omega} \nu f \, d\Omega. \quad (\text{B.18})$$

B.1.1. Absorbing Boundary Condition

In FEM of scattering problems, the Sommerfeld's radiation condition at infinity is approximated with an absorbing boundary condition (ABC) on the artificial boundary Γ_R . The general form of ABCs are as follows:

$$\frac{\partial u_s}{\partial r} = Bu_s, \quad \text{on } \Gamma_R, \quad (\text{B.19})$$

where the operator B is called the *Dirichlet-to-Neumann map*. So that relevant term in the weak form can also be written as:

$$\int_{\Gamma_R} \mathbf{n} \cdot \nu \nabla u_s \, d\Gamma_R = \int_{\Gamma_R} \nu Bu_s \, d\Gamma_R \quad (\text{B.20})$$

Sommerfeld, BGT-1 and BGT-2 applications on the operator B are given in Appendix B.1.1.1, Appendix B.1.1.2 and Appendix B.1.1.3, respectively.

B.1.1.1. Sommerfeld ABC

Sommerfeld radiation condition can be imposed on the truncation perimeter of the exterior domain as:

$$Bu_s = iku_s, \quad \text{on } \Gamma_R. \quad (\text{B.21})$$

So that the boundary condition on Γ_R becomes:

$$\int_{\Gamma_R} \nu Bu_s \, d\Gamma_R = ik \int_{\Gamma_R} \nu u_s \, d\Gamma_R \quad (\text{B.22})$$

For FEniCS, this term can be decomposed into its real and imaginary parts as follows:

$$\begin{aligned}
 ik \int_{\Gamma_R} v u_s \, d\Gamma_R &= ik \int_{\Gamma_R} (u_r v_r - u_i v_i) + i(u_r v_i + u_i v_r) \, d\Gamma_R \\
 &= k \int_{\Gamma_R} -(u_r v_i + u_i v_r) + i(u_r v_r - u_i v_i) \, d\Gamma_R \\
 &= -k \int_{\Gamma_R} (u_r v_i + u_i v_r) - i(u_r v_r - u_i v_i) \, d\Gamma_R,
 \end{aligned} \tag{B.23}$$

where $u_r = \Re\{u_s\}$, $u_i = \Im\{u_s\}$, $v_r = \Re\{v\}$, and $v_i = \Im\{v\}$. Note that the subscript s denoting the scattered wave has been removed from the decomposed terms for readability of the formula.

B.1.1.2. BGT-1 ABC

The general formulation for BGT ABC for two dimensional Helmholtz equation is as follows:

$$B_m = \prod_{j=1}^m \left(\frac{\partial}{\partial r} - ik + \frac{4j-3}{2r} \right), \quad m \geq 1. \tag{B.24}$$

Formulation for the first order BGT ABC can be obtained by expanding this formula using $m = 1$:

$$B_1 u_s = \left(ik - \frac{1}{2R} \right) u_s \quad \text{on } \Gamma_R. \tag{B.25}$$

So that the boundary condition on Γ_R becomes:

$$\int_{\Gamma_R} v B u_s \, d\Gamma_R = \int_{\Gamma_R} \left(ik - \frac{1}{2R} \right) u_s v \, d\Gamma_R = ik \int_{\Gamma_R} u_s v \, d\Gamma_R - \int_{\Gamma_R} \frac{u_s v}{2R} \, d\Gamma_R. \tag{B.26}$$

It can be decomposed into its real and imaginary parts for FEniCS, term by term. Decomposing the first term:

$$\begin{aligned}
 ik \int_{\Gamma_R} v u_s \, d\Gamma_R &= ik \int_{\Gamma_R} (u_r v_r - u_i v_i) + i(u_r v_i + u_i v_r) \, d\Gamma_R \\
 &= k \int_{\Gamma_R} -(u_r v_i + u_i v_r) + i(u_r v_r - u_i v_i) \, d\Gamma_R \\
 &= -k \int_{\Gamma_R} (u_r v_i + u_i v_r) - i(u_r v_r - u_i v_i) \, d\Gamma_R.
 \end{aligned} \tag{B.27}$$

Decomposing the second term:

$$\int_{\Gamma_R} \frac{u_s v}{2R} d\Gamma_R = \int_{\Gamma_R} \frac{1}{2R} ((u_r v_r - u_i v_i) + i(u_r v_i + u_i v_r)) d\Gamma_R. \quad (B.28)$$

Note that for $u_r = \Re\{u_s\}$, $u_i = \Im\{u_s\}$, $v_r = \Re\{v\}$, and $v_i = \Im\{v\}$, the subscript s denoting the scattered wave has been removed from the decomposed terms for readability of the formula.

B.1.1.3. BGT-2 ABC

Considering the general formulation for BGT ABC for two dimensional Helmholtz equation:

$$B_m = \prod_{j=1}^m \left(\frac{\partial}{\partial r} - ik + \frac{4j-3}{2r} \right), \quad m \geq 1. \quad (B.29)$$

The second order BGT-2 ABC can be expanded as:

$$B_2 u_s = ik u_s - \frac{u_s}{2R} + \frac{u_s}{8R(1-ikR)} + \frac{1}{2R-2R^2 ik} \frac{\partial^2 u_s}{\partial \theta^2}. \quad (B.30)$$

So that the boundary condition on Γ_R becomes:

$$\begin{aligned} \int_{\Gamma_R} B u_s v d\Gamma_R &= ik \int_{\Gamma_R} u_s v d\Gamma_R - \int_{\Gamma_R} \frac{u_s v}{2R} d\Gamma_R \\ &+ \int_{\Gamma_R} \frac{u_s v}{8R(1-ikR)} d\Gamma_R + \int_{\Gamma_R} \frac{v}{2R(1-ikR)} \frac{\partial^2 u_s}{\partial \theta^2}, \end{aligned} \quad (B.31)$$

where $\partial^2/\partial \theta^2$ term needs to be further expanded. Considering $u_s(x, y) = u_s(R \cos \theta, R \sin \theta)$ and using the chain rule for the first derivative,

$$\frac{\partial u_s(x, y)}{\partial \theta} = -R \sin \theta \frac{\partial u_s}{\partial x} + R \cos \theta \frac{\partial u_s}{\partial y}. \quad (B.32)$$

Introducing the test function $v(r, \theta)$, further expansion follows as:

$$v \frac{\partial^2 u_s}{\partial \theta^2} = \frac{\partial}{\partial \theta} \left(v \frac{\partial u_s}{\partial \theta} \right) = \frac{\partial v}{\partial \theta} \frac{\partial u_s}{\partial \theta} + v \frac{\partial^2 u_s}{\partial \theta^2}, \quad (B.33)$$

With further mathematical manipulation a more useful formulation is going to be:

$$\frac{\partial}{\partial \theta} \left(v \frac{\partial u_s}{\partial \theta} \right) = \frac{\partial v}{\partial \theta} \frac{\partial u_s}{\partial \theta} + v \frac{\partial^2 u_s}{\partial \theta^2} \quad \rightarrow \quad v \frac{\partial^2 u_s}{\partial \theta^2} = \frac{\partial}{\partial \theta} \left(v \frac{\partial u_s}{\partial \theta} \right) - \frac{\partial v}{\partial \theta} \frac{\partial u_s}{\partial \theta}. \quad (B.34)$$

Now it is possible to solve for $v\partial^2 u_s/\partial\theta^2$ term by term. Lets name the first term, I, the second, II so that $v\partial^2 u_s/\partial\theta^2 = I - II$. For the ease of readability of the equations lets name the first term of I as I_1 , and the second term as I_2 .

$$\begin{aligned} I_1 &= (-v_{,x}R \sin \theta + v_{,y}R \cos \theta) (-u_{s,x}R \sin \theta + u_{s,y}R \cos \theta) \\ &= u_{s,x}v_{,x}R^2 \sin^2 \theta - u_{s,y}v_{,x}R^2 \cos \theta \sin \theta - u_{s,x}v_{,y}R^2 \cos \theta \sin \theta \\ &\quad + u_{s,y}v_{,y}R^2 \cos^2 \theta, \end{aligned} \quad (B.35)$$

$$I_2 = v \left(-\frac{\partial^2 u_s}{\partial x \partial \theta} R \sin \theta - \frac{\partial u_s}{\partial x} R \cos \theta \right) + v \left(\frac{\partial^2 u_s}{\partial y \partial \theta} R \cos \theta - \frac{\partial u_s}{\partial y} R \sin \theta \right). \quad (B.36)$$

First term of I_2 , which can be named I_{2A} is:

$$\begin{aligned} I_{2A} &= v \left(-\frac{\partial^2 u_s}{\partial x^2} \frac{\partial x}{\partial \theta} R \sin \theta - \frac{\partial^2 u_s}{\partial x \partial y} \frac{\partial y}{\partial \theta} R \sin \theta - \frac{\partial u_s}{\partial x} R \cos \theta \right) \\ &= v \left(-u_{s,xx}(-R \sin \theta) R \sin \theta - u_{s,xy}R^2 \cos \theta \sin \theta - u_{s,x}R \cos \theta \right) \\ &= v \left(u_{s,xx}R^2 \sin^2 \theta - u_{s,xy}R^2 \cos \theta \sin \theta - u_{s,x}R \cos \theta \right). \end{aligned} \quad (B.37)$$

Second term of I_2 , which can be named I_{2B} is:

$$\begin{aligned} I_{2B} &= v \left(\frac{\partial^2 u_s}{\partial y \partial x} \frac{\partial x}{\partial \theta} R \cos \theta + \frac{\partial^2 u_s}{\partial y \partial y} \frac{\partial y}{\partial \theta} R \cos \theta - \frac{\partial u_s}{\partial y} R \sin \theta \right) \\ &= v \left(-u_{s,yx}R^2 \sin \theta \cos \theta + u_{s,yy}R^2 \cos^2 \theta - u_{s,y}R \sin \theta \right). \end{aligned} \quad (B.38)$$

Returning back to II,

$$\begin{aligned} II &= \frac{\partial u_s}{\partial \theta} \frac{\partial v}{\partial \theta} = (-u_{s,x}R \sin \theta + u_{s,y}R \cos \theta) (-v_{,x}R \sin \theta + v_{,y}R \cos \theta) \\ &= u_{s,x}v_{,x}R^2 \sin^2 \theta - u_{s,x}v_{,y}R^2 \cos \theta \sin \theta \\ &\quad - u_{s,y}v_{,x}R^2 \cos \theta \sin \theta + u_{s,y}v_{,y}R^2 \cos^2 \theta. \end{aligned} \quad (B.39)$$

Combining all the terms together results in $v\partial^2 u_s/\partial\theta^2 = I_1 + I_{2A} + I_{2B} - II$. Notice that $I_1 = II$. Therefore, $v\partial^2 u_s/\partial\theta^2 = I_{2A} + I_{2B}$ which is equal to:

$$\begin{aligned} v \frac{\partial^2 u_s}{\partial \theta^2} &= u_{s,xx}vR^2 \sin^2 \theta - u_{s,xy}vR^2 \cos \theta \sin \theta - u_{s,x}vR \cos \theta \\ &\quad - u_{s,yx}vR^2 \sin \theta \cos \theta + u_{s,yy}vR^2 \cos^2 \theta - u_{s,y}vR \sin \theta. \end{aligned} \quad (B.40)$$

Now that the degree of derivative is lowered and all variables can be expressed in FEniCS, complex numbers can be decomposed into real and imaginary parts.

Following the terms of Equation B.31 in order, the first term is going to be:

$$\begin{aligned}
 ik \int_{\Gamma_R} v u_s d\Gamma_R &= ik \int_{\Gamma_R} (u_r v_r - u_i v_i) + i(u_r v_i + u_i v_r) d\Gamma_R \\
 &= k \int_{\Gamma_R} -(u_r v_i + u_i v_r) + i(u_r v_r - u_i v_i) d\Gamma_R \\
 &= -k \int_{\Gamma_R} (u_r v_i + u_i v_r) - i(u_r v_r - u_i v_i) d\Gamma_R.
 \end{aligned} \tag{B.41}$$

The second term is going to be:

$$\int_{\Gamma_R} \frac{u_s v}{2R} d\Gamma_R = \int_{\Gamma_R} \frac{1}{2R} ((u_r v_r - u_i v_i) + i(u_r v_i + u_i v_r)) d\Gamma_R \tag{B.42}$$

The third term is going to be:

$$\begin{aligned}
 \int_{\Gamma_R} \frac{v u_s}{8R(1 - ikR)} d\Gamma_R &= - \int_{\Gamma_R} \frac{-u_r v_r + u_i v_i}{8R^3 k^2 + 8R} d\Gamma_R - \int_{\Gamma_R} k \frac{u_i v_r + u_r v_i}{8R^2 k^2 + 8} d\Gamma_R \\
 &\quad - i \int_{\Gamma_R} k \frac{-u_r v_r + u_i v_i}{8R^2 k^2 + 8} d\Gamma_R - i \int_{\Gamma_R} \frac{-u_i v_r - u_r v_i}{8R^3 k^2 + 8R} d\Gamma_R
 \end{aligned} \tag{B.43}$$

The fourth term is going to be:

$$\begin{aligned}
 \int_{\Gamma_R} \frac{v}{2R - 2R^2 ik} d\Gamma_R \frac{\partial^2 u}{\partial \theta^2} &= \int_{\Gamma_R} \frac{v}{2R - 2R^2 ik} \\
 &\quad (u_{,xx} R^2 \sin^2 \theta - u_{,xy} R^2 \cos \theta \sin \theta - u_{,x} R \cos \theta - u_{,yx} R^2 \sin \theta \cos \theta \\
 &\quad + u_{,yy} R^2 \cos^2 \theta - u_{,y} R \sin \theta) d\Gamma_R
 \end{aligned} \tag{B.44}$$

where

$$\begin{aligned}
 \frac{uv}{2R - 2R^2 ik} &= - \int_{\Gamma_R} 2R \frac{u_r v_r}{4R^4 k^2 + 4R^2} d\Gamma_R + \int_{\Gamma_R} 2R \frac{u_i v_i}{4R^4 k^2 + 4R^2} d\Gamma_R \\
 &\quad + \int_{\Gamma_R} 2R^2 k \frac{u_i v_r}{4R^4 k^2 + 4R^2} d\Gamma_R + \int_{\Gamma_R} 2R^2 k \frac{u_r v_i}{4R^4 k^2 + 4R^2} d\Gamma_R \\
 &\quad - i \int_{\Gamma_R} 2R^2 k \frac{u_r v_r}{4R^4 k^2 + 4R^2} d\Gamma_R + i \int_{\Gamma_R} 2R^2 k \frac{u_i v_i}{4R^4 k^2 + 4R^2} d\Gamma_R \\
 &\quad - i \int_{\Gamma_R} 2R \frac{u_i v_r}{4R^4 k^2 + 4R^2} d\Gamma_R - i \int_{\Gamma_R} 2R \frac{u_r v_i}{4R^4 k^2 + 4R^2} d\Gamma_R.
 \end{aligned} \tag{B.45}$$

Making the multiplication, the fourth term can be further expanded as in the example below:

$$\begin{aligned} \int_{\Gamma_R} \frac{v}{2R - 2R^2ik} d\Gamma_R \frac{\partial^2 u}{\partial \theta^2} &= \int_{\Gamma_R} \left(-2R \frac{u_{r,xx} v_r}{4R^4 k^2 + 4R^2} \right) R^2 \sin^2 \theta \\ &+ \dots + i \left(2R \frac{u_{r,y} v_i}{4R^4 k^2 + 4R^2} \right) (-R \sin \theta) d\Gamma_R. \end{aligned} \quad (\text{B.46})$$

Note that for $u_r = \Re\{u_s\}$, $u_i = \Im\{u_s\}$, $v_r = \Re\{v\}$, and $v_i = \Im\{v\}$, the subscript s denoting the scattered wave has been removed from the decomposed terms for readability of the formula.

B.1.2. Scattering From a Sound-Hard Body

The incident wave, u_{inc} , is defined on the scattering body in sound-hard problems. Considering this, the boundary integral for Γ_S can be taken one step further by the use of the boundary condition, $\partial u_s / \partial n = -\partial u_{\text{inc}} / \partial n$ on the sound-hard object:

$$\int_{\Gamma_S} \mathbf{n} \cdot v \nabla u_s d\Gamma_S = - \int_{\Gamma_S} v \frac{\partial u_{\text{inc}}}{\partial n} d\Gamma_S. \quad (\text{B.47})$$

Note that, a simplified expression can also be written using $u_{\text{inc}} = e^{ikx} = e^{ikr \cos \theta}$ since $\partial u_{\text{inc}} / \partial r = ik \cos \theta u_{\text{inc}}$. Substituting this into Equation B.47:

$$\int_{\Gamma_S} \mathbf{n} \cdot v \nabla u_s d\Gamma_S = ik \int_{\Gamma_S} v \cos \theta u_{\text{inc}} d\Gamma_S, \quad (\text{B.48})$$

which is valid only for a circular cylinder is found.

Since the incident wave, u_{inc} is defined on Γ_S , the source term is taken as $f = 0$. Applying the boundary condition, Equation B.47, to the general weak form, Equation B.18, gives:

$$\int_{\Omega} \nabla v \cdot \nabla u_s d\Omega - k^2 \int_{\Omega} v u_s d\Omega - \int_{\Gamma_R} v B u_s d\Gamma_R = - \int_{\Gamma_S} v \frac{\partial u_{\text{inc}}}{\partial n} d\Gamma_S. \quad (\text{B.49})$$

B.1.3. Scattering From a Sound-Soft Body

In sound-soft scattering problems, the incident wave, u_{inc} , is defined as a source term in the entire solution domain. From Equation B.18:

$$\int_{\Omega_{\text{R}}} (\nabla u_{\text{S}} \cdot \nabla v - k^2 u_{\text{S}} v) \, d\Omega_{\text{R}} - \int_{\Gamma_{\text{R}}} B u_{\text{S}} v \, d\Gamma_{\text{R}} - \int_{\Gamma_{\text{S}}} \mathbf{n} \cdot \nabla u_{\text{S}} v \, d\Gamma_{\text{S}} = - \int_{\Gamma_{\Omega}} u_{\text{inc}} v \, d\Gamma_{\Omega}, \quad (\text{B.50})$$

B.2. WEAK FORM: TOTAL WAVE SOLUTION

A plane wave $u_{\text{inc}} = e^{ikx}$ with wave number $k > 0$ is incident on Γ_S . The total field u satisfies the Helmholtz equation:

$$\nabla^2 u + k^2 u = f \quad \text{in } \Omega, \quad (\text{B.51})$$

subject to Neumann boundary condition,

$$\frac{\partial u}{\partial n} = 0, \quad \text{on } \Gamma_S, \quad (\text{B.52})$$

considering a sound-hard scattering body.

Problems with infinite exterior domains must satisfy the Sommerfeld radiation condition given in Equation B.15.

The weak form for the total scattering wave can be derived from:

$$\int_{\Omega} \bar{v} \nabla^2 u \, d\Omega + k^2 \int_{\Omega} \bar{v} u \, d\Omega = \int_{\Omega} \bar{v} f \, d\Omega. \quad (\text{B.53})$$

Using the identity $\bar{v} \nabla^2 u = \nabla \cdot (\bar{v} \nabla u) - \nabla \bar{v} \cdot \nabla u$, the weak form becomes,

$$\int_{\Omega} \nabla \cdot (\bar{v} \nabla u) \, d\Omega - \int_{\Omega} \nabla \bar{v} \cdot \nabla u \, d\Omega + k^2 \int_{\Omega} \bar{v} u \, d\Omega = \int_{\Omega} \bar{v} f \, d\Omega. \quad (\text{B.54})$$

Invoking the Green's theorem $\int_{\Omega} \nabla \cdot (\bar{v} \nabla u) \, d\Omega = \int_{\Gamma} \mathbf{n} \cdot \bar{v} \nabla u \, d\Gamma$:

$$\int_{\Gamma_R + \Gamma_S} \mathbf{n} \cdot \bar{v} \nabla u \, d\Gamma - \int_{\Omega} \nabla \bar{v} \cdot \nabla u \, d\Omega + k^2 \int_{\Omega} \bar{v} u \, d\Omega = \int_{\Omega} \bar{v} f \, d\Omega. \quad (\text{B.55})$$

In finite element modeling of scattering problems, the Sommerfeld's radiation condition at infinity is approximated with an absorbing boundary condition (ABC) on the artificial boundary Γ_R . The general form of ABCs are as follows:

$$\frac{\partial u_s}{\partial r} = B u_s, \quad \text{on } \Gamma_R, \quad (\text{B.56})$$

where the operator B is called the *Dirichlet-to-Neumann map*. Using the identities $\mathbf{n} \cdot \bar{v} \nabla u = \bar{v}(\mathbf{n} \cdot \nabla u)$, and $\frac{\partial u}{\partial r} = \mathbf{n} \cdot \nabla u$ in the relevant term of the weak form:

$$\int_{\Gamma_R} \mathbf{n} \cdot \bar{v} \nabla u \, d\Gamma_R = \int_{\Gamma_R} \mathbf{n} \cdot \bar{v} \nabla (u_s + u_{\text{inc}}) \, d\Gamma_R \quad (\text{B.57})$$

$$= \int_{\Gamma_R} \bar{v} B u_s \, d\Gamma_R + \int_{\Gamma_R} \bar{v}(\mathbf{n} \cdot \nabla u_{\text{inc}}) \, d\Gamma_R \quad (\text{B.58})$$

$$= \int_{\Gamma_R} \bar{v} B u \, d\Gamma_R - \int_{\Gamma_R} \bar{v} B u_{\text{inc}} \, d\Gamma_R + \int_{\Gamma_R} \bar{v}(\mathbf{n} \cdot \nabla u_{\text{inc}}) \, d\Gamma_R \quad (\text{B.59})$$

Boundary integral for the scattering obstacle Γ_S can be written as:

$$\int_{\Gamma_S} \mathbf{n} \cdot \bar{v} \nabla u \, d\Gamma_S = \int_{\Gamma_S} \bar{v} \frac{\partial u}{\partial n} \, d\Gamma_S = 0. \quad (\text{B.60})$$

General weak form for finding the total wave can be written as:

$$\int_{\Gamma_R} \bar{v} B u \, d\Gamma_R - \int_{\Gamma_R} \bar{v} B u_{\text{inc}} \, d\Gamma_R + \int_{\Gamma_R} \bar{v}(\mathbf{n} \cdot \nabla u_{\text{inc}}) \, d\Gamma_R - \int_{\Omega} \nabla \bar{v} \cdot \nabla u \, d\Omega + k^2 \int_{\Omega} \bar{v} u \, d\Omega = \int_{\Omega} \bar{v} f \, d\Omega. \quad (\text{B.61})$$

Rearranging the terms:

$$\int_{\Gamma_R} \bar{v} B u \, d\Gamma_R - \int_{\Omega} \nabla \bar{v} \cdot \nabla u \, d\Omega + k^2 \int_{\Omega} \bar{v} u \, d\Omega = \int_{\Omega} \bar{v} f \, d\Omega + \int_{\Gamma_R} \bar{v} B u_{\text{inc}} \, d\Gamma_R - \int_{\Gamma_R} \bar{v}(\mathbf{n} \cdot \nabla u_{\text{inc}}) \, d\Gamma_R. \quad (\text{B.62})$$

This time, a complex conjugate test function will be preferred to infer positive signs on the real and imaginary parts of the quantity $a(u, v)$.

The weak form of the Helmholtz equation with Sommerfeld absorbing boundary condition becomes,

$$-\int_{\Omega} \nabla \bar{v} \cdot \nabla u d\Omega + k^2 \int_{\Omega} \bar{v} u d\Omega + \int_{\Gamma_R} \bar{v} B u d\Gamma_R = \int_{\Omega} \bar{v} f d\Omega + \int_{\Gamma_R} \bar{v} B u_{\text{inc}} d\Gamma_R - \int_{\Gamma_R} \bar{v} (\mathbf{n} \cdot \nabla u_{\text{inc}}) d\Gamma_R. \quad (\text{B.63})$$

The weak form will be represented as in the format below to implement it in FEniCS,

$$a(u, v) = L(v), \quad (\text{B.64})$$

where the terms are:

$$\begin{aligned} a(u, v) &= \int_{\Gamma_R} \bar{v} B u d\Gamma_R - \int_{\Omega} \nabla \bar{v} \cdot \nabla u d\Omega + k^2 \int_{\Omega} \bar{v} u d\Omega, \\ L(v) &= \int_{\Omega} \bar{v} f d\Omega + \int_{\Gamma_R} \bar{v} B u_{\text{inc}} d\Gamma_R - \int_{\Gamma_R} \bar{v} (\mathbf{n} \cdot \nabla u_{\text{inc}}) d\Gamma_R. \end{aligned} \quad (\text{B.65})$$

FEniCS is not yet capable of making complex number calculations. Therefore, the weak form will be represented in the format below:

$$a_r(u, v) + i a_i(u, v) = L_r(v) + i L_i(v). \quad (\text{B.66})$$

The First term for $a(u, v)$ is going to be:

$$\begin{aligned} a_1(u, v) &= - \int_{\Omega} \nabla \bar{v} \cdot \nabla u d\Omega \\ &= - \int_{\Omega} (\nabla u_r \nabla v_r + \nabla u_i \nabla v_i) + i (\nabla u_i \nabla v_r - \nabla u_r \nabla v_i) d\Omega. \end{aligned} \quad (\text{B.67})$$

The second term for $a(u, v)$ is going to be:

$$\begin{aligned} a_2(u, v) &= k^2 \int_{\Omega} u \bar{v} d\Omega \\ &= k^2 \int_{\Omega} (u_r v_r + u_i v_i) + i (u_i v_r - u_r v_i) d\Omega \end{aligned} \quad (\text{B.68})$$

The second term for $a(u, v)$ for $B = ik$ is going to be:

$$\begin{aligned}
a_3(u, v) &= \int_{\Gamma_R} \bar{v} B u \, d\Gamma_R \\
&= ik \int_{\Gamma_R} \bar{v} u \, d\Gamma_R \\
&= ik \int_{\Gamma_R} (u_r v_r + u_i v_i) + i(u_i v_r - u_r v_i) \, d\Gamma_R \\
&= k \int_{\Gamma_R} -(u_i v_r - u_r v_i) + i(u_r v_r + u_i v_i) \, d\Gamma_R \\
&= k \int_{\Gamma_R} (u_r v_i - u_i v_r) + i(u_r v_r + u_i v_i) \, d\Gamma_R.
\end{aligned} \tag{B.69}$$

Whereas the first term for $L(v)$ with $f = u_{\text{inc}}$ is going to be:

$$\begin{aligned}
L_1(v) &= \int_{\Omega} \bar{v} f \, d\Omega \\
&= \int_{\Omega} \bar{v} u_{\text{inc}} \, d\Omega \\
&= \int_{\Omega} \cos(kx) v_r + \sin(kx) v_i + i(\sin(kx) v_r - \cos(kx) v_i) \, d\Omega.
\end{aligned} \tag{B.70}$$

The second term for $L(v)$ term is going to be:

$$\begin{aligned}
L_2(v) &= ik \int_{\Gamma_R} u_{\text{inc}} \bar{v} \, d\Gamma_R \\
&= ik \int_{\Gamma_R} \cos(kx) v_r + \sin(kx) v_i + i(\sin(kx) v_r - \cos(kx) v_i) \, d\Gamma_R \\
&= k \int_{\Gamma_R} \cos(kx) v_i - \sin(kx) v_r + i(\cos(kx) v_r + \sin(kx) v_i) \, d\Gamma_R
\end{aligned} \tag{B.71}$$

The third term for $L(v)$ term is going to be:

$$\begin{aligned}
L_3(v) &= - \int_{\Gamma_R} \mathbf{n} \cdot \bar{v} \nabla u_{\text{inc}} \, d\Gamma_R \\
&= - \int_{\Gamma_R} \mathbf{n} \cdot \left\{ (v_r \nabla \cos(kx) + v_i \nabla \sin(kx) + i(v_r \nabla \sin(kx) - v_i \nabla \cos(kx))) \right\} \, d\Gamma_R.
\end{aligned} \tag{B.72}$$

APPENDIX C: PYTHON AND FENICS CODES

C.1. CODES: EXACT SOLUTION

Algorithm C.1. Exact solution for a sound-hard cylinder.

```

1 def Exact_Hard(x,y,r_in, m_const):
2
3     mn      = k*r_in+m_const; Theta  = np.arctan2(y,x)
4     radius = np.sqrt(pow(x,2)+pow(y,2))
5     eps_n = 2; u_sum = 0
6
7     for n in range(1, mn+1, 1):
8         J = jvp(n, k*r_in, 1)
9         H = h1vp(n, k*r_in, 1)
10        u_sum = u_sum - eps_n*pow(1j,n)\
11            *J/H*hankel1(n,k*radius)*np.cos(n*Theta)
12        J_0 = jvp(0, k*r_in, 1)
13        H_0 = h1vp(0, k*r_in, 1)
14
15        u_ex = -J_0/H_0*hankel1(0,k*radius) + u_sum
16
17    return u_ex

```

Algorithm C.2. Exact solution for a sound-soft cylinder.

```

1 def Exact_Soft(x,y,r_in, m_const):
2
3     mn      = int(round(k*r_in+m_const,0)); Theta  = np.arctan2(y,x)
4     radius = np.sqrt(pow(x,2)+pow(y,2))
5     eps_n = 2; u_sum = 0
6
7     for n in range(1, mn+1, 1):
8         J_n = jv(n, k*r_in)
9         H_n = hankel1(n, k*r_in)
10        B_n = -J_n/H_n
11        u_sum = u_sum + eps_n*pow(1j,n)\
12            *B_n*hankel1(n,k*radius)*np.cos(n*Theta)
13        J_0 = jv(0, k*r_in)
14        H_0 = hankel1(0, k*r_in)
15        B_0 = -J_0/H_0
16
17        u_ex = B_0*hankel1(0,k*radius) + u_sum
18
19    return u_ex

```

C.2. CODES: MODIFIED EXACT SOLUTION

Algorithm C.3. Modified exact solution for a sound-hard cylinder adapted to BGT-2 ABC.

```

1  def Exact_Modified(x,y,r_in,r_out,k, m_const):
2
3      mn      = int(round((k*r_in+m_const)/2-1,0))
4
5      Theta   = np.arctan2(y,x)
6      radius  = np.sqrt(pow(x,2)+pow(y,2))
7
8      u_mex = 0
9      for n in range(-mn, mn+1,1):
10
11          alpha_m = -1/(2*1j*k)*pow((1+1j/(k*r_out)),-1)
12          beta_m = -1j*k+1/(2*r_out)+1/(8*1j*r_out*(1j+k*r_out))
13          B_m = -(alpha_m*pow(n/r_out,2)+beta_m)
14
15          A_11 = h1vp(n, k*r_in, 1)
16          A_12 = h2vp(n, k*r_in, 1)
17
18          A_21 = k*h1vp(n, k*r_out, 1)-B_m*hankel1(n,k*r_out)
19          A_22 = k*h2vp(n, k*r_out, 1)-B_m*hankel2(n,k*r_out)
20
21          D_m = A_11*A_22-A_21*A_12
22
23          Jp_m = jvp(n, k*r_in, 1)
24          a_m = -A_22*Jp_m/D_m
25          b_m = A_21*Jp_m/D_m
26
27          H1_m = hankel1(n,k*radius)
28          H2_m = hankel2(n,k*radius)
29
30          u_ex_m = (a_m*H1_m+b_m*H2_m)*np.exp(1j*n*Theta)
31          u_mex = u_mex + pow(1j,n)*u_ex_m
32
33      return u_mex

```

C.3. CODES: MESH GENERATION

Algorithm C.4. Circular cylinder structured mesh

```

1 def mesh_structured(n_lambda, k, r_out, r_in, s):
2
3     n_radial      = np.ceil(n_lambda*k*(r_out-r_in)/(2*np.pi))
4     n_tangential  = np.ceil(2*pi*n_radial)
5
6     nr = int(n_radial); nt = int(n_tangential); Theta = 2*pi
7     a_x = r_in; a_y = 0; b_x = r_out; b_y = r_in
8     a = Point(a_x, a_y); b = Point(b_x, b_y)
9     mesh = RectangleMesh(a, b, nr, nt)
10    mesh.init(mesh.topology().dim(), 0)
11    c_to_vertex = mesh.topology()(mesh.topology().dim(), 0)
12    x = mesh.coordinates()[:, 0]
13    y = mesh.coordinates()[:, 1]
14
15    mf = MeshFunction("size_t", mesh, 0, 0)
16
17    class top(SubDomain):
18        def inside(self, x, on_boundary):
19            return on_boundary and near(x[1], a[1])
20
21    class bottom(SubDomain):
22        def inside(self, x, on_boundary):
23            return on_boundary and near(x[1], b[1])
24
25    # Create map between top and bottom on the square
26    # before deforming it
27    top().mark(mf, 1)
28    bottom().mark(mf, 2)
29    top_vertices = np.flatnonzero(mf.array() == 1)
30    bottom_vertices = np.flatnonzero(mf.array() == 2)
31    top_to_bottom = {}
32    for top in top_vertices:
33        bottom = np.flatnonzero(x[bottom_vertices] == x[top])[0]
34        top_to_bottom[top] = bottom_vertices[bottom]
35
36    def denser(x, y):
37        return [a_x + (b_x-a_x)*((x-a_x)/(b_x-a_x))**s, y]
38
39    def cylinder(r, s):
40        return [r*np.cos(Theta*s), r*np.sin(Theta*s)]

```

```

41 # Convert the mesh to hollow circle
42 s = 1; x_bar, y_bar = denser(x, y)
43 xy_bar_coor = np.array([x_bar, y_bar]).transpose()
44 mesh.coordinates()[:] = xy_bar_coor
45 x_hat, y_hat = cylinder(x_bar, y_bar)
46 xy_hat_coor = np.array([x_hat, y_hat]).transpose()
47 mesh.coordinates()[:] = xy_hat_coor
48
49 # Create mesh object and open editor
50 new_mesh = Mesh()
51 editor = MeshEditor()
52 topological_dim = 2; geometrical_dim = 2
53 num_local_vertices = mesh.num_entities(0) - len(top_vertices)
54 num_global_vertices = num_local_vertices
55 num_local_cells = mesh.num_cells()
56 num_global_cells = num_local_cells
57 editor.open(new_mesh, "triangle", \
58 topological_dim, geometrical_dim)
59 editor.init_vertices_global(num_local_vertices, \
60 num_global_vertices)
61 editor.init_cells_global(num_local_cells, num_global_cells)
62
63 # Add new vertices
64 # (renumber as there are fewer vertices in new mesh)
65 map_vertex = {}
66 v_count = 0
67 for i, coordinate in enumerate(mesh.coordinates()):
68     if not (i in top_vertices):
69         editor.add_vertex(v_count, coordinate)
70         map_vertex[i] = v_count
71         v_count += 1
72     else:
73         map_vertex[i] = top_to_bottom[i]
74
75 # Add the cells
76 for i in range(num_global_cells):
77     vertices = c_to_vertex(i)
78     new_vertices = []
79     for vertex in vertices:
80         if vertex in top_vertices:
81             new_vertices.append(map_vertex[map_vertex[vertex]])
82         else:
83             new_vertices.append(map_vertex[vertex])
84     editor.add_cell(i, np.asarray(new_vertices, dtype='uint'))
85
86 editor.close()
87 return new_mesh

```

Algorithm C.5. Circular cylinder in a cylindrical domain with a non-structured mesh

```

1 def mesh_geometry(n_lambda, k, r_out, xcyl, ycyl, r_in):
2
3     N = int(round((n_lambda*(r_out-xcyl)*k/(2*np.pi))/r_out,0))
4
5     mesh_script = 'SetFactory("OpenCASCADE");\n'
6     mesh_script += \
7     "Circle(1) = {{{0}}, {1}}, {2}}, {3}}, {4}}, {5}}};\n"\
8     .format(xcyl, ycyl, 0, r_in, 0, "2*Pi")
9     mesh_script += \
10    "Circle(2) = {{{0}}, {1}}, {2}}, {3}}, {4}}, {5}}};\n"\
11    .format(xcyl, ycyl, 0, r_out, 0, "2*Pi")
12
13    mesh_script += "Characteristic Length {2, 4, 3, 1, 5}"
14    mesh_script += " = {0};\n".format(np.round(1/N,3))
15
16    # define solution domain
17    mesh_script += "Curve Loop(1) = {2};\n"
18    mesh_script += "Curve Loop(2) = {1};\n"
19    mesh_script += "Plane Surface(1) = {1, 2};\n"
20    mesh_script += "Physical Surface(101) = {1};\n"
21
22    # define absorbing boundary
23    mesh_script += "Physical Curve(201) = {2};\n"
24    # define scattering body
25    mesh_script += "Physical Curve(301) = {1};\n"
26
27    return mesh_script

```

Algorithm C.6. Circular cylinder in a rectangular domain

```

1 def mesh_geometry(n_lambda, k, xmin, xmax, ymin, ymax,\
2 rcyl, xcyl, ycyl):
3
4     N = int(round((n_lambda*(xmax-rcyl)*k/(2*np.pi))/xmax,0))
5
6     mesh_script = "SetFactory('OpenCASCADE');\n"
7     mesh_script += "Point(1) = {{{0}, {1}, {2}, {3}}};\n"\
8     .format(xmin, ymin, 0, 1)
9     mesh_script += "Point(2) = {{{0}, {1}, {2}, {3}}};\n"\
10    .format(xmax, ymin, 0, 1)
11    mesh_script += "Point(3) = {{{0}, {1}, {2}, {3}}};\n"\
12    .format(xmax, ymax, 0, 1)
13    mesh_script += "Point(4) = {{{0}, {1}, {2}, {3}}};\n"\
14    .format(xmin, ymax, 0, 1)
15
16    mesh_script += \
17    "Circle(1) = {{{0}, {1}, {2}, {3}, {4}, {5}}};\n"\
18    .format(xcyl, ycyl, 0, rcyl, 0, "2*Pi")
19
20    mesh_script += "Line(2) = {1, 2};\n"
21    mesh_script += "Line(3) = {2, 3};\n"
22    mesh_script += "Line(4) = {3, 4};\n"
23    mesh_script += "Line(5) = {4, 1};\n"
24
25    mesh_script += "Characteristic Length {1, 2, 3, 4, 5}"
26    mesh_script += " = {0};\n".format(np.round(1/N,3))
27
28    mesh_script += "Curve Loop(1) = {4, 5, 2, 3};\n"
29    mesh_script += "Curve Loop(2) = {1};\n"
30    mesh_script += "Plane Surface(1) = {1, 2};\n"
31
32    mesh_script += "Physical Surface(101) = {1};\n"
33
34    # domain boundaries
35    mesh_script += "Physical Curve(201) = {2};\n"
36    mesh_script += "Physical Curve(202) = {3};\n"
37    mesh_script += "Physical Curve(203) = {4};\n"
38    mesh_script += "Physical Curve(204) = {5};"
39
40    # circle at the center
41    mesh_script += "Physical Curve(301) = {1};\n"
42
43    return mesh_script

```

Algorithm C.7. Rectangular domain with half cylinder on top

```

1 def mesh_half_cylinder(n_lambda, k, r_in, xcyl,\
2 xmin, xmax, ymin, ymax):
3
4     N = int(round((n_lambda*(xmax-xcyl)\
5     * k/(2*np.pi))/xmax,0))
6
7     mesh_script = "Point(1) = {{{0},{1},{2},{3}}};\n"\
8     .format(xmin, ymin, 0, 1)
9     mesh_script += "Point(2) = {{{0},{1},{2},{3}}};\n"\
10    .format(xmax, ymin, 0, 1)
11    mesh_script += "Point(3) = {{{0},{1},{2},{3}}};\n"\
12    .format(xmax, ymax, 0, 1)
13    mesh_script += "Point(4) = {{{0},{1},{2},{3}}};\n"\
14    .format(xmin, ymax, 0, 1)
15    mesh_script += "Point(5) = {{{0},{1},{2},{3}}};\n"\
16    .format(xcyl, ymax, 0, 1)
17    mesh_script += "Point(6) = {{{0},{1},{2},{3}}};\n"\
18    .format(xcyl-r_in, ymax, 0, 1)
19    mesh_script += "Point(7) = {{{0},{1},{2},{3}}};\n"\
20    .format(xcyl, ymax-r_in, 0, 1)
21    mesh_script += "Point(8) = {{{0},{1},{2},{3}}};\n"\
22    .format(xcyl+r_in, ymax, 0, 1)
23
24    mesh_script += "Circle(1) = {6, 5, 7};\n"
25    mesh_script += "Circle(2) = {7, 5, 8};\n"
26    mesh_script += "Line(3) = {1, 2};\n"
27    mesh_script += "Line(4) = {2, 3};\n"
28    mesh_script += "Line(5) = {3, 8};\n"
29    mesh_script += "Line(6) = {6, 4};\n"
30    mesh_script += "Line(7) = {4, 1};\n"
31
32    mesh_script += "Characteristic Length {1, 2, 3, 4, 8, 7, 6}"
33    mesh_script += " = {0};\n".format(np.round(1/N,3))
34
35    mesh_script += "Curve Loop(1) = {1, 2, -5, -4, -3, -7, -6};\n"
36    mesh_script += "Plane Surface(1) = {1};\n"
37    mesh_script += "Physical Surface(101) = {1};\n"
38    mesh_script += "Physical Curve(201) = {3};\n"
39    mesh_script += "Physical Curve(202) = {4};\n"
40    mesh_script += "Physical Curve(203) = {5, 6};\n"
41    mesh_script += "Physical Curve(204) = {7};\n"
42    mesh_script += "Physical Curve(301) = {1, 2};\n"
43
44    return mesh_script

```

Algorithm C.8. Crescent in circular domain

```

1 def mesh_geometry(n_lambda, k, r_out, xcyl_1, ycyl_1, r_1, \
2 xcyl_2, ycyl_2, r_2):
3
4     N = int(round((n_lambda*(r_out-xcyl_1)*k/(2*np.pi))/r_out,0))
5
6     mesh_script = 'SetFactory("OpenCASCADE");\n'
7     mesh_script += \
8     "Circle(1) = {{0}, {1}, {2}, {3}, {4}, {5}};\n"\
9     .format(xcyl_1, ycyl_1, 0, r_1, 0, "2*Pi")
10    mesh_script += \
11    "Circle(2) = {{0}, {1}, {2}, {3}, {4}, {5}};\n"\
12    .format(xcyl_2, ycyl_2, 0, r_2, 0, "2*Pi")
13    mesh_script += "Coherence;\n"
14    mesh_script += "Recursive Delete {Curve{2}};\n"
15    mesh_script += "Recursive Delete {Curve{5}};\n"
16    mesh_script += "Physical Curve(301) = {4, 6, 1, 3};\n"
17
18    mesh_script += \
19    "Circle(7) = {{0}, {1}, {2}, {3}, {4}, {5}};\n"\
20    .format(xcyl_1, ycyl_1, 0, r_out, 0, "2*Pi")
21    mesh_script += "Characteristic Length {2, 4, 3, 1, 5}"
22    mesh_script += " = {0};\n".format(np.round(1/N,3))
23    mesh_script += "Curve Loop(1) = {7};\n"
24    mesh_script += "Curve Loop(2) = {1, -4, -6, 3};\n"
25    mesh_script += "Plane Surface(1) = {1, 2};\n"
26    mesh_script += "Physical Surface(101) = {1};\n"
27    mesh_script += "Physical Curve(201) = {7};\n"
28
29    return mesh_script

```

Algorithm C.9. Crescent in rectangular domain

```

1 def mesh_geometry(n_lambda, k, xcyl_1, ycyl_1, r_1, \
2 xcyl_2, ycyl_2, r_2, xmin, xmax, ymin, ymax):
3
4     N = int(round((n_lambda*(xmax-r_1)*k/(2*np.pi)),0))
5     mesh_script = 'SetFactory("OpenCASCADE");\n'
6
7     mesh_script += \
8     "Circle(1) = {{{0}}, {1}}, {2}}, {3}}, {4}}, {5}}};\n"\
9     .format(xcyl_1, ycyl_1, 0, r_1, 0, "2*Pi")
10    mesh_script += \
11    "Circle(2) = {{{0}}, {1}}, {2}}, {3}}, {4}}, {5}}};\n"\
12    .format(xcyl_2, ycyl_2, 0, r_2, 0, "2*Pi")
13
14    mesh_script += "Coherence;\n"
15    mesh_script += "Recursive Delete {Curve{2}};\n"
16    mesh_script += "Recursive Delete {Curve{5}};\n"
17    mesh_script += "Physical Curve(301) = {4, 6, 1, 3};\n"
18
19    mesh_script += \
20    "Point(5) = {{{0}}, {1}}, {2}}, {3}}};\n"\
21    .format(xmin, ymin, 0, 1)
22    mesh_script += \
23    "Point(6) = {{{0}}, {1}}, {2}}, {3}}};\n"\
24    .format(xmax, ymin, 0, 1)
25    mesh_script += \
26    "Point(7) = {{{0}}, {1}}, {2}}, {3}}};\n"\
27    .format(xmax, ymax, 0, 1)
28    mesh_script += \
29    "Point(8) = {{{0}}, {1}}, {2}}, {3}}};\n"\
30    .format(xmin, ymax, 0, 1)
31
32    mesh_script += "Line(7) = {5, 6};\n"
33    mesh_script += "Line(8) = {6, 7};\n"
34    mesh_script += "Line(9) = {7, 8};\n"
35    mesh_script += "Line(10) = {8, 5};\n"
36
37    mesh_script += \
38    "Characteristic Length {2, 4, 3, 1, 5, 6, 7, 8}"
39    mesh_script += " = {0};\n".format(np.round(1/N,3))
40    mesh_script += "Curve Loop(1) = {9, 10, 7, 8};\n"
41    mesh_script += "Curve Loop(2) = {1, -4, -6, 3};\n"
42    mesh_script += "Plane Surface(1) = {1, 2};\n"
43    mesh_script += "Physical Surface(101) = {1};\n"
44    mesh_script += "Physical Line(201) = {7};\n"
45    mesh_script += "Physical Line(202) = {8};\n"
46    mesh_script += "Physical Line(203) = {9};\n"
47    mesh_script += "Physical Line(204) = {10};\n"
48
49    return mesh_script

```

Algorithm C.10. Rectangular domain with half cylinders on top and bottom

```

1 def mesh_geometry_2(n_lambda, k, xmin, xmax, ymin, ymax, \
2 r_1, xcyl_1, r_2, xcyl_2):
3
4     N = int(round((n_lambda*(xmax-xcyl_1)*k/(2*np.pi))/xmax,0))
5
6     mesh_script = "Point(1) = {{{0}, {1}, {2}, {3}}};\n" \
7     .format(xmin, ymin, 0, 1)
8     mesh_script += "Point(2) = {{{0}, {1}, {2}, {3}}};\n" \
9     .format(xmax, ymin, 0, 1)
10    mesh_script += "Point(3) = {{{0}, {1}, {2}, {3}}};\n" \
11    .format(xmax, ymax, 0, 1)
12    mesh_script += "Point(4) = {{{0}, {1}, {2}, {3}}};\n" \
13    .format(xmin, ymax, 0, 1)
14
15    mesh_script += "Point(5) = {{{0}, {1}, {2}, {3}}};\n" \
16    .format(xcyl_1, ymax, 0, 1)
17    mesh_script += "Point(6) = {{{0}, {1}, {2}, {3}}};\n" \
18    .format(xcyl_1-r_1, ymax, 0, 1)
19    mesh_script += "Point(7) = {{{0}, {1}, {2}, {3}}};\n" \
20    .format(xcyl_1, ymax-r_1, 0, 1)
21    mesh_script += "Point(8) = {{{0}, {1}, {2}, {3}}};\n" \
22    .format(xcyl_1+r_1, ymax, 0, 1)
23
24    mesh_script += "Point(9) = {{{0}, {1}, {2}, {3}}};\n" \
25    .format(xcyl_2, ymin, 0, 1)
26    mesh_script += "Point(10) = {{{0}, {1}, {2}, {3}}};\n" \
27    .format(xcyl_2-r_2, ymin, 0, 1)
28    mesh_script += "Point(11) = {{{0}, {1}, {2}, {3}}};\n" \
29    .format(xcyl_2, ymin+r_2, 0, 1)
30    mesh_script += "Point(12) = {{{0}, {1}, {2}, {3}}};\n" \
31    .format(xcyl_2+r_2, ymin, 0, 1)
32
33    mesh_script += "Circle(1) = {8, 5, 7};\n"
34    mesh_script += "Circle(2) = {7, 5, 6};\n"
35    mesh_script += "Circle(3) = {10, 9, 11};\n"
36    mesh_script += "Circle(4) = {11, 9, 12};\n"
37
38    mesh_script += "Line(5) = {1, 10};\n"
39    mesh_script += "Line(6) = {12, 2};\n"
40    mesh_script += "Line(7) = {2, 3};\n"
41    mesh_script += "Line(8) = {3, 8};\n"
42    mesh_script += "Line(9) = {6, 4};\n"
43    mesh_script += "Line(10) = {4, 1};\n"

```

```

44 mesh_script += \
45 "Characteristic Length {1, 10, 11, 12, 2, 3, 8, 7, 6, 4}"
46 mesh_script += " = {0};\n".format(np.round(1/N,3))
47
48 mesh_script += \
49 "Curve Loop(1) = {10, 5, 3, 4, 6, 7, 8, 1, 2, 9};\n"
50 mesh_script += "Plane Surface(1) = {1};\n"
51
52 mesh_script += "Physical Surface(101) = {1};\n"
53
54 # domain boundaries
55 mesh_script += "Physical Curve(201) = {5, 6};\n"
56 mesh_script += "Physical Curve(202) = {7};\n"
57 mesh_script += "Physical Curve(203) = {8, 9};\n"
58 mesh_script += "Physical Curve(204) = {10};"
59
60 # half-circle
61 mesh_script += "Physical Curve(301) = {1, 2, 4, 3};\n"
62
63 return mesh_script

```

Algorithm C.11. Rectangular domain with half cylinder on top and full cylinder in the middle of the domain

```

1 def mesh_geometry(n_lambda, k, xcyl_1, ycyl_1, r_1,\
2 xcyl_2, ycyl_2, r_2, xmin, xmax, ymin, ymax):
3
4     N = int(round((n_lambda*(xmax-r_1)*k/(2*np.pi)),0))
5
6     mesh_script = "Point(1) = {{{0}}, {1}, {2}, {3}}};\n"\
7     .format(xmin, ymin, 0, 1)
8     mesh_script += "Point(2) = {{{0}}, {1}, {2}, {3}}};\n"\
9     .format(xmax, ymin, 0, 1)
10    mesh_script += "Point(3) = {{{0}}, {1}, {2}, {3}}};\n"\
11    .format(xmax, ymax, 0, 1)
12    mesh_script += "Point(4) = {{{0}}, {1}, {2}, {3}}};\n"\
13    .format(xmin, ymax, 0, 1)
14
15    mesh_script += "Point(5) = {{{0}}, {1}, {2}, {3}}};\n"\
16    .format(xcyl_1+r_1, ymax, 0, 1)
17    mesh_script += "Point(6) = {{{0}}, {1}, {2}, {3}}};\n"\
18    .format(xcyl_1-r_1, ymax, 0, 1)
19    mesh_script += "Point(7) = {{{0}}, {1}, {2}, {3}}};\n"\
20    .format(xcyl_1, ymax, 0, 1)
21    mesh_script += "Point(8) = {{{0}}, {1}, {2}, {3}}};\n"\
22    .format(xcyl_1, ymax-r_1, 0, 1)
23
24    mesh_script += "Point(9) = {{{0}}, {1}, {2}, {3}}};\n"\
25    .format(xcyl_2, ycyl_2, 0, 1)
26    mesh_script += "Point(10) = {{{0}}, {1}, {2}, {3}}};\n"\
27    .format(xcyl_2+r_2, ycyl_2, 0, 1)
28    mesh_script += "Point(11) = {{{0}}, {1}, {2}, {3}}};\n"\
29    .format(xcyl_2, ycyl_2+r_2, 0, 1)
30    mesh_script += "Point(12) = {{{0}}, {1}, {2}, {3}}};\n"\
31    .format(xcyl_2-r_2, ycyl_2, 0, 1)
32    mesh_script += "Point(13) = {{{0}}, {1}, {2}, {3}}};\n"\
33    .format(xcyl_2, ycyl_2-r_2, 0, 1)
34
35    mesh_script += "Line(1) = {1, 2};\n"
36    mesh_script += "Line(2) = {2, 3};\n"
37    mesh_script += "Line(3) = {3, 5};\n"
38    mesh_script += "Line(4) = {6, 4};\n"
39    mesh_script += "Line(5) = {4, 1};\n"
40
41    mesh_script += "Circle(6) = {5, 7, 8};\n"
42    mesh_script += "Circle(7) = {8, 7, 6};\n"
43
44    mesh_script += "Circle(8) = {10, 9, 11};\n"
45    mesh_script += "Circle(9) = {11, 9, 12};\n"
46    mesh_script += "Circle(10) = {12, 9, 13};\n"
47    mesh_script += "Circle(11) = {13, 9, 10};\n"

```

```
48 mesh_script += \  
49 "Characteristic Length {1,2,3,4,5,8,6,10,11,12,13}"  
50 mesh_script += " = {0};\n".format(np.round(1/N,3))  
51  
52 mesh_script += "Curve Loop(1) = {7, 4, 5, 1, 2, 3, 6};\n"  
53 mesh_script += "Curve Loop(2) = {9, 10, 11, 8};\n"  
54 mesh_script += "Plane Surface(1) = {1, 2};\n"  
55  
56 mesh_script += "Physical Surface(101) = {1};\n"  
57 mesh_script += "Physical Line(201) = {1};\n"  
58 mesh_script += "Physical Line(202) = {2};\n"  
59 mesh_script += "Physical Line(203) = {3, 4};\n"  
60 mesh_script += "Physical Line(204) = {5};\n"  
61 mesh_script += "Physical Line(301) = {6, 7, 8, 9, 10, 11};"  
62  
63 return mesh_script
```

Algorithm C.12. Rectangular domain with two cylinders in a vertical alignment

```

1 def mesh_geometry(n_lambda, k, xcyl_1, ycyl_1, r_1,\
2 xcyl_2, ycyl_2, r_2, xmin, xmax, ymin, ymax):
3
4     N = int(round((n_lambda*(xmax-xcyl_1)*k/(2*np.pi))/xmax,0))
5
6     mesh_script = 'SetFactory("OpenCASCADE");\n'
7     mesh_script += "Point(1) = {{{0}}, {1}, {2}, {3}}};\n"\
8     .format(xmin, ymin, 0, 1)
9     mesh_script += "Point(2) = {{{0}}, {1}, {2}, {3}}};\n"\
10    .format(xmax, ymin, 0, 1)
11    mesh_script += "Point(3) = {{{0}}, {1}, {2}, {3}}};\n"\
12    .format(xmax, ymax, 0, 1)
13    mesh_script += "Point(4) = {{{0}}, {1}, {2}, {3}}};\n"\
14    .format(xmin, ymax, 0, 1)
15
16    mesh_script += "Line(1) = {1, 2};\n"
17    mesh_script += "Line(2) = {2, 3};\n"
18    mesh_script += "Line(3) = {3, 4};\n"
19    mesh_script += "Line(4) = {4, 1};\n"
20
21    mesh_script += \
22    "Circle(5) = {{{0}}, {1}, {2}, {3}, {4}, {5}}};\n"\
23    .format(xcyl_1, ycyl_1, 0, r_1, 0, "2*Pi")
24    mesh_script += \
25    "Circle(6) = {{{0}}, {1}, {2}, {3}, {4}, {5}}};\n"\
26    .format(xcyl_2, ycyl_2, 0, r_2, 0, "2*Pi")
27
28    mesh_script += "Characteristic Length {1, 2, 3, 4, 5, 6}"
29    mesh_script += " = {0};\n".format(np.round(1/N,3))
30
31    mesh_script += "Curve Loop(1) = {3, 4, 1, 2};\n"
32    mesh_script += "Curve Loop(2) = {5};\n"
33    mesh_script += "Curve Loop(3) = {6};\n"
34    mesh_script += "Plane Surface(1) = {1, 2, 3};\n"
35
36    mesh_script += "Physical Surface(101) = {1};\n"
37    mesh_script += "Physical Line(201) = {1};\n"
38    mesh_script += "Physical Line(202) = {2};\n"
39    mesh_script += "Physical Line(203) = {3};\n"
40    mesh_script += "Physical Line(204) = {4};\n"
41    mesh_script += "Physical Line(301) = {5, 6};"
42
43    return mesh_script

```

Algorithm C.13. Louver section: Case 1

```

1  def mesh_case_1(l_w, l_h0, l_b, l_a, l_s, N):
2
3      P01 = [ l_w/2, l_h0]; P02 = [-l_w/2, l_h0]
4      P03 = [-l_w/2, -l_h0]; P04 = [ l_w/2, -l_h0]
5
6      P05 = [ l_w/2, l_h0-l_b]
7      P06 = [ l_w/2, l_h0-l_b-l_a]
8      P07 = [ l_w/2, l_h0-l_b-l_a-l_b]
9
10     P08 = [-l_w/2, l_h0-l_a]
11     P09 = [-l_w/2, l_h0-l_a-l_b]
12     P10 = [-l_w/2, l_h0-l_a-l_b-l_a]
13
14     P11 = [-l_w/2-l_s, -l_h0]; P12 = [ l_w/2+l_s, -l_h0]
15     P13 = [ l_w/2+l_s, l_h0]; P14 = [-l_w/2-l_s, l_h0]
16
17     mesh_script = "Point(1) = {{{0}}, {1}, {2}, {3}}};\n".\
18     .format(P01[0], P01[1], 0, 1)
19     mesh_script += "Point(2) = {{{0}}, {1}, {2}, {3}}};\n"\
20     .format(P02[0], P02[1], 0, 1)
21     mesh_script += "Point(3) = {{{0}}, {1}, {2}, {3}}};\n"\
22     .format(P03[0], P03[1], 0, 1)
23     mesh_script += "Point(4) = {{{0}}, {1}, {2}, {3}}};\n"\
24     .format(P04[0], P04[1], 0, 1)
25     mesh_script += "Point(5) = {{{0}}, {1}, {2}, {3}}};\n"\
26     .format(P05[0], P05[1], 0, 1)
27     mesh_script += "Point(6) = {{{0}}, {1}, {2}, {3}}};\n"\
28     .format(P06[0], P06[1], 0, 1)
29     mesh_script += "Point(7) = {{{0}}, {1}, {2}, {3}}};\n"\
30     .format(P07[0], P07[1], 0, 1)
31     mesh_script += "Point(8) = {{{0}}, {1}, {2}, {3}}};\n"\
32     .format(P08[0], P08[1], 0, 1)
33     mesh_script += "Point(9) = {{{0}}, {1}, {2}, {3}}};\n"\
34     .format(P09[0], P09[1], 0, 1)
35     mesh_script += "Point(10) = {{{0}}, {1}, {2}, {3}}};\n"\
36     .format(P10[0], P10[1], 0, 1)
37     mesh_script += "Point(11) = {{{0}}, {1}, {2}, {3}}};\n"\
38     .format(P11[0], P11[1], 0, 1)
39     mesh_script += "Point(12) = {{{0}}, {1}, {2}, {3}}};\n"\
40     .format(P12[0], P12[1], 0, 1)
41     mesh_script += "Point(13) = {{{0}}, {1}, {2}, {3}}};\n"\
42     .format(P13[0], P13[1], 0, 1)
43     mesh_script += "Point(14) = {{{0}}, {1}, {2}, {3}}};\n"\
44     .format(P14[0], P14[1], 0, 1)

```

```

45 # Bottom boundary
46 mesh_script += "Line(1) = {11, 3};\n"
47 mesh_script += "Line(2) = {4, 12};\n"
48
49 # Right boundary
50 mesh_script += "Line(3) = {12, 13};\n"
51
52 # Top boundary
53 mesh_script += "Line(4) = {13, 1};\n"
54 mesh_script += "Line(5) = { 2, 14};\n"
55
56 # Left boundary
57 mesh_script += "Line(6) = {14, 11};\n"
58
59 # Louver frame
60 mesh_script += "Line(7) = { 3,10};\n"
61 mesh_script += "Line(8) = {10, 4};\n"
62 mesh_script += "Line(9) = { 6, 8};\n"
63 mesh_script += "Line(10) = { 9, 7};\n"
64 mesh_script += "Line(11) = { 7, 6};\n"
65 mesh_script += "Line(12) = { 8, 9};\n"
66 mesh_script += "Line(13) = { 1, 5};\n"
67 mesh_script += "Line(14) = { 5, 2};\n"
68
69 mesh_script += \
70 "Characteristic Length {1,2,3,4,5,6,7,8,9,10,11,12,13,14}"
71 mesh_script += " = {0};\n".format(np.round(1/N,3))
72
73 mesh_script += \
74 "Curve Loop(1) = {5,6,1,7,8,2,3,4,13,14};\n"
75 mesh_script += "Curve Loop(2) = {9, 12, 10, 11};\n"
76 mesh_script += "Plane Surface(1) = {1, 2};\n"
77 mesh_script += "Physical Surface(101) = {1};\n"
78
79 mesh_script += "Physical Curve(201) = {1, 2};\n"
80 mesh_script += "Physical Curve(202) = {3};\n"
81 mesh_script += "Physical Curve(203) = {4, 5};\n"
82 mesh_script += "Physical Curve(204) = {6};\n"
83
84 mesh_script += \
85 "Physical Curve(301) = {7, 8, 9, 10, 11, 12, 13, 14};\n"
86
87 return mesh_script

```

Algorithm C.14. Louver section: Case 2

```

1  def mesh_case_2(l_alpha, l_w, l_h0, l_b, l_a, l_s, N):
2
3      l_m = l_b/(2*np.tan(l_alpha*np.pi/180))
4      l_e = l_w/2 - l_m
5      l_t = l_e*(np.tan(l_alpha*np.pi/180))
6      l_f = l_a-l_t
7
8      P01 = [ l_m, l_h0]; P02 = [-l_m, l_h0]
9      P03 = [-l_m,-l_h0]; P04 = [ l_m,-l_h0]
10
11     P05 = [ l_w/2, l_h0-l_t]
12     P06 = [ l_w/2, l_h0-l_t-l_b]
13     P07 = [ l_w/2, l_h0-l_t-l_b-l_a]
14     P08 = [ l_w/2, l_h0-l_t-l_b-l_a-l_b]
15
16     P09 = [-l_w/2, l_h0-l_f]
17     P10 = [-l_w/2, l_h0-l_f-l_b]
18     P11 = [-l_w/2, l_h0-l_f-l_b-l_a]
19     P12 = [-l_w/2, l_h0-l_f-l_b-l_a-l_b]
20
21     P13 = [-l_w/2-l_s,-l_h0]; P14 = [ l_w/2+l_s,-l_h0]
22     P15 = [ l_w/2+l_s, l_h0]; P16 = [-l_w/2-l_s, l_h0]
23
24     mesh_script = "Point(1) = {{{0}}, {1}, {2}, {3}}};\n" \
25     .format(P01[0], P01[1], 0, 1)
26     mesh_script += "Point(2) = {{{0}}, {1}, {2}, {3}}};\n" \
27     .format(P02[0], P02[1], 0, 1)
28     mesh_script += "Point(3) = {{{0}}, {1}, {2}, {3}}};\n" \
29     .format(P03[0], P03[1], 0, 1)
30     mesh_script += "Point(4) = {{{0}}, {1}, {2}, {3}}};\n" \
31     .format(P04[0], P04[1], 0, 1)
32
33     mesh_script += "Point(5) = {{{0}}, {1}, {2}, {3}}};\n" \
34     .format(P05[0], P05[1], 0, 1)
35     mesh_script += "Point(6) = {{{0}}, {1}, {2}, {3}}};\n" \
36     .format(P06[0], P06[1], 0, 1)
37     mesh_script += "Point(7) = {{{0}}, {1}, {2}, {3}}};\n" \
38     .format(P07[0], P07[1], 0, 1)
39     mesh_script += "Point(8) = {{{0}}, {1}, {2}, {3}}};\n" \
40     .format(P08[0], P08[1], 0, 1)
41
42     mesh_script += "Point(9) = {{{0}}, {1}, {2}, {3}}};\n" \
43     .format(P09[0], P09[1], 0, 1)
44     mesh_script += "Point(10) = {{{0}}, {1}, {2}, {3}}};\n" \
45     .format(P10[0], P10[1], 0, 1)
46     mesh_script += "Point(11) = {{{0}}, {1}, {2}, {3}}};\n" \
47     .format(P11[0], P11[1], 0, 1)
48     mesh_script += "Point(12) = {{{0}}, {1}, {2}, {3}}};\n" \
49     .format(P12[0], P12[1], 0, 1)

```

```

50 mesh_script += "Point(13) = {{{0}, {1}, {2}, {3}}};\n" \
51 .format(P13[0], P13[1], 0, 1)
52 mesh_script += "Point(14) = {{{0}, {1}, {2}, {3}}};\n" \
53 .format(P14[0], P14[1], 0, 1)
54 mesh_script += "Point(15) = {{{0}, {1}, {2}, {3}}};\n" \
55 .format(P15[0], P15[1], 0, 1)
56 mesh_script += "Point(16) = {{{0}, {1}, {2}, {3}}};\n" \
57 .format(P16[0], P16[1], 0, 1)
58
59 # Bottom boundary
60 mesh_script += "Line(1) = {13, 3};\n"
61 mesh_script += "Line(2) = {4, 14};\n"
62 # Right boundary
63 mesh_script += "Line(3) = {14, 15};\n"
64 # Top boundary
65 mesh_script += "Line(4) = {15, 1};\n"
66 mesh_script += "Line(5) = {2, 16};\n"
67 # Left boundary
68 mesh_script += "Line(6) = {16, 13};\n"
69 # Louver frame
70 mesh_script += "Line(7) = { 3, 12};\n"
71 mesh_script += "Line(8) = {12, 11};\n"
72 mesh_script += "Line(9) = {11, 4};\n"
73 mesh_script += "Line(10) = {7, 9};\n"
74 mesh_script += "Line(11) = {9, 10};\n"
75 mesh_script += "Line(12) = {10, 8};\n"
76 mesh_script += "Line(13) = { 8, 7};\n"
77 mesh_script += "Line(14) = { 1, 5};\n"
78 mesh_script += "Line(15) = { 5, 6};\n"
79 mesh_script += "Line(16) = { 6, 2};\n"
80
81 mesh_script += \
82 "Characteristic Length \
83 {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}"
84 mesh_script += " = {0};\n".format(np.round(1/N,3))
85
86 mesh_script += \
87 "Curve Loop(1) = {5, 6, 1, 7, 8, 9, 2, 3, 4, 14, 15, 16};\n"
88 mesh_script += "Curve Loop(2) = {10, 11, 12, 13};\n"
89 mesh_script += "Plane Surface(1) = {1, 2};\n"
90 mesh_script += "Physical Surface(101) = {1};\n"
91 mesh_script += "Physical Curve(201) = {1, 2};\n"
92 mesh_script += "Physical Curve(202) = {3};\n"
93 mesh_script += "Physical Curve(203) = {4, 5};\n"
94 mesh_script += "Physical Curve(204) = {6};\n"
95 mesh_script += \
96 "Physical Curve(301) = {7, 8, 9, 10, 11, 12, 13, 14, 15, 16};"
97
98 return mesh_script

```

Algorithm C.15. Louver section: Case 3

```

1 def mesh_case_3(l_alpha, l_w, l_h0, l_b, l_a, l_s, N):
2
3     l_m = l_b/(2*np.tan(l_alpha*np.pi/180))
4     l_e = l_w/2 - l_m
5     l_t = l_e*(np.tan(l_alpha*np.pi/180))
6     l_f = l_a-l_t
7
8     P01 = [ l_m, l_h0]; P02 = [-l_m, l_h0]
9     P03 = [-l_m,-l_h0]; P04 = [ l_m,-l_h0]
10
11     P05 = [ l_w/2, l_h0-l_t]
12     P06 = [ l_w/2, l_h0-l_t-l_b]
13     P07 = [ l_w/2, l_h0-l_t-l_b-l_a]
14     P08 = [ l_w/2, l_h0-l_t-l_b-l_a-l_b]
15
16     P09 = [-l_w/2, l_h0-l_f]
17     P10 = [-l_w/2, l_h0-l_f-l_b]
18     P11 = [-l_w/2, l_h0-l_f-l_b-l_a]
19     P12 = [-l_w/2, l_h0-l_f-l_b-l_a-l_b]
20
21     P13 = [-l_w/2-l_s,-l_h0]; P14 = [ l_w/2+l_s,-l_h0]
22     P15 = [ l_w/2+l_s, l_h0]; P16 = [-l_w/2-l_s, l_h0]
23
24     mesh_script = "Point(1) = {{{0}}, {1}, {2}, {3}}};\n" \
25     .format(P01[0], P01[1], 0, 1)
26     mesh_script += "Point(2) = {{{0}}, {1}, {2}, {3}}};\n" \
27     .format(P02[0], P02[1], 0, 1)
28     mesh_script += "Point(3) = {{{0}}, {1}, {2}, {3}}};\n" \
29     .format(P03[0], P03[1], 0, 1)
30     mesh_script += "Point(4) = {{{0}}, {1}, {2}, {3}}};\n" \
31     .format(P04[0], P04[1], 0, 1)
32
33     mesh_script += "Point(5) = {{{0}}, {1}, {2}, {3}}};\n" \
34     .format(P05[0], P05[1], 0, 1)
35     mesh_script += "Point(6) = {{{0}}, {1}, {2}, {3}}};\n" \
36     .format(P06[0], P06[1], 0, 1)
37     mesh_script += "Point(7) = {{{0}}, {1}, {2}, {3}}};\n" \
38     .format(P07[0], P07[1], 0, 1)
39     mesh_script += "Point(8) = {{{0}}, {1}, {2}, {3}}};\n" \
40     .format(P08[0], P08[1], 0, 1)
41
42     mesh_script += "Point(9) = {{{0}}, {1}, {2}, {3}}};\n" \
43     .format(P09[0], P09[1], 0, 1)
44     mesh_script += "Point(10) = {{{0}}, {1}, {2}, {3}}};\n" \
45     .format(P10[0], P10[1], 0, 1)
46     mesh_script += "Point(11) = {{{0}}, {1}, {2}, {3}}};\n" \
47     .format(P11[0], P11[1], 0, 1)
48     mesh_script += "Point(12) = {{{0}}, {1}, {2}, {3}}};\n" \
49     .format(P12[0], P12[1], 0, 1)

```

```

50 mesh_script += "Point(13) = {{{0}, {1}, {2}, {3}}};\n".\
51 format(P13[0], P13[1], 0, 1)
52 mesh_script += "Point(14) = {{{0}, {1}, {2}, {3}}};\n".\
53 format(P14[0], P14[1], 0, 1)
54 mesh_script += "Point(15) = {{{0}, {1}, {2}, {3}}};\n".\
55 format(P15[0], P15[1], 0, 1)
56 mesh_script += "Point(16) = {{{0}, {1}, {2}, {3}}};\n".\
57 format(P16[0], P16[1], 0, 1)
58
59 # Bottom boundary
60 mesh_script += "Line(1) = {13, 3};\n"
61 mesh_script += "Line(2) = {4, 14};\n"
62 # Right boundary
63 mesh_script += "Line(3) = {14, 15};\n"
64 # Top boundary
65 mesh_script += "Line(4) = {15, 1};\n"
66 mesh_script += "Line(5) = {2, 16};\n"
67 # Left boundary
68 mesh_script += "Line(6) = {16, 13};\n"
69 # Louver frame
70 mesh_script += "Line(7) = { 3, 12};\n"
71 mesh_script += "Line(8) = {12, 11};\n"
72 mesh_script += "Line(9) = {11, 4};\n"
73 mesh_script += "Line(10) = {7, 9};\n"
74 mesh_script += "Line(11) = {9, 10};\n"
75 mesh_script += "Line(12) = {10, 8};\n"
76 mesh_script += "Line(13) = { 8, 7};\n"
77 mesh_script += "Line(14) = { 1, 5};\n"
78 mesh_script += "Line(15) = { 5, 6};\n"
79 mesh_script += "Line(16) = { 6, 2};\n"
80
81 mesh_script += "Characteristic Length \
82 {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}"
83 mesh_script += " = {0};\n".format(np.round(1/N,3))
84
85 mesh_script += \
86 "Curve Loop(1) = {5,6,1,7,8,9,2,3,4,14,15,16};\n"
87 mesh_script += "Curve Loop(2) = {10, 11, 12, 13};\n"
88 mesh_script += "Plane Surface(1) = {1, 2};\n"
89 mesh_script += "Physical Surface(101) = {1};\n"
90
91 mesh_script += "Physical Curve(201) = {1, 2};\n"
92 mesh_script += "Physical Curve(202) = {3};\n"
93 mesh_script += "Physical Curve(203) = {4, 5};\n"
94 mesh_script += "Physical Curve(204) = {6};\n"
95
96 mesh_script += \
97 "Physical Curve(301) = {7, 8, 9, 10, 11, 12, 13, 14, 15, 16};"
98
99 return mesh_script

```

Algorithm C.16. Mesh file conversion routine

```

1 def mesh_generation(mesh_script):
2
3     print("Running Gmsh...")
4     geo_file=open("mesh.geo",'w')
5     geo_file.write(mesh_script)
6     geo_file.close()
7     gmsh_err=call(["gmsh", "-2", "mesh.geo", "-o", "mesh.msh"])
8     if gmsh_err:
9         print("...something bad happened \
10 when the mesh was generated...")
11     else:
12         print("...mesh generated successfully!")
13
14     msh = meshio.read("mesh.msh")
15
16     line_cells = []
17     for cell in msh.cells:
18         if cell.type == "triangle":
19             triangle_cells = cell.data
20         elif cell.type == "line":
21             if len(line_cells) == 0:
22                 line_cells = cell.data
23             else:
24                 line_cells = np.vstack([line_cells, cell.data])
25
26     line_data = []
27     for key in msh.cell_data_dict["gmsh:physical"].keys():
28         if key == "line":
29             if len(line_data) == 0:
30                 line_data = \
31                     msh.cell_data_dict["gmsh:physical"][key]
32             else:
33                 line_data = \
34                     np.vstack([line_data, \
35                             msh.cell_data_dict["gmsh:physical"][key]])
36         elif key == "triangle":
37             triangle_data = \
38                 msh.cell_data_dict["gmsh:physical"][key]

```

```

39     triangle_mesh = \
40     meshio.Mesh(points=msh.points[:, :2], \
41     cells={"triangle": triangle_cells})
42     line_mesh = \
43     meshio.Mesh(points=msh.points, \
44     cells=[("line", line_cells)], \
45     cell_data={"name_to_read": [line_data]})
46     meshio.write("mesh.xdmf", triangle_mesh)
47
48     meshio.xdmf.write("mf.xdmf", line_mesh)
49
50     mesh = Mesh()
51     with XDMFFile("mesh.xdmf") as infile:
52         infile.read(mesh)
53     mvc = MeshValueCollection("size_t", mesh, 2)
54     with XDMFFile("mf.xdmf") as infile:
55         infile.read(mvc, "name_to_read")
56     mf = cpp.mesh.MeshFunctionSizet(mesh, mvc)
57
58     mesh_data = [mesh, mvc, mf]
59
60     return mesh_data

```

Algorithm C.17. Mesh data extraction routine.

```

1  F = FunctionSpace(mesh, 'P', p)
2
3  n = F.dim()
4  d = mesh.geometry().dim()
5
6  F_dof_coordinates = F.tabulate_dof_coordinates()
7  F_dof_coordinates.resize((n, d))
8  F_dof_x = F_dof_coordinates[:, 0]
9  F_dof_y = F_dof_coordinates[:, 1]

```

C.4. CODES: FINITE ELEMENT MODELS

C.4.1. Cylindrical Domain, Sound Hard, Cylinder Body

Algorithm C.18. Scattering wave solution for a sound-hard cylinder in a cylindrical solution domain, utilizing Sommerfeld ABC

```

1 def FEM_Hard_Sommerfeld(mesh, k, r_in, r_out, p, q):
2
3     # Define a mixed function space
4     Er = FiniteElement('P', triangle, p)
5     Ei = FiniteElement('P', triangle, p)
6     Ec = Er * Ei
7     V = FunctionSpace(mesh, Ec)
8
9     # Initialize mesh function for boundary domains
10    boundaries = \
11    MeshFunction('size_t', mesh, mesh.topology().dim()-1)
12
13    # Define boundaries
14    class Out(SubDomain):
15        def inside(self, x, on_boundary):
16            return on_boundary and \
17            pow(x[0],2)+pow(x[1],2)>=pow(r_out,2)-.01
18
19    class In(SubDomain):
20        def inside(self, x, on_boundary):
21            return on_boundary and \
22            pow(x[0],2)+pow(x[1],2)<=pow(r_in,2)+.01
23
24    # Initialize sub-domain instances
25    out = Out()
26    inc = In()
27
28    # Mark the boundaries
29    boundaries.set_all(0)
30    out.mark(boundaries, 1)
31    inc.mark(boundaries, 2)
32
33    # Define measures associated with
34    # the interior domains and exterior boundaries
35    ds = Measure('ds', domain=mesh, subdomain_data=boundaries)
36
37    # Define basis functions
38    (u_r, u_i) = TrialFunction(V)
39    (v_r, v_i) = TestFunction(V)

```

```

40 # Define the weak form
41 C_1 = \
42 Expression('k*cos(atan2(x[1],x[0]))*cos(k*x[0])',degree=q, k=k)
43 C_2 = \
44 Expression('k*cos(atan2(x[1],x[0]))*sin(k*x[0])',degree=q, k=k)
45
46 a_r = \
47 + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
48 - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
49 - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
50 + k*(inner(u_r,v_i) + inner(u_i,v_r))*ds(1)
51
52 a_i = \
53 + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
54 + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
55 - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
56 - k*(inner(u_r,v_r) - inner(u_i,v_i))*ds(1)
57
58 a = a_r + a_i
59
60 L_r = -(C_1*v_i + C_2*v_r)*ds(2)
61 L_i = (C_1*v_r - C_2*v_i)*ds(2)
62
63 L = L_r + L_i
64
65 # Solve the FEM
66 u = Function(V)
67 solve(a == L, u )
68
69 return u

```

Algorithm C.19. Scattering wave solution for a sound-hard cylinder in a cylindrical solution domain, utilizing BGT-1 ABC

```

1 def FEM_Hard_BGT1(mesh, k, r_in, r_out, p, q, tol):
2
3     # Define a mixed function space
4     Er = FiniteElement('P', triangle, p)
5     Ei = FiniteElement('P', triangle, p)
6     Ec = Er * Ei
7     V = FunctionSpace(mesh, Ec)
8
9     # Initialize mesh function for boundary domains
10    boundaries = \
11    MeshFunction('size_t', mesh, mesh.topology().dim()-1)
12
13    # Define boundaries
14    class Out(SubDomain):
15        def inside(self, x, on_boundary):
16            return on_boundary and \
17            pow(x[0],2)+pow(x[1],2)>=pow(r_out,2)-tol
18    class In(SubDomain):
19        def inside(self, x, on_boundary):
20            return on_boundary and \
21            pow(x[0],2)+pow(x[1],2)<=pow(r_in,2)+tol
22
23    # Initialize sub-domain instances
24    out = Out()
25    inc = In()
26
27    # Mark the boundaries
28    boundaries.set_all(0)
29    out.mark(boundaries, 1)
30    inc.mark(boundaries, 2)
31
32    # Define new measures associated with
33    # the interior domains and exterior boundaries
34    ds = Measure('ds', domain=mesh, subdomain_data=boundaries)
35
36    # Define basis functions
37    (u_r, u_i) = TrialFunction(V)
38    (v_r, v_i) = TestFunction(V)

```

```

39 # Define the weak form
40 C_1 = \
41 Expression('k*cos(atan2(x[1],x[0]))*cos(k*x[0])',degree=q, k=k)
42 C_2 = \
43 Expression('k*cos(atan2(x[1],x[0]))*sin(k*x[0])',degree=q, k=k)
44
45 a_r = \
46 + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
47 - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
48 - (k**2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
49 + k*(inner(u_r,v_i) + inner(u_i,v_r))*ds(1)\
50 + 1/(2*r_out)*(inner(u_r,v_r)-inner(u_i,v_i))*ds(1)
51
52 a_i = \
53 + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
54 + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
55 - (k**2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
56 - k*(inner(u_r,v_r) - inner(u_i,v_i))*ds(1)\
57 + 1/(2*r_out)*(inner(u_r,v_i)+inner(u_i,v_r))*ds(1)
58
59 a = a_r + a_i
60
61 L_r = -(C_1*v_i + C_2*v_r)*ds(2)
62 L_i = (C_1*v_r - C_2*v_i)*ds(2)
63
64 L = L_r + L_i
65
66 # Solve the FEM
67 u = Function(V)
68 solve(a == L, u )
69
70 return u

```

Algorithm C.20. Scattering wave solution for a sound-hard cylinder in a cylindrical solution domain, utilizing BGT-2 ABC

```

1 def FEM_Hard_BGT2(mesh, k, r_in, r_out, p, q, tol):
2
3     # Define a mixed function space
4     Er = FiniteElement('P', triangle, p)
5     Ei = FiniteElement('P', triangle, p)
6     Ec = Er * Ei
7     V = FunctionSpace(mesh, Ec)
8
9     # Initialize mesh function for boundary domains
10    boundaries = \
11    MeshFunction('size_t', mesh, mesh.topology().dim()-1)
12
13    class In(SubDomain):
14        def inside(self, x, on_boundary):
15            return on_boundary and \
16            pow(x[0], 2) + pow(x[1], 2) <= pow(r_in, 2) + tol
17
18    class Out(SubDomain):
19        def inside(self, x, on_boundary):
20            return on_boundary and \
21            pow(x[0], 2) + pow(x[1], 2) >= pow(r_out, 2) - tol
22
23    # Initialize sub-domain instances
24    out = Out()
25    inc = In()
26
27    # Mark the boundaries
28    boundaries.set_all(0)
29    out.mark(boundaries, 1)
30    inc.mark(boundaries, 2)
31
32    # Define new measures associated with
33    # the interior domains and exterior boundaries
34    ds = Measure('ds', domain=mesh, subdomain_data=boundaries)
35
36    # Define basis functions
37    (u_r, u_i) = TrialFunction(V)
38    (v_r, v_i) = TestFunction(V)

```

```

39 # Define the Weak Form
40 C_1 = \
41 Expression('k*cos(atan2(x[1],x[0]))*cos(k*x[0])',degree=q, k=k)
42 C_2 = \
43 Expression('k*cos(atan2(x[1],x[0]))*sin(k*x[0])',degree=q, k=k)
44
45 T_1 = \
46 Expression('pow(r_out,2)*pow(sin(atan2(x[1],x[0])),2)'\
47 ,degree=q, r_out=r_out)
48 T_2 = \
49 Expression('-pow(r_out,2)*cos(atan2(x[1],x[0]))'\
50 '*sin(atan2(x[1],x[0]))', degree=q, r_out=r_out)
51 T_3 = \
52 Expression('-r_out*cos(atan2(x[1],x[0]))',\
53 degree=q, r_out=r_out)
54 T_4 = \
55 Expression('-pow(r_out,2)*cos(atan2(x[1],x[0]))'\
56 '*sin(atan2(x[1],x[0]))', degree=q, r_out=r_out)
57 T_5 = \
58 Expression('pow(r_out,2)*pow(cos(atan2(x[1],x[0])),2)'\
59 , degree=q, r_out=r_out)
60 T_6 = \
61 Expression('-r_out*sin(atan2(x[1],x[0]))'\
62 , degree=q, r_out=r_out)
63
64 EX11 = \
65 Expression('1/(8*pow(r_out,3)*pow(k,2)+8*r_out)'\
66 , degree=q, r_out=r_out, k=k)
67 EX12 = Expression('k/(8*pow(r_out,2)*pow(k,2)+8)'\
68 , degree=q, r_out=r_out, k=k)
69
70 EX21 = \
71 Expression('2*r_out/(4*pow(r_out,4)*pow(k,2)+4*pow(r_out,2))'\
72 , degree=q, r_out=r_out, k=k)
73 EX22 = \
74 Expression('2*pow(r_out,2)*k/(4*pow(r_out,4)'\
75 '*pow(k,2)+4*pow(r_out,2))', degree=q, r_out=r_out, k=k)

```

```

76 # Real part of the sesquilinear form
77 a_r = \
78 + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
79 - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
80 - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
81 + k*(inner(u_r,v_i) + inner(u_i,v_r))*ds(1)\
82 + 1/(2*r_out)*(inner(u_r,v_r)-inner(u_i,v_i))*ds(1)\
83 - inner(u_r,v_r)*EX11*ds(1)\
84 + inner(u_i,v_i)*EX11*ds(1)\
85 + inner(u_i,v_r)*EX12*ds(1)\
86 + inner(u_r,v_i)*EX12*ds(1)\
87 -T_1*inner(u_r.dx(0).dx(0),v_r)*EX21*ds(1)\
88 +T_1*inner(u_i.dx(0).dx(0),v_i)*EX21*ds(1)\
89 +T_1*inner(u_r.dx(0).dx(0),v_i)*EX22*ds(1)\
90 +T_1*inner(u_i.dx(0).dx(0),v_r)*EX22*ds(1)\
91 -T_2*inner(u_r.dx(0).dx(1),v_r)*EX21*ds(1)\
92 +T_2*inner(u_i.dx(0).dx(1),v_i)*EX21*ds(1)\
93 +T_2*inner(u_r.dx(0).dx(1),v_i)*EX22*ds(1)\
94 +T_2*inner(u_i.dx(0).dx(1),v_r)*EX22*ds(1)\
95 -T_3*inner(u_r.dx(0),v_r)*EX21*ds(1)\
96 +T_3*inner(u_i.dx(0),v_i)*EX21*ds(1)\
97 +T_3*inner(u_r.dx(0),v_i)*EX22*ds(1)\
98 +T_3*inner(u_i.dx(0),v_r)*EX22*ds(1)\
99 -T_4*inner(u_r.dx(1).dx(0),v_r)*EX21*ds(1)\
100 +T_4*inner(u_i.dx(1).dx(0),v_i)*EX21*ds(1)\
101 +T_4*inner(u_r.dx(1).dx(0),v_i)*EX22*ds(1)\
102 +T_4*inner(u_i.dx(1).dx(0),v_r)*EX22*ds(1)\
103 -T_5*inner(u_r.dx(1).dx(1),v_r)*EX21*ds(1)\
104 +T_5*inner(u_i.dx(1).dx(1),v_i)*EX21*ds(1)\
105 +T_5*inner(u_r.dx(1).dx(1),v_i)*EX22*ds(1)\
106 +T_5*inner(u_i.dx(1).dx(1),v_r)*EX22*ds(1)\
107 -T_6*inner(u_r.dx(1),v_r)*EX21*ds(1)\
108 +T_6*inner(u_i.dx(1),v_i)*EX21*ds(1)\
109 +T_6*inner(u_r.dx(1),v_i)*EX22*ds(1)\
110 +T_6*inner(u_i.dx(1),v_r)*EX22*ds(1)

```

```

111 # Imaginary part of the sesquilinear form
112 a_i = \
113 + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
114 + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
115 - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
116 - k*(inner(u_r,v_r) - inner(u_i,v_i))*ds(1)\
117 + 1/(2*r_out)*(inner(u_r,v_i)+inner(u_i,v_r))*ds(1)\
118 - inner(u_r,v_r)*EX12*ds(1)\
119 + inner(u_i,v_i)*EX12*ds(1)\
120 - inner(u_i,v_r)*EX11*ds(1)\
121 - inner(u_r,v_i)*EX11*ds(1)\
122 -T_1*inner(u_r.dx(0).dx(0),v_r)*EX22*ds(1)\
123 +T_1*inner(u_i.dx(0).dx(0),v_i)*EX22*ds(1)\
124 -T_1*inner(u_r.dx(0).dx(0),v_i)*EX21*ds(1)\
125 -T_1*inner(u_i.dx(0).dx(0),v_r)*EX21*ds(1)\
126 -T_2*inner(u_r.dx(0).dx(1),v_r)*EX22*ds(1)\
127 +T_2*inner(u_i.dx(0).dx(1),v_i)*EX22*ds(1)\
128 -T_2*inner(u_r.dx(0).dx(1),v_i)*EX21*ds(1)\
129 -T_2*inner(u_i.dx(0).dx(1),v_r)*EX21*ds(1)\
130 -T_3*inner(u_r.dx(0),v_r)*EX22*ds(1)\
131 +T_3*inner(u_i.dx(0),v_i)*EX22*ds(1)\
132 -T_3*inner(u_r.dx(0),v_i)*EX21*ds(1)\
133 -T_3*inner(u_i.dx(0),v_r)*EX21*ds(1)\
134 -T_4*inner(u_r.dx(1).dx(0),v_r)*EX22*ds(1)\
135 +T_4*inner(u_i.dx(1).dx(0),v_i)*EX22*ds(1)\
136 -T_4*inner(u_r.dx(1).dx(0),v_i)*EX21*ds(1)\
137 -T_4*inner(u_i.dx(1).dx(0),v_r)*EX21*ds(1)\
138 -T_5*inner(u_r.dx(1).dx(1),v_r)*EX22*ds(1)\
139 +T_5*inner(u_i.dx(1).dx(1),v_i)*EX22*ds(1)\
140 -T_5*inner(u_r.dx(1).dx(1),v_i)*EX21*ds(1)\
141 -T_5*inner(u_i.dx(1).dx(1),v_r)*EX21*ds(1)\
142 -T_6*inner(u_r.dx(1),v_r)*EX22*ds(1)\
143 +T_6*inner(u_i.dx(1),v_i)*EX22*ds(1)\
144 -T_6*inner(u_r.dx(1),v_i)*EX21*ds(1)\
145 -T_6*inner(u_i.dx(1),v_r)*EX21*ds(1)
146
147 # Real part of the antilinear form
148 L_r = -(C_1*v_i + C_2*v_r)*ds(2)
149 # Imaginary part of the antilinear form
150 L_i = (C_1*v_r - C_2*v_i)*ds(2)
151
152 a = a_r + a_i
153 L = L_r + L_i
154
155 # Solve the FEM
156 u = Function(V)
157 solve(a == L, u )
158
159 return u

```

C.4.2. Cylindrical Domain, Sound Soft, Cylinder Body

Algorithm C.21. Scattering wave solution for a sound-soft cylinder in a cylindrical solution domain utilizing Sommerfeld ABC

```

1  def FEM_Soft_Sommerfeld(mesh, k, r_in, r_out, p, q, tol):
2
3      # Define a mixed function space
4      Er = FiniteElement('P', triangle, p)
5      Ei = FiniteElement('P', triangle, p)
6      Ec = Er * Ei
7      V = FunctionSpace(mesh, Ec)
8
9      # Initialize mesh function to define boundaries
10     boundaries = \
11     MeshFunction('size_t', mesh, mesh.topology().dim()-1)
12
13     class Out(SubDomain):
14         def inside(self, x, on_boundary):
15             return on_boundary and \
16                 pow(x[0],2)+pow(x[1],2)>=pow(r_out,2) - tol
17
18     class In(SubDomain):
19         def inside(self, x, on_boundary):
20             return on_boundary and \
21                 pow(x[0],2)+pow(x[1],2)<=pow(r_in,2) + tol
22
23     # Initialize sub-domain instances
24     out = Out()
25     inc = In()
26
27     # Define Dirichlet boundary conditions
28     u_D_re = Expression('-cos(k*x[0])', degree=q, k=k)
29     u_D_im = Expression('-sin(k*x[0])', degree=q, k=k)
30
31     bc = [DirichletBC(V.sub(0), u_D_re, inc),
32           DirichletBC(V.sub(1), u_D_im, inc)]
33
34     # Mark the boundaries
35     boundaries.set_all(0)
36     out.mark(boundaries, 1)
37     inc.mark(boundaries, 2)
38
39     # Define new measures associated with
40     # the interior domains and exterior boundaries
41     ds = Measure('ds', domain=mesh, subdomain_data=boundaries)
42     n = FacetNormal(mesh)
43
44     # Define basis functions
45     (u_r, u_i) = TrialFunction(V)
46     (v_r, v_i) = TestFunction(V)

```

```

47 # Define the weak form
48 a_r = \
49 + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
50 - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
51 - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
52 + k*(inner(u_r,v_i) + inner(u_i,v_r))*ds(1)\
53 + dot(n, v_r*nabla_grad(u_r))*ds(2)\
54 - dot(n, v_i*nabla_grad(u_i))*ds(2)
55
56 a_i = \
57 + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
58 + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
59 - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
60 - k*(inner(u_r,v_r) - inner(u_i,v_i))*ds(1)\
61 + dot(n, v_r*nabla_grad(u_i))*ds(2)\
62 + dot(n, v_i*nabla_grad(u_r))*ds(2)
63
64 a = a_r + a_i
65
66 L=inner(Constant(0), v_r)*dx(0)
67
68 u = Function(V)
69 solve(a == L, u, bc)
70
71 return u

```

C.4.3. Cylindrical Domain, Sound Hard, Arbitrary-Shapes

Algorithm C.22. Scattering wave solution for a sound-hard arbitrarily-shaped body in a cylindrical solution domain utilizing BGT-2 ABC

```

1 def FEM_Hard_BGT2(mesh, mf, k, p, q, r_out):
2
3     x = SpatialCoordinate(mesh)
4     n = FacetNormal(mesh)
5
6     ds_1 = \
7     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=201)
8     ds_2 = \
9     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=301)
10
11     V_1 = FiniteElement("CG", mesh.ufl_cell(), p)
12     V_2 = FiniteElement("CG", mesh.ufl_cell(), p)
13     V = FunctionSpace(mesh, MixedElement([V_1, V_2]))
14
15     (u_r, u_i) = TrialFunction(V)
16     (v_r, v_i) = TestFunction(V)
17
18     C_1 = \
19     Expression('k*cos(atan2(x[1],x[0]))*cos(k*x[0])', degree=q, k=k)
20     C_2 = \
21     Expression('k*cos(atan2(x[1],x[0]))*sin(k*x[0])', degree=q, k=k)
22
23     T_1 = \
24     Expression('pow(r_out,2)*pow(sin(atan2(x[1],x[0])),2)'\
25     , degree=q, r_out=r_out)
26     T_2 = \
27     Expression('-pow(r_out,2)*cos(atan2(x[1],x[0]))'\
28     '*sin(atan2(x[1],x[0]))', degree=q, r_out=r_out)
29     T_3 = \
30     Expression('-r_out*cos(atan2(x[1],x[0]))'\
31     , degree=q, r_out=r_out)
32     T_4 = \
33     Expression('-pow(r_out,2)*cos(atan2(x[1],x[0]))'\
34     '*sin(atan2(x[1],x[0]))', degree=q, r_out=r_out)
35     T_5 = \
36     Expression(' pow(r_out,2)*pow(cos(atan2(x[1],x[0])),2)'\
37     , degree=q, r_out=r_out)
38     T_6 = \
39     Expression('-r_out*sin(atan2(x[1],x[0]))'\
40     , degree=q, r_out=r_out)

```

```

41 EX11 = \
42 Expression('1/(8*pow(r_out,3)*pow(k,2)+8*r_out)'\
43 , degree=q, r_out=r_out, k=k)
44 EX12 = \
45 Expression('k/(8*pow(r_out,2)*pow(k,2)+8)'\
46 , degree=q, r_out=r_out, k=k)
47 EX21 = \
48 Expression('2*r_out/(4*pow(r_out,4)*pow(k,2)+4*pow(r_out,2))'\
49 , degree=q, r_out=r_out, k=k)
50 EX22 = \
51 Expression('2*pow(r_out,2)*k/(4*pow(r_out,4)*pow(k,2)'\
52 '+4*pow(r_out,2))', degree=q, r_out=r_out, k=k)
53
54 a_r = \
55 + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
56 - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
57 - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
58 + k*(inner(u_r,v_i) + inner(u_i,v_r))*ds_1\
59 + 1/(2*r_out)*(inner(u_r,v_r)-inner(u_i,v_i))*ds_1\
60 - inner(u_r,v_r)*EX11*ds_1\
61 + inner(u_i,v_i)*EX11*ds_1\
62 + inner(u_i,v_r)*EX12*ds_1\
63 + inner(u_r,v_i)*EX12*ds_1\
64 -T_1*inner(u_r.dx(0).dx(0),v_r)*EX21*ds_1\
65 +T_1*inner(u_i.dx(0).dx(0),v_i)*EX21*ds_1\
66 +T_1*inner(u_r.dx(0).dx(0),v_i)*EX22*ds_1\
67 +T_1*inner(u_i.dx(0).dx(0),v_r)*EX22*ds_1\
68 -T_2*inner(u_r.dx(0).dx(1),v_r)*EX21*ds_1\
69 +T_2*inner(u_i.dx(0).dx(1),v_i)*EX21*ds_1\
70 +T_2*inner(u_r.dx(0).dx(1),v_i)*EX22*ds_1\
71 +T_2*inner(u_i.dx(0).dx(1),v_r)*EX22*ds_1\
72 -T_3*inner(u_r.dx(0),v_r)*EX21*ds_1\
73 +T_3*inner(u_i.dx(0),v_i)*EX21*ds_1\
74 +T_3*inner(u_r.dx(0),v_i)*EX22*ds_1\
75 +T_3*inner(u_i.dx(0),v_r)*EX22*ds_1\
76 -T_4*inner(u_r.dx(1).dx(0),v_r)*EX21*ds_1\
77 +T_4*inner(u_i.dx(1).dx(0),v_i)*EX21*ds_1\
78 +T_4*inner(u_r.dx(1).dx(0),v_i)*EX22*ds_1\
79 +T_4*inner(u_i.dx(1).dx(0),v_r)*EX22*ds_1\
80 -T_5*inner(u_r.dx(1).dx(1),v_r)*EX21*ds_1\
81 +T_5*inner(u_i.dx(1).dx(1),v_i)*EX21*ds_1\
82 +T_5*inner(u_r.dx(1).dx(1),v_i)*EX22*ds_1\
83 +T_5*inner(u_i.dx(1).dx(1),v_r)*EX22*ds_1\
84 -T_6*inner(u_r.dx(1),v_r)*EX21*ds_1\
85 +T_6*inner(u_i.dx(1),v_i)*EX21*ds_1\
86 +T_6*inner(u_r.dx(1),v_i)*EX22*ds_1\
87 +T_6*inner(u_i.dx(1),v_r)*EX22*ds_1

```

```

88     a_i = \
89     + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
90     + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
91     - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
92     - k*(inner(u_r,v_r) - inner(u_i,v_i))*ds_1\
93     + 1/(2*r_out)*(inner(u_r,v_i)+inner(u_i,v_r))*ds_1\
94     - inner(u_r,v_r)*EX12*ds_1\
95     + inner(u_i,v_i)*EX12*ds_1\
96     - inner(u_i,v_r)*EX11*ds_1\
97     - inner(u_r,v_i)*EX11*ds_1\
98     -T_1*inner(u_r.dx(0).dx(0),v_r)*EX22*ds_1\
99     +T_1*inner(u_i.dx(0).dx(0),v_i)*EX22*ds_1\
100    -T_1*inner(u_r.dx(0).dx(0),v_i)*EX21*ds_1\
101    -T_1*inner(u_i.dx(0).dx(0),v_r)*EX21*ds_1\
102    -T_2*inner(u_r.dx(0).dx(1),v_r)*EX22*ds_1\
103    +T_2*inner(u_i.dx(0).dx(1),v_i)*EX22*ds_1\
104    -T_2*inner(u_r.dx(0).dx(1),v_i)*EX21*ds_1\
105    -T_2*inner(u_i.dx(0).dx(1),v_r)*EX21*ds_1\
106    -T_3*inner(u_r.dx(0),v_r)*EX22*ds_1\
107    +T_3*inner(u_i.dx(0),v_i)*EX22*ds_1\
108    -T_3*inner(u_r.dx(0),v_i)*EX21*ds_1\
109    -T_3*inner(u_i.dx(0),v_r)*EX21*ds_1\
110    -T_4*inner(u_r.dx(1).dx(0),v_r)*EX22*ds_1\
111    +T_4*inner(u_i.dx(1).dx(0),v_i)*EX22*ds_1\
112    -T_4*inner(u_r.dx(1).dx(0),v_i)*EX21*ds_1\
113    -T_4*inner(u_i.dx(1).dx(0),v_r)*EX21*ds_1\
114    -T_5*inner(u_r.dx(1).dx(1),v_r)*EX22*ds_1\
115    +T_5*inner(u_i.dx(1).dx(1),v_i)*EX22*ds_1\
116    -T_5*inner(u_r.dx(1).dx(1),v_i)*EX21*ds_1\
117    -T_5*inner(u_i.dx(1).dx(1),v_r)*EX21*ds_1\
118    -T_6*inner(u_r.dx(1),v_r)*EX22*ds_1\
119    +T_6*inner(u_i.dx(1),v_i)*EX22*ds_1\
120    -T_6*inner(u_r.dx(1),v_i)*EX21*ds_1\
121    -T_6*inner(u_i.dx(1),v_r)*EX21*ds_1
122
123    L_r = \
124    - dot( n, v_r*nabla_grad(cos(k*x[0])))*ds_2\
125    + dot( n, v_i*nabla_grad(sin(k*x[0])))*ds_2
126
127    L_i = \
128    - dot( n, v_i*nabla_grad(cos(k*x[0])))*ds_2\
129    - dot( n, v_r*nabla_grad(sin(k*x[0])))*ds_2
130
131    a = a_r + a_i; L = L_r + L_i
132    u = Function(V); solve(a == L, u)
133
134    return u

```

Algorithm C.23. Total wave solution for a sound-hard arbitrarily-shaped body in a cylindrical solution domain utilizing Sommerfeld ABC

```

1 def FEM_Hard_Sommerfeld(mesh, mf, p, k):
2
3     x = SpatialCoordinate(mesh); n = FacetNormal(mesh)
4
5     ds_1 = \
6     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=201)
7     ds_2 = \
8     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=301)
9
10
11     V_1 = FiniteElement("CG", mesh.ufl_cell(), p)
12     V_2 = FiniteElement("CG", mesh.ufl_cell(), p)
13     V = FunctionSpace(mesh, MixedElement([V_1, V_2]))
14
15     (u_r, u_i) = TrialFunction(V); (v_r, v_i) = TestFunction(V)
16
17     a_r = \
18     + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
19     + inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
20     - pow(k,2)*( inner(u_r,v_r) + inner(u_i,v_i) )*dx\
21     + k*( inner(u_i,v_r) - inner(u_r,v_i) )*ds_1
22
23     a_i = \
24     + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
25     - inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
26     - pow(k,2)*( inner(u_i,v_r) - inner(u_r,v_i) )*dx\
27     - k*( inner(u_r,v_r) + inner(u_i,v_i) )*ds_1
28
29     L_r = \
30     + dot(n, v_r*nabla_grad(cos(k*x[0])))*ds_1\
31     + dot(n, v_i*nabla_grad(sin(k*x[0])))*ds_1\
32     - k*(cos(k*x[0])*v_i - sin(k*x[0])*v_r)*ds_1
33
34     L_i = \
35     + dot(n, v_r*nabla_grad(sin(k*x[0])))*ds_1\
36     - dot(n, v_i*nabla_grad(cos(k*x[0])))*ds_1\
37     - k*(cos(k*x[0])*v_r + sin(k*x[0])*v_i)*ds_1
38
39     a = a_r + a_i; L = L_r + L_i
40
41     u = Function(V)
42     solve(a == L, u)
43
44     return u

```

C.4.4. Rectangular Domain, Sound-Hard, Arbitrary-Shapes, Single Body

Algorithm C.24. Scattering wave solution for a sound-hard arbitrarily-shaped body in a rectangular solution domain utilizing Sommerfeld ABC

```

1 def FEM_Hard_Robin(mesh, mf, k, p):
2
3     # Find the boundary normals
4     x = SpatialCoordinate(mesh);
5     n = FacetNormal(mesh)
6
7     # Define measures and identify boundaries
8     ds_do_1 = \
9     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=201)
10    ds_do_2 = \
11    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=202)
12    ds_do_3 = \
13    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=203)
14    ds_do_4 = \
15    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=204)
16    ds_in_1 = \
17    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=301)
18
19    # Define a mixed function space
20    V_1 = FiniteElement("CG", mesh.ufl_cell(), p)
21    V_2 = FiniteElement("CG", mesh.ufl_cell(), p)
22    V = FunctionSpace(mesh, MixedElement([V_1, V_2]))
23
24    # Define basis functions
25    (u_r, u_i) = TrialFunction(V)
26    (v_r, v_i) = TestFunction(V)
27
28    # Define the weak form
29    a_r = \
30    + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
31    - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
32    - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
33    + k*(inner(u_r,v_i) + inner(u_i,v_r))\
34    * (ds_do_1 + ds_do_2 + ds_do_3 + ds_do_4)
35
36    a_i = \
37    + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
38    + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
39    - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
40    - k*(inner(u_r,v_r) - inner(u_i,v_i))\
41    * (ds_do_1 + ds_do_2 + ds_do_3 + ds_do_4)
42
43    a = a_r + a_i

```

```
44     L_r = \  
45     - dot( n, v_r*nabla_grad(cos(k*x[0])))*ds_in_1\  
46     + dot( n, v_i*nabla_grad(sin(k*x[0])))*ds_in_1\  
47     L_i = \  
48     - dot( n, v_i*nabla_grad(cos(k*x[0])))*ds_in_1\  
49     - dot( n, v_r*nabla_grad(sin(k*x[0])))*ds_in_1\  
50     L = L_r + L_i\  
51  
52     u_s = Function(V)  
53     solve(a == L, u_s )  
54  
55     return u_s
```

C.4.5. Rectangular Domain, Sound-Hard, Arbitrary-Shapes, Symmetric Bodies

Algorithm C.25. Scattering wave solution for a sound-hard symmetric body in a rectangular solution domain utilizing homogeneous Neumann boundary conditions on the horizontal, Robin boundary conditions on the vertical outer domain boundaries

```

1 def FEM_Hard_HNBC(mesh, mf, k, p):
2
3     x = SpatialCoordinate(mesh); n = FacetNormal(mesh)
4
5     ds_do_1 = \
6     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=202)
7     ds_do_2 = \
8     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=204)
9     ds_di_1 = \
10    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=301)
11
12    V_1 = FiniteElement("CG", mesh.ufl_cell(), p)
13    V_2 = FiniteElement("CG", mesh.ufl_cell(), p)
14    V = FunctionSpace(mesh, MixedElement([V_1, V_2]))
15
16    (u_r, u_i) = TrialFunction(V)
17    (v_r, v_i) = TestFunction(V)
18
19    a_r = \
20    + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
21    - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
22    - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
23    + k*(inner(u_r,v_i) + inner(u_i,v_r))*(ds_do_1 + ds_do_2)
24    a_i = \
25    + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
26    + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
27    - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
28    - k*(inner(u_r,v_r) - inner(u_i,v_i))*(ds_do_1 + ds_do_2)
29
30    L_r = \
31    - dot(n, v_r*nabla_grad(cos(k*x[0])))*ds_di_1\
32    + dot(n, v_i*nabla_grad(sin(k*x[0])))*ds_di_1
33    L_i = \
34    - dot(n, v_i*nabla_grad(cos(k*x[0])))*ds_di_1\
35    - dot(n, v_r*nabla_grad(sin(k*x[0])))*ds_di_1
36
37    a = a_r + a_i; L = L_r + L_i
38
39    u = Function(V); solve(a == L, u)
40
41    return u

```

Algorithm C.26. Scattering wave solution for a sound-hard symmetric body in a rectangular solution domain utilizing homogeneous Neumann boundary condition on the top, Robin boundary conditions on the left, bottom and right outer domain boundaries

```

1 def FEM_Hard_HNBC(mesh, mf, k, p):
2
3     x = SpatialCoordinate(mesh); n = FacetNormal(mesh)
4
5     ds_do_1 = \
6     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=201)
7     ds_do_2 = \
8     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=202)
9     ds_do_3 = \
10    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=204)
11    ds_di_1 = \
12    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=301)
13
14    V_1 = FiniteElement("CG", mesh.ufl_cell(), p)
15    V_2 = FiniteElement("CG", mesh.ufl_cell(), p)
16    V = FunctionSpace(mesh, MixedElement([V_1, V_2]))
17
18    (u_r, u_i) = TrialFunction(V); (v_r, v_i) = TestFunction(V)
19
20    a_r = \
21    + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
22    - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
23    - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
24    + k*(inner(u_r,v_i) + inner(u_i,v_r))\
25    * (ds_do_1 + ds_do_2 + ds_do_3)
26
27    a_i = \
28    + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
29    + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
30    - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
31    - k*(inner(u_r,v_r) - inner(u_i,v_i))\
32    * (ds_do_1 + ds_do_2 + ds_do_3)
33
34    L_r = \
35    - dot(n, v_r*nabla_grad(cos(k*x[0])))*ds_di_1\
36    + dot(n, v_i*nabla_grad(sin(k*x[0])))*ds_di_1
37
38    L_i = \
39    - dot(n, v_i*nabla_grad(cos(k*x[0])))*ds_di_1\
40    - dot(n, v_r*nabla_grad(sin(k*x[0])))*ds_di_1
41
42    a = a_r + a_i; L = L_r + L_i
43
44    u = Function(V); solve(a == L, u)
45
46    return u

```

C.4.6. Rectangular Domain, Sound-Hard, Arbitrary-Shapes, Periodic Bodies

Algorithm C.27. Scattering wave solution for sound-hard vertically periodic bodies in a rectangular solution domain utilizing periodic boundary condition on the top and bottom, Robin boundary condition on the left and right outer domain boundaries

```

1 def FEM_Hard_PBC(mesh, mf, ymax, tol, p, k):
2
3     x = SpatialCoordinate(mesh)
4     n = FacetNormal(mesh)
5
6     ds_do_1 = \
7     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=202)
8     ds_do_2 = \
9     Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=204)
10    ds_di_1 = \
11    Measure("ds", domain=mesh, subdomain_data=mf, subdomain_id=301)
12
13    class PeriodicBoundary(SubDomain):
14        def inside(self, x, on_boundary):
15            return bool(-ymax + tol > x[1] > -ymax - tol and\
16                        on_boundary)
17        def map(self, x, y):
18            y[0] = x[0]
19            y[1] = x[1] - 2*ymax
20
21    V_1 = FiniteElement("CG", mesh.ufl_cell(), p)
22    V_2 = FiniteElement("CG", mesh.ufl_cell(), p)
23    V = FunctionSpace(mesh, MixedElement([V_1, V_2]),\
24                      constrained_domain=PeriodicBoundary())
25
26    (u_r, u_i) = TrialFunction(V)
27    (v_r, v_i) = TestFunction(V)
28
29    a_r = \
30    + inner(nabla_grad(u_r), nabla_grad(v_r))*dx\
31    - inner(nabla_grad(u_i), nabla_grad(v_i))*dx\
32    - pow(k,2)*( inner(u_r,v_r) - inner(u_i,v_i))*dx\
33    + k*(inner(u_r,v_i) + inner(u_i,v_r))*(ds_do_1 + ds_do_2)
34
35    a_i = \
36    + inner(nabla_grad(u_r), nabla_grad(v_i))*dx\
37    + inner(nabla_grad(u_i), nabla_grad(v_r))*dx\
38    - pow(k,2)*( inner(u_r,v_i) + inner(u_i,v_r))*dx\
39    - k*(inner(u_r,v_r) - inner(u_i,v_i))*(ds_do_1 + ds_do_2)

```

```
40     L_r = \  
41     - dot(n, v_r*nabla_grad(cos(k*x[0])))*ds_di_1\  
42     + dot(n, v_i*nabla_grad(sin(k*x[0])))*ds_di_1\  
43  
44     L_i = \  
45     - dot(n, v_i*nabla_grad(cos(k*x[0])))*ds_di_1\  
46     - dot(n, v_r*nabla_grad(sin(k*x[0])))*ds_di_1\  
47  
48     a = a_r + a_i; L = L_r + L_i\  
49     u = Function(V); solve(a == L, u)  
50  
51     return u
```

C.5. CODES: NELDER-MEAD OPTIMIZATION

Algorithm C.28. Objective function for the Nelder-Mead optimization

```

1 def Obj_func(x):
2     r_2, x_offset = x
3     geo_type = constraints(r_1, r_2, x_offset)
4     if geo_type == 1:
5         u_m_R = 10
6     elif geo_type == 2:
7         u_m_R = 10
8     elif geo_type == 3:
9         u_m_R = 10
10    else:
11        r_out = r_1 + 2*np.pi/k
12        xcyl_2 = xcyl_1 - x_offset
13        Mesh_Data = \
14        mesh_generation(k, n_lambda, xcyl_1, ycyl_1, r_1, \
15        xcyl_2, ycyl_2, r_2, r_out)
16        mesh = Mesh_Data[0]
17        mf = Mesh_Data[2]
18        u_s = FEM_Hard_BGT2(mesh, mf, k, r_out, p, q)
19        (u_s_r, u_s_i) = u_s.split(True)
20        F = FunctionSpace(mesh, 'P', p)
21        n = F.dim()
22        d = mesh.geometry().dim()
23        F_dof_coordinates = F.tabulate_dof_coordinates()
24        F_dof_coordinates.resize((n, d))
25        F_dof_x = F_dof_coordinates[:, 0]
26        F_dof_y = F_dof_coordinates[:, 1]
27        u_p_temp = Plane_Wave(k, F_dof_x)
28        u_p_r_temp = u_p_temp.real
29        u_p_i_temp = u_p_temp.imag
30        u_p_r = Function(F)
31        u_p_i = Function(F)
32        u_p_r.vector().set_local(u_p_r_temp)
33        u_p_i.vector().set_local(u_p_i_temp)
34        u_p_m = sqrt(pow(u_p_r, 2) + pow(u_p_i, 2))
35        u_s_m = sqrt(pow(u_s_r, 2) + pow(u_s_i, 2))
36        u_m = sqrt(pow(u_p_r + u_s_r, 2) + pow(u_p_i + u_s_i, 2))
37        mf_I = \
38        MeshFunction('size_t', mesh, mesh.topology().dim())
39        region = AutoSubDomain(lambda x: \
40        pow(x[0], 2) + pow(x[1], 2) >= pow(r_1, 2) - tol \
41        and pow(x[0], 2) + pow(x[1], 2) <= pow(r_out, 2) + tol \
42        and x[0] >= 0)
43        region.mark(mf_I, 1)
44        dx_sub = Measure('dx', subdomain_data=mf_I)
45        u_m_R = assemble(u_m * dx_sub(1))
46    return u_m_R

```

Algorithm C.29. Nelder-Mead optimization.

```

1 Inputs = []
2 Outputs = []
3 for x_1_dummy in range(1, 6, 1): # crop radius
4     x_1 = round(0.2*x_1_dummy, 2)
5     for x_2_dummy in range(1, 5, 1): # crop distance
6         x_2 = round(0.5*x_2_dummy, 2)
7
8         x0 = (x_1, x_2)
9         opt_u_m_R = minimize(Obj_func, x0, method='Nelder-Mead')
10
11         opt_result = opt_u_m_R.x
12         opt_fun = opt_u_m_R.fun
13
14         if opt_fun >= 10:
15             opt_status = False
16         else:
17             opt_status = opt_u_m_R.success
18
19         run_no = run_no+1
20         Inputs.append ([x_1, x_2])
21         Outputs.append([opt_result[0], \
22             opt_result[1], opt_fun, opt_status])
23
24 Input = np.matrix(Inputs)
25 Output = np.matrix(Outputs)

```

C.6. CODES: INSERTION LOSS

Algorithm C.30. Decomposing sound pressure data to sound pressure spectrum levels

```

1  # Define constants
2  df_ref = 1; P_ref = 20*pow(10,-6)
3  SPL_P = int(round(20*np.log10(1/P_ref),0))
4  # 1/1 octave band center frequencies
5  f_11_c = [63, 125, 250, 500, 1000, 2000, 4000, 8000]
6  # 1/1 octave band lower limit frequencies
7  f_11_l = [44, 88, 177, 355, 710, 1420, 2840, 5680]
8  # 1/1 octave band upper limit frequencies
9  f_11_u = [88, 177, 355, 710, 1420, 2840, 5680, 11360]
10 # 1/3 octave band center frequencies
11 f_13_c = [50, 63, 80, 100, 125, 160, 200, 250, 315, \
12           400, 500, 630, 800, 1000, 1250, 1600, 2000, \
13           2500, 3150, 4000, 5000, 6300, 8000, 10000]
14 # 1/3 octave band lower limit frequencies
15 f_13_l = [44.7, 56.2, 70.8, 89.1, 112, 141, 178, 224, 282, \
16           355, 447, 562, 708, 891, 1122, 1413, 1778, \
17           2239, 2818, 3548, 4467, 5623, 7079, 8913]
18 # 1/3 octave band upper limit frequencies
19 f_13_u = [56.2, 70.8, 89.1, 112, 141, 178, 224, 282, \
20           355, 447, 562, 708, 891, 1122, 1413, 1778, \
21           2239, 2818, 3548, 4467, 5623, 7079, 8913, 11220]
22
23 # Decompose to 1/3 octave band frequencies
24 # according to the frequency resolution given
25 f_13 = list(range(np.size(f_13_c)))
26 for j in range (0, np.size(f_13_c), 1):
27     f_13_n=[]
28     for i in range (0, int(11360/del_f), 1):
29         if f_13_l[j] <= f_n[i] < f_13_u[j]:
30             f_13_n.append(f_n[i].tolist())
31     f_13[j] = f_13_n
32
33 # Calculate the sound pressure spectrum levels
34 Pink_Band = np.array([1]*np.size(f_13_c))*SPL_P
35 White_del = (1/3)*10*np.log10(2)
36 White_Filter = np.cumsum(np.array([1]*np.size(f_13_c))*White_del)
37 White_Normal = \
38 np.cumsum(np.array([1]*np.size(f_13_c))*White_del)[13]
39 White_Band = White_Filter - White_Normal
40 SPL_P_Normalized = Pink_Band + White_Band

```

Algorithm C.31. Calculation of 1/3 and 1/1 octave band insertion loss levels

```

1  # 1/3 octave band insertion loss
2  SPL_13 = []
3  IL = []
4  for i in range(0, np.size(f_13_c), 1):
5      # Spectral density function
6      if f_13_min[i] == f_13_max[i]:
7          SDF = np.sum(np.square(u_m_R[f_13_min[i]]))
8      else:
9          SDF = np.sum(np.square(u_m_R[f_13_min[i]:f_13_max[i]]))
10
11     # Bandwidth
12     df_b = f_13_u[i] - f_13_l[i]
13
14     # Sound Pressure Levels in 1/3 Octave Bands
15     SPL_13.append(10*np.log10(SDF*(df_ref/df_b)/pow(P_ref,2)))
16
17 IL_13 = (SPL_P_Normalized - np.array(SPL_13)).clip(min=0)
18
19 # 1/1 octave band insertion loss
20 IL_13_split = np.array_split(IL_13, np.size(f_11_c))
21 IL_11 = []
22 for i in range(0, np.size(f_11_c), 1):
23     IL_11.append(10*np.log10((1/3)*(\
24         pow(10, IL_13_split[i][0]/10) + \
25         pow(10, IL_13_split[i][1]/10) + \
26         pow(10, IL_13_split[i][2]/10))))

```

APPENDIX D: COMPLEMENTARY RESULTS

D.1. OPTIMIZATION WITH PARAMETER-SWEEP ALGORITHM

D.1.1. Cylindrical Domain, Sound-Hard, Single Body

Results obtained for a cylindrical solution domain utilizing BGT-2 ABC on the outer boundary.

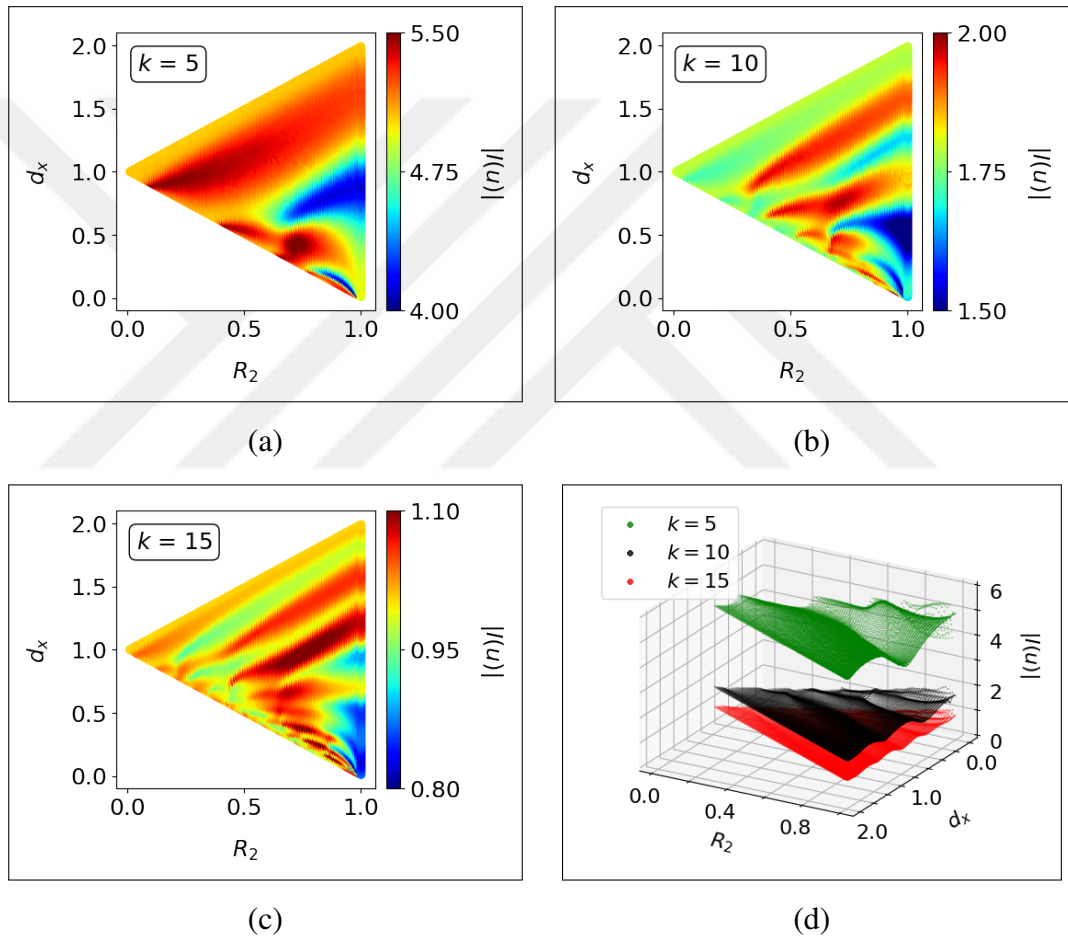


Figure D.1. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the parameter sweep algorithm.

D.1.2. Cylindrical Domain, Sound-Soft, Single Body

Parameter-sweep optimization results for sound-soft obstacle in a cylindrical domain utilizing BGT-2 ABC.

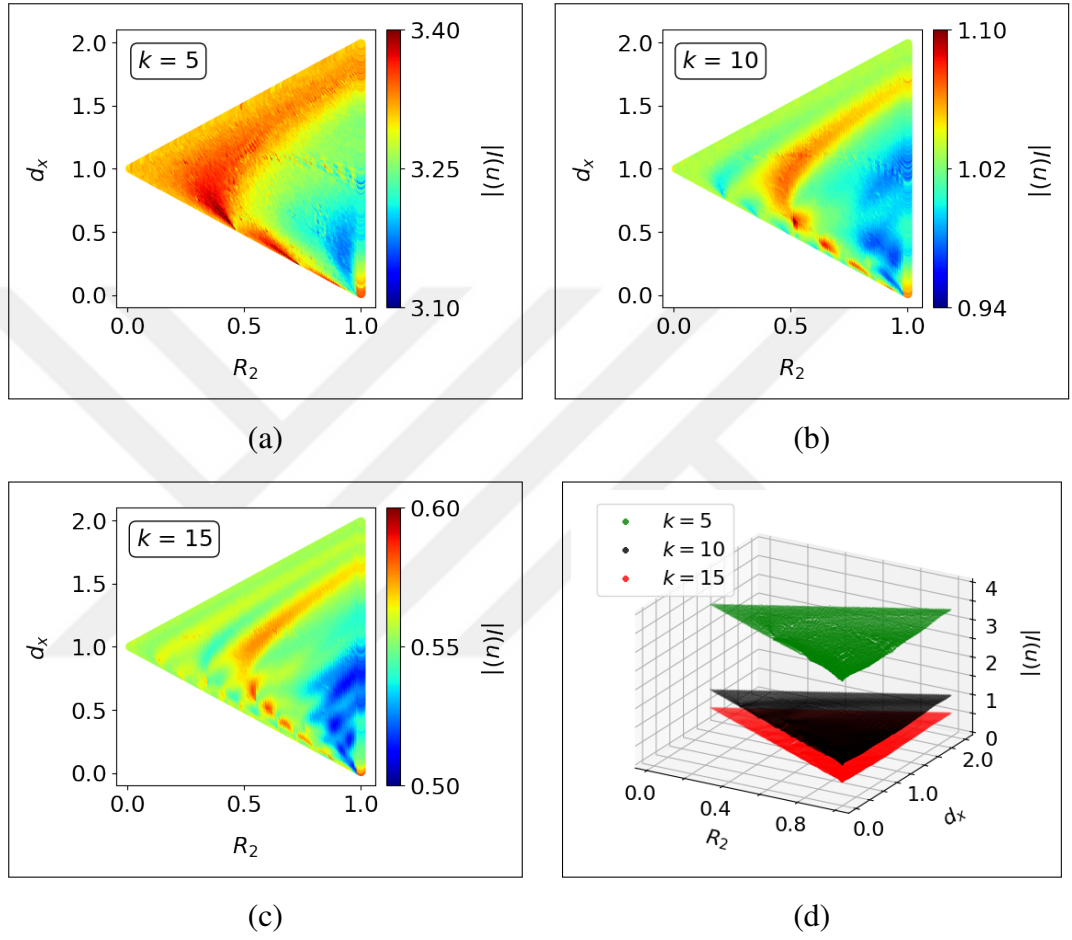


Figure D.2. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the parameter sweep algorithm.

D.1.3. Rectangular Domain, Sound-Hard, Single Body

Results obtained for a rectangular solution domain utilizing Sommerfeld ABC on the outer boundary.

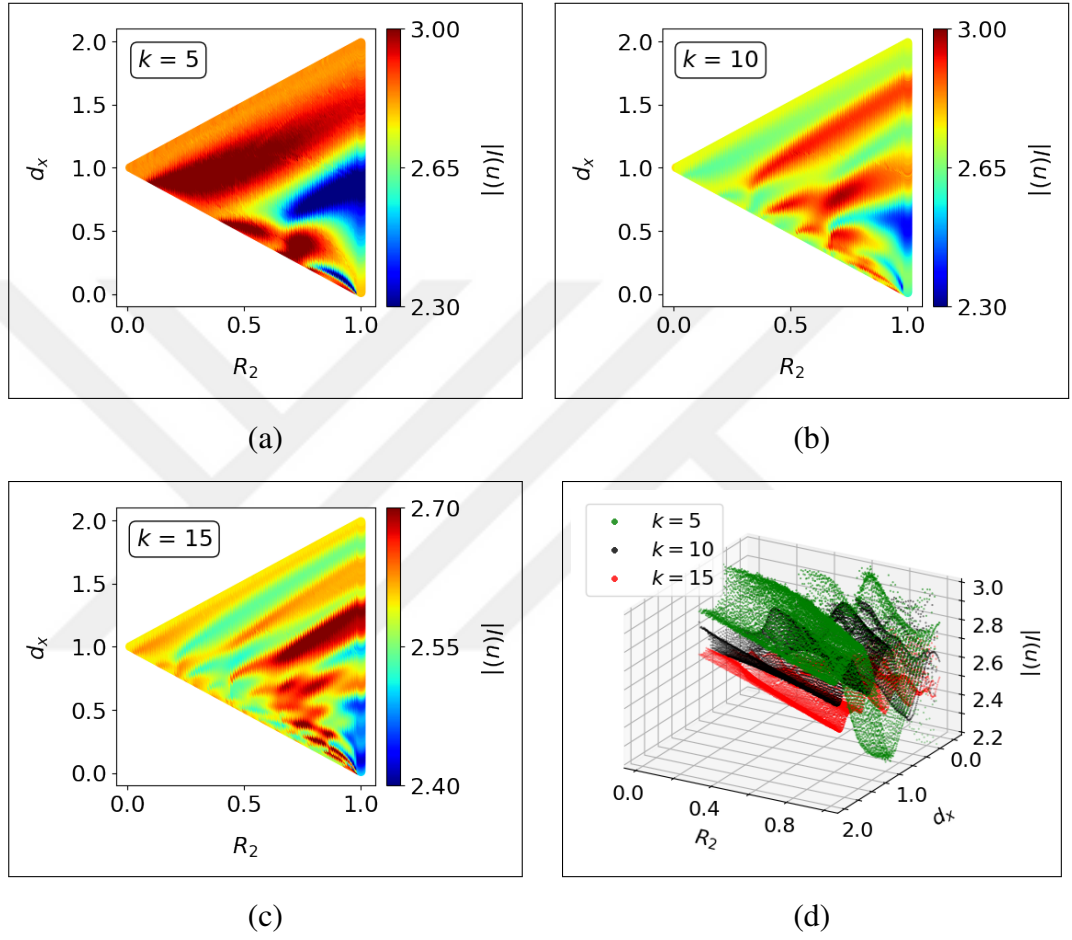


Figure D.3. Optimization results for a single crescent-shaped obstacle in a rectangular solution domain utilizing Sommerfeld ABC with the parameter sweep algorithm.

D.1.4. Rectangular Domain, Sound-Soft, Single Body

Parameter-sweep optimization results for sound-soft obstacle in a rectangular solution domain utilizing BGT-2 ABC.

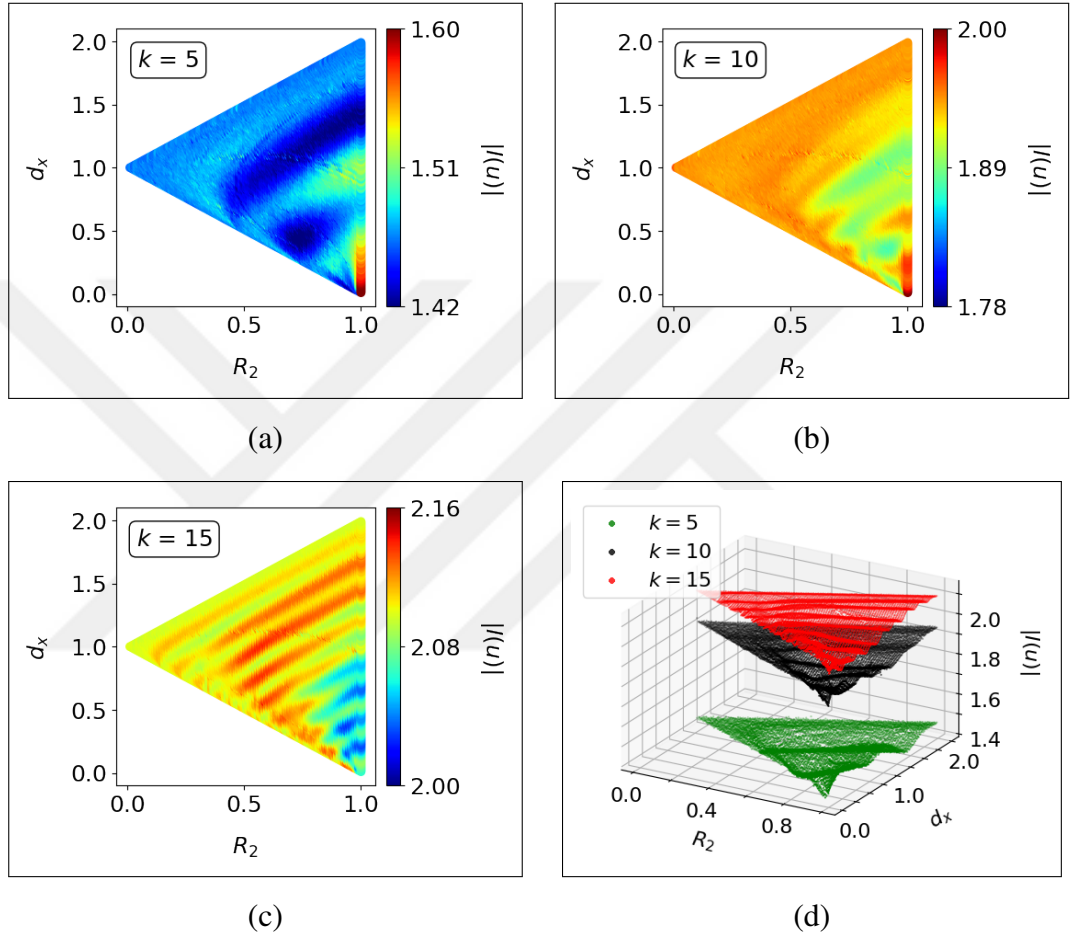


Figure D.4. Optimization results for a single sound-soft crescent-shaped obstacle in a rectangular solution domain utilizing Sommerfeld ABC with the parameter sweep algorithm.

D.1.5. Rectangular Domain, Sound-Hard, Periodic Bodies

Parameter-sweep optimization results for sound-hard obstacle in a rectangular solution domain utilizing periodic BC on top and bottom, Robin BC on left and right boundaries.

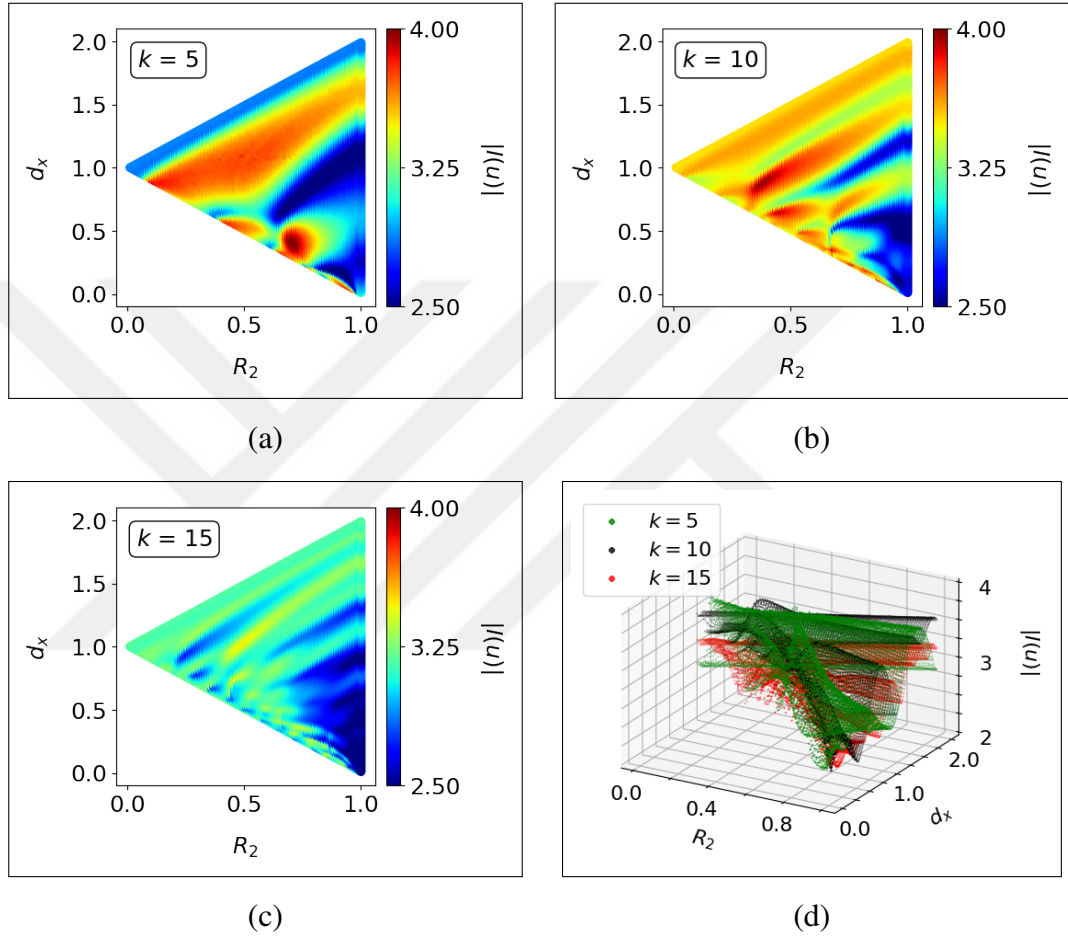


Figure D.5. Optimization results for a periodic sound-hard crescent-shaped obstacle in a rectangular solution domain utilizing PBC and Robin BC with the parameter sweep algorithm.

D.1.6. Rectangular Domain, Sound-Soft, Periodic Bodies

D.1.7. Rectangular Domain, Sound-Hard, Periodic Bodies

Parameter-sweep optimization results for sound-soft obstacle in a rectangular solution domain utilizing periodic BC on top and bottom, Robin BC on left and right boundaries.

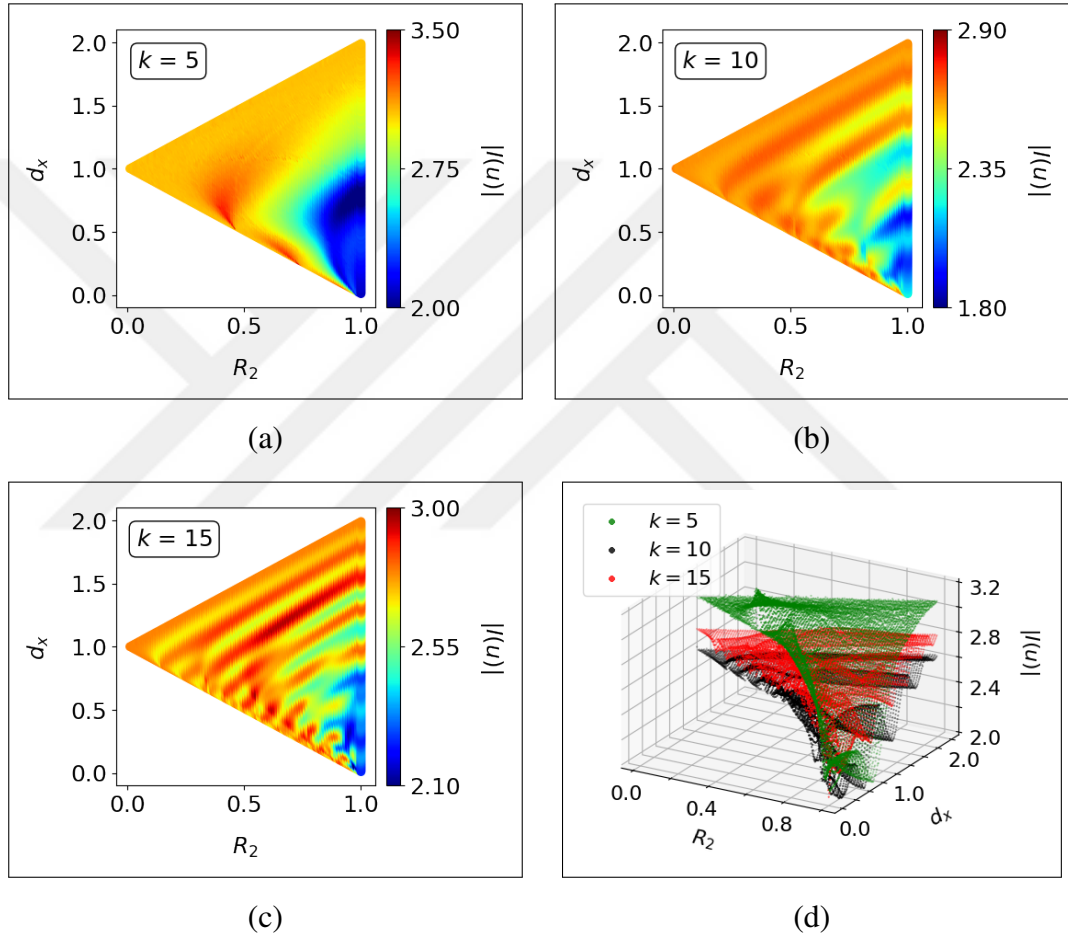


Figure D.6. Optimization results for a periodic sound-soft crescent-shaped obstacle in a rectangular solution domain utilizing PBC and Robin BC with the parameter sweep algorithm.

D.2. OPTIMIZATION WITH NELDER-MEAD ALGORITHM

D.2.1. Cylindrical Domain, Sound-Hard, Single Body

Results obtained for a sound-hard obstacle in a cylindrical solution domain utilizing BGT-2 ABC.

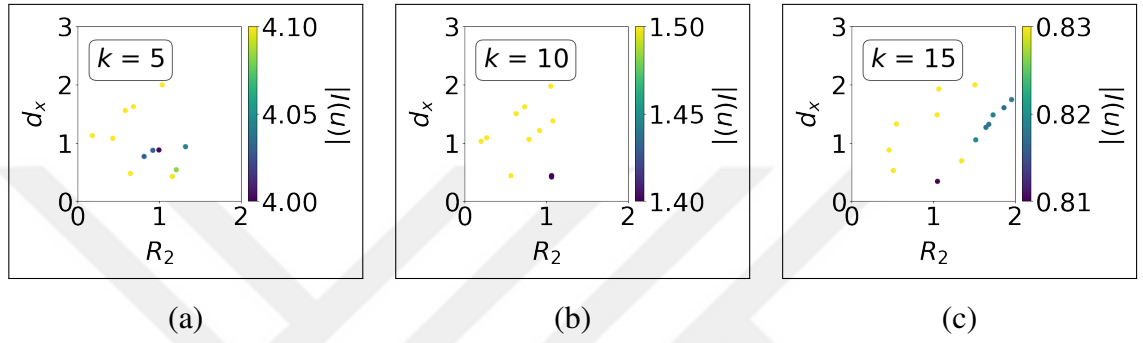


Figure D.7. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the Nelder-Mead algorithm.

D.2.2. Cylindrical Domain, Sound-Soft, Single Body

Results obtained for a sound-soft obstacle in a cylindrical solution domain utilizing BGT-2 ABC.

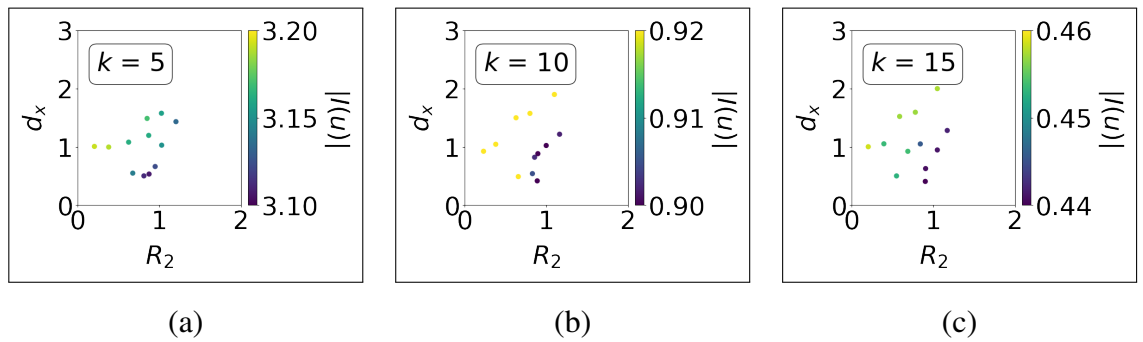


Figure D.8. Optimization results for a single crescent-shaped obstacle in a cylindrical solution domain utilizing BGT-2 ABC with the Nelder-Mead algorithm.

D.2.3. Rectangular Domain, Sound-Hard, Single Body

Results obtained for a sound-hard obstacle in a rectangular solution domain.

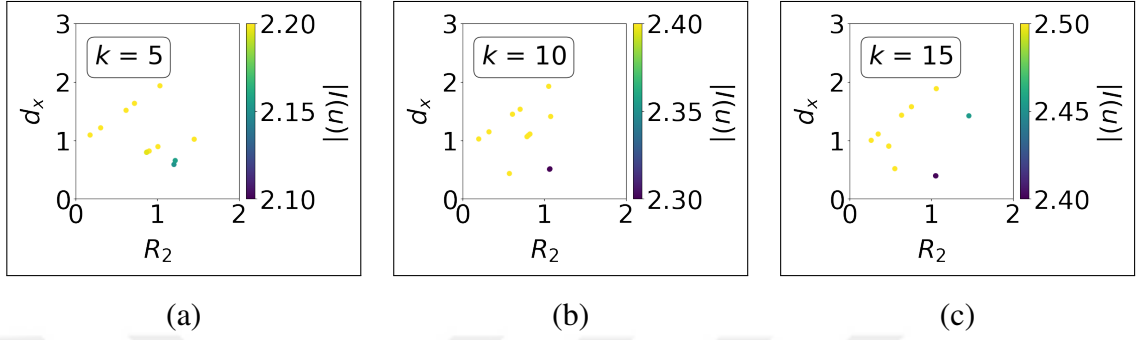


Figure D.9. Optimization results for a single crescent-shaped obstacle in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.

D.2.4. Rectangular Domain, Sound-Soft, Single Body

Results obtained for a sound-soft obstacle in a rectangular solution domain.

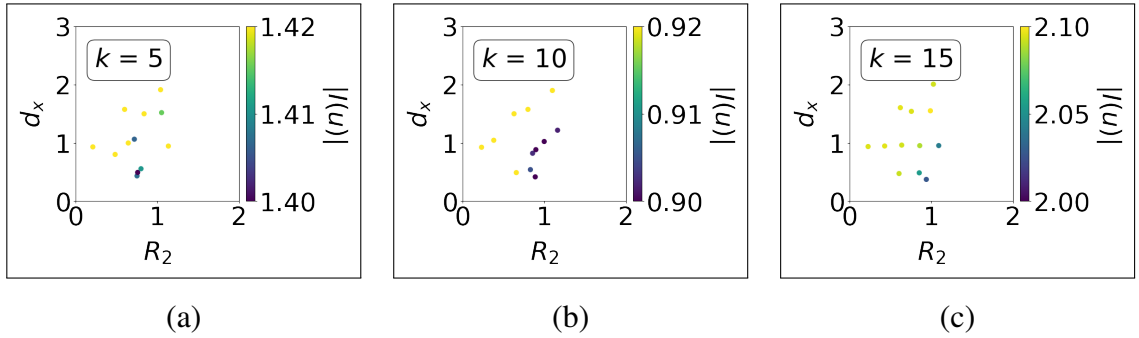


Figure D.10. Optimization results for periodic crescent-shaped obstacles in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.

D.2.5. Rectangular Domain, Sound-Hard, Periodic Bodies

Results obtained for sound-hard obstacles with 1-D periodic arrangement in a rectangular solution domain.

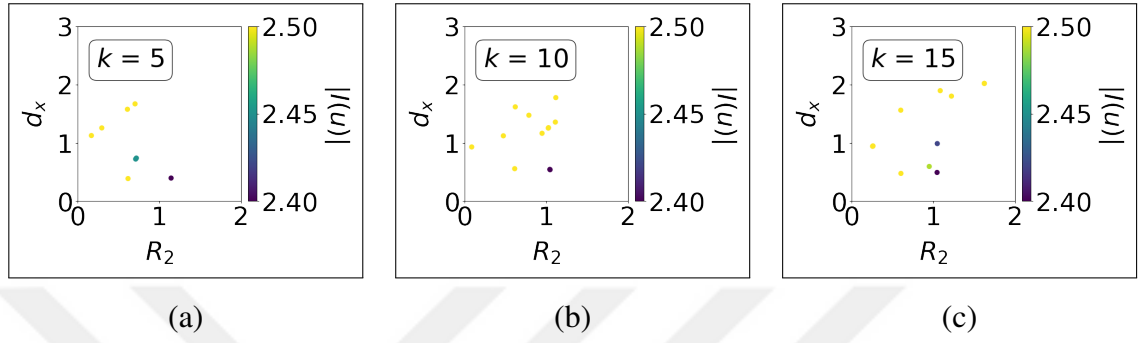


Figure D.11. Optimization results for periodic crescent-shaped obstacles in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.

D.2.6. Rectangular Domain, Sound-Soft, Periodic Bodies

Results obtained for sound-soft obstacles with 1-D periodic arrangement in a rectangular solution domain.

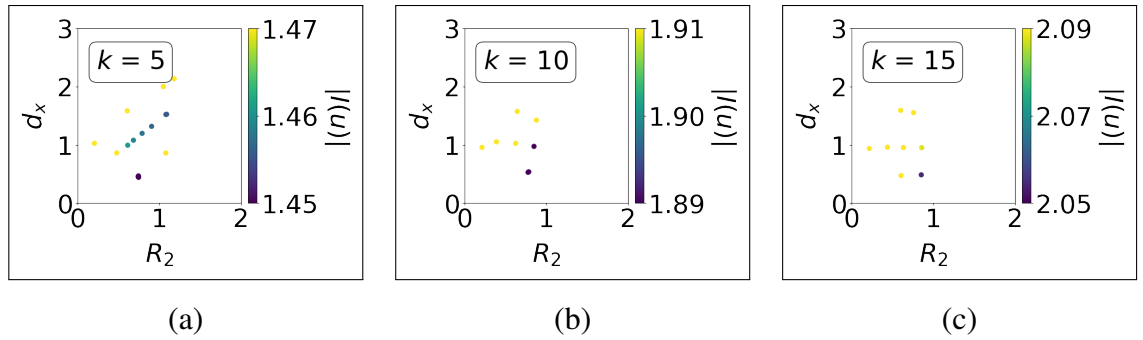


Figure D.12. Optimization results for periodic crescent-shaped obstacles in a rectangular solution domain utilizing Robin ABC with the Nelder-Mead algorithm.

D.3. ACOUSTIC LOUVER BENCHMARK DATA

Table D.1. Octave Band Insertion Loss Data from Manufacturers' Brochures

	Insertion Loss (dB)								w	OAR
	63 Hz	125 Hz	250 Hz	500 Hz	1 kHz	2 kHz	4 kHz	8 kHz		
Acoustic Louver 1 [132]	7	7	9	12	16	19	17	16	0.25	30 ≤ OAR ≤ 40
Acoustic Louver 2 [133]	5	5	7	9	13	13	12	12	0.15	N/A
Acoustic Louver 3 [133]	7	7	9	14	21	22	16	14	0.3	N/A
Acoustic Louver 4 [134]	8	11	13	16	19	23	24	24	0.3	N/A
Acoustic Louver 5 [134]	9	13	15	17	20	19	21	21	0.3	N/A
Acoustic Louver 6 [135]	3	4	6	10	12	13	13	14	0.15	15 ≤ OAR ≤ 30
Acoustic Louver 7 [135]	4	5	7	12	16	16	16	18	0.15	8 ≤ OAR ≤ 20
Acoustic Louver 8 [135]	5	6	9	14	20	20	20	20	0.3	10 ≤ OAR ≤ 30
Acoustic Louver 9 [135]	6	7	10	18	31	28	26	25	0.3	6 ≤ OAR ≤ 20
Acoustic Louver 10 [136]	2	3	5	7	10	12	12	11	0.15	N/A
Acoustic Louver 11 [137]	9	11	9	11	15	17	16	16	0.1	15 ≤ OAR ≤ 30
Acoustic Louver 12 [138]	4	6	7	12	12	13	14	14	0.2	20 ≤ OAR ≤ 40
Acoustic Louver 13 [138]	7	8	8	17	18	19	18	19	0.3	20 ≤ OAR ≤ 40
Acoustic Louver 14 [139]	2	5	7	12	18	21	16	16	0.3	10 ≤ OAR ≤ 25
Acoustic Louver 15 [139]	3	9	12	21	32	34	32	32	0.3	10 ≤ OAR ≤ 25

D.4. ACOUSTIC LOUVER FEA RESULTS

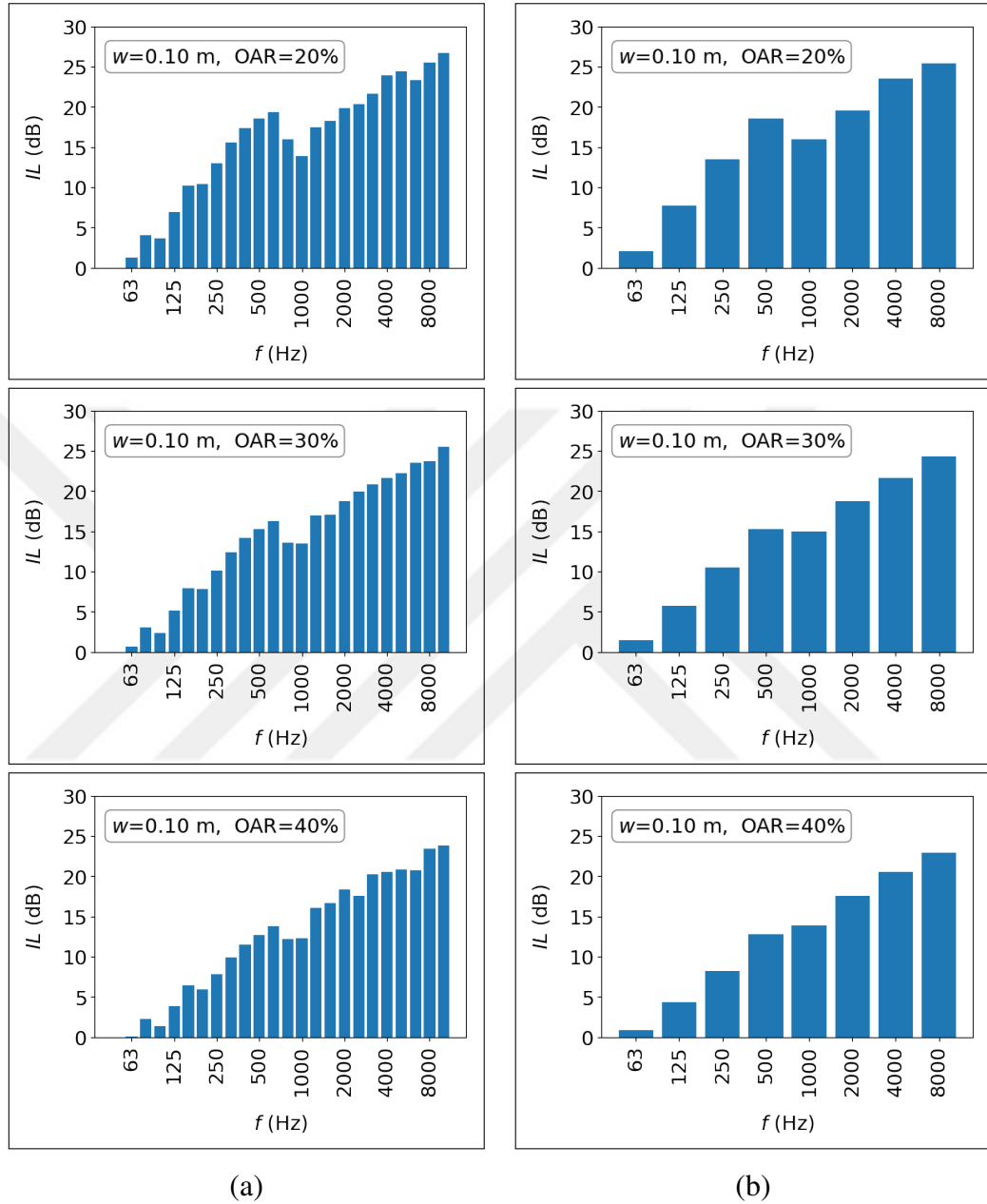


Figure D.13. Insertion loss for a conventional acoustic louver with $w = 100\text{ mm}$ thickness
(a) 1/3 octave band and (b) 1/1 octave band results.

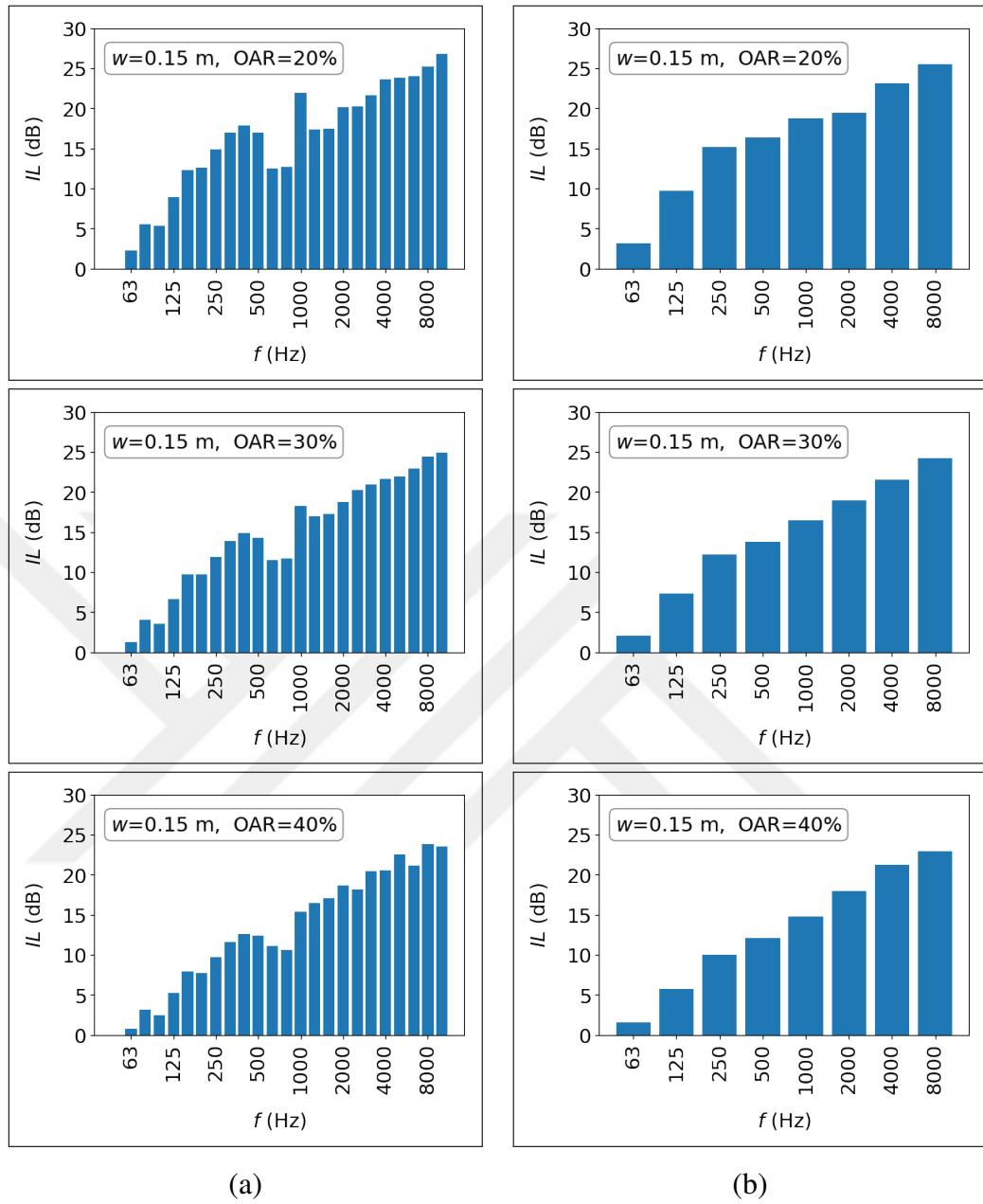


Figure D.14. Insertion loss for a conventional acoustic louver with $w = 150$ mm thickness
 (a) 1/3 octave band and (b) 1/1 octave band results.

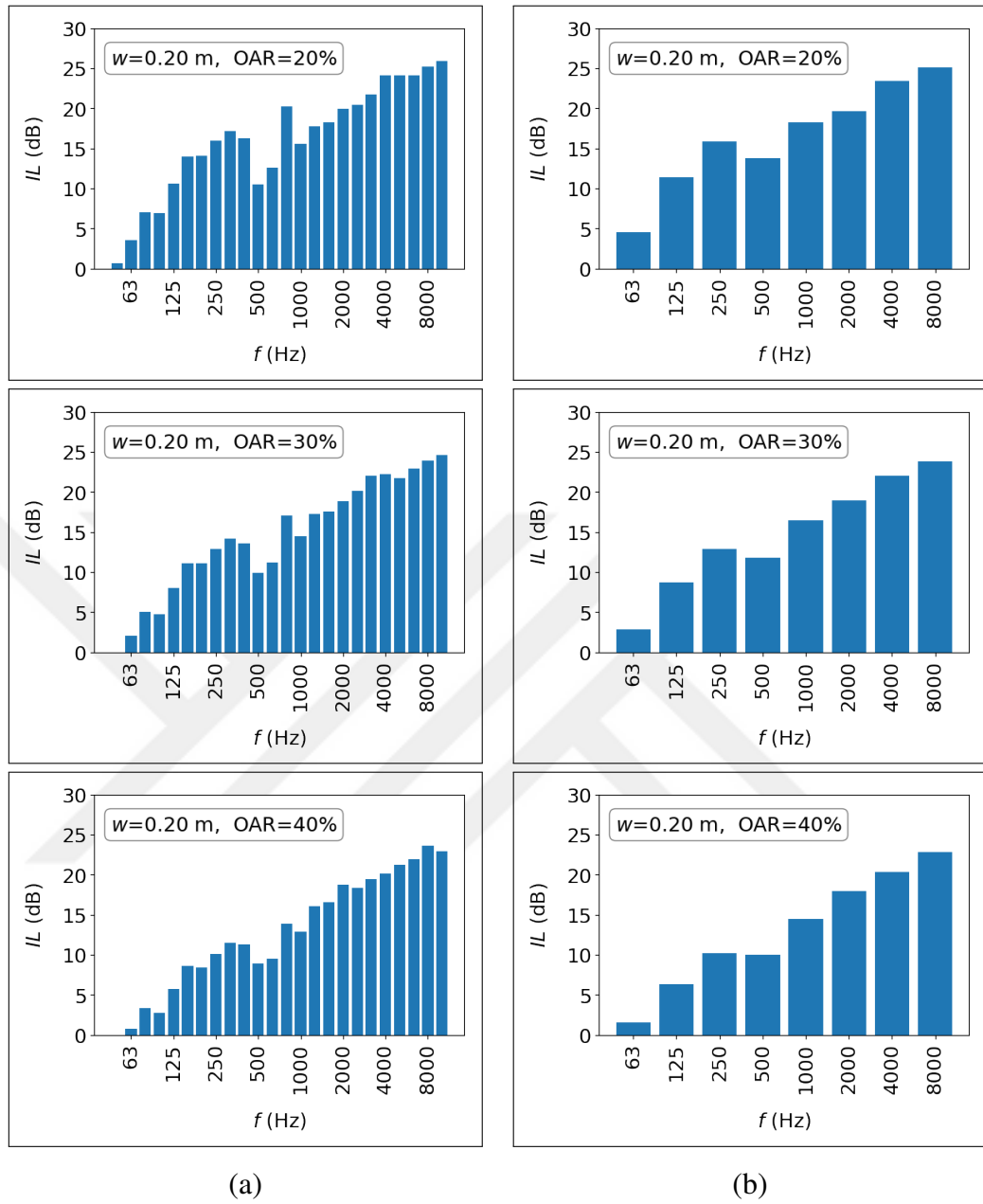


Figure D.15. Insertion loss for a conventional acoustic louver with $w = 200$ mm thickness
(a) 1/3 octave band and (b) 1/1 octave band results.

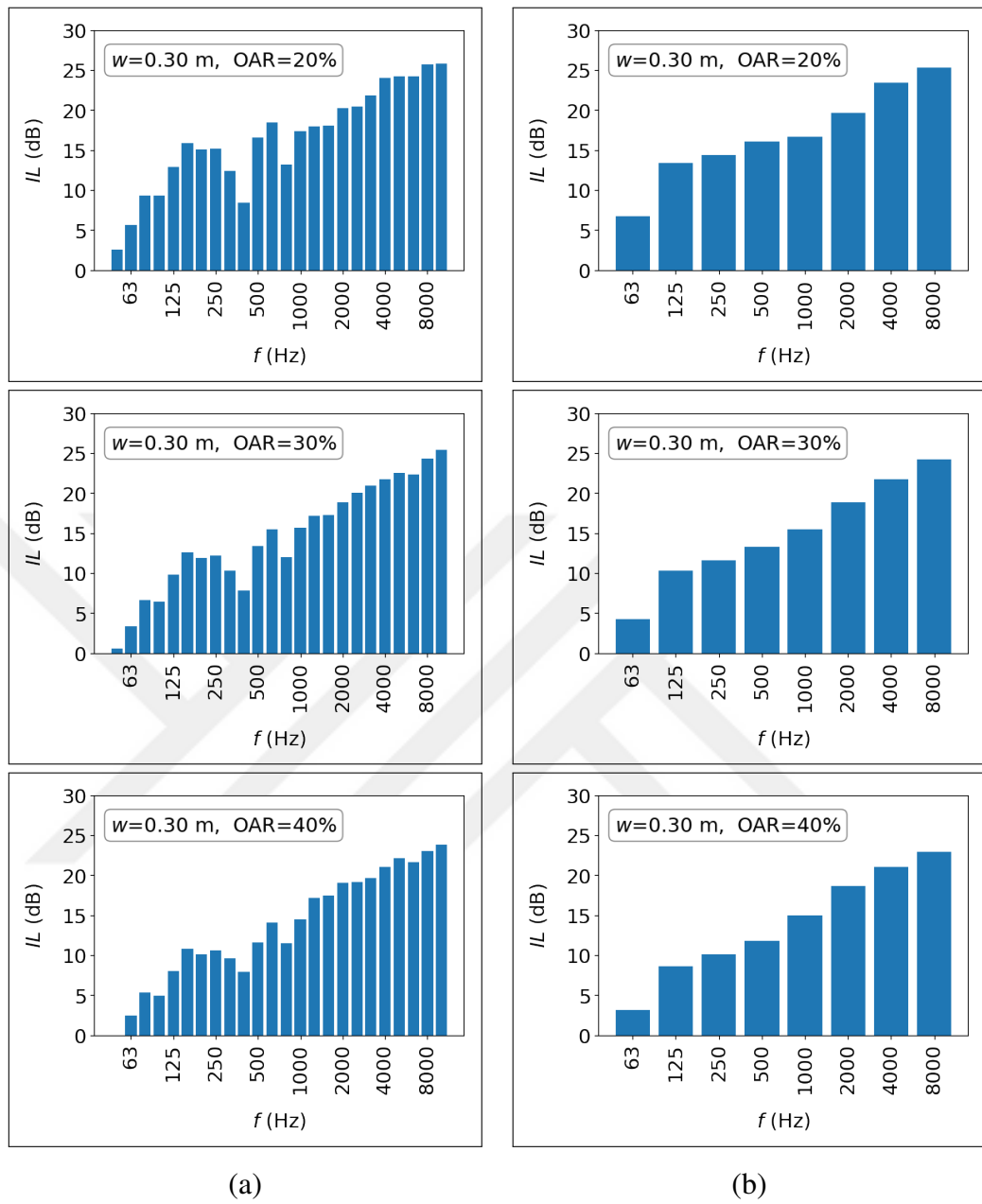


Figure D.16. Insertion loss for a conventional acoustic louver with $w = 300$ mm thickness
(a) 1/3 octave band and (b) 1/1 octave band results.