POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Efficient Techniques for Accurate Sound Propagation Modeling in Virtual Acoustics

Tesi di Laurea Magistrale in
Music and Acoustic Engineering - Ingegneria Acustica e Musicale

Author: **Gerardo Cicalese**

Student ID: 10776504
Advisor: Prof. Ilario Mazzieri
Co-advisor: Prof. Gabriele Ciaramella
Academic Year: 2022-2023

Abstract

The modeling of sound propagation in acoustical spaces has garnered interest with the advent of virtual acoustics and the subsequent rise in popularity of virtual reality. Virtual acoustics, in fact, finds applications in interactive multimedia platforms such as virtual and augmented reality (VR and AR), video games, VoIP-enabled virtual environments, and the Metaverse. In the field of architecture and building design, VR technology has proven invaluable, enabling enhanced design comparison and facilitating communication among stakeholders. Moreover, VR technology aids in predicting noise levels in diverse environments, contributing to improved acoustical comfort and the formulation of effective noise mitigation strategies.

Our research is centered around the modeling of sound propagation in acoustical spaces, with a particular focus on exploring innovative wave-based techniques. We delve into the Adaptive Rectangular Decomposition (ARD) method, renowned for its computational efficiency and its capability to deliver accurate acoustic responses. Our objective is to overcome certain limitations associated with the ARD method, namely the absence of air damping support and the lack of realistic boundary conditions.

Keywords: virtual acoustics, sound propagation modeling, auralization

Abstract in lingua italiana

La modellazione della propagazione sonora negli ambienti acustici ha guadagnato una grande popolarità con l'avvento dell'acustica virtuale e il grande successo della realtà virtuale. L'acustica virtuale, infatti, trova applicazioni in piattaforme multimediali interattive come la realtà virtuale e aumentata (VR e AR), videogiochi, ambienti virtuali VoIP-enabled e il Metaverso. Nel campo dell'architettura e del design degli edifici, la tecnologia VR si è rivelata preziosa, consentendo un miglior confronto dei progetti e facilitando la comunicazione tra gli stakeholder. Inoltre, la tecnologia VR rappresenta un ausilio per la previsione dei livelli di rumore in diversi ambienti, contribuendo a migliorare il comfort acustico e a formulare efficaci strategie di mitigazione del rumore.
La nostra ricerca si concentra sulla modellazione della propagazione del suono negli ambienti acustici, con particolare attenzione all'esplorazione di innovative tecniche wave-based. Approfondiremo il metodo dell'Adaptive Rectangular Decomposition (ARD), noto per la sua efficienza computazionale e la sua capacità di fornire risposte impulsive acustiche accurate. Il nostro obiettivo è quello di superare alcune limitazioni associate al metodo ARD, in particolare l'assenza di supporto per l'assorbimento dell'aria e la mancanza di condizioni al contorno realistiche.

Parole chiave: acustica virtuale, modellazione della propagazione sonora, auralizzazione

Contents

1 | Introduction

The task of modeling sound propagation behavior in acoustical spaces has garnered increasing interest in recent years due to the rising popularity of virtual reality, leading to the development of *virtual acoustics*. Virtual acoustics refers to the three major subsystems involved in an acoustical communication task: source modeling, transmission medium (room acoustics) modeling, and receiver (listener) modeling (Savioja et al. [1999]).
Virtual acoustics finds application in several interactive multimedia applications, such as VoIP-enabled virtual environments, video games, and, more recently, in *virtual and augmented reality* (VR and AR). In the latter case, VR and AR experiences are delivered via Head-Mounted Display devices (HMD) for visual rendering and for head-tracking, while headphones are employed for audio reproduction: the audio-visual presentation must take into account the user's position in a 3D space and its head's orientation, effectively simulating a *6-Degree-of-Freedom* (6DoF) navigation inside a virtual world which, recently, has lead to the development of the *Metaverse*. The Metaverse is a simulated world, in which inanimate objects and players coexist in an environment that manifests itself through realistic audio-visual scene rendering (Jot et al. [2021]).
The technology developed to build the Metaverse has made possible to extend the applicability of AR and VR to many other cases of use, not limited to entertainment as before. These include virtual meetings, assisted surgery, virtual travel, and "time machines". In fact, in the aftermath of the Notre-Dame cathedral fire, a team of researchers and sound engineers created a virtual reconstruction of a concert in the cathedral based on old recordings. This innovative approach offers users an immersive audio experience, transporting them to a specific moment in the past (Brian et al. [2021]).
Creating a convincing illusion for the human brain is no simple task. To achieve suspension of disbelief, a strong consistency must be maintained between the audio and visual components of the user's experienced scene, known as audio/visual congruence. Additionally, congruence between virtual elements and the real world, as previously experienced by the user, is crucial (virtual/real congruence) (Jot et al. [2021]).
For instance, when a user is in a cathedral, they naturally expect to hear sounds with noticeable reverberations. Conversely, in a library setting, a sense of intimacy should be conveyed through the soundscape. Merely simulating sound propagation from sources to the listener is insufficient. Factors such as the variations in sound heard by each ear and

the scattering process from the user's body, head, and ears contribute to the externalization of sounds. Neglecting these phenomena results in the sound emitted from headphones appearing to originate from within the listener's head, leading to a lack of spatial localization of sounds. Throughout this work, we will briefly explore some techniques used to model these acoustic effects (Kosikowski [2018]).

In recent years, VR has gained significant traction in the field of architecture and building design. VR technology proves invaluable in the design process, simplifying the comparison of different design choices and enhancing communication among stakeholders. In the past, the evaluation of design choices from an aesthetic perspective relied on scale models, while assessing acoustic aspects involved analyzing room acoustics parameters such as reverberation time, clarity, and speech intelligibility index. These evaluations were conducted either on scale models or through numerical simulations.

Advancing beyond these methods, auralizations emerged as a novel approach. Auralizations allow listeners to experience the sound of a simulated room by utilizing room impulse responses, given a source-receiver configuration and a virtual environment. However, the use of VR takes the evaluation of acoustic qualities in a space to a new level. By immersing users in a VR environment, enabling them to freely navigate the modeled space, and providing the ability to switch between different proposed designs, a more comprehensive understanding of the space's acoustics can be achieved. This immersive experience can be likened to an interactive auralization, where both the source and receiver have the capability to move. It is crucial to assess how the acoustic quality within a space varies based on the positions of the source and listener (Kosikowski [2018]).

In addition to the previous applications, VR proves to be valuable in predicting noise levels within different environments, aiming to improve the overall comfort of occupants. By simulating and analyzing acoustic scenarios, such as urban settings, workplaces, and public spaces, our methodology can assist in identifying potential noise concerns and devising effective strategies for noise mitigation. This utilization has the potential to enhance living conditions, promote healthier environments, and contribute to the fields of urban planning and design. In this regard, it is essential to not only focus on auralization alone but also consider the broader concept of soundscape composition. Predicting noise levels within a room necessitates modeling all ambient sounds that contribute to the soundscape. For example, when simulating an office environment, it becomes crucial to reproduce sounds such as keyboard clicks, ambient noise from outside, conversations among individuals, impact noise from walking on the upper floor, and so on (Kosikowski [2018]).

Furthermore, this methodology finds application in the automotive and aerospace industries, where it assists in simulating and analyzing airborne noise and vibration patterns.

By employing VR, designers can effectively enhance the design of vehicles, aiming for reduced noise levels and increased comfort for passengers.
Throughout this work, our focus will be extensively on modeling sound propagation in acoustical spaces. Ray-based methods, such as ray tracing and the image-source method, are the most commonly used modeling techniques. However, wave-based techniques, such as the finite-element method (FEM), boundary-element method (BEM), and finite-difference time-domain (FDTD) method, which directly solve the acoustic wave equation, have gained interest recently. These wave-based techniques provide a full transient solution that accurately accounts for all wave phenomena, including diffraction. However, they are computationally expensive and suitable for simulating low frequencies only.
Recently, an Adaptive Rectangular Decomposition (ARD) technique was proposed, which achieves two orders of magnitude speed-up over FDTD methods and other state-of-the-art numerical techniques, while guaranteeing the computation of an accurate acoustic response. This simulation method is still too computationally expensive to unveil the possibilities of developing a fully real-time auralization technique; many interactive applications like video games only devote 10% of the total computation and memory resources to auralization. Hence, precomputation-based methods using ARD were developed and are well documented in Raghuvanshi et al. [2011] and Chandak et al. [2011], either for the trivial Static Source - Static Receiver (SS-SR) case and for the more general Moving Source - Moving Receiver (MS-MR) case, and will be presented in this work.
It is a common practice in video game engines to model reverberation by means of digital filters (reverb filters), whose parameters depend on the room's characteristics and on the source-receiver configuration. However, these filters are not physically based and only provide a rough approximation of acoustical spaces with different sizes. They are unable to capture the full range of acoustical effects observed in real life. Additionally, manually assigning these reverb filters to different parts of the environment requires a significant amount of time and effort. ARD can be used to precompute high-quality reverb filters for arbitrary scenes without any human intervention. This technique provides physically motivated reverb filters parameters, ensuring that the audio pipeline remains unchanged.
In the current state of the art, the ARD method exhibits several limitations. Firstly, it is impractical to model enclosures with irregular boundaries due to the algorithm's reliance on rectangular decomposition within the volume. Moreover, the supported boundary conditions are limited to partial and full absorption, which do not account for frequency-dependent absorption characteristics commonly observed in physical walls and boundaries. Additionally, the neglect of air absorption is acceptable only in small spaces, while in larger spaces, such as cathedrals, air damping becomes a significant factor. In our research, we will present a modified version of the ARD algorithm that incorporates air

damping, addressing this limitation. Furthermore, we will offer insights into effectively modeling various types of boundary conditions.

2 | Mathematical modeling and solution techniques for acoustics

In this chapter, we explore the mathematical foundations and some solution techniques used in acoustics.
We begin by formulating our acoustic problem using the unviscid and viscous wave equations, which are the most important mathematical models in acoustics.
Next, we introduce analytical tools for solving partial differential equations. Specifically, we explore Fourier analysis, which allows us to decompose complex waveforms, and the separation of variables technique, which enables modal analysis of the wave equation.
Numerical methods play a crucial role in solving complex acoustic problems. We explore the finite difference method, a popular numerical technique for solving partial differential equations. We present a well known Finite Difference Time Domain (FDTD) scheme for the wave equation, with a focus on its performance and stability.
Furthermore, we investigate the harmonic oscillator, a model that provides insights into various systems, including the evolution of the normal modes of the wave equation.

2.1. Acoustic problem formulation

In this section we introduce the wave equation, the most important partial differential equation in acoustics. We will also formulate the problem of acoustic wave propagation which will be used throughout this text. The following dissertation is based on Asmar [2010] and Bilbao [2009].

2.1.1. Unviscid wave equation

The propagation of sound waves in an isentropic fluid within a three-dimensional domain Ω is characterized by the *three-dimensional wave equation*, which is commonly referred to as the (unviscid) *acoustic wave equation*:

$$\frac{\partial^2 p(\underline{x},t)}{\partial t^2} - c^2 \Delta p(\underline{x},t) = f(\underline{x},t), \quad t \in \mathbb{R}^+, \quad \underline{x} \in \mathbb{R}^3, \tag{2.1.1}$$

where $p = p(\underline{x}, t)$ denotes the pressure at the point $\underline{x}$ at time t, c is the *propagation speed*, and $f = f(\underline{x}, t)$ denotes an external force at the point $\underline{x}$ at time t.
We also introduce the *pressure velocity* v, which is defined as

$$v(\underline{x}, t) \triangleq \frac{\partial p}{\partial t}. \tag{2.1.2}$$

Please note that the pressure velocity, defined as the time derivative of the pressure, does not correspond to the particle velocity, which is instead proportional to the Laplacian of the pressure. The latter won't be considered throughout this work.

2.1.2. Viscous wave equation

The attenuation of sound waves in the free medium becomes significant in large rooms, in particular at high frequencies. Thus, in reverberation calculations it has to be taken into account, Cfr. Kuttruff and Everton [2010].
We introduce frequency-independent, mass-proportional damping to model viscous dissipation in air. As the treatment is analogous to the unviscid case, we'll only report the differences.
Sound propagation in a fluid, considering viscous damping, is governed by the *viscous acoustic wave equation*:

$$\frac{\partial^2 p(\underline{x}, t)}{\partial t^2} + 2\alpha \frac{\partial p(\underline{x}, t)}{\partial t} - c^2 \Delta p(\underline{x}, t) = F(x, t), \quad t \in \mathbb{R}^+, \quad \underline{x} \in \mathbb{R}^3, \tag{2.1.3}$$

where α is the *absorption coefficient*.

2.1.3. Initial conditions

The wave equation is a second-order (in time) PDE, and, as such, requires the specification of two initial conditions. Normally, these are the values of the pressure p and of the pressure velocity v at time $t = 0$, i.e.,

$$\begin{cases} p(\underline{x}, 0) = p_0(\underline{x}), \\ \dfrac{\partial p}{\partial t}(\underline{x}, 0) = v_0(\underline{x}), \end{cases} \tag{2.1.4}$$

where p_0 and v_0 are given functions.

2.1.4. Boundary conditions

The wave equation involves a second-order differentiation in space, and, as such, requires the specification of boundary conditions on the boundary of the spatial domain.
If we consider Eq. (2.1.1) or Eq. (2.1.3) in a bounded domain Ω, we can impose the *Neumann boundary conditions* on the Lipschitz boundary $\Gamma_v = \partial\Omega$, namely,

$$\frac{\partial p(\underline{x}, t)}{\partial \underline{n}} = \overline{g}, \quad \underline{x} \in \Gamma_v, \tag{2.1.5}$$

where $\overline{g}$ is a given function. For $\overline{g} = 0$ (homogeneous case), it corresponds to a closed end; a propagating wave impinging a closed end will be reflected losslessly with no change of sign.
Another type are the *Dirichlet boundary conditions*, namely,

$$p(\underline{x}, t) = \overline{g}, \quad \underline{x} \in \Gamma_v. \tag{2.1.6}$$

For $\overline{g} = 0$, such a condition corresponds to an open end; a propagating wave impinging an open end will be reflected losslessly with a change of sign.

2.1.5. Force envelope modeling

In the study of acoustics, it is necessary to develop mathematical representations of sound fields. To accurately model the forces acting on the pressure field, a common approach is to utilize force envelope modeling. This technique involves defining a spatial region within which the force is assumed to act.
We can express the force as

$$f(\underline{x}, t) = f_{env}(\underline{x}) f_{time}(t),$$

where $f_{env}(\underline{x})$ is the force envelope and $f_{time}(t)$ is the time evolution term of the force.
One example of a force envelope is a Gaussian envelope which, in the three-dimensional case, can be defined as

$$f_{env}(\underline{x}) \triangleq \frac{1}{(2\pi)^{3/2}\sigma_x\sigma_y\sigma_z} e^{-\frac{(x-\mu_x)^2}{2\sigma_x^2} - \frac{(y-\mu_y)^2}{2\sigma_y^2} - \frac{(z-\mu_z)^2}{2\sigma_z^2}},$$

where μ, σ represent respectively the *mean* and the *standard deviation*. This function has a bell-shaped profile, with the maximum amplitude at the center and decreasing rapidly as the distance from the center increases, as shown in Fig. 2.1.1.
Concerning the time evolution term $f_{time}(t)$, it is common, as discussed in Sec. 7.6.1, for it to be an impulse signal with a short duration. This characteristic ensures that its

spectrum remains flat across a wide frequency range.

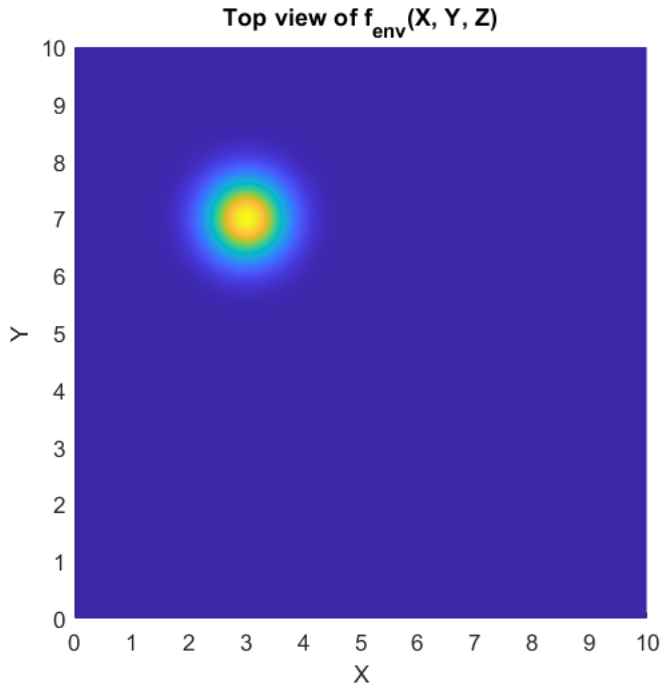


Figure 2.1.1: Top view ($z = 0$) of a three-dimensional Gaussian with $\mu_x = 3, \mu_y = 7, \mu_z = 0, \sigma_x = \sigma_y = \sigma_z = 0.5$.

2.1.6. Problem formulation

The problem of wave propagation that we will consider from now on is the wave equation with arbitrary initial conditions and the homogeneous Neumann boundary conditions, i.e., Eq. (2.1.5) with $\overline{g} = 0$. This choice of boundary conditions is necessary to develop the Fourier method (Cfr. Ch. 4). The problem can be formulated as follows:

$$\begin{cases} \frac{\partial^2 p(\underline{x},t)}{\partial t^2} + 2\alpha \frac{\partial p(\underline{x},t)}{\partial t} - c^2 \Delta p(\underline{x},t) = f(\underline{x},t), & t \in \mathbb{R}^+, \ \underline{x} \in \mathbb{R}^3, \\ p(\underline{x},0) = p_0(\underline{x}), & \underline{x} \in \mathbb{R}^3, \\ \frac{\partial p}{\partial t}(\underline{x},0) = v_0(\underline{x}), & \underline{x} \in \mathbb{R}^3, \\ \frac{\partial p(\underline{x},t)}{\partial \underline{n}} = 0, & t \in \mathbb{R}^+, \ \underline{x} \in \Gamma_v. \end{cases} \tag{2.1.7}$$

2.1.7. Setup of test cases

In this section, we'll introduce some test cases whose analytical solution is known, in order to perform testing of the numerical simulation methods. We'll consider the 1D wave equation.

The first one will be the *standing wave test case* in the domain $\Omega = [0, L]$ in the time range $[0, T]$:

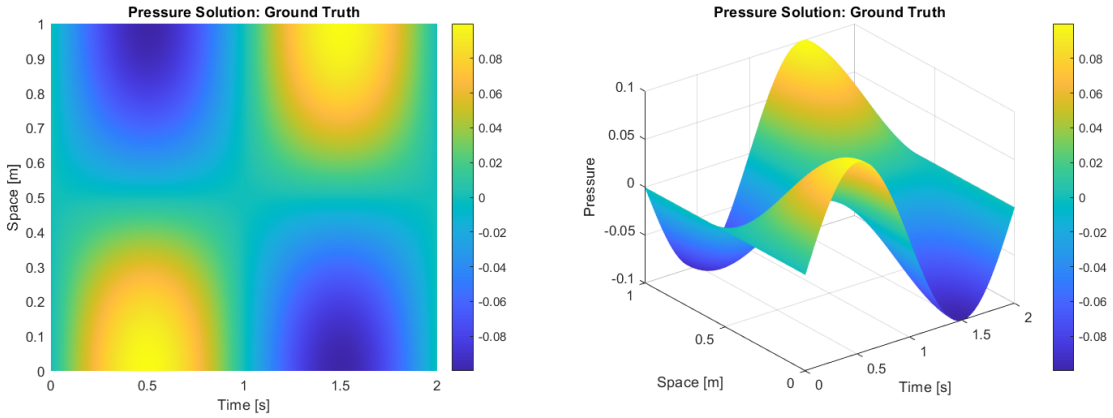
$$\begin{cases} p_0(x) = 0, \\ v_0(x) = A\omega \cos(kx), \\ f(x) = 0, \end{cases}$$

with the homogeneous Neumann boundary conditions on both ends. The parameters are configured as

$$\begin{cases} L = 1, \\ T = 2, \\ c_0 = 1, \\ \alpha = 0, \\ A = 0.1, \\ k = \pi, \\ \omega = kc_0 = \pi c_0. \end{cases}$$

The ground truth solution, shown in Fig. 2.1.2, is

$$p(x) = A\cos(kx)\sin(\omega t),$$
$$v(x) = A\omega\cos(kx)\cos(\omega t).$$



(a) Top view. (b) 3D view.

Figure 2.1.2: Ground truth solution of the standing wave test case.

The second one will be the *propagating wave test case* in the domain $\Omega = [0, L]$ in the

time range $[0, T]$:

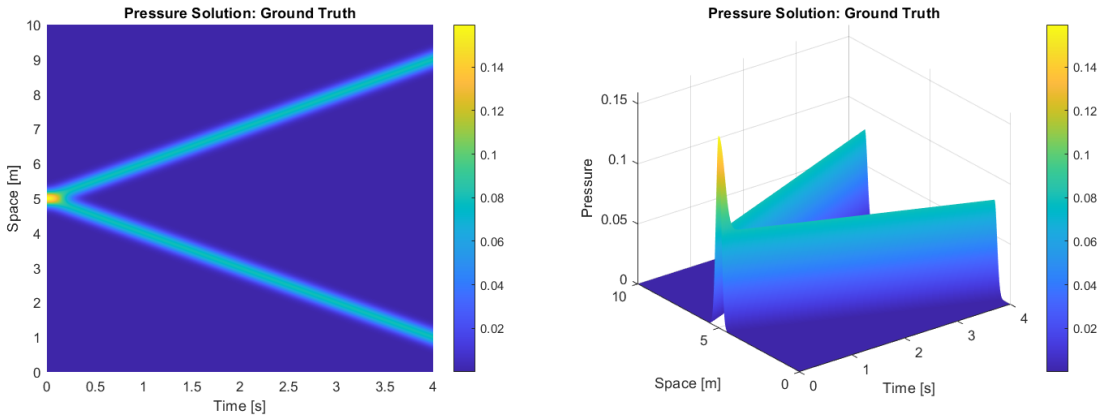
$$\begin{cases} y(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \\ y'(x) = \frac{dy}{dx} = \frac{-x}{\sigma^2} y(x), \\ p_0(x) = A\left(y(-kx) + y(kx)\right), \\ v_0(x) = A\omega\left(y'(-kx) + y'(kx)\right), \\ f(x) = 0, \end{cases}$$

with the homogeneous Neumann boundary conditions on both ends. The parameters are configured as

$$\begin{cases} L = 10, \\ T = 4, \\ c_0 = 1, \\ \alpha = 0, \\ A = 0.1, \\ k = \pi, \\ \omega = kc_0 = \pi c_0, \\ \mu = 5, \\ \sigma = 0.5. \end{cases}$$

The ground truth solution, shown in Fig. 2.1.3, is

$$\begin{aligned} p(x) &= A\left(y(\omega t - kx) + y(\omega t + kx)\right), \\ v(x) &= A\omega\left(y'(\omega t - kx) + y'(\omega t + kx)\right). \end{aligned}$$



(a) Top view.

(b) 3D view.

Figure 2.1.3: Ground truth solution of the propagating wave test case.

2.2. Modal analysis of the wave equation

This section introduces the fundamentals of Fourier analysis, which is a mathematical tool used to represent a signal as a sum of trigonometric functions. Furthermore, it will present the separation of variables technique applied to the wave equation. These tools constitute a framework which let us perform modal analysis and obtain the standing wave solution to the wave equation.

2.2.1. Continuous Fourier analysis

A Fourier series represents a periodic signal or a finite-length signal as a sum of sine and cosine functions.
The following dissertation is based on Asmar [2010].

Fourier Series

Suppose that $f(x)$ is a $2p$-periodic piecewise smooth function. The Fourier series of f is given by:

$$s_f(x) \triangleq a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{n\pi}{p}x\right) + b_n \sin\left(\frac{n\pi}{p}x\right) \right), \tag{2.2.1}$$

where

$$a_0 \triangleq \frac{1}{2p} \int_{-p}^{p} f(x)dx, \tag{2.2.2}$$

$$a_n \triangleq \frac{1}{p} \int_{-p}^{p} f(x) \cos\left(\frac{n\pi}{p}x\right) dx, \quad n = 1, 2, \ldots, \tag{2.2.3}$$

$$b_n \triangleq \frac{1}{2p}\int_{-p}^{p} f(x)\sin\left(\frac{n\pi}{p}x\right)dx, \quad n = 1, 2, \ldots . \tag{2.2.4}$$

Theorem 2.2.1: Convergence of the Fourier Series

- If f is continuous at x, the Fourier series $s_f(x)$ converges to $f(x)$;
- If f presents a jump discontinuity at x, the Fourier series $s_f(x)$ converges to $\frac{f(x-)+f(x+)}{2}$.

Definition 2.2.1: Compact form of the Fourier Series

Suppose that $f(x)$ is a **real** $2p$-periodic piecewise smooth function. The Fourier series of f in Eq. (2.2.1) can be rewritten in a *compact form* as

$$s_f(x) \triangleq C_0 + \sum_{n=1}^{\infty} C_n \cos\left(\frac{n\pi}{p}x + \theta_n\right), \tag{2.2.5}$$

where C_n and θ_n are related to a_n and b_n as

$$C_0 = a_0, \quad C_n = \sqrt{a_n^2 + b_n^2}, \quad \theta_n = \arctan\left(\frac{-b_n}{a_n}\right).$$

Note that the compact form in Eq. (2.2.5) uses the cosine form. We could just have used the sine form, with terms $\sin\left(\frac{n\pi}{p}x + \theta_n\right)$ instead of $\cos\left(\frac{n\pi}{p}x + \theta_n\right)$. The literature overwhelmingly favors the cosine form (Lathi and Green [2018]).

Definition 2.2.2: Complex form of Fourier Series

Let f be a $2p$-periodic smooth function. The *complex form of the Fourier series* of f is

$$\widetilde{s}_f(x) \triangleq \sum_{n=-\infty}^{\infty} c_n e^{j\frac{n\pi}{p}x}, \tag{2.2.6}$$

where the *Fourier coefficients* c_n are given by

$$c_n \triangleq \frac{1}{2p}\int_{-p}^{p} f(x)e^{-j\frac{n\pi}{p}x}dx, \quad n = 0, \pm 1, \pm 2, \ldots . \tag{2.2.7}$$

Theorem 2.2.2: Fourier Series of an even function

Suppose that $f(x)$ is $2p$-periodic and has the Fourier series representation as in Eq. (2.2.1). Then, f is even if and only if $b_n = 0$ for all $n \in \mathbb{N}$. In this case

$$s_f(x) \triangleq a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi}{p}x\right), \tag{2.2.8}$$

where

$$a_0 \triangleq \frac{1}{p}\int_0^p f(x)dx, \tag{2.2.9}$$

$$a_n \triangleq \frac{2}{p}\int_0^p f(x)\cos\left(\frac{n\pi}{p}x\right)dx, \quad n = 1, 2, \dots . \tag{2.2.10}$$

Theorem 2.2.3: Fourier Series of an odd function

Suppose that $f(x)$ is $2p$-periodic and has the Fourier series representation as in Eq. (2.2.1). Then, f is odd if and only if $a_n = 0$ for all $n \in \mathbb{N}$ (including a_0). In this case

$$s_f(x) \triangleq \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi}{p}x\right), \tag{2.2.11}$$

where

$$b_n \triangleq \frac{2}{p}\int_0^p f(x)\cos\left(\frac{n\pi}{p}x\right)dx, \quad n = 1, 2, \dots . \tag{2.2.12}$$

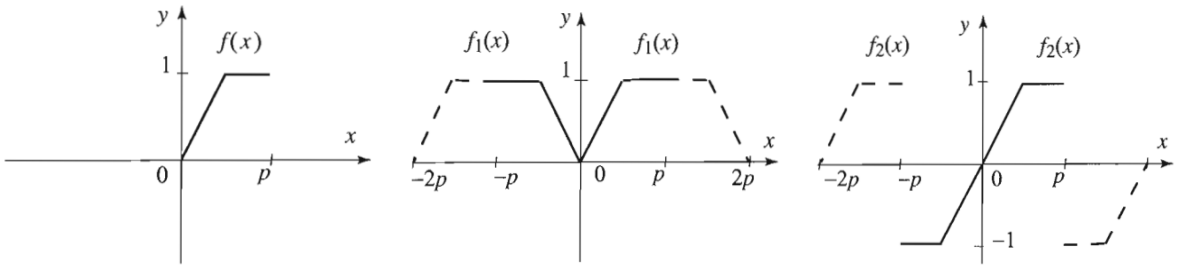
Half-range expansions: Cosine Series and Sine Series

In many applications, we are interested in representing a function $f(x)$ by a Fourier series that is defined only in a finite interval, say $0 < x < p$. Since f is clearly not periodic, the results of the previous sections are not readily applicable. However, we can represent f by a Fourier series s_f, after extending it to a periodic function.
We define the *even periodic extension* of f:

$$f_e(x) \triangleq \begin{cases} f(x), & 0 < x < p, \\ f(-x), & -p < x < 0, \\ f_e(x + 2p), & \textit{otherwise.} \end{cases}$$

We define the *odd periodic extension* of f:

$$f_o(x) \triangleq \begin{cases} f(x), & 0 < x < p, \\ -f(-x), & -p < x < 0, \\ f_o(x + 2p), & \textit{otherwise.} \end{cases}$$



(a) $f(x), 0 < x < p$. (b) Even $2p$-periodic extension. (c) Odd $2p$-periodic extension.

Figure 2.2.1: Example of extensions of a function f. Source: Asmar [2010].

By the way they are constructed, the function f_e is even and $2p$-periodic, and the function f_o is odd and $2p$-periodic. Both functions agree with f on the interval $0 < x < p$, which justifies calling them *extensions* of f (Fig. 2.2.1). Since f is piecewise smooth, it follows that f_e and f_o are both piecewise smooth.

Applying Th. 2.2.2 and Th. 2.2.3, we find respectively that f_e has a cosine series expansion (Th. 2.2.4) and f_o has a sine series expansion (Th. 2.2.5). Now, $f(x) = f_e(x)$ for all $0 < x < p$, and so the cosine series represents f on this interval. Similar reasoning, using f_o yields the sine series expansion of f.

Theorem 2.2.4: Cosine Series expansion

Suppose that $f(x)$ is a piecewise smooth function defined on an interval $0 < x < p$. Then, f has a *cosine series expansion*:

$$s_f(x) \triangleq a_0 + \sum_{n=1}^{\infty} a_n \cos\left(\frac{n\pi}{p}x\right), \tag{2.2.13}$$

where

$$a_0 \triangleq \frac{1}{p}\int_0^p f(x)dx, \tag{2.2.14}$$

$$a_n \triangleq \frac{2}{p}\int_0^p f(x)\cos\left(\frac{n\pi}{p}x\right)dx, \quad n = 1, 2, \ldots . \tag{2.2.15}$$

Theorem 2.2.5: Sine Series expansion

Suppose that $f(x)$ is a piecewise smooth function defined on an interval $0 < x < p$. Then, f has a *sine series expansion*:

$$s_f(x) \triangleq \sum_{n=1}^{\infty} b_n \sin\left(\frac{n\pi}{p}x\right), \tag{2.2.16}$$

where

$$b_n \triangleq \frac{2}{p}\int_0^p f(x)\cos\left(\frac{n\pi}{p}x\right)dx, \quad n = 1, 2, \dots . \tag{2.2.17}$$

2.2.2. Discrete Fourier analysis

By now, it is clear that Fourier analysis is an essential tool for decomposing complex signals in easily manageable trigonometric or complex exponential functions. However, in realistic applications, functions are measured over discrete sets of values, and hence they are usually represented by sequences of values. To analyze these discrete functions, we will introduce and use the Discrete Fourier Transform, along with some modified versions usable when some requirements are met.
The following dissertation is based on Asmar [2010] and Bull and Zhang [2021].

Discrete Fourier Transform

Given a discrete-time signal $x = x_n$ defined over the domain $n = 0, 1, \dots, N-1$, we define the N-point (direct) *discrete Fourier transform* of x, abbreviated DFT, by:

$$X_k = \mathrm{DFT}\{x_n\} \triangleq \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x_n e^{j2\pi n\frac{k}{N}}, \quad k = 0, 1, \dots, N-1. \tag{2.2.18}$$

Thus, the DFT of an N-length signal $x = x_n$ is an N-length complex signal X_k, where k is the index of the *frequency bin*. We will also use the notation $X_k = \mathcal{F}_N(x)$.
The inverse discrete Fourier transform (iDFT) is defined as

$$x_n = \mathrm{iDFT}\{X_k\} \triangleq \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} X_k e^{-j2\pi n\frac{k}{N}}, \quad n = 0, 1, \dots, N-1. \tag{2.2.19}$$

We will also use the notation $x_n = \mathcal{F}_N^{-1}(X)$.

Discrete Cosine Transform

For a one-dimensional array x_n, the *Discrete Cosine Transform* (DCT), in its most popular form (DCT-II), is given by

$$X_k = \sqrt{\frac{2}{N}} \varepsilon_k \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi k}{N}\left(n + \frac{1}{2}\right)\right), \tag{2.2.20}$$

where

$$\varepsilon_k = \begin{cases} \frac{1}{\sqrt{2}}, & k = 0, \\ 1, & \text{otherwise.} \end{cases}$$

Similarly, the inverse DCT (also known as DCT-III), is defined by

$$x_n = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \varepsilon_k X_k \cos\left(\frac{\pi n}{N}\left(k + \frac{1}{2}\right)\right). \tag{2.2.21}$$

The DCT is derived by performing an even extension y_n of the original signal x_n and then applying the DFT to y_n, which corresponds to the DCT of x_n. In case of DCT-II, the even extension is performed as follows:

$$y_n \triangleq \begin{cases} x_n, & 0 \leqslant n \leqslant N-1, \\ x_{2N-1-n}, & N \leqslant n \leqslant 2N-1. \end{cases} \tag{2.2.22}$$

An example is shown in Fig. 2.2.2. The other variants of the DCT perform the even extension differently, sometimes not duplicating the value x_{N-1}.

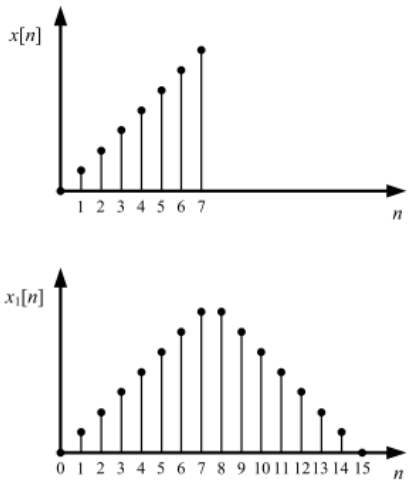


Figure 2.2.2: Symmetrical signal extension for the DCT. Source: Bull and Zhang [2021].

Using Eq. (2.2.20), we can see that the DCT basis functions are given by

$$a(k,n) \triangleq \sqrt{\frac{2}{N}}\varepsilon_k \cos\left(\frac{\pi k}{N}\left(n+\frac{1}{2}\right)\right), \quad 0 \leqslant k, \;\; n \leqslant N-1.$$

For $N = 16$, the basis functions for DCT-II are shown in Fig. 2.2.3.

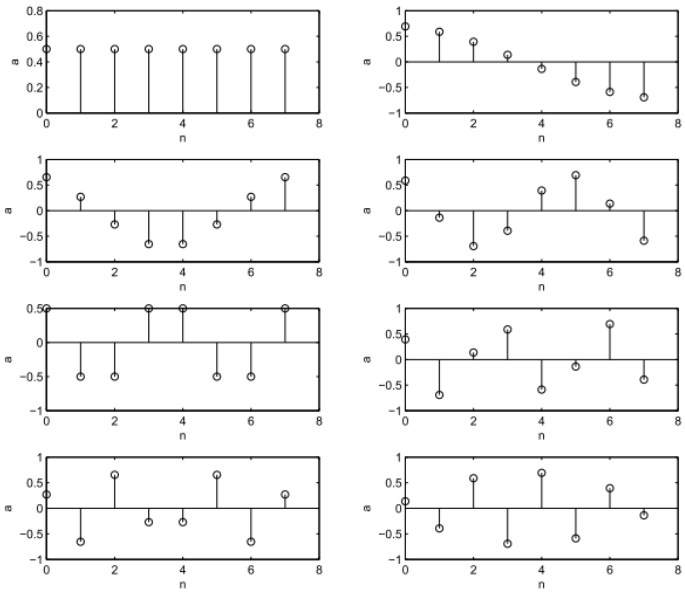


Figure 2.2.3: Basis functions for DCT-II for $N = 16$. Source: Bull and Zhang [2021].

2.2.3. Separation of variables

The method of separation of variables is used to solve a wide range of linear partial differential equations with boundary and initial conditions, including the homogeneous wave equation.

Consider the acoustic wave propagation problem (2.1.7) in a rectangular domain of dimensions L_x, L_y, L_z. We'll first consider the unviscid case ($\alpha = 0$). The acoustic wave equation (2.1.1), employing Cartesian coordinates, becomes

$$\frac{\partial^2 p(\underline{x},t)}{\partial t^2} - c^2\left(\frac{\partial^2 p}{\partial x^2}+\frac{\partial^2 p}{\partial y^2}+\frac{\partial^2 p}{\partial z^2}\right) = f(\underline{x},t),$$

$$t \in \mathbb{R}^+, \;\; 0 < x < L_x, \;\; 0 < y < L_y, \;\; 0 < z < L_z. \quad (2.2.23)$$

The homogeneous Neumann boundary conditions, in Cartesian coordinates, reduce to

$$\begin{cases} \frac{\partial p(\underline{x},t)}{\partial x}\big|_{x=0} = 0, \quad \frac{\partial p(\underline{x},t)}{\partial x}\big|_{x=L_x} = 0, & 0 \leqslant y \leqslant L_y, \quad 0 \leqslant z \leqslant L_z, \quad t \geqslant 0, \\ \frac{\partial p(\underline{x},t)}{\partial y}\big|_{y=0} = 0, \quad \frac{\partial p(\underline{x},t)}{\partial y}\big|_{y=L_y} = 0, & 0 \leqslant x \leqslant L_x, \quad 0 \leqslant z \leqslant L_z, \quad t \geqslant 0, \\ \frac{\partial p(\underline{x},t)}{\partial z}\big|_{z=0} = 0, \quad \frac{\partial p(\underline{x},t)}{\partial z}\big|_{z=L_z} = 0, & 0 \leqslant x \leqslant L_x, \quad 0 \leqslant y \leqslant L_y, \quad t \geqslant 0. \end{cases} \tag{2.2.24}$$

We want to solve the associated homogeneous equation:

$$\frac{\partial^2 p(\underline{x},t)}{\partial t^2} = c^2 \left(\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} \right), \qquad t \in \mathbb{R}^+, \;\; 0 < x < L_x, \;\; 0 < y < L_y, \;\; 0 < z < L_z. \tag{2.2.25}$$

Following the suggestion of Asmar [2010], we apply separation of variables, obtaining

$$p(x,y,z,t) = X(x)Y(y)Z(z)T(t).$$

Thus, the PDE in Eq. (2.2.25) can be rewritten as four independent ODEs:

$$\begin{cases} X'' + \mu^2 X = 0, & \frac{dX}{dx}\big|_0 = 0, & \frac{dX}{dx}\big|_{L_x} = 0, \\ Y'' + \nu^2 Y = 0, & \frac{dY}{dy}\big|_0 = 0, & \frac{dY}{dy}\big|_{L_y} = 0, \\ Z'' + \xi^2 Z = 0, & \frac{dZ}{dz}\big|_0 = 0, & \frac{dZ}{dz}\big|_{L_z} = 0, \\ T'' + c^2 k^2 T = 0, & k^2 = \mu^2 + \nu^2 + \xi^2. & \end{cases}$$

A guess solution to the last four differential equations is

$$X_m(x) = \cos\left(\frac{m\pi}{L_x}x\right),$$

$$Y_n(y) = \cos\left(\frac{n\pi}{L_y}y\right),$$

$$Z_p(z) = \cos\left(\frac{p\pi}{L_z}z\right),$$

$$T_{mnp}(t) = \widetilde{P}_{mnp}\, e^{j\omega_{mnp}t},$$

where we put

$$\omega_{mnp} = c\, k_{mnp} = c\pi\sqrt{\frac{m^2}{L_x^2} + \frac{n^2}{L_y^2} + \frac{p^2}{L_z^2}}, \tag{2.2.26}$$

with $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+$, $p \in \mathbb{N}^+$. The ω_{mnp}'s are called the *characteristic frequencies*, or *natural frequencies*, of the volume, while the k_{mnp}'s are the *natural wavenumbers*.
We can derive a product solution satisfying Eq. (2.1.1) and Eq. (2.2.24):

$$\widetilde{p}(\underline{x}, t) = \widetilde{P}_{mnp}\, \widetilde{p}_{mnp}(\underline{x}, t),$$

where the functions $\widetilde{p}_{mnp}(\underline{x}, t)$ are called the *normal modes* of the three-dimensional wave equation, expressed as a multiplication between the *mode shapes* $p_{mnp}(\underline{x})$ and the time evolution term

$$\begin{aligned} \widetilde{p}_{mnp}(\underline{x}, t) &\triangleq p_{mnp}(\underline{x})\, e^{j\omega_{mnp}t}, \\ p_{mnp}(\underline{x}) &= \cos\left(\frac{m\pi}{L_x}x\right) \cos\left(\frac{n\pi}{L_y}y\right) \cos\left(\frac{p\pi}{L_z}z\right), \end{aligned} \tag{2.2.27}$$

while $\widetilde{P}_{mnp}$ are the *modal coefficients* which depend on the initial conditions (they can be interpreted as phasors):

$$\widetilde{P}_{mnp} \triangleq \left|\widetilde{P}_{mnp}\right| e^{j\angle\widetilde{P}_{mnp}} = P_{mnp}\, e^{j\phi_{mnp}}.$$

In Fig. 2.2.4 we depict the first mode shapes of the two-dimensional wave equation in a rectangular domain of dimensions $L_x = 1\,m$, $L_y = 1\,m$. We also specify the value of $k_{mn} = \omega_{mn}/c$ for each mode (m, n); notice that, as $L_x = L_y$, we have that $k_{mn} = k_{nm}$ for each $m, n \in \mathbb{N}^+$.

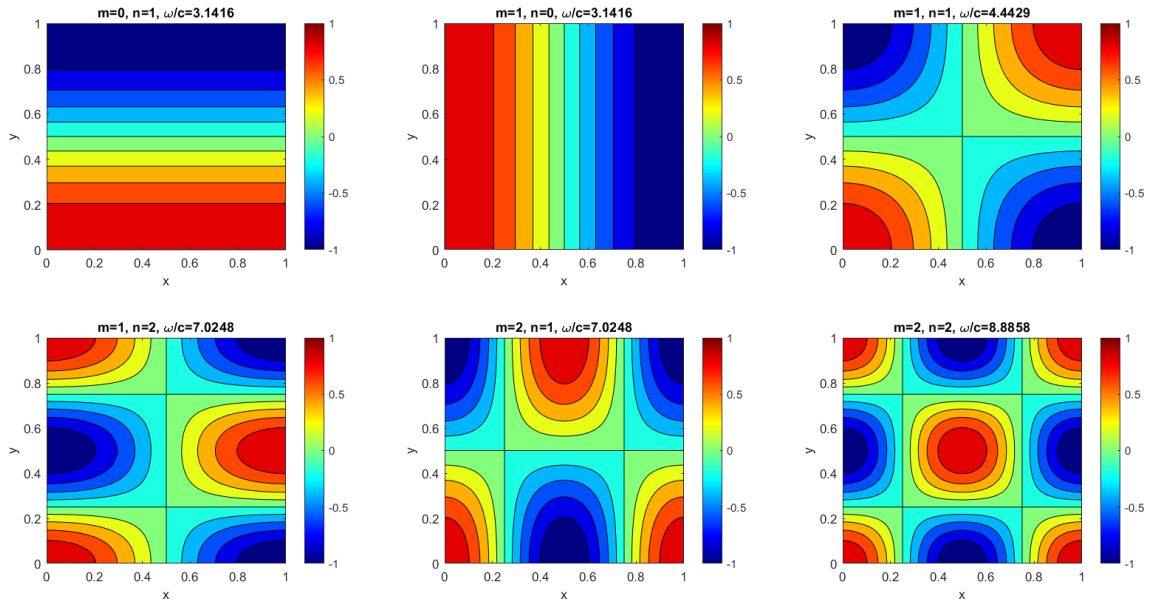


Figure 2.2.4: First mode shapes of the two-dimensional wave equation.

The general solution, known as *standing wave solution*, is

$$\widetilde{p}(\underline{x},t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{p=1}^{\infty} \widetilde{P}_{mnp} \cos\left(\frac{m\pi}{L_x}x\right)\cos\left(\frac{n\pi}{L_y}y\right)\cos\left(\frac{p\pi}{L_z}z\right)\, e^{j\omega_{mnp}t}.$$

The above equation represents a *triple Cosine Series expansion* with respect to the spatial coordinates x, y, z (generalization of Th. 2.2.4). Instead, the time dependence is expressed as a Fourier Series in complex form (Def. 2.2.2). However, as the pressure $p(\underline{x},t)$ is a real function, we can rewrite it in compact form (Def. 2.2.1):

$$p(\underline{x},t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{p=1}^{\infty} P_{mnp} \cos\left(\frac{m\pi}{L_x}x\right)\cos\left(\frac{n\pi}{L_y}y\right)\cos\left(\frac{p\pi}{L_z}z\right)\, \cos\left(\omega_{mnp}t + \phi_{mnp}\right). \quad (2.2.28)$$

Let's now consider the viscous problem. It can be proven that the mode shapes are not affected by the viscous damping, while the normal modes are characterized by an exponential decay:

$$\widetilde{p}_{mnp}(\underline{x},t) = p_{mnp}(\underline{x})\, e^{-\alpha t}\, e^{j\omega_{mnp}t},$$

The separation of variables method alone cannot be used to solve the inhomogeneous wave equation, but it provides the basis for the formulation of the modal superposition approach which we'll present in Subsec. 2.4.3.

2.3. Finite difference method for the solution of the wave equation

In this section, we introduce a popular numerical method for the solution of ordinary and partial differential equations: the *finite difference method*. The following dissertation is based on Fornberg [1988].
Consider the following discretization of the x-axis with constant step size dh:

$$x_n = n \cdot dh, \quad n \in \mathbb{Z},$$

where x_n is the n-th grid point.
We can approximate a derivative $f^{(m)}(x)$ of a sufficiently smooth function f by defining a set of $2N+1$ grid points $\{x_n\}_{n=-N}^{N}$ and employing a *centered finite difference scheme*

δ_x (also known as *centered finite difference stencil* or *centered difference operator*):

$$\left.\frac{d^m f}{dx^m}\right|_{x=x_0} \approx \delta_x f(x) \triangleq \frac{\sum_{n=-N}^{N} \delta_n f(x_n)}{dh^m},$$

where δ_n is the n-th weight (also known as finite difference coefficient). Some weights for the centered difference operator are reported in Tab. 2.3.1, for order of derivation one and two, also specifying the order of accuracy (see Subsec. 2.3.1).

Derivative	Accuracy	-3	-2	-1	0	1	2	3
$f^{(1)}$	2			-1/2	0	1/2		
	4		1/12	-2/3	0	2/3	-1/12	
	6	-1/60	3/20	-3/4	0	3/4	-3/20	1/60
$f^{(2)}$	2			1	-2	1		
	4		-1/12	4/3	-5/2	4/3	-1/12	
	6	1/90	-3/20	3/2	-49/18	3/2	-3/20	1/90

Table 2.3.1: Finite difference coefficients. Source: Fornberg [1988].

For example, to approximate the first derivative ($m = 1$) with second order accuracy, we employ the following centered stencil with coefficients $\delta_{-1} = -1/2,\ \delta_0 = 0,\ \delta_1 = 1/2$:

$$\left.\frac{df}{dx}\right|_{x=x_0} \approx \frac{\sum_{n=-1}^{1} \delta_n f(x_n)}{dh} = \frac{f(x_1) - f(x_{-1})}{2\,dh}.$$

2.3.1. Accuracy of the finite difference stencil

Accuracy is a metric used to describe the proportionality between the step size and truncation error of a finite difference operator. For example, consider the following operator, known as *forward difference operator*, applied to a function $f(x)$:

$$\delta_{x+} f(x) \triangleq \frac{1}{dx}\left(f(x + dx) - f(x)\right),$$

where dx is the step size or increment.
Assuming $f(x)$ to be infinitely differentiable, and expanding $f(x + dx)$ in Taylor series about x, one has

$$\delta_{x+} f(x) = \frac{df}{dx} + \frac{dx}{2}\frac{d^2 f}{dx^2} + o(dx^2).$$

The operator δ_{x+} thus approximates the first derivative to an *accuracy* which depends on the first power of dx; as dx is made small, the difference approximation approaches the exact value of the derivative with an error proportional to dx. Such an approximation is thus often called first-order accurate.
In general, the better the accuracy of the approximation, the more adjacent values of the function f will be required (higher operator width). This leads to a trade-off between performance and accuracy (Bilbao [2009]).

2.3.2. An example of a FDTD scheme for the wave equation

We want to derive a finite difference scheme to solve the 1D unviscid acoustic wave equation (Cfr. Eq. 2.1.1):

$$\begin{cases} \frac{\partial^2 p}{\partial t^2} - c^2 \frac{\partial^2 p}{\partial x^2} = F(x,t), & t \in \mathbb{R}^+, \;\; x \in \mathbb{R}, \\ p(x,0) = p_0(x), & x \in \mathbb{R}. \end{cases} \tag{2.3.1}$$

We preliminarly introduce the notation: p_i^n represents an approximation to the solution of the wave equation at $x = i\,dh, t = n\,dt$, where dh is the spacing between adjacent grid points, and dt is the time step.
We approximate the second order time derivative with a centered second order finite difference:

$$\frac{\partial^2 p}{\partial t^2} \approx \frac{p_i^{n+1} - 2p_i^n + p_i^{n-1}}{dt^2}, \tag{2.3.2}$$

and, analogously, we approximate the second order space derivative:

$$\frac{\partial^2 p}{\partial x^2} \approx \frac{p_{i+1}^n - 2p_i^n + p_{i-1}^n}{dh^2}. \tag{2.3.3}$$

Substituting Eq. (2.3.2) and Eq. (2.3.3) in Eq. (2.3.1), we obtain

$$p_i^{n+1} = 2(1-\lambda^2)p_i^n + \lambda^2(p_{i+1}^n + p_{i-1}^n) - p_i^{n-1}, \tag{2.3.4}$$

where λ is a dimensionless parameter known as *Courant number*, defined as

$$\lambda \triangleq c\frac{dt}{dh}. \tag{2.3.5}$$

The scheme (2.3.4) is known as *Leap-Frog* (Quarteroni et al. [2007]); the solution may be updated, explicitly, at each time step n, from previously computed values at the previous two time steps. It is perhaps easiest to see the behavior of this algorithm through a

dependence plot showing the "footprint" of the scheme, shown in Fig. 2.3.1.
For the moment, it is assumed that the spatial domain of the problem is infinite; the analysis of boundary conditions is postponed to Sec. 3.3.

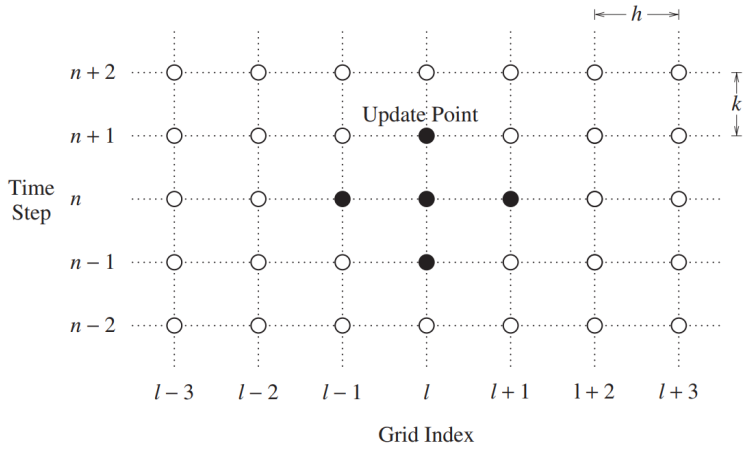


Figure 2.3.1: Computational footprint of the scheme in Eq. (2.3.4): a value of the grid function p_i^n at the update location $(i, n+1)$, as indicated, is updated using values at the previous two time steps. The set of active points of the scheme at this update location is indicated in black. Source: Bilbao [2009].

2.3.3. Review of the Leap-Frog method

In this section we analyze the finite difference scheme introduced in Sec. 2.3.2 from the point of view of the accuracy, stability, and convergence. We will also examine the impact of numerical dispersion on the accuracy of the finite difference method.
As a reference, we will consider the Leap-Frog scheme in Eq. (2.3.4) used to solve the 1D unviscid wave equation.
The following dissertation is based on Quarteroni et al. [2007], Bilbao [2009], and Smith [1985].

Consistency

The local truncation error of a numerical scheme is the residual that is generated by pretending the exact solution to satisfy the numerical method itself.
Denoting by p the solution of the exact problem in Eq. (2.3.1), and considering the FDTD scheme (2.3.4), we define the *local truncation error* τ_i^n at (x_i, t^n) as follows

$$\tau_i^n \triangleq \frac{p(x_i, t^{n+1}) - 2p(x_i, t^n) + p(x_i, t^{n-1})}{dt^2} - c^2 \frac{p(x_{i+1}, t^n) - 2p(x_i, t^n) + p(x_{i-1}, t^n)}{dh^2}.$$

The *truncation error* is

$$\tau(dt, dh) \triangleq \max_{i,n} |\tau_i^n|.$$

When $\tau(dt, dh)$ goes to zero as dt and dh tend to zero independently, the numerical scheme is said to be *consistent*.
Moreover, we say that it is of *order* p in time and of *order* q in space (for suitable integers p and q), if, for a sufficiently smooth solution of the exact problem, we have

$$\tau(dt, dh) = \mathcal{O}(dt^p + dh^q).$$

Scheme (2.3.4) is second order accurate in both time and space, i.e., $\mathcal{O}(dt^2 + dh^2)$ (Quarteroni et al. [2007]).
Finally, we say that a numerical scheme is *convergent* if

$$\lim_{dt,dh \to 0} \max_{i,n} |p(x_i, t^n) - p_i^n| = 0.$$

Stability

A numerical method for a hyperbolic problem (linear or nonlinear) is said to be *stable*, for any time T, if there exist two constants $C_T > 0$ (possibly depending on T) and $\delta_0 > 0$, such that

$$\left\| \underline{p}^n \right\|_\Delta \leqslant C_T \left\| \underline{p}^0 \right\|_\Delta ,$$

for any n such that $ndt \leqslant T$ and for any dt, dx such that $0 < dt \leqslant \delta_0$, $0 < dh \leqslant \delta_0$. We have denoted by $\|\cdot\|_\Delta$ a suitable discrete norm as, for instance, one of those indicated below

$$\left\| \underline{p} \right\|_{\Delta,r} = \left(dh \sum_{j=-\infty}^{\infty} |p_j|^r \right)^{\frac{1}{r}} \quad \text{for } r = 1, 2,$$

$$\left\| \underline{p} \right\|_{\Delta,\infty} = \sup_j |p_j|.$$

Note that $\|\cdot\|_{\Delta,r}$ is an approximation of the norm of $L^r(\mathbb{R})$ defined in Def. 2.3.1.

Definition 2.3.1: L^p norms

Consider a vector $\underline{x}$ of components $\{x_i\}$ in a normed space, e.g., $\mathbb{R}^n$. According to Quarteroni et al. [2007], the *p-norm* or L^p norm is defined as

$$\|\underline{x}\|_p \triangleq \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}, \quad \text{for } 1 \leqslant p < \infty.$$

The most common p-norms are L^1 norm, known as *taxicab norm*, and L^2 norm, known as *Euclidean norm*.
Notice that the limit as p goes to infinity of $\|\underline{x}\|_p$ exists, is finite, and equals the maximum module of the components of $\underline{x}$. Such a limit defines in turn a norm, called the *infinity norm* (or *maximum norm*), and is known as L^∞ norm:

$$\|\underline{x}\|_\infty = \max_{1 \leqslant i \leqslant n} |x_i|.$$

CFL condition

Courant, Friedrichs and Lewy have shown that a necessary and sufficient condition for any explicit scheme (including the Leap-Frog scheme (2.3.4)) to be stable is that the time and space discretization steps must obey the following condition:

$$|\lambda| = \left|c\frac{dt}{dh}\right| \leqslant \lambda_{max}, \tag{2.3.6}$$

which is known as the *CFL condition*, and λ_{max} is a given constant. In particular, for the Leap-Frog scheme we have that $\lambda_{max} = 1$, i.e.,

$$|\lambda| = \left|c\frac{dt}{dh}\right| \leqslant 1, \tag{2.3.7}$$

This has an interesting geometrical interpretation, as illustrated in Fig. 2.3.2. At a given update point, the value of the solution to the continuous-time wave equation depends on values traveling on solution characteristics (solid dark lines), defined by $x - ct =$ constant and $x + ct =$ constant. The cone of dependence of the solution may be illustrated as the interior of this region (in grey). The scheme in Eq. (2.3.4) at the update point possesses a numerical cone of dependence, illustrated by black points, and bounded by dashed black lines.

- At left, $dh = c\,dt$, and the characteristics align exactly with values on the grid; in this case, the numerical solution is exact.

- At center, a value $dh < c\,dt$ is chosen, violating stability condition in Eq. (2.3.7); the numerical cone of dependence lies strictly within the region of the dependence of the wave equation, and the scheme cannot compute correctly the solution.
- At right, with a choice of $dh > c\,dt$, the numerical cone of dependence of the scheme includes that of the wave equation, and the scheme is stable.

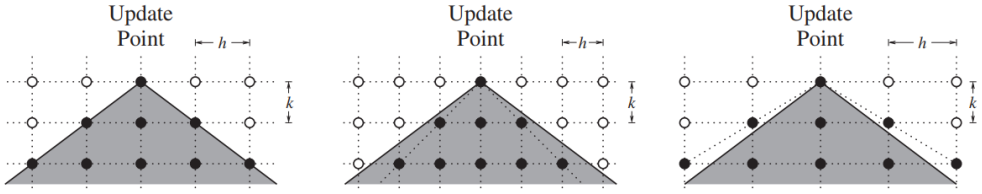


Figure 2.3.2: A geometrical interpretation of the Courant-Friedrichs-Lewy stability condition. Source: Bilbao [2009].

Von Neumann stability analysis

We plug in the Leap-Frog scheme (2.3.4) a general solution of the form

$$p_i^n = z^n e^{ji\beta dh} \tag{2.3.8}$$

where $z = e^{sdt}$, where $s = \sigma + j\omega$ is a complex frequency variable, and β is a real wavenumber. This may be thought as a wavelike solution, which corresponds to the sampled propagating wave solution:

$$p(x,t) = e^{st+j\beta x} = e^{\sigma t}e^{j(\beta x+\omega t)}. \tag{2.3.9}$$

By substituting Eq. (2.3.8) in Eq. (2.3.4), we obtain

$$z^{n+1}e^{ji\beta dh} = 2(1-\lambda^2)z^n e^{ji\beta dh} + \lambda^2\left(z^n e^{j(i+1)\beta dh} + z^n e^{j(i-1)\beta dh}\right) - z^{n-1}e^{ji\beta dh}.$$

By dividing by z^n we get

$$z = 2(1-\lambda^2) + \lambda^2\left(e^{j\beta dh} + e^{-j\beta dh}\right) - z^{-1}$$

$$z - 2\left(1 - \lambda^2(1-\cos(\beta dh))\right) + z^{-1} = 0,$$

from which the following characteristic equation results:

$$z + 2\left(2\lambda^2\sin^2\left(\frac{1}{2}\beta dh\right) - 1\right) + z^{-1} = 0. \tag{2.3.10}$$

Eq. (2.3.10) is analogous to the continuous characteristic equation:

$$s^2 + c^2\beta^2 = 0,$$

which is obtained by replacing Eq. (2.3.9) in Eq. (2.3.1).
The roots of the FDTD characteristic equation in Eq. (2.3.10) are given by

$$z_{\pm} = 1 - 2\lambda^2 \sin^2\left(\frac{1}{2}\beta dh\right) \pm \sqrt{\Delta},$$

where Δ is defined as

$$\Delta \triangleq \left(1 - 2\lambda^2 \sin^2(\beta dh/2)\right)^2 - 1,$$

while the roots of the characteristic equation of the wave equation itself are

$$s_{\pm}(\beta) = \pm jc\beta. \tag{2.3.11}$$

We notice that, in both cases, there are two solutions, representing the propagation of the wave in opposite directions.
In order for a solution such as Eq. (2.3.8) to behave as a solution to the wave equation, one should have $|z| \leqslant 1$ for any value of the wavenumber β, otherwise such a solution will experience exponential growth or damping. From inspection of the FDTD scheme characteristic equation in Eq. (2.3.10), one may deduce that the roots are complex conjugates of magnitude less than 1 when Δ is non-positive:

$$\begin{aligned}
&\left(1 - 2\lambda^2 \sin^2(\beta dh/2)\right)^2 - 1 \leqslant 0,\\
&1 + \left(2\lambda^2 \sin^2(\beta dh/2)\right)^2 - 2 \cdot 2\lambda^2 \sin^2(\beta dh/2) - 1 \leqslant 0,\\
&\left(2\lambda^2 \sin^2(\beta dh/2)\right)^2 - 2 \cdot 2\lambda^2 \sin^2(\beta dh/2) \leqslant 0,
\end{aligned}$$

that is

$$\lambda^2 \sin^2(\beta dh/2) \leqslant 1,$$

which must be satisfied for any possible value of β, hence we obtain

$$\begin{aligned}
&\max\left\{\lambda^2 \sin^2(\beta dh/2)\right\} \leqslant 1,\\
&|\lambda| \leqslant 1,
\end{aligned}$$

recalling that $0 \leqslant \sin^2(\beta dh/2) \leqslant 1$. This condition is the CFL condition introduced in Eq. (2.3.7).

Note that for λ slightly greater than unity, the condition $\lambda^2 \sin^2(\beta dh/2) \leqslant 1$ will be violated near the maximum of the squared sine, which occurs at the wavenumber $\beta = \pi/dh$, corresponding to a wavelength of $2\,dh$, which is the shortest wavelength which may be represented on a grid of spacing dh (spatial Nyquist wavelength).

Dissipation

The Leap-Frog scheme (2.3.4) is exactly conservative, in a numerical sense, over the unbounded spatial domain $\mathcal{D} = \mathbb{Z}$. This is true regardless of the values chosen for the time step dt and the grid spacing dh, and whether or not the scheme is stable (Bilbao [2009]).

Dispersion

Consider the Leap-Frog scheme (2.3.4) under the stability condition in Eq. (2.3.7). Using $z = e^{j\omega dt}$, the characteristic equation in Eq. (2.3.10) may be written as

$$e^{j\omega dt} + 2\left(2\lambda^2 \sin^2\left(\frac{1}{2}\beta dh\right) - 1\right) + e^{-j\omega dt} = 0,$$

$$2\left(2\lambda^2 \sin^2\left(\frac{1}{2}\beta dh\right) - 1\right) = -2\cos(\omega dt),$$

or

$$\pm\lambda \sin\left(\frac{1}{2}\beta dh\right) = \sin\left(\frac{1}{2}\omega dt\right),$$

that is

$$\frac{1}{2}\omega dt = \pm \arcsin\left(\lambda \sin\left(\frac{1}{2}\beta dh\right)\right),$$

which results in the *dispersion relation* for the FDTD scheme:

$$\omega = \pm\frac{2}{dt}\arcsin\left(\lambda \sin\left(\frac{1}{2}\beta dh\right)\right). \tag{2.3.12}$$

The above equation relates the frequency ω and the wavenumber β in complete analogy with the dispersion relation for the wave equation itself, obtained imposing $\sigma = 0$ in the characteristic equation (2.3.11):

$$\omega(\beta) = \pm c\beta. \tag{2.3.13}$$

To compare the two dispersion relations, we define the phase velocity v_ϕ and the group velocity v_g:

$$v_\phi \triangleq \frac{\omega}{\beta}, \tag{2.3.14a}$$

$$v_g \triangleq \frac{d\omega}{d\beta}. \tag{2.3.14b}$$

In the case under consideration, these velocities are constant and equal to c for all wavenumbers. Instead, for the Leap-Frog scheme (2.3.4), these velocities are, in general, functions of the wavenumber: different wavelengths travel at different speeds, making the scheme dispersive. Dispersion leads to a progressive distortion of a pulse as it travels, as illustrated in Fig. 2.3.3. This type of anomalous behavior is purely a result of the discretization, and it is known as numerical dispersion; it should be carefully distinguished from physical dispersion of a model problem itself, which will arise when systems are subject to stiffness. It is also true that the numerical velocities will also depend on the

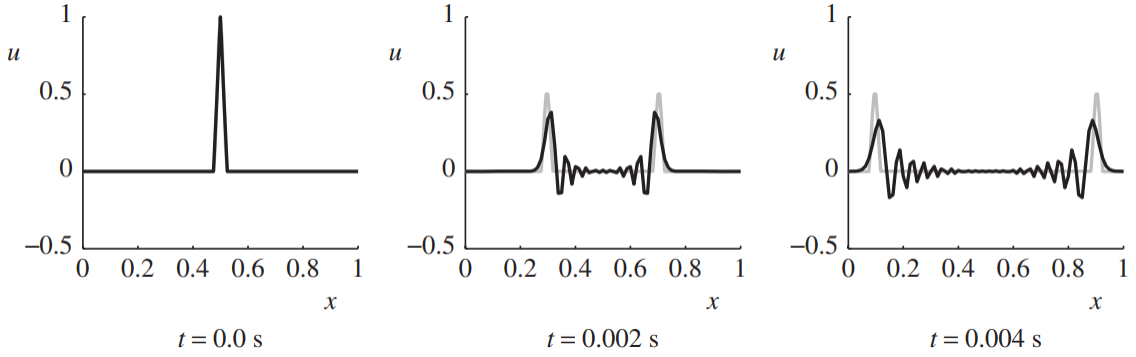


Figure 2.3.3: Numerical dispersion. Output from the scheme in Eq. (2.3.4) for the wave equation, at times as indicated, for $\lambda = 1$ (in grey) and $\lambda = 0.5$ (in black). c is chosen as 100, the sample rate is 16000 Hz, and the initial conditions are set according to a narrow cosine distribution, of width 1/40. Notice that, for $\lambda < 1$, the higher-frequency components lag the wavefront, illustrating the phase velocity characteristic of the scheme. Notice also that the gross speed of the wave packet is slower as well, illustrating the group velocity characteristic. Source: Bilbao [2009].

choice of the parameter λ. The velocity curves, as functions of frequency $f = \omega/(2\pi)$ for different values of λ, are shown in Fig. 2.3.4.

Now, consider the very special case of $\lambda = 1$. Under this condition, the dispersion relation in Eq. (2.3.12) reduces to

$$\omega = \pm\frac{2}{dt}\arcsin\left(\sin\left(\frac{1}{2}\beta dh\right)\right) = \pm\frac{2}{dt}\frac{1}{2}\beta dh = \pm\frac{\beta dh}{dt} = \pm c\beta.$$

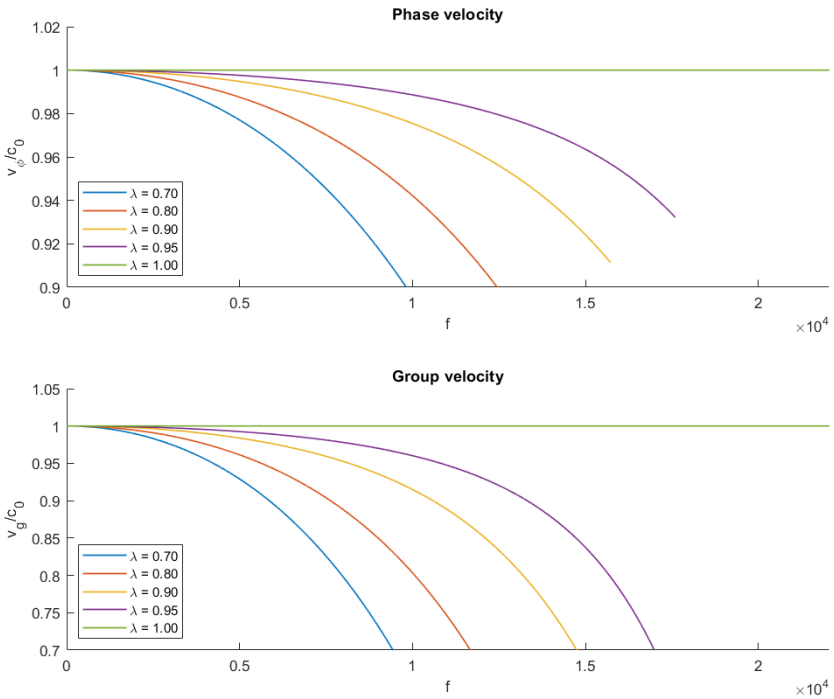


Figure 2.3.4: Numerical phase velocity (up) and group velocity (down) as a function of frequency f, normalized by the model velocity c_0, for the Leap-Frog scheme (2.3.4), for a variety of values of λ, as indicated. The sample rate is chosen as 44100 Hz.

Now, the numerical dispersion relation is exactly the same of the continuous one, and the phase and group velocities are both equal to c, independently of β. Thus, there is no numerical dispersion for this choice of λ. In fact, as a rule of thumb, for any FDTD explicit numerical method, the best numerical behavior (i.e., the least numerical dispersion) is achieved when the stability condition is satisfied as near to equality as possible (Bilbao [2009]).

The bandwidth of the scheme can be computed from Eq. (2.3.12). Considering only the positive solution, using $\omega = 2\pi f$, defining the sampling frequency $f_s = 1/dt$, and maximizing the term $\sin(\beta dh/2)$, the maximum frequency f_{max} will be given by

$$2\pi f_{max} = 2 f_s \arcsin(\lambda),$$

$$f_{max} = \frac{f_s}{\pi} \arcsin(\lambda).$$

2.4. Solution techniques for the harmonic oscillator

In this section, we discuss the discretization of the classical harmonic oscillator equation, which includes the inhomogeneous and damped cases. Finally, we will introduce the modal superposition approach, which demonstrates that each mode of the wave equation can be represented as an independent harmonic oscillator. The following dissertation is based on Cieśliński [2011] and Asmar [2010].

2.4.1. Harmonic oscillator equation

The equation of motion of a forced harmonic oscillator with a driving force $f(t) \in \mathbb{R}$ is

$$\begin{cases} \ddot{x}(t) + \omega^2 x(t) = f(t), \quad t > 0, \\ x(0) = x_0, \\ v(0) = \frac{dx}{dt}\big|_{t=0} = v_0, \end{cases} \tag{2.4.1}$$

where $x = x(t) \in \mathbb{R}$, $\omega \in \mathbb{R}^+$. It is convenient to represent Eq. (2.4.1) as the following first order system

$$\begin{cases} \dot{x}(t) = v(t), \quad t > 0, \\ \dot{v}(t) = -\omega^2 x(t) + f(t), \quad t > 0, \\ x(0) = x_0, \\ v(0) = \frac{dx}{dt}\big|_{t=0} = v_0, \end{cases} \tag{2.4.2}$$

where v represents the velocity.

Analytical solution

The general solution to the harmonic oscillator equation in Eq. (2.4.1), for the case of constant driving force, is given by

$$\begin{aligned} x(t) &= c_1 e^{j\omega t} + c_2 e^{-j\omega t} + \frac{f}{\omega^2}, \\ v(t) &= j\omega \left(c_1 e^{j\omega t} - c_2 e^{-j\omega t} \right), \end{aligned} \tag{2.4.3}$$

where $c_1, c_2 \in \mathbb{R}$.

Numerical solution

We can consider Eq. (2.4.3) at a given instant $t_n \in \mathbb{R}$ by setting $x_n := x(t_n)$, obtaining the discretized solution

$$\begin{aligned} x_n &= c_1 e^{j\omega t_n} + c_2 e^{-j\omega t_n} + \frac{f}{\omega^2}, \\ v_n &= j\omega \left(c_1 e^{j\omega t_n} - c_2 e^{-j\omega t_n} \right). \end{aligned} \tag{2.4.4}$$

From Eq. (2.4.4) we obtain

$$\begin{aligned} c_1 e^{j\omega t_n} &= \frac{1}{2} \left(x_n - \frac{f}{\omega^2} - j\frac{v_n}{\omega} \right), \\ c_2 e^{-j\omega t_n} &= \frac{1}{2} \left(x_n - \frac{f}{\omega^2} + j\frac{v_n}{\omega} \right). \end{aligned} \tag{2.4.5}$$

Denoting $dt_n = t_{n+1} - t_n$ (the time step, in general variable) and evaluating Eq. (2.4.4) at t_{n+1}, we obtain

$$\begin{aligned} x_{n+1} &= c_1 e^{j\omega t_n + j\omega dt_n} + c_2 e^{-j\omega t_n - j\omega dt_n} + \frac{f}{\omega^2}, \\ v_{n+1} &= j\omega \left(c_1 e^{j\omega t_n + j\omega dt_n} - c_2 e^{-j\omega t_n - j\omega dt_n} \right), \end{aligned} \tag{2.4.6}$$

where c_1 and c_2 depend on the initial conditions.

Proposition 2.4.1

For any $n = 0, 1, \ldots$, the solution of Eq. (2.4.2) is given by

$$\begin{pmatrix} x_{n+1} \\ v_{n+1} \end{pmatrix} = \begin{pmatrix} \cos(\omega dt_n) & \sin(\omega dt_n)/\omega \\ -\omega \sin(\omega dt_n) & \cos(\omega dt_n) \end{pmatrix} \begin{pmatrix} x_n \\ v_n \end{pmatrix} + \begin{pmatrix} \frac{2}{\omega^2} \sin^2\left(\frac{\omega dt_n}{2}\right) f \\ \frac{1}{\omega} \sin(\omega dt_n) f \end{pmatrix}, \tag{2.4.7}$$

where dt_n is an arbitrary variable time step (in particular, we can take $dt_n = dt = \text{const}$).

Proof: It is enough to substitute Eq. (2.4.5) into Eq. (2.4.6). This result holds for any value of the coefficients c_1, c_2. □

Proposition 2.4.2

The solution in Eq. (2.4.7) of the harmonic oscillator equation with constant driving

force f is equivalent to the system

$$\begin{aligned} \frac{x_{n+1}-x_n}{\delta_n} &= \frac{1}{2}(v_{n+1}+v_n), \\ \frac{v_{n+1}-v_n}{\delta_n} &= -\frac{1}{2}\omega^2(x_{n+1}+x_n)+f, \end{aligned} \tag{2.4.8}$$

where

$$\delta_n = \frac{2}{\omega}\tan\left(\frac{\omega dt_n}{2}\right).$$

Proof: From Eq. (2.4.7) we compute:

$$\begin{aligned} x_{n+1}+x_n &= (1+\cos(\omega dt_n))x_n + \frac{1}{\omega}\sin(\omega dt_n)v_n + \frac{1}{\omega^2}(1-\cos(\omega dt_n))f, \\ x_{n+1}-x_n &= -(1-\cos(\omega dt_n))x_n + \frac{1}{\omega}\sin(\omega dt_n)v_n + \frac{1}{\omega^2}(1-\cos(\omega dt_n))f, \\ v_{n+1}+v_n &= -\omega(\sin(\omega dt_n))x_n + (1+\cos(\omega dt_n))v_n + \frac{1}{\omega}\sin(\omega dt_n)f, \\ v_{n+1}-v_n &= -\omega(\sin(\omega dt_n))x_n - (1-\cos(\omega dt_n))v_n + \frac{1}{\omega}\sin(\omega dt_n)f. \end{aligned}$$

Hence

$$\begin{aligned} \omega\sin\left(\frac{\omega dt_n}{2}\right)(x_{n+1}+x_n) + \cos\left(\frac{\omega dt_n}{2}\right)(v_{n+1}-v_n) &= \frac{2}{\omega}\sin\left(\frac{\omega dt_n}{2}\right)f, \\ \omega\cos\left(\frac{\omega dt_n}{2}\right)(x_{n+1}-x_n) &= \sin\left(\frac{\omega dt_n}{2}\right)(v_{n+1}+v_n), \end{aligned}$$

which is equivalent to Eq. (2.4.8). □

Proposition 2.4.3

If the time step is constant, i.e., $dt_n = dt\ \forall n = 0, 1, \ldots$, then Eq. (2.4.7) can be rewritten in the following equivalent form

$$\begin{aligned} x_{n+1} - 2\cos(\omega dt)x_n + x_{n-1} &= \frac{2}{\omega^2}(1-\cos(\omega dt))f, \\ v_n &= \frac{\omega}{\sin(\omega dt)}(x_{n+1}-\cos(\omega dt)x_n) - \frac{1}{\omega}\tan\left(\frac{\omega dt}{2}\right)f. \end{aligned} \tag{2.4.9}$$

Proof: From Eq. (2.4.7) we get:

$$\begin{aligned} v_n &= \frac{\omega}{\sin(\omega dt)}(x_{n+1} - \cos(\omega dt)x_n) - \frac{1}{\omega}\tan\left(\frac{\omega dt}{2}\right) f \\ &= -\omega\sin(\omega dt)x_{n-1} + \cos(\omega dt)v_{n-1} + \frac{1}{\omega}\sin(\omega dt)f, \\ v_{n-1} &= \frac{\omega}{\sin(\omega dt)}(x_n - \cos(\omega dt)x_{n-1}) - \frac{1}{\omega}\tan\left(\frac{\omega dt}{2}\right) f, \end{aligned} \tag{2.4.10}$$

for which it follows Eq. (2.4.9) for v_n. Eliminating v_n and v_{n-1} from the system (2.4.10) we get the first equation of Eq. (2.4.9). □

2.4.2. Damped harmonic oscillator equation

If viscous damping is considered, the equation of motion for the harmonic oscillator with constant driving force f is given by

$$\begin{cases} \ddot{x}(t) + 2\alpha\dot{x}(t) + \omega_0^2 x(t) = f, \\ x(0) = x_0, \\ v(0) = \frac{dx}{dt}\big|_{t=0} = v_0. \end{cases} \tag{2.4.11}$$

We can reduce it to the homogeneous harmonic oscillator without damping.

Proposition 2.4.4

Consider the transformation

$$X = e^{\alpha t}(x - x_e), \tag{2.4.12}$$

where x_e is the steady-state solution, obtained when all the transients are extinguished, i.e.,

$$\omega_0^2 x_e = f \quad \rightarrow \quad x_e = f\omega_0^{-2},$$

Replacing Eq. (2.4.12) into Eq. (2.4.11) we get

$$\begin{cases} \ddot{X} + \omega^2 X = 0, \\ X(0) = X_0, \\ V(0) = \frac{dX}{dt}\big|_{t=0} = V_0, \end{cases} \tag{2.4.13}$$

where $\omega^2 = \omega_0^2 - \alpha^2$ is the *damped natural frequency*.

Proof: Straightforward computation. □

Defining the dummy variables $p = m\dot{x} = mv$ and $P = m\dot{X}$ with $m \in \mathbb{R}\backslash 0$, we can express P in terms of x, p, namely,

$$P = e^{\alpha t}(p + m\alpha(x - x_e)).$$

Analytical solution

The general solution to the forced damped harmonic oscillator equation, which is transformed to the undamped harmonic oscillator equation (2.4.13) with $f = 0$, is given by

$$\begin{aligned} X(t) &= c_1 e^{j\omega t} + c_2 e^{-j\omega t}, \\ V(t) &= j\omega \left(c_1 e^{j\omega t} - c_2 e^{-j\omega t}\right), \end{aligned} \tag{2.4.14}$$

where $c_1, c_2 \in \mathbb{R}$.

Numerical solution

Recalling Eq. (2.4.7), the solution of Eq. (2.4.11) in terms of variables X, P is given by

$$\begin{pmatrix} X_{n+1} \\ P_{n+1} \end{pmatrix} = \begin{pmatrix} \cos(\omega dt) & \frac{1}{m\omega}\sin(\omega dt) \\ -m\omega \sin(\omega dt) & \cos(\omega dt) \end{pmatrix} \begin{pmatrix} X_n \\ P_n \end{pmatrix}, \tag{2.4.15}$$

where $X_{n+1} = X(t_{n+1})$ and $P_{n+1} = P(t_{n+1})$ for $n = 0, 1, \ldots$. Finally, substituting

$$\begin{aligned} X_n &= e^{\alpha t_n}(x_n - x_e), \\ P_n &= e^{\alpha t_n}(p_n + m\alpha(x_n - x_e)), \end{aligned}$$

into Eq. (2.4.15), we obtain the following result.

Corollary 2.4.1

The solution of the damped harmonic oscillator equation with constant driving force f (see Eq. 2.4.11) is given by

$$\begin{aligned} x_{n+1} &= x_e + e^{-\alpha dt}\left((x_n - x_e)\left(\cos(\omega dt) + \frac{\alpha}{\omega}\sin(\omega dt)\right) + \frac{\sin(\omega dt)}{\omega} p_n\right), \\ p_{n+1} &= e^{-\alpha dt}\left(p_n\left(\cos(\omega dt) - \frac{\alpha}{\omega}\sin(\omega dt)\right) - \left(\omega + \frac{\alpha^2}{\omega}\right)(x_n - x_e)\sin(\omega dt)\right), \\ &\forall n = 0, 1, \ldots . \end{aligned} \tag{2.4.16}$$

2.4.3. Modal superposition for the wave equation

In this paragraph, we use the modal superposition approach to solve the acoustic wave equation with arbitrary initial conditions and the homogeneous Neumann boundary conditions. Our goal is to derive the Fourier-time domain acoustic wave equation: we will see that, in the Fourier-time domain, the infinite degrees-of-freedom system reduces to an infinite number of single degree-of-freedom systems, which can be solved independently and effortlessly.

Consider a rectangular domain of dimensions L_x, L_y, L_z. Recalling Eq. (2.2.27), the mode shapes are

$$p_{mnp}(\underline{x}) \triangleq \cos\left(\frac{m\pi}{L_x}x\right)\cos\left(\frac{n\pi}{L_y}y\right)\cos\left(\frac{p\pi}{L_z}z\right).$$

This method corresponds exactly to the *method of eigenfunction expansion* proposed in Asmar [2010]. In fact, analogously to Eq. (2.2.28), we can express $p(\underline{x},t)$ as a *triple Cosine Series expansion* as follows:

$$\begin{aligned} p(\underline{x},t) &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{p=1}^{\infty} P_{mnp}(t)\, p_{mnp}(\underline{x}) \\ &= \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{p=1}^{\infty} P_{mnp}(t)\, \cos\left(\frac{m\pi}{L_x}x\right)\cos\left(\frac{n\pi}{L_y}y\right)\cos\left(\frac{p\pi}{L_z}z\right), \end{aligned} \quad (2.4.17)$$

where the modal coefficients $P_{mnp}(t)$ are interpreted as *triple Fourier cosine series coefficients*, and can be computed as

$$P_{mnp}(t) = \frac{8}{L_xL_yL_z}\int_0^{L_z}\int_0^{L_y}\int_0^{L_x} p(\underline{x},t)\, \cos\left(\frac{m\pi}{L_x}x\right)\cos\left(\frac{n\pi}{L_y}y\right)\cos\left(\frac{p\pi}{L_z}z\right)\, dxdydz.$$

We express the force analogously:

$$f(\underline{x},t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty}\sum_{p=1}^{\infty} F_{mnp}(t)\, p_{mnp}(\underline{x}).$$

We can replace $p(\underline{x},t)$ and $f(\underline{x},t)$ with the triple Fourier cosine series in the acoustic wave equation (2.2.23), obtaining

$$\begin{aligned} &\frac{d^2P_{mnp}(t)}{dt^2}p_{mnp}(\underline{x}) - c^2P_{mnp}(t)\left(\frac{\partial^2 p_{mnp}(\underline{x})}{\partial x^2} + \frac{\partial^2 p_{mnp}(\underline{x})}{\partial y^2} + \frac{\partial^2 p_{mnp}(\underline{x})}{\partial z^2}\right) \\ &\qquad = F_{mnp}(t)\, p_{mnp}(\underline{x}). \end{aligned}$$

We proceed differentiating the normal modes $p_{mnp}(\underline{x})$ with respect to the space coordinates:

$$\frac{\partial^2 p_{mnp}(\underline{x})}{\partial x^2} = -\mu_m^2 p_{mnp}(\underline{x}),$$
$$\frac{\partial^2 p_{mnp}(\underline{x})}{\partial y^2} = -\nu_n^2 p_{mnp}(\underline{x}),$$
$$\frac{\partial^2 p_{mnp}(\underline{x})}{\partial z^2} = -\xi_p^2 p_{mnp}(\underline{x}).$$

So that, remembering that $k^2 = \mu^2 + \nu^2 + \xi^2$, we get the acoustic wave equation in the Fourier-time domain:

$$\frac{d^2 P_{mnp}(t)}{dt^2} + k_{mnp}^2 c^2 P_{mnp}(t) = F_{mnp}(t), \quad m \in \mathbb{N}^+, \ n \in \mathbb{N}^+, \ p \in \mathbb{N}^+. \tag{2.4.18}$$

For each tuple (m, n, p), this represents the equation of motion for a single degree-of-freedom system (forced harmonic oscillator), whose solution $P_{mnp}(t)$ is trivial. Employing Eq. (2.4.17), we can go back to the solution in the space-time domain.
Let's now consider the viscous case. We recall that the mode shapes are not affected by the viscous damping; hence, the Fourier-time domain equation with the homogeneous Neumann boundary conditions becomes

$$\frac{d^2 P_{mnp}(t)}{dt^2} + 2\alpha \frac{dP_{mnp}(t)}{dt} + k_{mnp}^2 c^2 P_{mnp}(t) = F_{mnp}(t),$$
$$m \in \mathbb{N}^+, \ n \in \mathbb{N}^+, \ p \in \mathbb{N}^+, \tag{2.4.19}$$

which, for each tuple (m, n, p), represents the equation of motion for a single degree-of-freedom system (forced damped harmonic oscillator).

3 | Simulation of the wave equation through the FDTD method

The numerical simulation of the wave equation is of fundamental importance in the field of acoustics, as it allows us to study the behavior of sound waves in different environments. In this chapter, we focus on the Finite-Difference Time-Domain (FDTD) method for the discretization of the wave equation. In Sec. 2.3.2 we have already presented a well known finite difference scheme of order $(2, 2)$ to solve the wave equation; in this chapter, we'll derive a higher order finite difference scheme to solve the viscous acoustic wave equation, both in second order representation (see Eq. 2.1.3) and in first order representation. We also explain the implementation of boundary conditions, which are necessary to ensure that the simulated sound waves behave correctly at the boundaries of the computational domain.

3.1. Discretization of the second order equation

Consider the viscous acoustic wave equation in Eq. (2.1.3),

$$\frac{\partial^2 p(\underline{x}, t)}{\partial t^2} + 2\alpha \frac{\partial p(\underline{x}, t)}{\partial t} - c^2 \Delta p(\underline{x}, t) = f(x, t), \quad t \in \mathbb{R}^+, \quad \underline{x} \in \mathbb{R}^N.$$

The unviscid case can be seen as a particular case of the viscous case.
Consider a space-time grid with constant step size in both space (dh) and time (dt). Thus:

$$\begin{aligned} \underline{x}_{i,j,k} &= x_i\, \hat{i} + y_j\, \hat{j} + z_k\, \hat{k} \\ &= i\, dh\, \hat{i} + j\, dh\, \hat{j} + k\, dh\, \hat{k}, \quad i, j, k \in \mathbb{N}, \\ t^n &= n\, dt, \qquad\qquad\qquad\qquad n \in \mathbb{N}, \end{aligned}$$

where $\hat{i}, \hat{j}, \hat{k}$ are the *unit vectors* in Cartesian coordinates. We introduce the following notation for simplicity:

$$p(\underline{x}_{i,j,k}, t^n) = p^n_{i,j,k}.$$

We consider a second order accurate centered stencil for the approximation of the first time derivative:

$$\left.\frac{\partial p}{\partial t}\right|_{\underline{x}=\underline{x}_{i,j,k},t=t^n} \approx \frac{p^{n+1}_{i,j,k} - p^{n-1}_{i,j,k}}{2\,dt},$$

and the second time derivative, respectively,

$$\left.\frac{\partial^2 p}{\partial t^2}\right|_{\underline{x}=\underline{x}_{i,j,k},t=t^n} \approx \frac{p^{n-1}_{i,j,k} - 2p^{n}_{i,j,k} + p^{n+1}_{i,j,k}}{dt^2}.$$

Next, we approximate each second order space derivatives with a sixth order accurate centered stencil:

$$\left.\frac{\partial^2 p}{\partial x^2}\right|_{\underline{x}=\underline{x}_{i,j,k},t=t^n} \approx \frac{Ap^n_{i-3,j,k} + Bp^n_{i-2,j,k} + Cp^n_{i-1,j,k} + Dp^n_{i,j,k} + Cp^n_{i+1,j,k} + Bp^n_{i+2,j,k} + Ap^n_{i+3,j,k}}{dh^2}, \quad (3.1.1)$$

which, in vector notation, becomes

$$\left.\frac{\partial^2 p}{\partial x^2}\right|_{\underline{x}=\underline{x}_{i,j,k},t=t^n} \approx \frac{1}{dh^2}\begin{bmatrix} A & B & C & D & C & B & A \end{bmatrix}\begin{bmatrix} p^n_{i-3,j,k} \\ p^n_{i-2,j,k} \\ p^n_{i-1,j,k} \\ p^n_{i,j,k} \\ p^n_{i+1,j,k} \\ p^n_{i+2,j,k} \\ p^n_{i+3,j,k} \end{bmatrix},$$

where the stencil coefficients A, B, C, D, found in Tab. 2.3.1, are

$$\begin{aligned} A &= \frac{1}{90}, \\ B &= -\frac{3}{20}, \\ C &= \frac{3}{2}, \\ D &= -\frac{49}{18}. \end{aligned} \quad (3.1.2)$$

We would like to note here that a sixth order scheme was chosen as it will give sufficiently low interface errors in the context of Rectangular Domain Decomposition (which will be

introduced in Ch. 5) and Adaptive Rectangular Decomposition (Ch. 7), while being reasonably efficient. Lower (second/fourth) order schemes would be more efficient and much easier to implement, but they would result in more prominent spurious reflections (Cfr. Sec. 5.5.2), which would appear as undesirable and audible high frequency noise (Raghuvanshi et al. [2011]).
In the 1D case, the finite difference scheme can be expressed as

$$\underline{p}^{n+1} = 2\underline{p}^{n} - \underline{p}^{n-1} - \alpha\, dt\, (\underline{p}^{n+1} - \underline{p}^{n-1}) + \left(\frac{cdt}{dh}\right)^2 [K]\, \underline{p}^{n} + dt^2 \underline{f}^{n} + O(dt^2) + O(dh^6),$$

which can be rewritten by expliciting $\underline{p}^{n+1}$:

$$\underline{p}^{n+1} = \frac{2\underline{p}^{n} - (1 - \alpha\, dt)\underline{p}^{n-1} + \left(\frac{cdt}{dh}\right)^2 [K]\, \underline{p}^{n} + dt^2 \underline{f}^{n}}{1 + \alpha\, dt} + O(dt^2) + O(dh^6). \tag{3.1.3}$$

This equation represents an explicit finite difference scheme which computes the solution at time t_{n+1} starting from the solution at times t_n and t_{n-1}. $\underline{p} = [p_1,\ p_2,\ \ldots]$ is the vector storing the pressure values at each space step, $\underline{f} = [f_1,\ f_2,\ \ldots]$ is the vector storing the force values at each space step, while $[K]$ is the *stiffness matrix*, defined as

$$[K] \triangleq \begin{bmatrix} \ddots & & & & & & & & \\ & A & B & C & D & C & B & A & \\ & & A & B & C & D & C & B & A \\ & & & A & B & C & D & C & B & A \\ & & & & & & & & & \ddots \end{bmatrix}. \tag{3.1.4}$$

We can approximate the pressure velocity (defined in Eq. (2.1.2)) employing the *implicit Euler* method, i.e.,

$$\underline{v}^{n+1} = \frac{\underline{p}^{n+1} - \underline{p}^{n}}{dt}. \tag{3.1.5}$$

3.1.1. Analysis of the scheme

In this subsection, we will perform the von Neumann analysis of the derived finite difference method as in Subsec. 2.3.3.

Stability analysis

Consider the FDTD $(2,6)$ scheme (3.1.3), which we report for readability:

$$\underline{p}^{n+1} = \frac{2\underline{p}^n - (1-\alpha\, dt)\underline{p}^{n-1} + \left(\frac{cdt}{dh}\right)^2 [K]\, \underline{p}^n + dt^2\, \underline{f}^n}{1+\alpha\, dt}. \tag{3.1.6}$$

Recalling the definition of the Courant number λ in Eq. (2.3.5), we rewrite Eq. (3.1.6) as

$$(1+\alpha\, dt)\underline{p}^{n+1} = 2\underline{p}^n - (1-\alpha\, dt)\underline{p}^{n-1} + \lambda^2[K]\, \underline{p}^n + dt^2\, \underline{f}^n,$$

which, in scalar form, becomes

$$\begin{aligned}(1+\alpha\, dt)p_i^{n+1} &= 2p_i^n - (1-\alpha\, dt)p_i^{n-1} - \alpha\, dt\, (p_i^{n+1} - p_i^{n-1}) \\ &+ \lambda^2 \begin{bmatrix} A & B & C & D & C & B & A \end{bmatrix} \begin{bmatrix} p_{i-3}^n \\ p_{i-2}^n \\ p_{i-1}^n \\ p_i^n \\ p_{i+1}^n \\ p_{i+2}^n \\ p_{i+3}^n \end{bmatrix} + dt^2 f_i^n. \end{aligned} \tag{3.1.7}$$

Imposing $p_i^n = z^n e^{ji\beta dh}$ as in Eq. (2.3.8) and $f_i^n = 0$ in the above equation, we get

$$(1+dt\,\alpha)z = 2 + \lambda^2 \begin{bmatrix} A & B & C & D & C & B & A \end{bmatrix} \begin{bmatrix} e^{-3j\beta dh} \\ e^{-2j\beta dh} \\ e^{-j\beta dh} \\ 1 \\ e^{j\beta dh} \\ e^{2j\beta dh} \\ e^{3j\beta dh} \end{bmatrix} - (1-dt\,\alpha)z^{-1},$$

which results in the characteristic equation

$$(1+dt\,\alpha)z - 2\left(1+\lambda^2\gamma\right) + (1-dt\,\alpha)z^{-1} = 0. \tag{3.1.8}$$

where $\gamma(\beta dh)$ is defined as

$$\gamma(\beta dh) \triangleq A\cos(3\beta dh) + B\cos(2\beta dh) + C\cos(\beta dh) + \frac{D}{2}. \tag{3.1.9}$$

The plot of the function $\gamma(\beta dh)$ is shown in Fig. 3.1.1. We notice that the minimum of the function is $-A'$ where $A' \triangleq |A|+|B|+|C|+|D|/2 = 136/45 = 3.0\overline{2}$ and the maximum is 0, hence $-A' \leqslant \gamma \leqslant 0,\ \forall \lambda, \beta dh$. Furthermore, the function is periodic of period 2π.

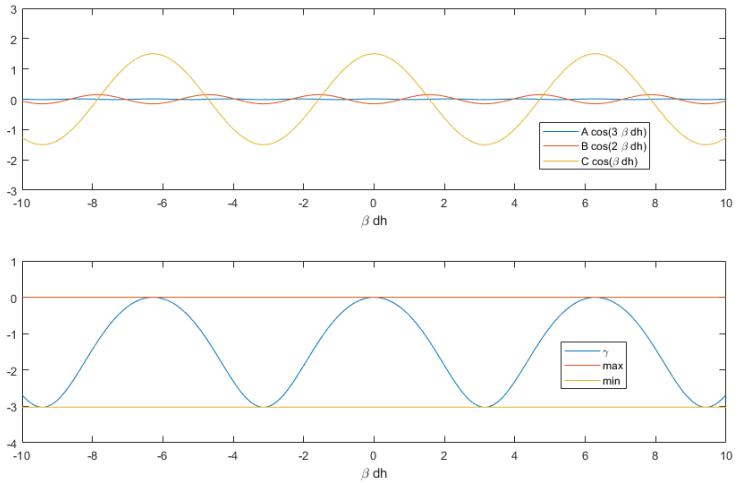


Figure 3.1.1: Plot of the function $\gamma(\beta dh)$ and of the single harmonics.

The roots of the characteristic equation (3.1.8) are

$$z_{\pm} = \frac{1 + \lambda^2\gamma \pm \sqrt{\Delta}}{1 + dt\,\alpha},$$

where Δ is defined as

$$\Delta \triangleq \left(1 + \lambda^2\gamma\right)^2 - (1 + dt\,\alpha)(1 - dt\,\alpha).$$

The scheme is conditionally stable: we require that, for each possible value of βdh, both roots have magnitude less than 1. In Fig. 3.1.2 we plot the maximum absolute value of the roots $z_{\pm}$ as a function of λ for different values of α. Notice how, for $\lambda \lesssim 0.8$, both roots have magnitude less than 1 for all the considered values of α: this means that the CFL condition might be $|\lambda| \lesssim 0.8$; a more accurate bound will be computed below. We also notice that it would reasonable to consider the case $\alpha = 0$ as the worst case, i.e., the case for which the stability criterion would be more strict.
In Fig. 3.1.3 we plot the value of Δ as a function of βdh for different values of λ and α.

We notice that, in the undamped case ($\alpha = 0$), $\Delta < 0$ for $\lambda \lesssim 0.8$.

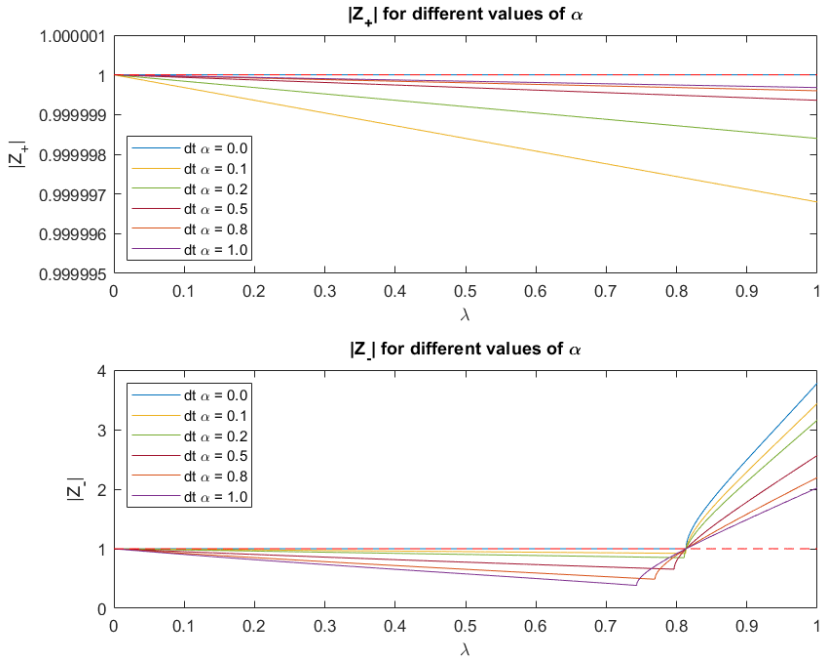


Figure 3.1.2: Maximum absolute value of the roots as a function of the Courant number λ for different values of the absorption coefficient α. The dashed curve in red represents the stability limit $|z_{\pm}| = 1$.

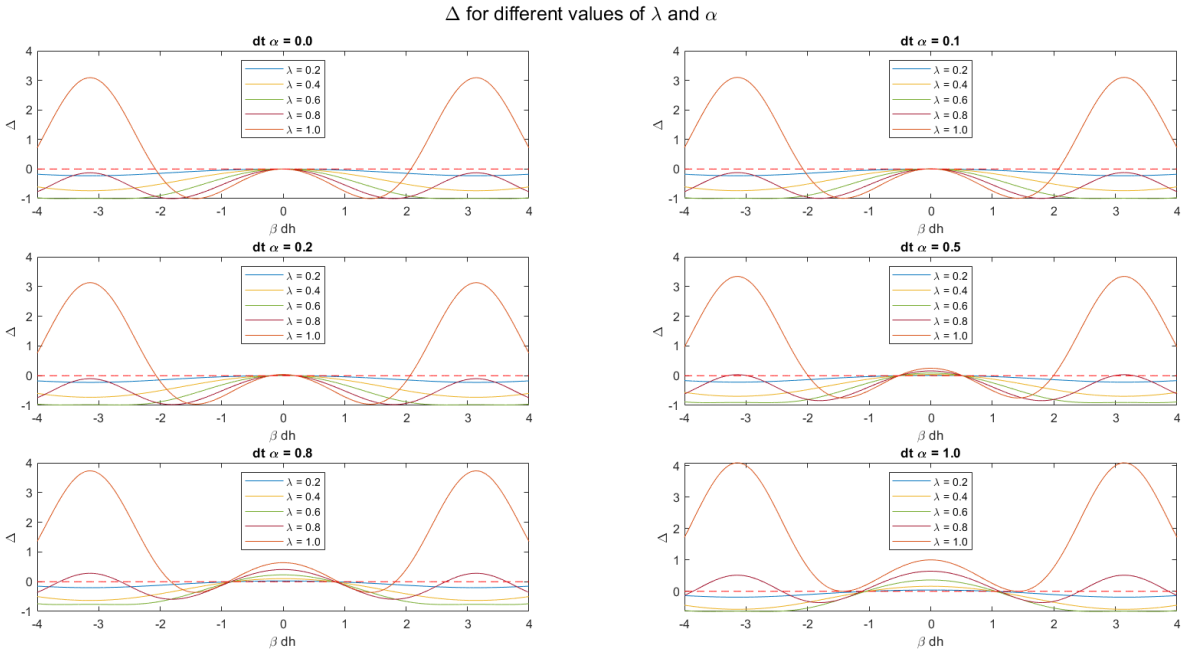


Figure 3.1.3: Δ as a function of βdh for different values of the Courant number λ and of the absorption coefficient α. The dashed curve in red represents $\Delta = 0$.

We want to compute the stability criterion more accurately and analytically. First, we compute the magnitude of the roots for the case of non-positive Δ:

$$\begin{aligned} |z_{\pm}| &= \left| \frac{1 + \lambda^2\gamma \pm j\sqrt{(1 + dt\,\alpha)(1 - dt\,\alpha) - (1 + \lambda^2\gamma)^2}}{1 + dt\,\alpha} \right| \\ &= \frac{\sqrt{(1 + \lambda^2\gamma)^2 + (1 + dt\,\alpha)(1 - dt\,\alpha) - (1 + \lambda^2\gamma)^2}}{|1 + dt\,\alpha|} \\ &= \frac{\sqrt{(1 + dt\,\alpha)(1 - dt\,\alpha)}}{|1 + dt\,\alpha|} = \sqrt{\frac{(1 + dt\,\alpha)(1 - dt\,\alpha)}{(1 + dt\,\alpha)^2}} = \sqrt{\frac{1 - dt\,\alpha}{1 + dt\,\alpha}} \leqslant 1, \end{aligned} \tag{3.1.10}$$

since $\alpha \geqslant 0$. The equality holds for $\alpha = 0$.
Now, we compute the magnitude of the roots for the case of positive Δ:

$$|z_{\pm}| = \left| \frac{1 + \lambda^2\gamma \pm \sqrt{(1 + \lambda^2\gamma)^2 - (1 + dt\,\alpha)(1 - dt\,\alpha)}}{1 + dt\,\alpha} \right|.$$

This means that the scheme is stable for the values of the Courant number λ such that, for each value of βdh, either $\Delta \leqslant 0$ or, if $\Delta > 0$, the magnitude of $z_{\pm}$ is less or equal than 1 (i.e., $\Delta > 0 \wedge |z_{\pm}| \leqslant 1$).
In the undamped case, the condition $\Delta \leqslant 0$ reduces to

$$\begin{aligned} &\Delta = \left(1 + \lambda^2\gamma\right)^2 - 1 \leqslant 0, \\ &\left(1 + \lambda^2\gamma\right)^2 \leqslant 1, \\ &-1 \leqslant 1 + \lambda^2\gamma \leqslant 1, \\ &-2 \leqslant \lambda^2\gamma \leqslant 0, \end{aligned}$$

that is

$$\lambda^2\gamma \geqslant -2 \wedge \gamma \leqslant 0,$$

which, recalling the bounds on γ, can be rewritten as

$$\begin{aligned} &\min \lambda^2\gamma \geqslant -2 \wedge \max \gamma \leqslant 0, \\ &\lambda^2 A' \leqslant 2 \wedge 0 \leqslant 0, \end{aligned}$$

which results in

$$|\lambda| \leqslant \sqrt{\frac{2}{A'}}. \tag{3.1.11}$$

We also need to consider the condition $\Delta > 0 \wedge |z_\pm| \leqslant 1$. First, examine the positive solution, for which the condition becomes

$$\left|1 + \lambda^2\gamma + \sqrt{(1+\lambda^2\gamma)^2}\right| \leqslant 1,$$

$$-1 \leqslant 2 + 2\lambda^2\gamma \leqslant 1,$$

$$2 + 2\lambda^2\gamma \geqslant -1 \wedge 2 + 2\lambda^2\gamma \leqslant 1,$$

that is

$$\lambda^2\gamma \geqslant -\frac{3}{2} \wedge \lambda^2\gamma \leqslant -\frac{1}{2},$$

which again, recalling the bounds on γ, can be rewritten as

$$\min \lambda^2\gamma \geqslant -\frac{3}{2} \wedge \max \lambda^2\gamma \leqslant -\frac{1}{2},$$

$$\lambda^2 A' \leqslant \frac{3}{2} \wedge \lambda^2 \cdot 0 \leqslant -\frac{1}{2},$$

which is never satisfied.
By excess of zeal, we consider the negative solution, for which the condition becomes

$$\left|1 + \lambda^2\gamma - \sqrt{(1+\lambda^2\gamma)^2}\right| \leqslant 1,$$

$$0 \leqslant 1,$$

which is always satisfied.
This means that, in the undamped case, only with $\Delta \leqslant 0$ we can guarantee that $|z_\pm| \leqslant 1$. Hence, from Eq. (3.1.11) we obtain the CFL condition for the undamped FDTD $(2,6)$ scheme:

$$|\lambda| \leqslant \sqrt{\frac{2}{A'}} = \sqrt{\frac{2}{|A| + |B| + |C| + \frac{|D|}{2}}} = \sqrt{\frac{45}{68}} = \frac{3\sqrt{85}}{34} \approx 0.8135. \tag{3.1.12}$$

In the damped case, we already know from Eq. (3.1.10) that, in case of non-positive Δ, we have complex conjugate roots of unit magnitude less than 1 (see Eq. 3.1.10): this means that, for values of λ such that $\Delta \leqslant 0$, $\forall \beta dh$, the scheme is stable. This condition can be

expressed as

$$\left(1+\lambda^2\gamma\right)^2-(1+dt\,\alpha)(1-dt\,\alpha)\leqslant 0,$$
$$1+\left(\lambda^2\gamma\right)^2+2\lambda^2\gamma\leqslant 1-(dt\,\alpha)^2,$$
$$\left(\lambda^2\gamma\right)^2+2\left(\lambda^2\gamma\right)+(dt\,\alpha)^2\leqslant 0.$$

First, we find the root of the associated equation:

$$\left(\lambda^2\gamma\right)^2+2\left(\lambda^2\gamma\right)+(dt\,\alpha)^2=0,$$
$$\left(\lambda^2\gamma\right)_{\pm}=-1\pm\sqrt{1-(dt\,\alpha)^2}.$$

For $(dt\,\alpha)^2>1\rightarrow dt\,\alpha>1$, we have that $\Delta'\triangleq 1-(dt\,\alpha)^2$ is negative and the inequality has no solution. If $\Delta'\geqslant 0$, i.e., $dt\,\alpha\leqslant 1$, the solution is

$$-1-\sqrt{1-(dt\,\alpha)^2}\leqslant\lambda^2\gamma\leqslant-1+\sqrt{1-(dt\,\alpha)^2},$$
$$\lambda^2\gamma\geqslant-1-\sqrt{1-(dt\,\alpha)^2}\wedge\lambda^2\gamma\leqslant-1+\sqrt{1-(dt\,\alpha)^2},$$

which, recalling the bounds on γ, can be rewritten as

$$\min\lambda^2\gamma\geqslant-1-\sqrt{1-(dt\,\alpha)^2}\wedge\max\lambda^2\gamma\leqslant-1+\sqrt{1-(dt\,\alpha)^2},$$
$$\lambda^2A'\leqslant 1+\sqrt{1-(dt\,\alpha)^2}\wedge 0\leqslant-1+\sqrt{1-(dt\,\alpha)^2},$$
$$\lambda^2\leqslant\frac{1+\sqrt{1-(dt\,\alpha)^2}}{A'}\wedge\sqrt{1-(dt\,\alpha)^2}\geqslant 1,$$

that is

$$|\lambda|\leqslant\sqrt{\frac{1+\sqrt{1-(dt\,\alpha)^2}}{A'}}\wedge(dt\,\alpha)^2\leqslant 0,$$

which is satisfied only for $\alpha=0$. In fact, inspecting Fig. 3.1.3, we notice that, for $0<dt\,\alpha\leqslant 1$, there's no λ such that the condition $\Delta\leqslant 0$ is satisfied for all the values of βdh.

However, recalling Fig. 3.1.2, we know that there are some values of λ such that, for each value of βdh, either $\Delta\leqslant 0$ or $\Delta>0\wedge|z_{\pm}|\leqslant 1$. In particular, it can be proven that a CFL condition of the damped $(2,6)$ FDTD scheme is, again, given by Eq. (3.1.12), which we report for readability:

$$|\lambda|\leqslant\sqrt{\frac{2}{A'}}=\frac{3\sqrt{85}}{34}\approx 0.8135.$$

For the sake of clarity, the CFL condition could be higher for higher values of α, but we

prefer to have a stability criterion valid for the worst case ($\alpha = 0$).

Dissipation and dispersion analysis

Consider the characteristic equation of the FDTD $(2, 6)$ scheme in Eq. (3.1.8) with $\alpha = 0$, and impose $z = e^{j\omega dt}$:

$$e^{j\omega dt} - 2\left(1 + \lambda^2\gamma\right) + e^{-j\omega dt} = 0,$$

$$\cos(\omega dt) = 1 + \lambda^2\gamma,$$

which, recalling the definition of γ in Eq. (3.1.9), becomes

$$\begin{aligned} 1 - 2\sin^2\left(\frac{1}{2}\omega dt\right) &= 1 + \lambda^2\left[A + B + C + \frac{D}{2}\right] \\ &- 2\lambda^2\left[A\sin^2\left(\frac{3}{2}\beta dh\right) + B\sin^2\left(\beta dh\right) + C\sin^2\left(\frac{1}{2}\beta dh\right)\right]. \end{aligned}$$

Noticing that $A + B + C + D/2 = 0$ (see Eq. 3.1.2), the above equation can be rewritten as

$$\sin^2\left(\frac{1}{2}\omega dt\right) = \lambda^2\left[A\sin^2\left(\frac{3}{2}\beta dh\right) + B\sin^2\left(\beta dh\right) + C\sin^2\left(\frac{1}{2}\beta dh\right)\right],$$

$$\sin\left(\frac{1}{2}\omega dt\right) = \lambda\sqrt{A\sin^2\left(\frac{3}{2}\beta dh\right) + B\sin^2\left(\beta dh\right) + C\sin^2\left(\frac{1}{2}\beta dh\right)},$$

which results in the dispersion relation for the FDTD $(2, 6)$ scheme:

$$\omega = \pm\frac{2}{dt}\arcsin\left(\lambda\sqrt{A\sin^2\left(\frac{3}{2}\beta dh\right) + B\sin^2\left(\beta dh\right) + C\sin^2\left(\frac{1}{2}\beta dh\right)}\right). \tag{3.1.13}$$

The numerical phase and group velocities (respectively defined in Eq. (2.3.14a) and Eq. (2.3.14b)) depend on the choice of the parameter λ. The velocity curves, as functions of frequency $f = \omega/(2\pi)$ for different values of λ, are shown in Fig. 3.1.4. Notice that, when $\lambda = \lambda_{max}$, there is a discontinuity in the group velocity.

For the FDTD $(2, 6)$ scheme, the best numerical behavior (less dispersion) is obtained for low values of λ in a limited frequency band where the response is nearly flat; in general, there is a trade-off between the bandwidth of the scheme and the amount of numerical dispersion, thus the most suitable choice of λ depends on the application. For example, if we want the phase velocity to be limited in a range $[1 - \varepsilon, 1 + \varepsilon]$, we choose the value of λ that satisfies this requirement while maximizing the scheme's bandwidth as much as possible.

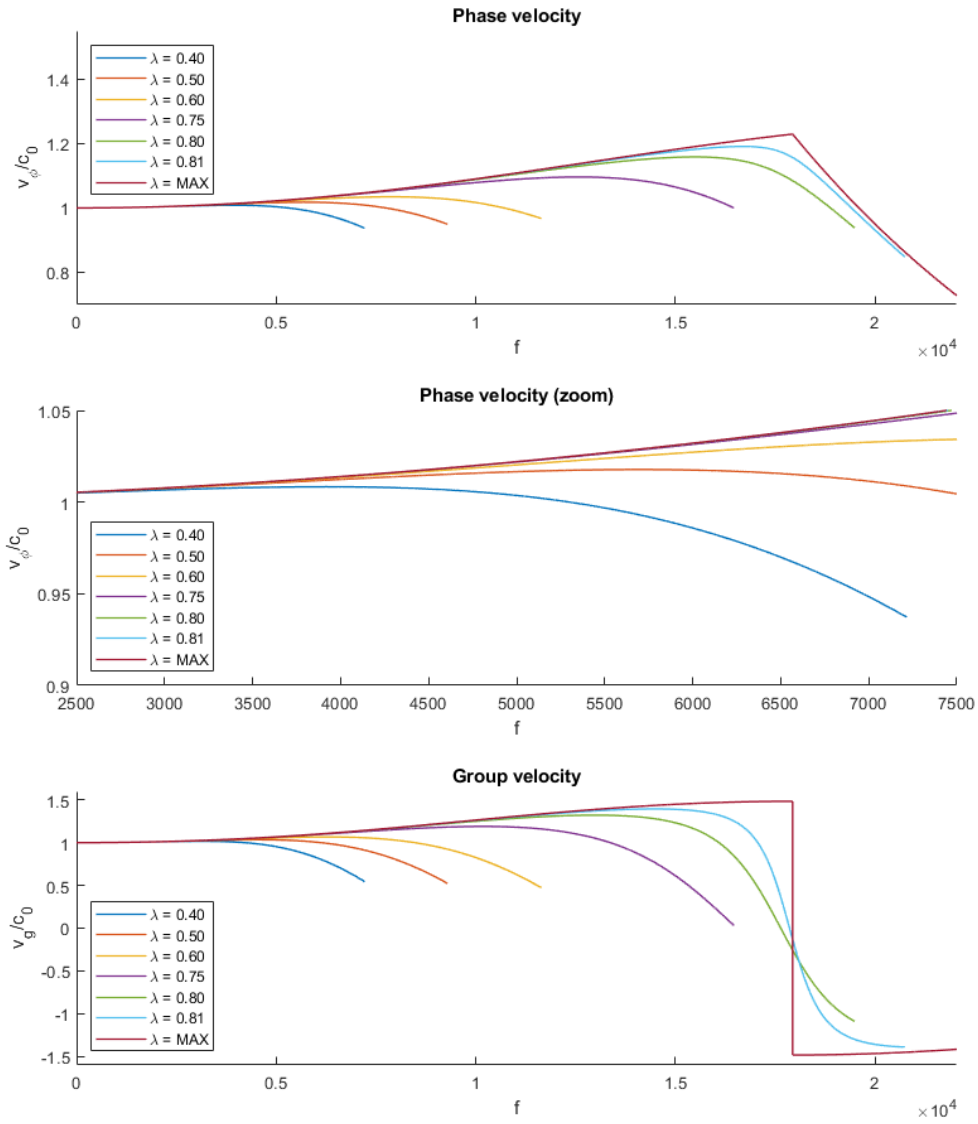


Figure 3.1.4: Numerical phase velocity (up) and group velocity (down) as a function of frequency f, normalized by the model velocity c_0, for the FDTD $(2, 6)$ scheme in Eq. (3.1.8), for a variety of values of λ, as indicated. The sample rate is chosen as 44100 Hz.

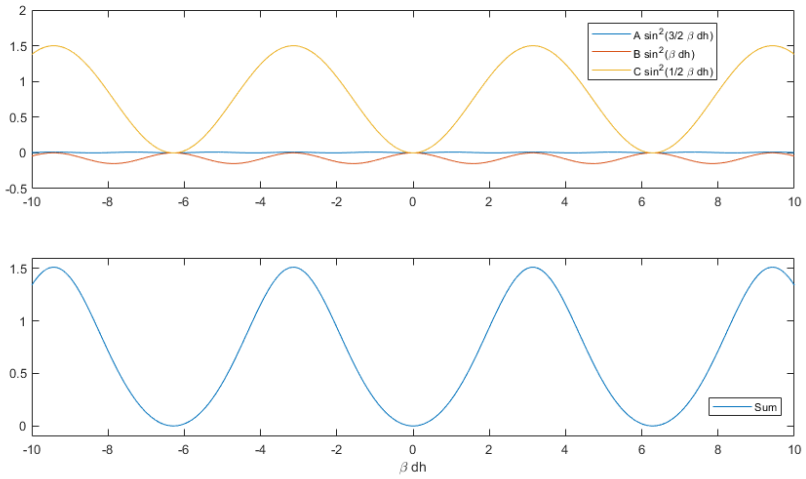


Figure 3.1.5: Plot of the argument of the square root in Eq. 3.1.13. We notice that the maximum corresponds to the constructive interference of the first and third "harmonic" when the second harmonic is zero.

The effects of dispersion are illustrated in Fig. 3.1.6, in which we show the wave packet at $t = 0$ (a Gaussian distribution of width 1/500) and at $t = 1$ for different values of λ. Notice that, for $\lambda \to 0.8135\ldots$, the higher-frequency components surpass the wavefront, illustrating the phase velocity characteristic of the scheme. Notice also that the gross speed of the wave packet is faster as well, illustrating the group velocity characteristic. We recall that the expected amplitude of the wave packet at the final time of simulation is half of that at the start, because the wave is split in two propagating waves.
In Fig. 3.1.7 and Fig. 3.1.8 we show the frequency response (respectively magnitude and phase) for different values of λ, from which we can infer the dissipation and dispersion characteristic of the scheme. They are obtained by performing the ratio between the FFT of the wave packet at $t = 1$ and the one at $t = 0$. We notice that the best results (less dissipation and less dispersion) are obtained for low values of λ, and the artifacts gets worse for $\lambda \to 0.8135\ldots$.
Further confirmations of the behavior of the scheme (3.1.3) are found in Fig. 3.1.9, in which we show the DCT of the wave packet at $t = 0$ and at $t = 1$ for different values of λ.

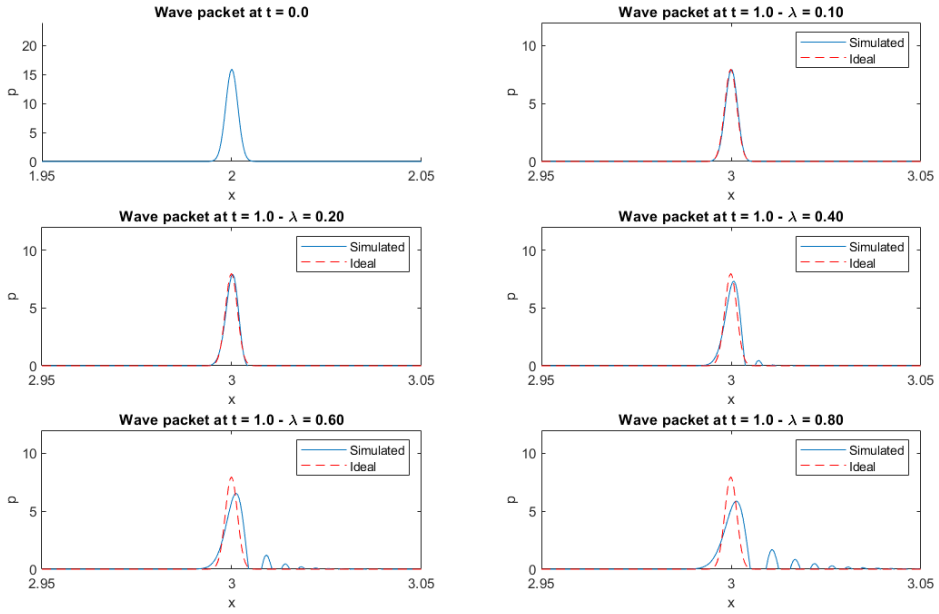


Figure 3.1.6: The snapshot in the northwest corner depicts the wave packet at $t = 0$, a Gaussian distribution with a width of 1/500. The remaining snapshots depict the output of second order FDTD $(2, 6)$ at $t = 1$ for different values of λ. The wave speed c is set to 1, $\alpha = 0$, and the spatial step size is $dh = 4e - 4$. The ground truth solution is displayed using dashed lines.

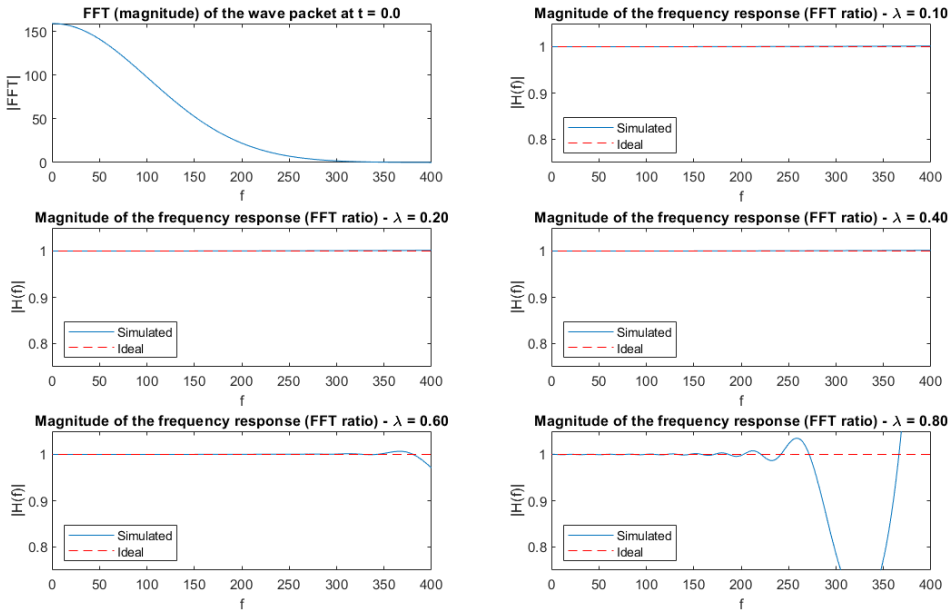


Figure 3.1.7: The northwest image represents the magnitude of the FFT of the wave packet at $t = 0$. The other figures depict the magnitude of the frequency response of second order FDTD $(2, 6)$ for different values of λ. The ground truth solution is displayed using dashed lines.

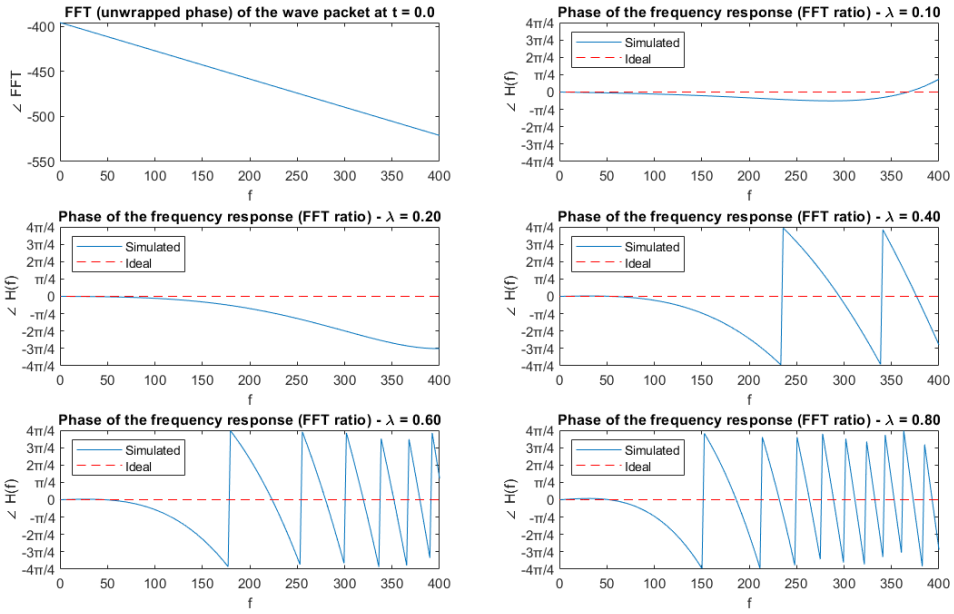


Figure 3.1.8: The northwest image represents the unwrapped phase of the FFT of the wave packet at $t = 0$. The other figures depict the phase of the frequency response of second order FDTD $(2, 6)$ for different values of λ. The ground truth solution is displayed using dashed lines.

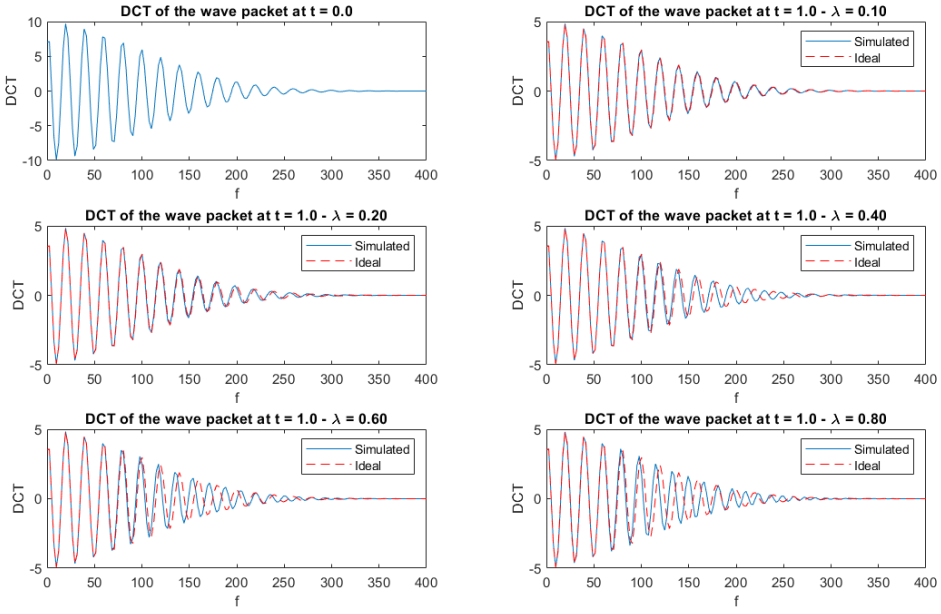


Figure 3.1.9: The northwest snapshot represents the DCT of the wave packet at $t = 0$. The other snapshots depict the DCT of the output from second order FDTD $(2, 6)$ for different values of λ. The ground truth solution is displayed using dashed lines.

The bandwidth of the scheme can be computed analytically from Eq. (3.1.13). Considering only the positive solution, using $\omega = 2\pi f$, defining the sampling frequency $f_s = 1/dt$, and maximizing the argument of the square root (see Fig. 3.1.5), the maximum frequency f_{max} will be given by

$$2\pi f_{max} = 2f_s \arcsin\left(\lambda\sqrt{A+C}\right),$$

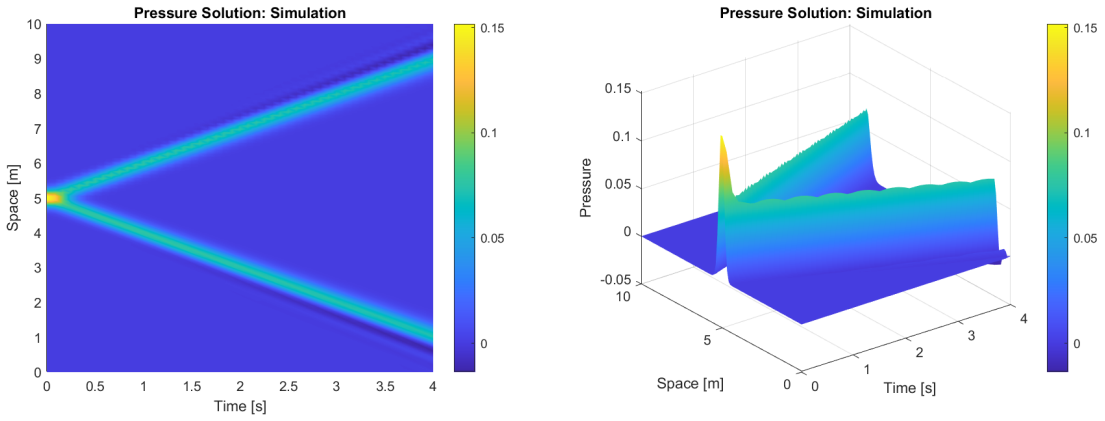
$$f_{max} = \frac{f_s}{\pi} \arcsin\left(\lambda\sqrt{A+C}\right),$$

$$f_{max} = \frac{f_s}{\pi} \arcsin\left(\lambda\sqrt{\frac{68}{45}}\right),$$

which is defined for any value of λ which satisfies the CFL condition in Eq. (3.1.12).

3.1.2. Simulation of a test case

Consider the propagating wave test case in Subsec. 2.1.7. The results of the simulation using second order FDTD with $dh = 0.1$ and $\lambda = 0.8$ are shown in Fig. 3.1.10.



(a) Top view.

(b) 3D view.

Figure 3.1.10: propagating wave test case simulated with second order FDTD $(2, 6)$ with parameters $dh = 0.1$ and $\lambda = 0.8$.

3.1.3. Convergence test

Consider the standing wave test case in Subsec. 2.1.7. The results of convergence test on second order FDTD with $dh = 1e-2$ and different values of dt are shown in Fig. 3.1.11: for each value of dt, we show the values of L^1, L^2, and L^∞ norm (Def. 2.3.1) of the error (difference between numerical solution and ground truth at the last time instant). Analogously, in Fig. 3.1.12, we show the results of convergence test with $dt = 1e-3$ and

different values of dh.

Notice that the theoretical order of accuracy is not $o(dt^2 + dh^6)$ as one would expect: in fact, the order of accuracy of a numerical scheme is, in general different from the order of accuracy of the finite difference operators employed to approximate the partial derivatives. It has been found that the best fitting order of accuracy for second order FDTD $(2, 6)$ is $o(dt + dh^3)$.

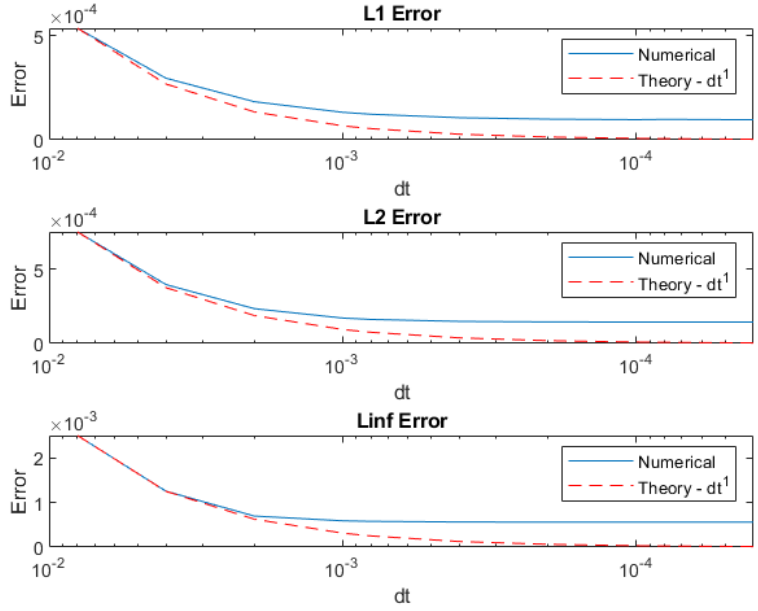


Figure 3.1.11: Convergence test on second order FDTD $(2, 6)$ with $dh = 1e - 2$ and different values of dt. The red curve represents the theoretical convergence behavior.

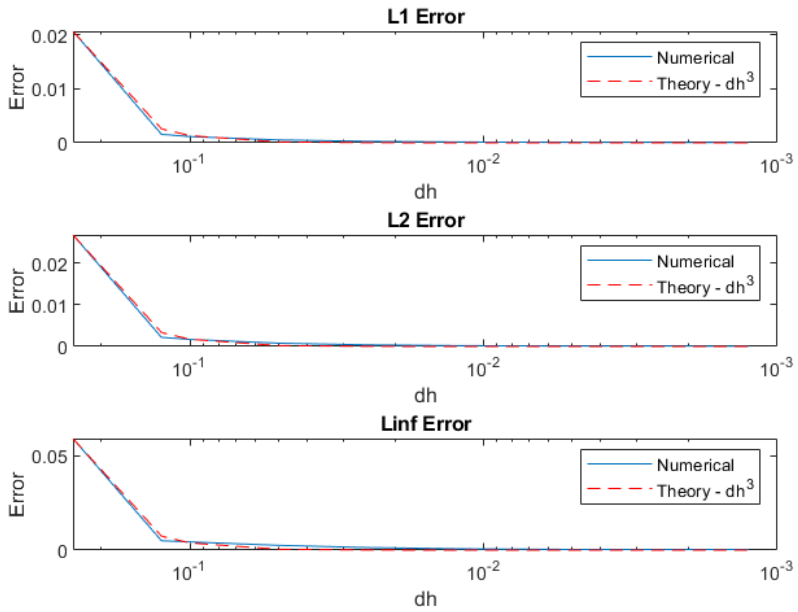


Figure 3.1.12: Convergence test on second order FDTD $(2,6)$ with $dt = 1e-3$ and different values of dh. The red curve represents the theoretical convergence behavior.

3.2. Discretization of the first order equation

We employ a first order representation for the wave equation:

$$\begin{cases} \dot{v}(\underline{x},t) = c^2 \Delta p(\underline{x},t) - 2\alpha v(\underline{x},t) + f(\underline{x},t), & t \in \mathbb{R}^+, \quad \underline{x} \in \mathbb{R}^3, \\ \dot{p}(\underline{x},t) = v(\underline{x},t), & t \in \mathbb{R}^+, \quad \underline{x} \in \mathbb{R}^3. \end{cases} \tag{3.2.1}$$

Analogously to the discretization of the second-order wave equation in Sec. 3.1, we consider the following discretization of the x-axis with a constant step size dh, focusing on the 1D case for simplicity:

$$x_i = i \cdot dh, \quad i = 0, \ldots, I-1,$$

where x_i is the i-th grid point.
Furthermore, we proceed by approximating the partial derivatives with finite differences. For the pressure velocity we employ *implicit Euler*, while for the pressure we use *explicit*

Euler, obtaining

$$\begin{cases} \dfrac{\underline{v}^{n+1}-\underline{v}^{n}}{dt} = \dfrac{c^2}{dh^2}[K]\underline{p}^{n+1} - 2\alpha\underline{v}^{n+1} + \underline{f}^{n+1}, \\ \dfrac{\underline{p}^{n+1}-\underline{p}^{n}}{dt} = \underline{v}^{n}, \end{cases}$$

where $\underline{p}$, $\underline{v}$ and $\underline{f}$ are vectors of dimension I, while the matrix $[K]$ is of dimension $I \times I$. It can be rewritten as

$$\begin{cases} \underline{v}^{n+1} = \dfrac{\underline{v}^{n} + c^2\frac{dt}{dh^2}[K]\underline{p}^{n+1} + dt\,\underline{f}^{n+1}}{1+2dt\;\alpha}, \\ \underline{p}^{n+1} = dt\;\underline{v}^{n} + \underline{p}^{n}, \end{cases} \tag{3.2.2}$$

or, making both equations independent of each other, as

$$\begin{cases} \underline{v}^{n+1} = \dfrac{\left(1 + c^2\frac{dt^2}{dh^2}[K]\right)\underline{v}^{n} + c^2\frac{dt}{dh^2}[K]\underline{p}^{n} + dt\,\underline{f}^{n+1}}{1+2dt\;\alpha}, \\ \underline{p}^{n+1} = dt\;\underline{v}^{n} + \underline{p}^{n}, \end{cases} \tag{3.2.3}$$

To make the notation more compact, we define the *state vector* $\underline{z}$ of dimension $2I$ as

$$\underline{z} \triangleq \begin{bmatrix} \underline{v} \\ \underline{p} \end{bmatrix},$$

so that Eq. (3.2.2) can be rewritten as

$$\underline{z}^{n+1} = \begin{bmatrix} \frac{[I]_{I\times I} + c^2\frac{dt^2}{dh^2}[K]}{1+2dt\;\alpha} & \frac{c^2\frac{dt}{dh^2}[K]}{1+2dt\;\alpha} \\ dt[I]_{I\times I} & [I]_{I\times I} \end{bmatrix}\underline{z}^{n} + \begin{bmatrix} dt\,\underline{f}^{n+1} \\ \underline{0}_I \end{bmatrix}. \tag{3.2.4}$$

We define the *system matrix* $\left[\overline{K}\right]$ of dimension $2I \times 2I$ as

$$\left[\overline{K}\right] \triangleq \begin{bmatrix} \frac{[I]_{I\times I} + c^2\frac{dt^2}{dh^2}[K]}{1+2dt\;\alpha} & \frac{c^2\frac{dt}{dh^2}[K]}{1+2dt\;\alpha} \\ dt[I]_{I\times I} & [I]_{I\times I} \end{bmatrix}, \tag{3.2.5}$$

and the *system force vector* $\overline{\underline{f}}$ of dimension $2I$ as

$$\overline{\underline{f}} \triangleq \begin{bmatrix} dt\,\underline{f}^{n+1} \\ \underline{0}_I \end{bmatrix},$$

so that Eq. (3.2.4) can be rewritten in a compact form as

$$\underline{z}^{n+1} = \left[\overline{K}\right] \underline{z}^{n} + \overline{\underline{f}}^{n+1}. \tag{3.2.6}$$

This equation represents an explicit finite difference scheme of order $(2, 6)$ which computes the solution at time t_{n+1} starting from the solution at time t_n.

3.2.1. Analysis of the scheme

In this subsection, we'll perform the analysis of the derived finite difference method.

Stability analysis

In this subsection, we will perform the stability analysis of the derived finite difference method (see Eq. 3.2.6), which we report for readability:

$$\underline{z}^{n+1} = \left[\overline{K}\right] \underline{z}^{n} + \overline{\underline{f}}^{n}. \tag{3.2.7}$$

Theorem 3.2.1: Convergence of a linear system

Consider the following homogeneous system of first order linear difference equations with constant coefficients:

$$\underline{x}^{n+1} = [A]\underline{x}^{n},$$

where the matrix $[A]_{N\times N}$ and the initial vector $\underline{x}(0)$ are known.
The system is asymptotically stable, i.e.,

$$\lim_{n\to\infty} [A]^{n} = \underline{0},$$

if and only if the *spectral radius* $\rho([A])$, defined as

$$\rho([A]) \triangleq \max|\lambda|, \tag{3.2.8}$$

is less than 1 (Meyer [2000]).

From Th. 3.2.1 follows that we need to guarantee that the spectral radius of $\left[\overline{K}\right]$ is less than 1, recalling that $\left[\overline{K}\right]$ is defined in Eq. (3.2.5) as

$$\left[\overline{K}\right] \triangleq \begin{bmatrix} \frac{[I]_{I\times I} + c^2 \frac{dt^2}{dh^2}[K]}{1+2dt\ \alpha} & \frac{c^2 \frac{dt}{dh^2}[K]}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & [I]_{I\times I} \end{bmatrix}. \tag{3.2.9}$$

To compute the spectral radius of $\left[\overline{K}\right]$, we first compute that of $[K]$ which, recalling Eq. (3.1.4), is defined as

$$[K] \triangleq \begin{bmatrix} \ddots & & & & & & & & & \\ & A & B & C & D & C & B & A & & \\ & & A & B & C & D & C & B & A & \\ & & & A & B & C & D & C & B & A \\ & & & & & & & & & \ddots \end{bmatrix}.$$

It can be proven that

$$\max|\lambda_k| \leqslant 2|A| + 2|B| + 2|C| + |D| = \frac{272}{45} \approx 6.0444\,, \tag{3.2.10}$$

implying that

$$\rho([K]) \leqslant 2|A| + 2|B| + 2|C| + |D| \triangleq \rho([K])_{max}.$$

It has been numerically verified that

$$\lim_{I\to\infty} \rho([K]) = \rho([K])_{max}. \tag{3.2.11}$$

Furthermore, from Eq. (3.2.10) and the definition of spectral radius in Eq. (3.2.8) follows that, for all eigenvalues λ_k of the matrix $[K]$ we have that

$$|\lambda_k| \leqslant \rho([K])_{max},$$

and it has been numerically verified that

$$\lim_{I\to\infty} \lambda_k = -\rho([K])_{max}. \tag{3.2.12}$$

To prove this result we can use the Gershgorin Circle Theorem in 3.2.2.

Theorem 3.2.2: Gershgorin circle theorem

The Gershgorin circle theorem (Bell [1965]) states that every eigenvalue of a square matrix $[A]$ of dimension $N \times N$ lies in the union of the circles (Gershgorin discs)

$$|z - a_{ii}| \leqslant R_i, \tag{3.2.13}$$

where the radius R_i is computed summing the moduli of the off-diagonal elements in the i-th row or column

$$R_i = \sum_{j \neq i} |a_{ij}|.$$

Analogously, we can define an upper bound on the larger circles

$$|z| \leqslant |a_{ii}| + R_i. \tag{3.2.14}$$

To sum up, according to this theorem, every eigenvalue of a matrix must lie within at least one of the circles in the complex plane centered at the diagonal entries of the matrix and with radius equal to the sum of the absolute values of the off-diagonal entries in the corresponding row. Equivalently, every eigenvalue lies within the union of the circles centered in zero and with radius $|a_{ii}| + R_i$.
For our matrix, the diagonal entries are all equal to D, and the off-diagonal entries in the each row are A for the first and last entries, B for the second and second-to-last entries, C for the remaining entries. Therefore, every eigenvalue of the matrix must lie within the circle in the complex plane centered at D with radius $2|A| + 2|B| + 2|C|$. Equivalently, recalling Eq. (3.2.14), we state that

$$\max |\lambda_k| \leqslant |D| + 2|A| + 2|B| + 2|C|,$$

which is exactly the upper bound in Eq. (3.2.10). □
To compute an upper bound on the spectral radius of the matrix $\left[\overline{K}\right]$, we can exploit its property of being a block matrix. We start by computing the characteristic polynomial

of $\left[\overline{K}\right]$:

$$
\begin{aligned}
&\left|\left[\overline{K}\right]-\left[\overline{I}\right]\right| \\
&=\left|\begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]}{1+2dt\ \alpha} & \frac{c^2\frac{dt}{dh^2}[K]}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & [I]_{I\times I} \end{bmatrix}-\lambda_{\overline{k}}\left[\overline{I}\right]\right| \\
&=\left|\begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]}{1+2dt\ \alpha}-\lambda_{\overline{k}}[I]_{I\times I} & \frac{c^2\frac{dt}{dh^2}[K]}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & [I]_{I\times I}-\lambda_{\overline{k}}[I]_{I\times I} \end{bmatrix}\right| \\
&=\left|\begin{bmatrix} (1-(1+2dt\ \alpha))\frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]}{1+2dt\ \alpha} & \frac{c^2\frac{dt}{dh^2}[K]}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & (1-\lambda_{\overline{k}})[I]_{I\times I} \end{bmatrix}\right|,
\end{aligned}
$$

that is

$$
\begin{aligned}
&\left|\left[\overline{K}\right]-\left[\overline{I}\right]\right| \\
&=\left|\frac{1}{1+2dt\ \alpha}\left(\left(1-\lambda_{\overline{k}}\right)\left(1-(1+2dt\ \alpha)\lambda_{\overline{k}}\right)[I]+(1-\lambda_{\overline{k}})c^2\frac{dt^2}{dh^2}[K]-c^2\frac{dt^2}{dh^2}[K]\right)\right| \\
&=\left(\frac{1}{1+2dt\ \alpha}\right)^I\left|\left(1-\lambda_{\overline{k}}\right)\left(1-(1+2dt\ \alpha)\lambda_{\overline{k}}\right)[I]-\lambda_{\overline{k}}c^2\frac{dt^2}{dh^2}[K]\right| \\
&=\left(\frac{1}{1+2dt\ \alpha}\lambda_{\overline{k}}c^2\frac{dt^2}{dh^2}\right)^I\left|\frac{\left(1-\lambda_{\overline{k}}\right)\left(1-(1+2dt\ \alpha)\lambda_{\overline{k}}\right)}{\lambda_{\overline{k}}c^2\frac{dt^2}{dh^2}}[I]-[K]\right|,
\end{aligned}
$$

thus, the characteristic equation of $\left[\overline{K}\right]$ is

$$
\left|\frac{\left(1-\lambda_{\overline{k}}\right)\left(1-(1+2dt\ \alpha)\lambda_{\overline{k}}\right)}{\lambda_{\overline{k}}c^2\frac{dt^2}{dh^2}}[I]-[K]\right|=0. \tag{3.2.15}
$$

We also know that the characteristic equation of $[K]$ is

$$
\left|\lambda_k[I]-[K]\right|=0. \tag{3.2.16}
$$

Comparing the characteristic equations of $[K]$ and $\left[\overline{K}\right]$, namely Eq. (3.2.16) and Eq. (3.2.15), we derive that

$$
\lambda_k=\frac{\left(1-\lambda_{\overline{k}}\right)\left(1-(1+2dt\ \alpha)\lambda_{\overline{k}}\right)}{\lambda_{\overline{k}}c^2\frac{dt^2}{dh^2}},
$$

which, recalling the definition of the Courant number $\lambda \triangleq cdt/dh$, can be rewritten as

$$\lambda_k = \frac{(1-\lambda_{\overline{k}})\,(1-(1+2dt\;\alpha)\lambda_{\overline{k}})}{\lambda_{\overline{k}}\lambda^2},$$

$$\begin{aligned}\lambda^2\lambda_k\lambda_{\overline{k}} &= (1-\lambda_{\overline{k}})\,(1-(1+2dt\;\alpha)\lambda_{\overline{k}})\\ &= 1-(1+2dt\;\alpha)\lambda_{\overline{k}}-\lambda_{\overline{k}}+(1+2dt\;\alpha)\lambda_{\overline{k}}^2,\end{aligned}$$

which results in the following quadratic equation:

$$(1+2dt\;\alpha)\lambda_{\overline{k}}^2-(2+2dt\;\alpha+\lambda_k\lambda^2)\lambda_{\overline{k}}+1=0. \tag{3.2.17}$$

Let's now compute the roots of Eq. (3.2.17):

$$\begin{aligned}\lambda_{\overline{k},\pm} &= 1+dt\;\alpha+\frac{1}{2}\lambda_k\lambda^2\pm\sqrt{\left(1+dt\;\alpha+\frac{1}{2}\lambda_k\lambda^2\right)^2-1}\\ &= 1+dt\;\alpha+\frac{1}{2}\lambda_k\lambda^2\pm\sqrt{dt^2\alpha^2+2dt\;\alpha+\frac{1}{4}\lambda_k^2\lambda^4+(1+dt\;\alpha)\lambda_k\lambda^2}.\end{aligned}$$

Recalling Th. 3.2.1, the scheme is stable if the absolute value of $\lambda_{\overline{k},\pm}$ is less than 1 for $\lambda_k=-\rho([K])_{max}$ (see Eq. 3.2.12). Hence, the roots can be expressed as

$$\lambda_{\overline{k},\pm}=1+dt\;\alpha-\frac{\rho([K])_{max}}{2}\lambda^2\pm\sqrt{\left(1+dt\;\alpha-\frac{\rho([K])_{max}}{2}\lambda^2\right)^2-1}.$$

Let

$$\Delta\triangleq\left(1+dt\;\alpha-\frac{\rho([K])_{max}}{2}\lambda^2\right)^2-1.$$

In Fig. 3.2.1 we plot the absolute value of the roots $z_\pm$ and Δ as a function of λ for different values of α. Notice how, for $\lambda\lesssim 0.8$, both roots have magnitude less than 1 and $\Delta<0$ for all the considered values of α: this means that the CFL condition might be $|\lambda|\lesssim 0.8$; a more accurate bound will be computed below. We also notice that it would reasonable to consider the case $\alpha=0$ as the worst case, i.e., the case for which the stability criterion would be more strict.

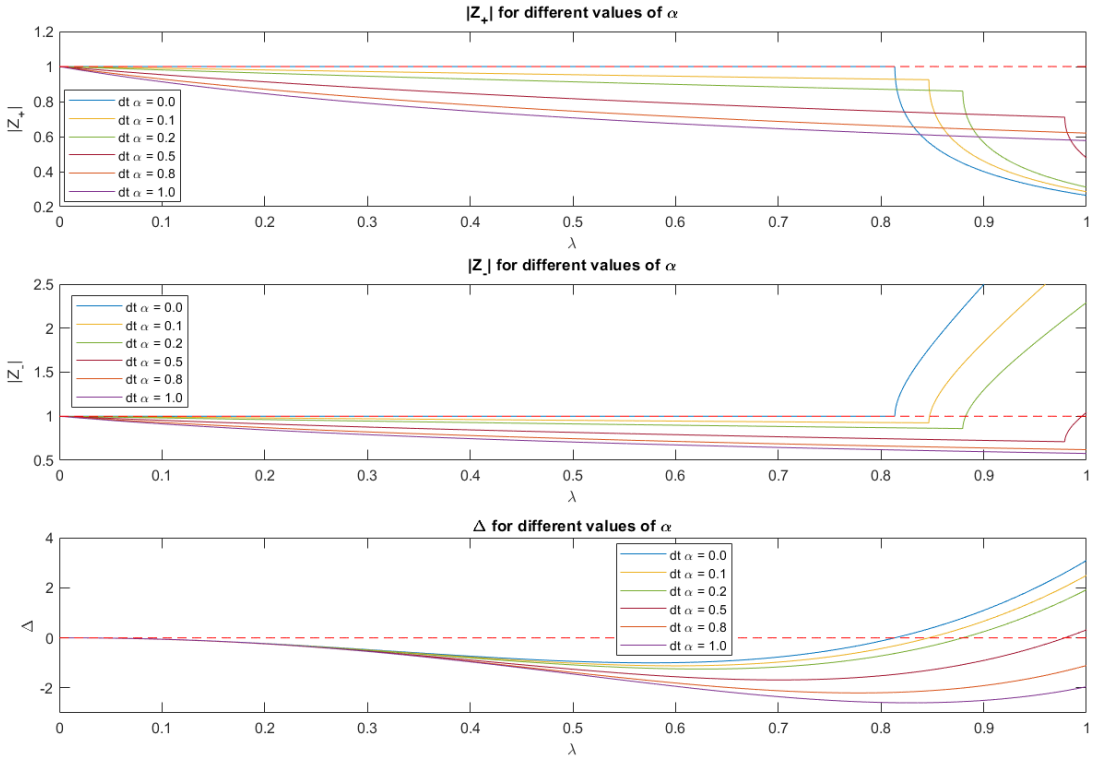


Figure 3.2.1: Absolute value of the roots and Δ as a function of the Courant number λ for different values of the absorption coefficient α. The dashed curve in red in the roots plots represents the stability limit $|z_{\pm}| = 1$, while in the Δ plot it represents $\Delta = 0$.

We want to compute a more accurate stability criterion: we'll consider the worst case $\alpha = 0$. If $\Delta \leqslant 0$, the absolute value of the roots, which are complex conjugate, is

$$\begin{aligned}
\left|\lambda_{\overline{k},\pm}\right| &= \left|1 - \frac{\rho([K])_{max}}{2}\lambda^2 \pm j\sqrt{-\frac{(\rho([K])_{max})^2}{4}\lambda^4 + \rho([K])_{max}\lambda^2}\right| \\
&= \sqrt{\left(1 - \frac{\rho([K])_{max}}{2}\lambda^2\right)^2 - \frac{(\rho([K])_{max})^2}{4}\lambda^4 + \rho([K])_{max}\lambda^2} \\
&= \sqrt{1 + \frac{(\rho([K])_{max})^2}{4}\lambda^4 - \rho([K])_{max}\lambda^2 - \frac{(\rho([K])_{max})^2}{4}\lambda^4 + \rho([K])_{max}\lambda^2} = 1.
\end{aligned}$$

It can be proven that, for $\Delta \geqslant 0$, the condition $|\lambda_{\overline{k},\pm}| < 1$ cannot be satisfied. This means that, to ensure the stability of the scheme in the undamped case, we must satisfy the

condition $\Delta \leqslant 0$, i.e.,

$$\left(1 - \frac{\rho([K])_{max}}{2}\lambda^2\right)^2 - 1 \leqslant 0,$$
$$-1 \leqslant 1 - \frac{1}{2}\rho([K])_{max}\lambda^2 \leqslant 1,$$
$$1 - \frac{1}{2}\rho([K])_{max}\lambda^2 \geqslant -1 \quad \wedge \quad 1 - \frac{\rho([K])_{max}}{2}\lambda^2 \leqslant 1,$$
$$-\rho([K])_{max}\lambda^2 \geqslant -4 \quad \wedge \quad \rho([K])_{max} \geqslant 0,$$

that is

$$\lambda^2 \leqslant \frac{4}{\rho([K])_{max}},$$
$$|\lambda| \leqslant \frac{2}{\sqrt{\rho([K])_{max}}},$$

which results in the stability criterion of first order FDTD:

$$|\lambda| \leqslant \frac{2}{\sqrt{2|A| + 2|B| + 2|C| + |D|}} = \sqrt{\frac{2}{|A| + |B| + |C| + \frac{|D|}{2}}} = \frac{3\sqrt{85}}{34} \approx 0.8135. \quad (3.2.18)$$

Eq. (3.2.18) corresponds exactly to the stability criterion of the second order FDTD scheme in Eq. (3.1.12). With $\alpha > 0$, the CFL condition would be higher, however we're considering the worst case.

Dissipation and dispersion analysis

The effects of dispersion of the first order FDTD $(1, 6)$ scheme in Eq. (3.2.6) are illustrated in Fig. 3.2.2, in which we show the wave packet at $t = 0$ (a Gaussian distribution of width 1/500) and at $t = 1$ for different values of λ. In Fig. 3.2.3 and Fig. 3.2.4 we show the frequency response (respectively magnitude and phase) for different values of λ. In Fig. 3.2.5 we show the DCT of the wave packet at $t = 0$ and at $t = 1$ for different values of λ. We notice that the dissipation and dispersion characteristic of this scheme are worse than that of the second order FDTD $(2, 6)$ scheme in Eq. (3.1.3), but similar considerations hold. Again, we notice that, for $\lambda \to 0.8135\ldots$, the higher-frequency components surpass the wavefront and the gross speed of the wave packet is faster as well. The best results (less dissipation and less dispersion) are obtained for low values of λ, and the artifacts gets worse for $\lambda \to 0.8135\ldots$.

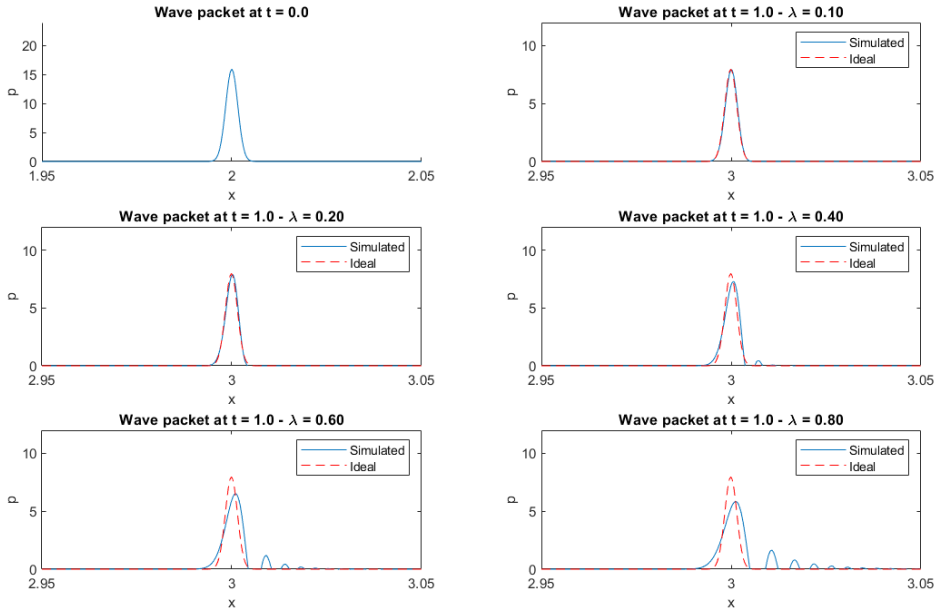


Figure 3.2.2: The snapshot in the northwest corner depicts the wave packet at $t = 0$, a Gaussian distribution with a width of 1/500. The remaining snapshots depict the output of first order FDTD $(1, 6)$ at $t = 1$ for different values of λ. The wave speed c is set to 1, $\alpha = 0$ is applied, and the spatial step size is $dh = 4e - 4$. The ground truth solution is displayed using dashed lines.

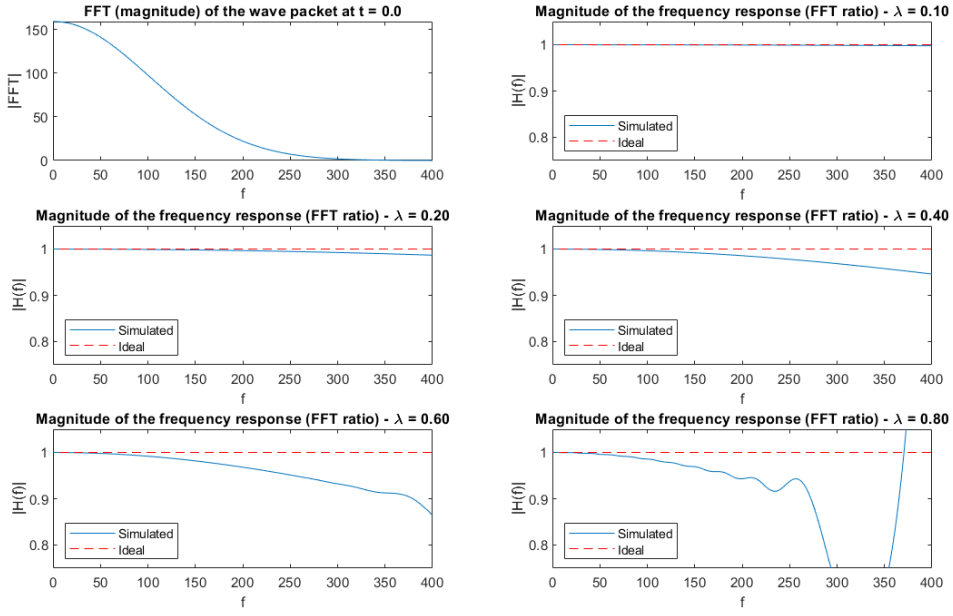


Figure 3.2.3: The northwest image represents the magnitude of the FFT of the wave packet at $t = 0$. The other figures depict the magnitude of the frequency response of first order FDTD $(1, 6)$ for different values of λ. The ground truth solution is displayed using dashed lines.

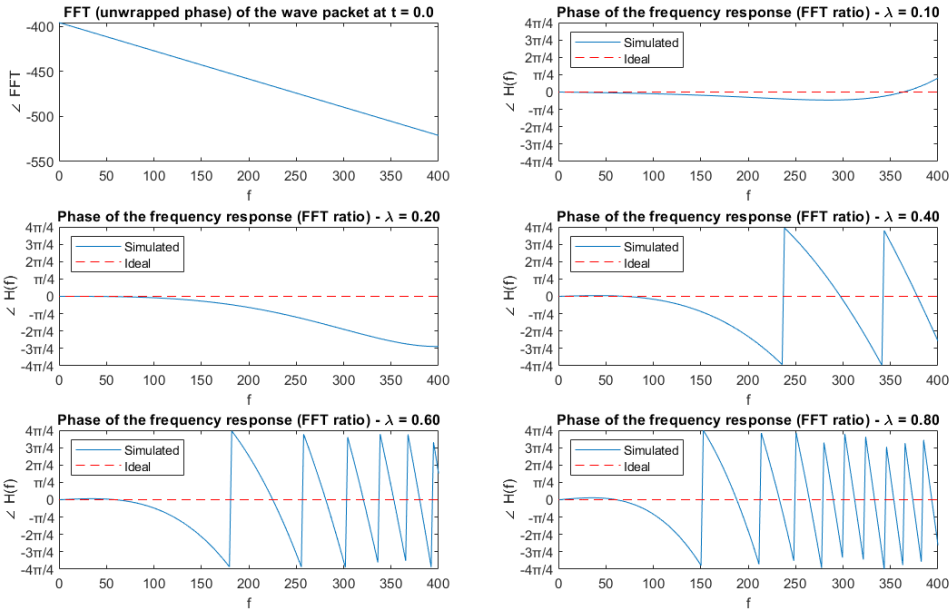


Figure 3.2.4: The northwest image represents the unwrapped phase of the FFT of the wave packet at $t = 0$. The other figures depict the phase of the frequency response of first order FDTD $(1, 6)$ for different values of λ. The ground truth solution is displayed using dashed lines.

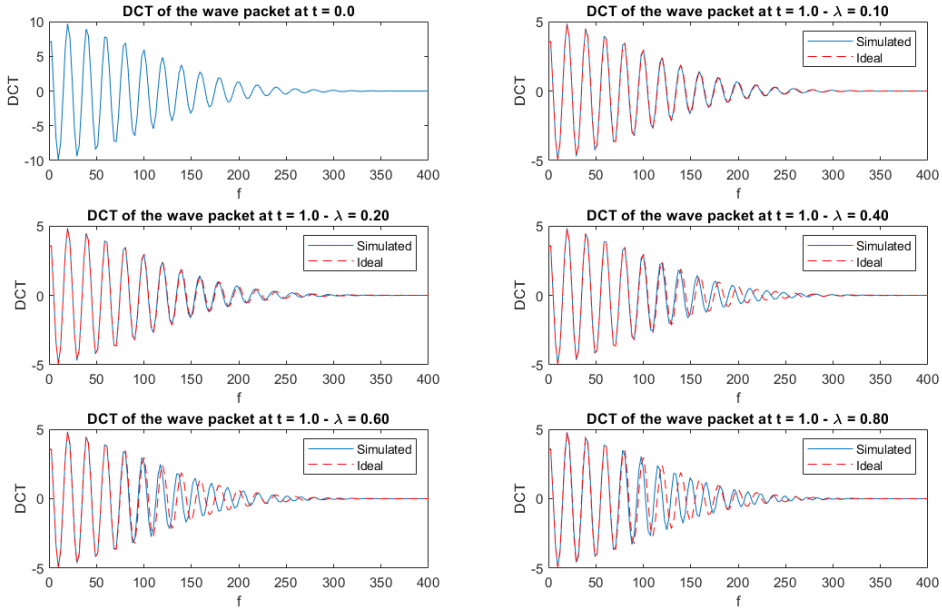
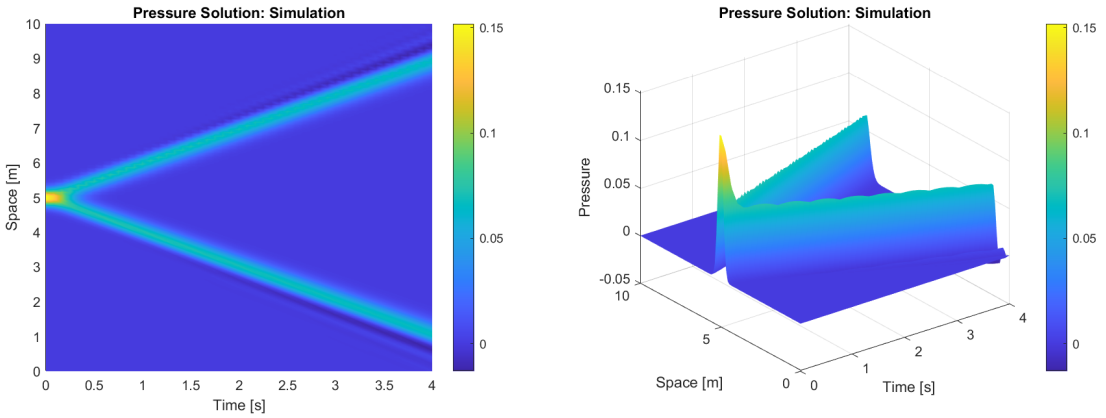


Figure 3.2.5: The northwest snapshot represents the DCT of the wave packet at $t = 0$. The other snapshots depict the DCT of the output from first order FDTD $(1, 6)$ for different values of λ. The ground truth solution is displayed using dashed lines.

3.2.2. Simulation of a test case

Consider the propagating wave test case in Subsec. 2.1.7. The results of the simulation using first order FDTD with $dh = 0.1$ and $\lambda = 0.8$ are shown in Fig. 3.2.6.



(a) Top view.

(b) 3D view.

Figure 3.2.6: propagating wave test case simulated with first order FDTD $(1, 6)$ with parameters $dh = 0.1$ and $\lambda = 0.8$.

3.2.3. Convergence test

Consider the standing wave test case in Subsec. 2.1.7. The results of convergence test on first order FDTD with $dh = 1e-2$ and different values of dt are shown in Fig. 3.2.7: for each value of dt, we show the values of L^1, L^2, and L^∞ norm (Def. 2.3.1) of the error (difference between numerical solution and ground truth at the last time instant). Analogously, in Fig. 3.2.8, we show the results of convergence test with $dt = 1e-3$ and different values of dh.

Notice that the theoretical order of accuracy is not $\mathcal{O}(dt + dh^6)$ as one would expect. It has been found that the best fitting order of accuracy for first order FDTD $(1, 6)$ is $\mathcal{O}(dt + dh^3)$.

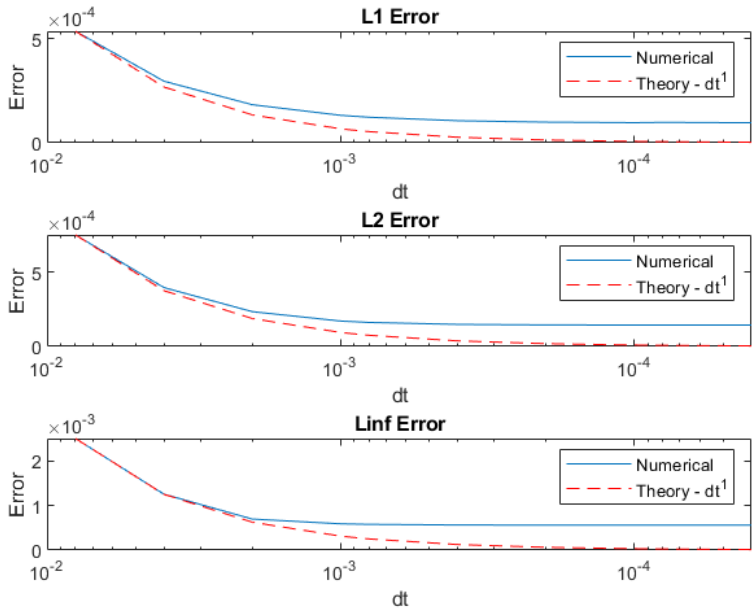


Figure 3.2.7: Convergence test on first order FDTD $(1, 6)$ with $dh = 1e-2$ and different values of dt. The red curve represents the theoretical convergence behavior.

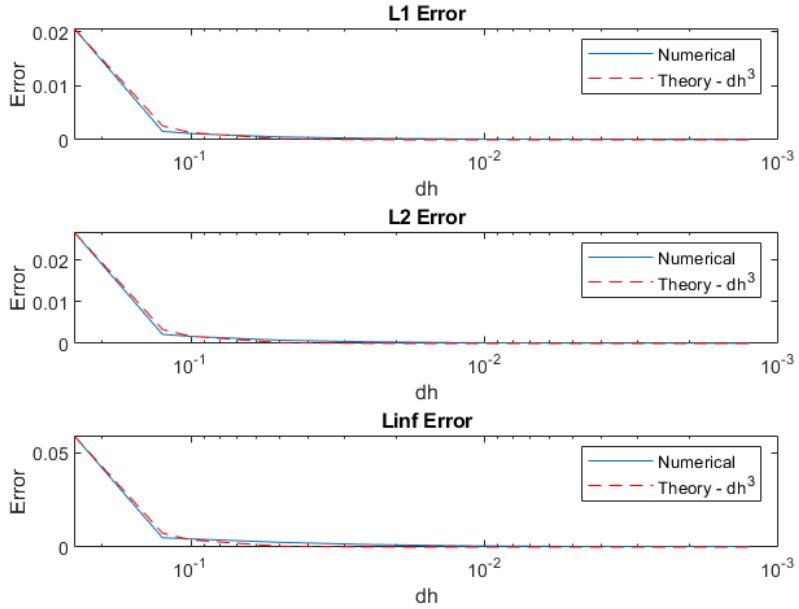


Figure 3.2.8: Convergence test on first order FDTD $(1, 6)$ with $dt = 1e-3$ and different values of dh. The red curve represents the theoretical convergence behavior.

3.3. Treatment of boundary conditions

In this paragraph, we show how to impose the Neumann boundary conditions for the previously derived FDTD schemes in the 1D case.
Consider the following discretization of the x-axis with constant step size dh:

$$x_i = i \cdot dh, \quad i = 0, \dots, I-1,$$

where x_i is the i-th grid point. We want to impose the following homogeneous conditions at the left boundary:

$$\left.\frac{\partial p}{\partial x}\right|_{x=x_0,t} = 0.$$

The right boundary condition can be treated similarly.
Consider the 6-th order accurate centered stencil for the second space derivative of Eq. (3.1.1). The second space derivative evaluated at the leftmost point x_0 is approximated as follows:

$$\left.\frac{\partial^2 p}{\partial x^2}\right|_{\underline{x}=\underline{x}_1,t=t^n} \approx \frac{Ap^n_{-3} + Bp^n_{-2} + Cp^n_{-1} + Dp^n_0 + Cp^n_1 + Bp^n_2 + Ap^n_3}{dh^2}.$$

As the points x_{-3}, x_{-2}, x_{-1} are not defined, we impose even symmetry about $x_{-1/2}$ [†]:

$$\begin{cases} p^n_{-1} = p^n_0, \\ p^n_{-2} = p^n_1, \\ p^n_{-3} = p^n_2, \end{cases} \tag{3.3.1}$$

so that the stencil becomes

$$\left.\frac{\partial^2 p}{\partial x^2}\right|_{\underline{x}=\underline{x}_0,t=t^n} \approx \frac{(D+C)p^n_0 + (C+B)p^n_1 + (B+A)p^n_2 + Ap^n_3}{dh^2}.$$

[†]As will become clearer in Ch. 5, to derive a Domain Decomposition scheme, the type of even symmetry assumed for the FDTD method must be the same as the one defined by the DCT-II (see Eq. 2.2.22) employed for the Fourier method (Ch. 4).

The stencils to evaluate the second space derivative at the points x_1, x_2 can be derived analogously. Thus, it can be proven that the matrix $[K]$ becomes

$$\begin{bmatrix} D+C & C+B & B+A & A & & & & \\ C+B & D+A & C & B & A & & & \\ B+A & C & D & C & B & A & & \\ A & B & C & D & C & B & A & \\ & & & & & & & \ddots \end{bmatrix}.$$

4 | Simulation of the wave equation through the Fourier method

In this chapter, we focus on the Fourier method, an efficient numerical technique that relies on the use of discrete transforms to simulate the acoustic wave equation. We will focus on the Discrete Cosine Transform (DCT), a transform that is particularly well-suited for simulating waves that are a combination of cosines. We will explore how to derive the acoustic wave equation in the DCT-time domain, and how to discretize it. We will explain why the homogeneous Neumann boundary conditions are necessary for this method, as other boundary conditions can result in spectral leakage and cause numerical inaccuracies.

4.1. Choice of the discrete transform

We want to select a discrete transform which let us express the solution $p(\underline{x}, t)$ in terms of the mode shapes for the homogeneous Neumann boundary conditions. As explained in Bull and Zhang [2021], the discrete Fourier transform (DFT, introduced in Subsec. 2.2.2) can serve as a suitable starting point for our purpose. After all, we know that the DFT is well matched to sinusoidal data. However, some complications arise.
With the DFT, a windowed [†] finite-length data sequence is naturally extended by periodic extension prior to transformation. Applying the discrete-time Fourier series (DTFS) to this new periodic function produces the DFT coefficients: this is quite useful for capturing the magnitude and phase of the underlying signal in the frequency domain.
The extension process using a rectangular window introduces discontinuities, which cause spectral leakage or ripples in the frequency domain: this can be addressed by using smoother windows, which however produce a smearing effect worse than in the case of rectangular window which has a narrower main lobe. All these effects distort the characteristics of the underlying signal, and the original signal is not recoverable anymore, which is against our goal of accurate simulation.

[†]The DFT implicitly applies a rectangular window on the time-domain signal.

We now state the desirable characteristic of our discrete transform:

- Good energy compaction, so that truncating the coefficients introduces negligible errors.
- Orthonormality, to support energy balancing and for a fast implementation of the inverse transform.
- Basis functions corresponding to the mode shapes obtained with the homogeneous Neumann boundary conditions. As a bonus, the basis functions will be independent of the signal, avoiding the need to recompute the basis functions at each iteration.
- Minimal spectral leakage, to not degradate the accuracy of the simulation.
- Separability, to reduce complexity for fast 3D computations.
- Real-valued basis functions and coefficients, to ease arithmetic complexity and support simple matrix transposition.

We anticipate that the DCT, introduced in Subsec. 2.2.2, satisfies all these requirements. Like the DFT, the DCT provides information about a signal in the frequency domain. However, unlike the DFT, the DCT of a real-valued signal is itself real-valued and importantly it also does not introduce artifacts due to periodic extension of the input data. A comparison is shown in Fig. 4.1.1.

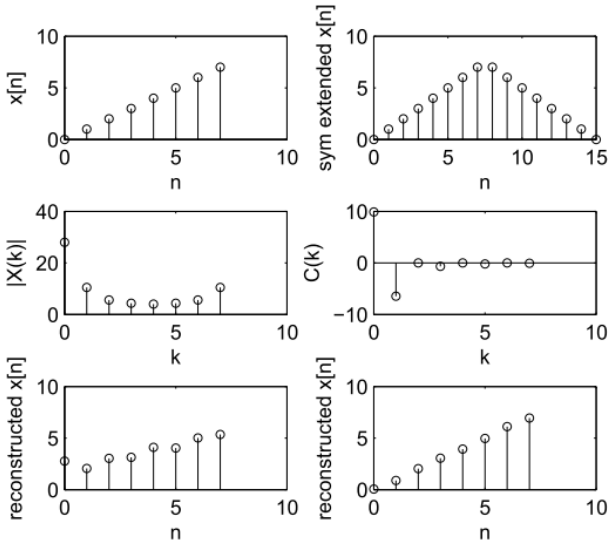


Figure 4.1.1: Comparison of DFT and DCT. Top: Input signal $x[n]$ and the symmetrically extended sequence $x_1[n]$. Middle: Eight-point DFT (magnitude only) and the eight-point DCT. Bottom: inverse DFT (magnitude only) and inverse DCT. Source: Bull and Zhang [2021].

With the DFT, a finite-length data sequence is naturally extended by periodic extension. The DCT avoids the artifacts introduced by this periodic extension by symmetrically extending the signal prior to application of the DFT. This produces an even sequence which has the added benefit of yielding real-valued coefficients.
As shown in Fig. 2.2.3, the basis functions are analogous to the mode shapes with the homogeneous Neumann boundary conditions and are independent of the signal. As the solution of the acoustic wave equation is a linear combination of the mode shapes, and these mode shapes correspond exactly to the basis functions of the DCT, spectral leakage is dramatically reduced.
The proof of the remaining characteristics is out of the scope of this work.

4.2. Discretization of the unviscid wave equation in the Fourier-time domain

In Subsec. 2.4.3 we have derived the acoustic wave equation in the Fourier-time domain with the homogeneous Neumann boundary conditions (see Eq. 2.4.18), which we report for readability:

$$\frac{d^2 P_{mnp}(t)}{dt^2} + k_{mnp}^2 c^2 P_{mnp}(t) = F_{mnp}(t), \quad m \in \mathbb{N}^+, \;\; n \in \mathbb{N}^+, \;\; p \in \mathbb{N}^+.$$

To represent the DCT coefficients $P_{mnp}(t)$ and $F_{mnp}(t)$ in a calculator, it's convenient to introduce the global index i defined as

$$i \triangleq p\frac{L_y L_x}{dh^2} + n\frac{L_x}{dh} + m + 1.$$

Then, we truncate the index sequence so that $i = 1, 2, \ldots, I$ with $\omega_I \geqslant \omega_{max}$, where ω_{max} is the highest radian frequency we're interested in.
We get

$$\frac{d^2 P_i(t)}{dt^2} + \omega_i^2 P_i(t) = F_i(t), \quad i = 1, \ldots, I, \tag{4.2.1}$$

where ω_i can be expressed as

$$\omega_i = \omega_{mnp} = c\pi\sqrt{\frac{m^2}{L_x^2} + \frac{n^2}{L_y^2} + \frac{p^2}{L_z^2}},$$

with $\omega_1 = \omega_{0,0,0} = 0$, Cfr. Eq. (2.2.26).
Eq. (4.2.1) in vector notation becomes

$$\frac{d^2\underline{P}(t)}{dt^2} + \underline{\omega}^2 \odot \underline{P}(t) = \underline{F}(t), \tag{4.2.2}$$

where $\underline{P}$, $\underline{F}$, and $\underline{\omega}^2$ [†] are I-length vectors, while $\odot$ represents the *Hadamard product* (also known as elementwise product). In case we are interested in the pressure velocity (defined in Eq. (2.1.2)), we introduce the I-length vector $\underline{V}$ storing the DCT coefficients of the pressure velocity.
To convert the pressure field values at a given time $p(\underline{x}, \hat{t})$ to the modal coefficients $\underline{P}(\hat{t})$ and back, we employ respectively the DCT and the iDCT:

$$\underline{P}(\hat{t}) = \text{DCT}\left\{p(\underline{x}, \hat{t})\right\},$$
$$p(\underline{x}, \hat{t}) = \text{iDCT}\left\{\underline{P}(\hat{t})\right\}.$$

An analogous transformation is defined for the force terms and for the pressure velocity:

$$\underline{F}(\hat{t}) = \text{DCT}\left\{f(\underline{x}, \hat{t})\right\},$$
$$f(\underline{x}, \hat{t}) = \text{iDCT}\left\{\underline{F}(\hat{t})\right\},$$

$$\underline{V}(\hat{t}) = \text{DCT}\left\{v(\underline{x}, \hat{t})\right\},$$
$$v(\underline{x}, \hat{t}) = \text{iDCT}\left\{\underline{V}(\hat{t})\right\}.$$

4.2.1. Time integration

In Sec. 2.4 we have introduced the exact time integration of the forced harmonic oscillator. As the DCT-time domain wave equation 4.2.2 represents I independent single degree-of-freedom systems, its integration is trivial. In particular:

- The equation associated to the first mode (where $\omega_1 = 0$) is

$$\ddot{P}_1(t) = F_1(t),$$

  which is solved by the following (second order finite-difference) scheme:

$$P_1^{n+1} = 2P_1^n - P_1^{n-1} + dt^2 F_1^n, \quad n = 1, 2, \dots \ ,$$

[†]The notation can be confusing; the square is applied elementwise. In other words, $\underline{\omega}^2$ is the vector storing the squared natural radian frequencies.

where the coefficients have been inferred from Tab. 2.3.1. Note that this time-stepping scheme is exact in time.

- The remaining equations represent independent forced harmonic oscillators of natural frequency ω_i, $i = 2, \dots, I$, whose time-stepping scheme is defined in Eq. (2.4.9). However, it's important to note that this time-stepping scheme is exact under the assumption of $F(t)$ constant over a time step dt: this is not a problem for input source sounds, as the time step is necessarily below the sampling rate of the input signal (Raghuvanshi et al. [2011]).

Hence, exact time integration of the unviscid acoustic wave equation in the DCT-time domain is formulated as

$$\begin{cases} P_1^{n+1} = 2P_1^n - P_1^{n-1} + dt^2 F_1^n, \\ P_i^{n+1} = 2P_i^n \cos(\omega_i dt) - P_i^{n-1} + \dfrac{2}{\omega_i^2} F_i^n (1 - \cos(\omega_i dt)), \quad i = 2, \dots, I, \end{cases} \tag{4.2.3}$$

for $n = 1, 2, \dots$.
To update the pressure velocity, we employ *explicit Euler* for the first mode and Prop. 2.4.3 for the remaining modes, obtaining the following scheme:

$$\begin{cases} V_1^n = \dfrac{P_1^{n+1} - P_1^n}{dt}, \\ V_i^n = \dfrac{\omega_i}{\sin(\omega_i dt)} (P_i^{n+1} - \cos(\omega_i dt) P_i^n) - \dfrac{1}{\omega_i} \tan\left(\dfrac{\omega_i dt}{2}\right) F_i^n, \quad i = 2, \dots, I, \end{cases} \tag{4.2.4}$$

for $n = 1, 2, \dots$.

4.2.2. Stability, dissipation and dispersion analysis

We perform Von Neumann analysis (see Subsec. 2.3.3) on the second order Fourier scheme in Eq. (4.2.3). We obtain the following characteristic equation:

$$z - 2\cos(\omega_i dt) + z^{-1} = 0, \tag{4.2.5}$$

hence the roots are

$$z_\pm = \cos(\omega_i dt) \pm \sqrt{\cos^2(\omega_i dt) - 1} = \cos(\omega_i dt) \pm j \sin(\omega_i dt).$$

Stability is achieved when the roots are of magnitude less or equal than 1. We have that $|z_{\pm}| = 1$, in fact

$$|z_{\pm}| = \cos^2(\omega_i dt) + \sin^2(\omega_i dt) = 1.$$

This means that the scheme is unconditionally stable.
Starting from the characteristic equation in Eq. (4.2.5), we want to derive the dispersion relation imposing $z = e^{j\omega dt}$:

$$2\cos(\omega dt) - 2\cos(\omega_i dt) = 0,$$

$$\cos(\omega dt) = \cos(\omega_i dt),$$

$$\omega = \omega_i,$$

hence the second order Fourier scheme is not dispersive, meaning that $v_\phi = v_g = c$.
The effects of dispersion are illustrated in Fig. 4.2.1, in which we show the wave packet at $t = 0$ (a Gaussian distribution of width 1/500) and at $t = 1$ for different values of λ. Notice that, for all values of λ, there are no artifacts (except, for $\lambda \neq 0.5$, for a delay inherent to the discrete time nature of the numerical solution) and the numerical solution closely matches the ground truth solution. Notice also that the speed of the wave packet is c as desired. We recall that the expected amplitude of the wave packet at the final time of simulation is half of that at the start, because the wave is split in two propagating waves.
In Fig. 4.2.2 and Fig. 4.2.3 we show the frequency response (respectively magnitude and phase) for different values of λ, from which we can infer the dissipation and dispersion characteristic of the scheme. They are obtained by performing the ratio between the FFT of the wave packet at $t = 1$ and the one at $t = 0$. We notice that, except for a linear phase difference for $\lambda \neq 0.5$, there's no dispersion and no dissipation.
Further confirmations of the behavior of the scheme (3.1.3) are found in Fig. 4.2.4, in which we show the DCT of the wave packet at $t = 0$ and at $t = 1$ for different values of λ.

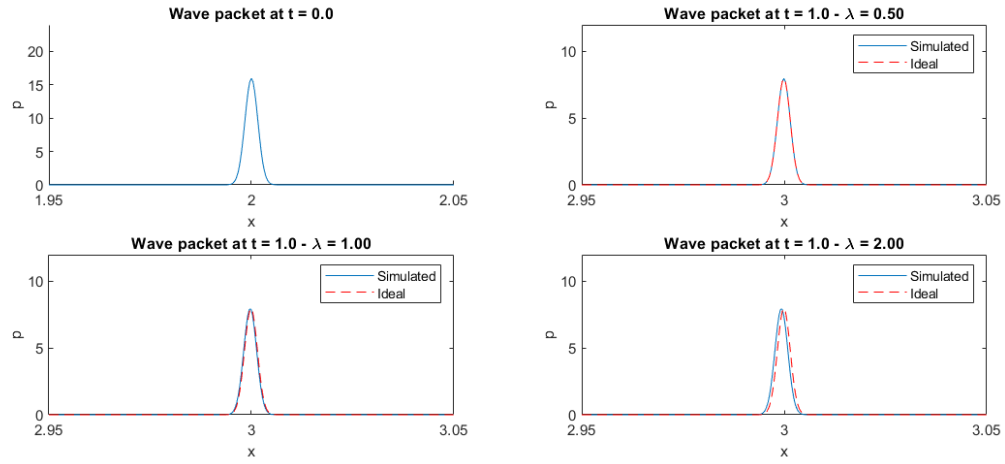


Figure 4.2.1: The snapshot in the northwest corner depicts the wave packet at $t = 0$, a Gaussian distribution with a width of 1/500. The remaining snapshots depict the output of the second order Fourier method at $t = 1$ for different values of λ. The wave speed c is set to 1, $\alpha = 0$, and the spatial step size is $dh = 4e - 4$. The ground truth solution is displayed using dashed lines.

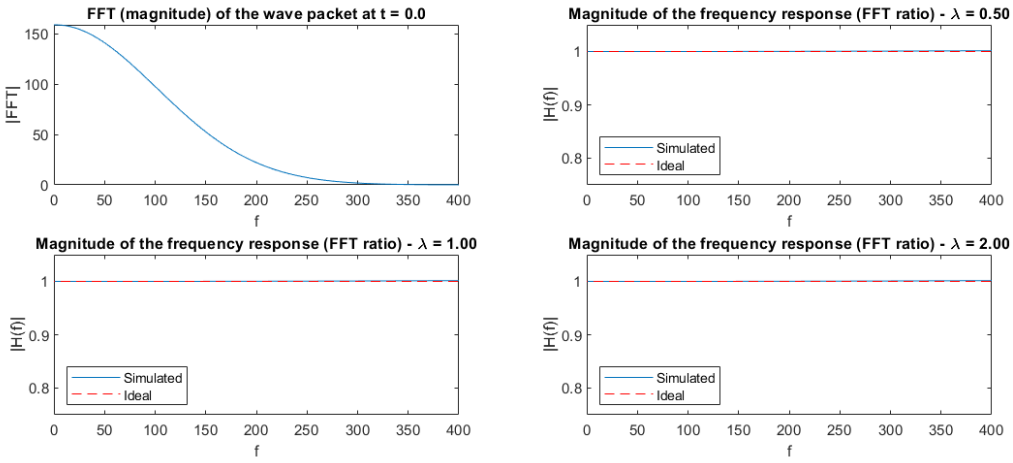


Figure 4.2.2: The northwest image represents the magnitude of the FFT of the wave packet at $t = 0$. The other figures depict the magnitude of the frequency response of the second order Fourier method for different values of λ. The ground truth solution is displayed using dashed lines.

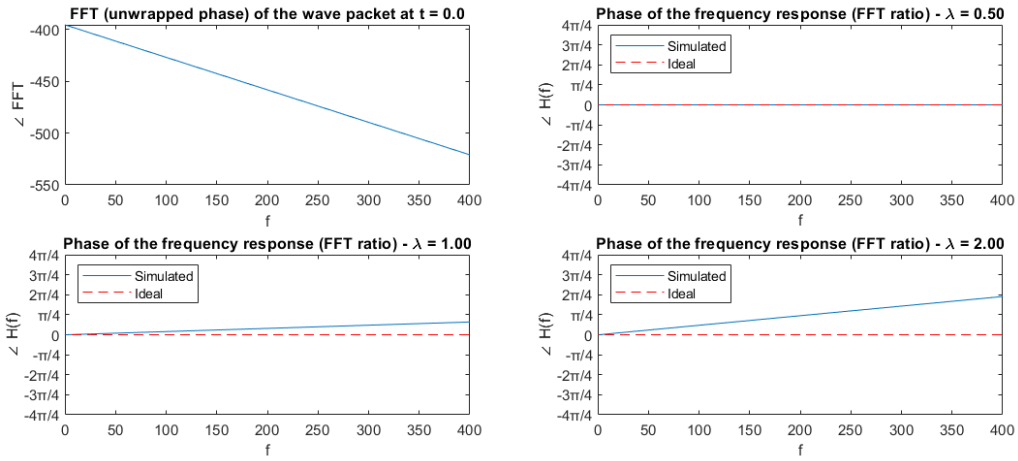


Figure 4.2.3: The northwest image represents the unwrapped phase of the FFT of the wave packet at $t = 0$. The other figures depict the phase of the frequency response of the second order Fourier method for different values of λ. The ground truth solution is displayed using dashed lines.

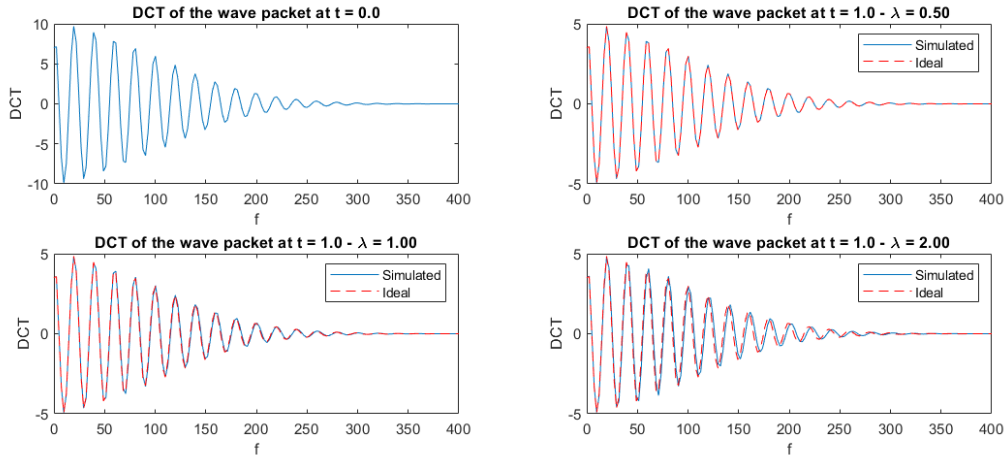
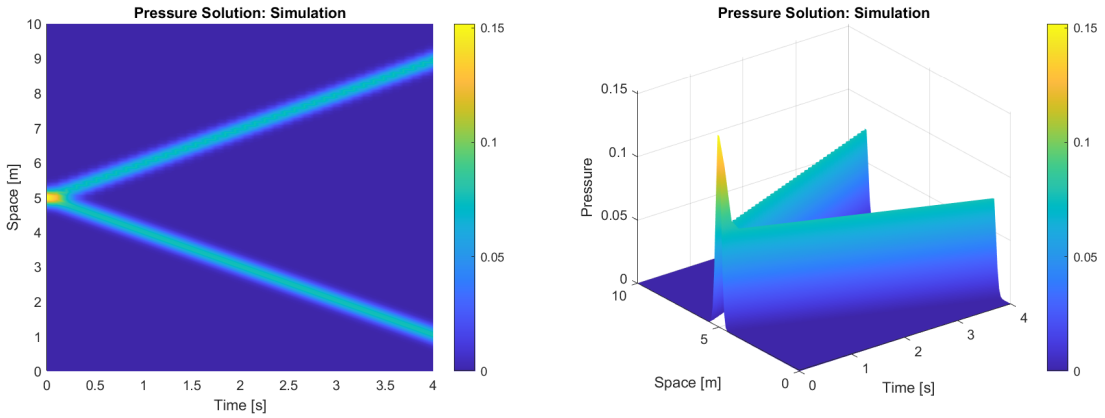


Figure 4.2.4: The northwest snapshot represents the DCT of the wave packet at $t = 0$. The other snapshots depict the DCT of the output from the second order Fourier method for different values of λ. The ground truth solution is displayed using dashed lines.

4.2.3. Simulation of a test case

Consider the propagating wave test case in Subsec. 2.1.7. The results of the simulation using the second order Fourier method with $dh = 0.1$ and $\lambda = 1$ are shown in Fig. 4.2.5.



(a) Top view.

(b) 3D view.

Figure 4.2.5: propagating wave test case simulated with the second order Fourier method with parameters $dh = 0.1$ and $\lambda = 1$.

4.2.4. Convergence test

Consider the standing wave test case in Subsec. 2.1.7. The results of convergence test on the second order Fourier method with $dh = 1e-2$ and different values of dt are shown in Fig. 4.2.6: for each value of dt, we show the values of L^1, L^2, and L^∞ norm (Def. 2.3.1) of the error (difference between numerical solution and ground truth at the last time instant). Analogously, in Fig. 4.2.7, we show the results of convergence test with $dt = 1e-3$ and different values of dh.

It has been found that the best fitting order of accuracy for the second order Fourier method is $\mathcal{O}(dt + dh)$.

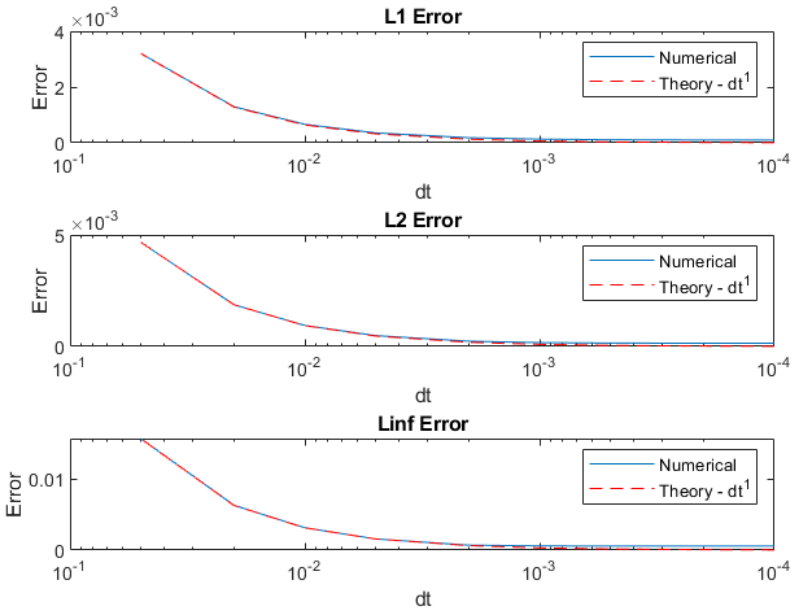


Figure 4.2.6: Convergence test on the second order Fourier method with $dh = 1e-2$ and different values of dt. The red curve represents the theoretical convergence behavior.

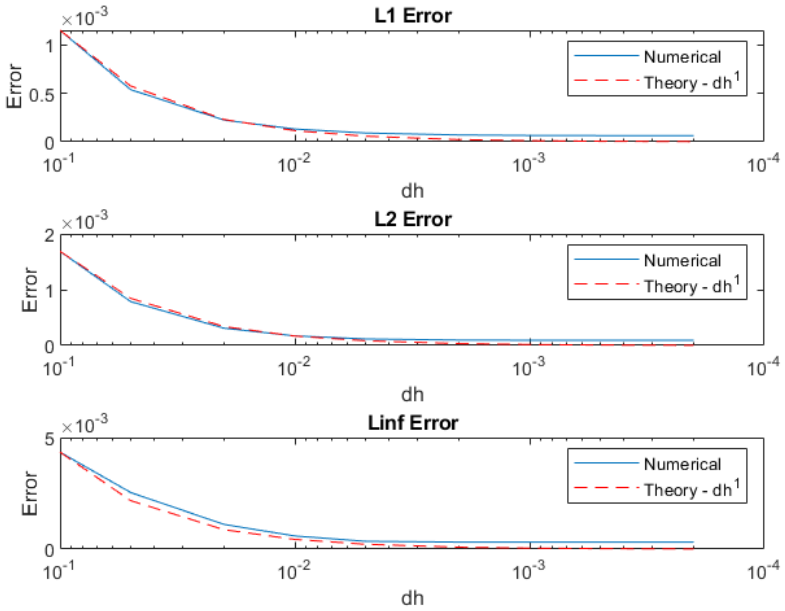


Figure 4.2.7: Convergence test on the second order Fourier method with $dt = 1e-3$ and different values of dh. The red curve represents the theoretical convergence behavior.

4.3. Discretization of the viscous wave equation in the Fourier-time domain

From analogous considerations to Subsec. 4.2, starting from the viscous acoustic wave equation in the Fourier-time domain (see Eq. 2.4.19), we obtain the following DCT-time domain equation:

$$\frac{d^2P_i(t)}{dt^2} + 2\alpha\frac{dP_i(t)}{dt} + \omega_i^2 P_i(t) = F_i(t), \quad i = 1, \dots, I, \tag{4.3.1}$$

which, in vector notation becomes

$$\frac{d^2\underline{P}(t)}{dt^2} + 2\alpha\frac{d\underline{P}(t)}{dt} + \underline{\omega}^2\underline{P}(t) = \underline{F}(t). \tag{4.3.2}$$

This represents I independent forced damped harmonic oscillators. However, in Sec. 2.4, we have shown that to perform time integration of damped harmonic oscillators, it's convenient to express its equation of motion as a first order system. Hence, we employ a first order representation of Eq. (4.3.1), namely

$$\begin{bmatrix} \dot{V}_i(t) \\ \dot{P}_i(t) \end{bmatrix} = \begin{bmatrix} -2\alpha & -\omega_i^2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} V_i(t) \\ P_i(t) \end{bmatrix} + \begin{bmatrix} F_i(t) \\ 0 \end{bmatrix}, \quad i = 1, \dots, I.$$

4.3.1. Time integration

Analogously to Subsec. 4.2.1, we perform exact time integration of the DCT-time domain wave equation in Eq. (4.3.2), which represents I independent single degree-of-freedom systems. In particular:

- The equation associated to the first mode (where $\omega_1 = 0$) is

  $$\ddot{P}_1(t) + 2\alpha\dot{P}_1(t) = F_1(t),$$

  which, expressed as a first order system, corresponds to

  $$\begin{cases} \dot{V}_1(t) = -2\alpha V_1(t) + F_1(t), \\ \dot{P}_1(t) = V_1(t), \end{cases}$$

  which, in turn, is discretized by the following finite-difference scheme (implicit Euler for the pressure velocity DCT coefficient V_1 and explicit Euler for the pressure DCT

coefficient P_1):

$$\begin{cases} \dfrac{V_1^{n+1} - V_1^n}{dt} = -2\alpha V_1^{n+1} + F_1^{n+1}, \\ \dfrac{P_1^{n+1} - P_1^n}{dt} = V_1^n, \end{cases} \qquad n = 0, 1, \dots \ ,$$

or equivalently

$$\begin{cases} V_1^{n+1} = \dfrac{V_1^n + dt F_1^{n+1}}{1 + 2\alpha dt}, \\ P_1^{n+1} = P^n + dt\, V_1^n, \end{cases} \qquad n = 0, 1, \dots \ .$$

- The remaining equations represent independent forced damped harmonic oscillators of natural frequency ω_i, $i = 2, \dots, I$, whose time-stepping scheme is defined in Eq. (2.4.16).

Hence, exact time discretization of the viscous acoustic wave equation in the DCT-time domain is formulated as the two following algorithms, respectively used to update the pressure field and the pressure velocity:

$$\begin{cases} P_1^{n+1} = P_1^n + dt\, V_1^n, \\ P_{i,e} = \dfrac{F_i^n}{\omega_i^2}, & i = 2, \dots, I, \\ P_i^{n+1} = P_{i,e} + e^{-\alpha dt} \left((P_i^n - P_{i,e}) \left(\cos(\omega_i dt) + \dfrac{\alpha}{\omega_i} \sin(\omega_i dt) \right) + \dfrac{\sin(\omega_i dt)}{\omega_i} V_i^n \right), & i = 2, \dots, I, \end{cases} \tag{4.3.3}$$

for $n = 1, 2, \dots$.

$$\begin{cases} V_1^{n+1} = \dfrac{V_1^n + dt F_1^n}{1 + 2\alpha dt}, \\ P_{i,e} = \dfrac{F_i^{n+1}}{\omega_i^2}, & i = 2, \dots, I, \\ V_i^{n+1} = e^{-\alpha dt} \left(V_i^n \left(\cos(\omega_i dt) - \dfrac{\alpha}{\omega_i} \sin(\omega_i dt) \right) - \left(\omega_i + \dfrac{\alpha^2}{\omega_i} \right) (P_i^n - P_{i,e}) \sin(\omega_i dt) \right), & i = 2, \dots, I, \end{cases} \tag{4.3.4}$$

for $n = 1, 2, \dots$.

4.3.2. Stability, dissipation and dispersion analysis

It can be proven that the first order Fourier scheme in Eq. (4.3.3) is unconditionally stable, not dispersive and not dissipative analogously to the second order scheme (see Subsec. 4.2.2). The effects of dispersion are illustrated in Fig. 4.3.1, in which we show the wave packet at $t = 0$ (a Gaussian distribution of width 1/500) and at $t = 1$ for different values of λ. In Fig. 4.3.2 and Fig. 4.3.3 we show the frequency response (respectively magnitude

and phase) for different values of λ. In Fig. 3.2.5 we show the DCT of the wave packet at $t = 0$ and at $t = 1$ for different values of λ.

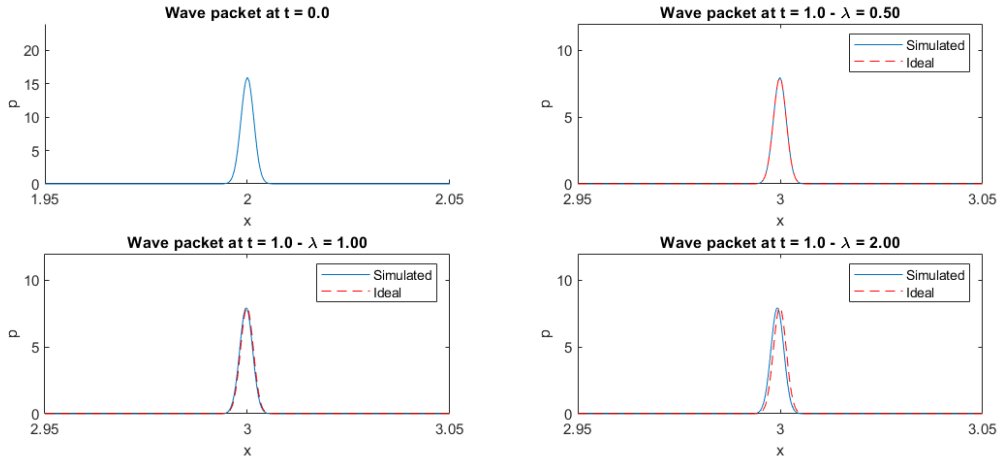


Figure 4.3.1: The snapshot in the northwest corner depicts the wave packet at $t = 0$, a Gaussian distribution with a width of 1/500. The remaining snapshots depict the output of the first order Fourier method at $t = 1$ for different values of λ. The wave speed c is set to 1, $\alpha = 0$, and the spatial step size is $dh = 4e - 4$. The ground truth solution is displayed using dashed lines.

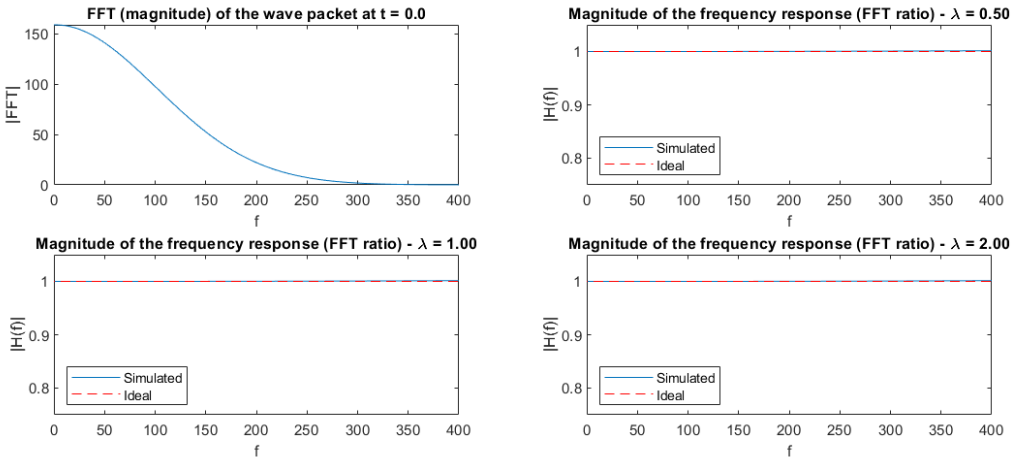


Figure 4.3.2: The northwest image represents the magnitude of the FFT of the wave packet at $t = 0$. The other figures depict the magnitude of the frequency response of the first order Fourier method for different values of λ. The ground truth solution is displayed using dashed lines.

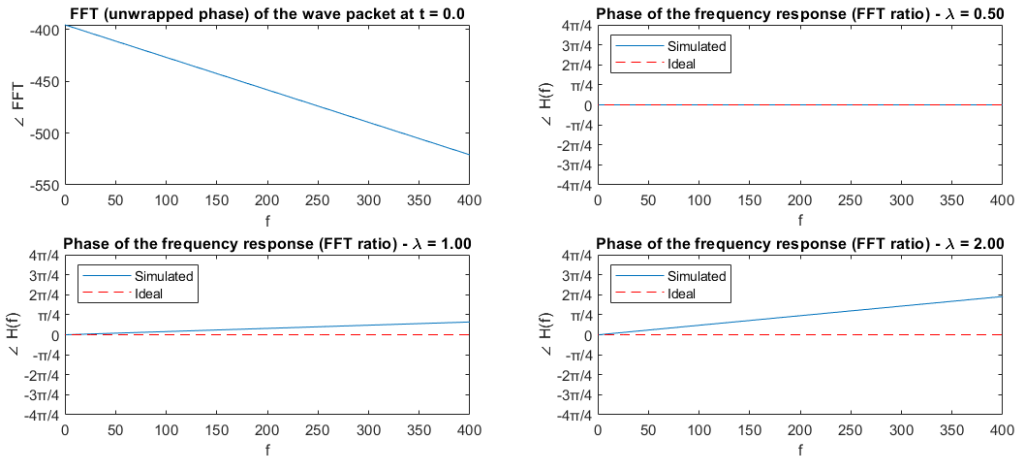


Figure 4.3.3: The northwest image represents the unwrapped phase of the FFT of the wave packet at $t = 0$. The other figures depict the phase of the frequency response of the first order Fourier method for different values of λ. The ground truth solution is displayed using dashed lines.

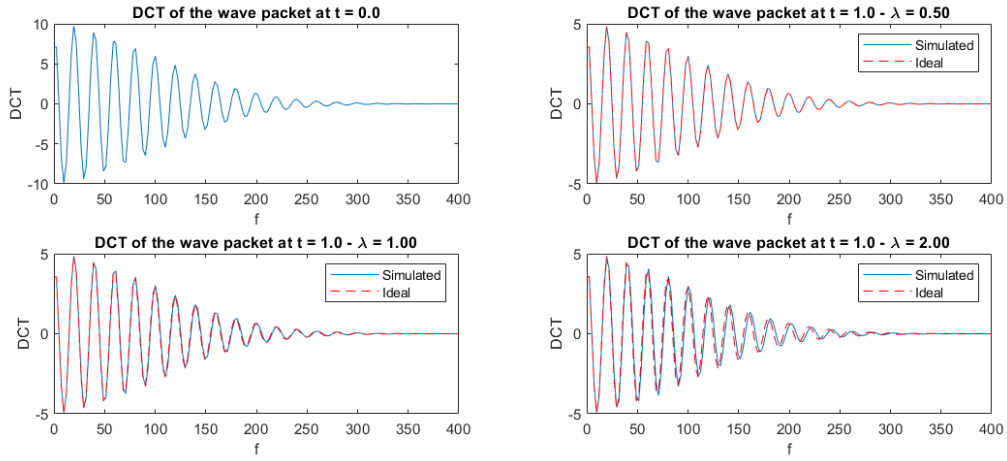
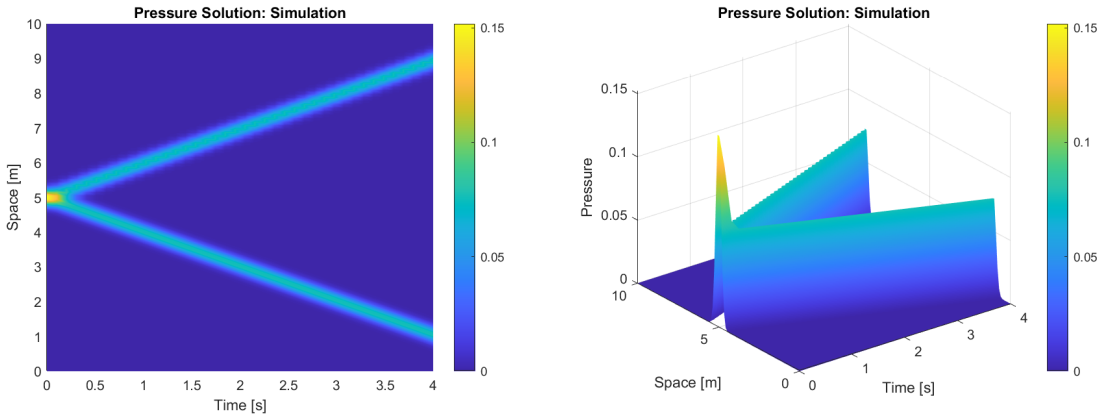


Figure 4.3.4: The northwest snapshot represents the DCT of the wave packet at $t = 0$. The other snapshots depict the DCT of the output from the first order Fourier method for different values of λ. The ground truth solution is displayed using dashed lines.

4.3.3. Simulation of a test case

Consider the propagating wave test case in Subsec. 2.1.7. The results of the simulation using the second order Fourier method with $dh = 0.1$ and $\lambda = 0.5$ are shown in Fig. 4.3.5.



(a) Top view.

(b) 3D view.

Figure 4.3.5: propagating wave test case simulated with the first order Fourier method with parameters $dh = 0.1$ and $\lambda = 0.5$.

4.3.4. Convergence test

Consider the standing wave test case in Subsec. 2.1.7. The results of convergence test on the first order Fourier method with $dh = 1e-2$ and different values of dt are shown in Fig. 4.3.6: for each value of dt, we show the values of L^1, L^2, and L^∞ norm (Def. 2.3.1) of the error (difference between numerical solution and ground truth at the last time instant). Analogously, in Fig. 4.3.7, we show the results of convergence test with $dt = 1e-3$ and different values of dh.

It has been found that the best fitting order of accuracy for the first order Fourier method is $\mathcal{O}(dt + dh)$.

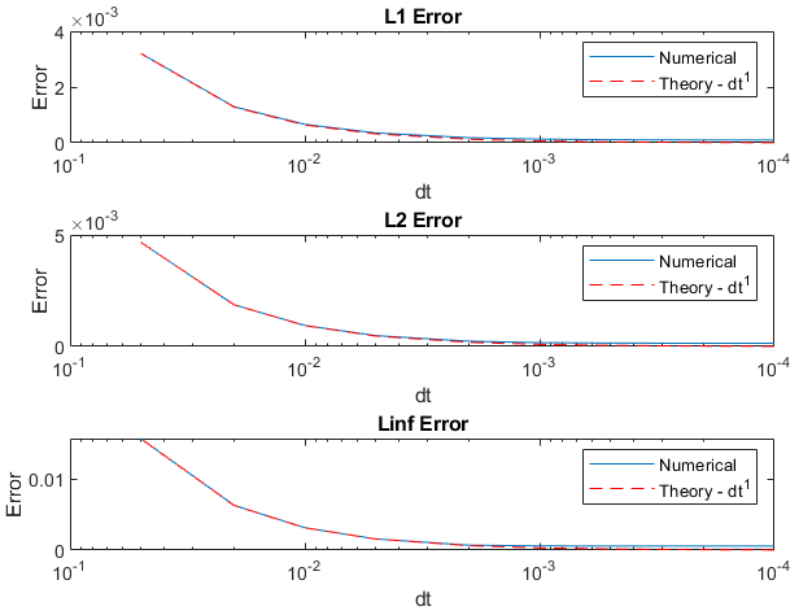


Figure 4.3.6: Convergence test on the first order Fourier method with $dh = 1e-2$ and different values of dt. The red curve represents the theoretical convergence behavior.

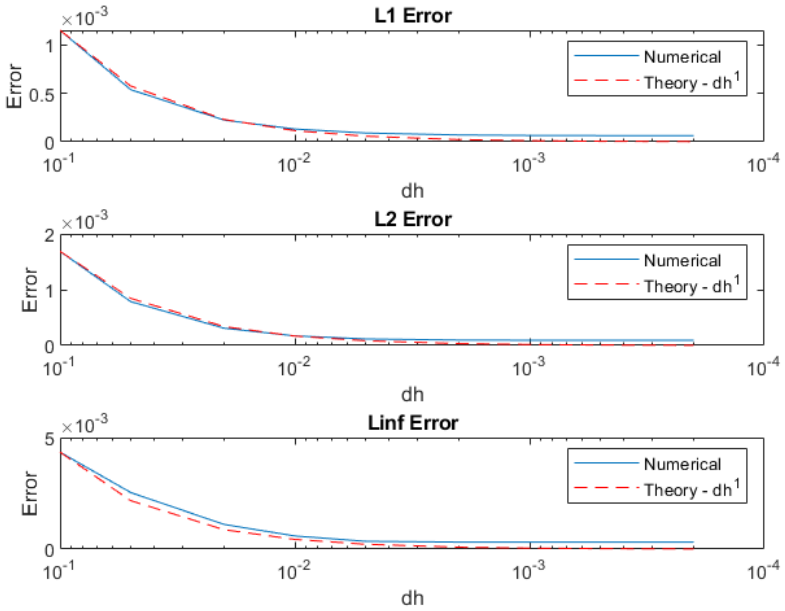


Figure 4.3.7: Convergence test on the first order Fourier method with $dt = 1e-3$ and different values of dh. The red curve represents the theoretical convergence behavior.

4.4. Treatment of boundary conditions

In the current state, since we are considering the Fourier method, we are limited to imposing the Neumann boundary conditions, as we rely on the DCT to obtain independent single degree-of-freedom equations. This is because the DCT employs cosines as basis functions, which coincide with the mode shapes obtained imposing the Neumann boundary conditions. Imposing a different type of boundary conditions, such as Dirichlet conditions, would lead to a non-negligible spectral leakage which would negatively affect the accuracy of the solution.

To impose Dirichlet boundary conditions, one could replace the DCT with the DST, which uses sines basis functions instead of cosine ones as the kernel. In general, for more complex boundary conditions, a different discrete transform is required, since the mode shapes will not be trigonometric functions anymore: these transforms are likely more computationally expensive.

Finally, it is possible to impose the non-homogeneous Dirichlet or Neumann boundary conditions using modified versions of the DCT/DST (analogous to the Shifted Discrete Fourier Transform) that shift the basis functions to satisfy the boundary conditions. However, this approach is out of the scope of this work (and is computationally expensive) and would lack clear advantages in terms of physical modeling over homogeneous boundary conditions.

5 | Simulation of the wave equation through domain decomposition

In the previous chapters, we have introduced two numerical methods, namely the Finite-Difference Time-Domain (FDTD) method in Ch. 3 and the Fourier method in Ch. 4, for solving the wave equation. Although these methods possess straightforward formulations, they exhibit certain limitations, such as constraints on geometry and a lack of ease in parallelization. Consequently, the purpose of this chapter is to address and overcome these limitations.
To achieve this goal, we will first provide a comprehensive explanation of Domain Decomposition, a technique utilized to partition a domain into multiple independent subdomains that can be updated separately. Subsequently, we will present the Rectangular Domain Decomposition approach, which is a specific type of domain decomposition suitable for the Fourier method. Finally, we'll present a simulation algorithm for the wave equation based on Rectangular Domain Decomposition.

5.1. Fundamentals of Domain Decomposition

Domain decomposition methods are effective techniques for solving problems that involve different shapes and sizes of domains, including in the field of acoustics.
This approach is especially helpful when working with large and complex domains like concert halls or auditoriums. It would be too computationally expensive to solve the entire problem all at once in such cases. Instead, we divide the domain into smaller parts, solve each part separately, and then bring together the results to find a solution for the whole domain. This method allows us to handle the computational demands of large and complex domains more efficiently.

5.1.1. Definition

The following dissertation is based on Quarteroni and Valli [1999] and Dolean et al. [2015].
The *domain decomposition* methods involve dividing the computational domain into sub-

domains, which may or may not overlap.
Let Ω be a domain in d dimensions with a Lipschitz boundary $\partial\Omega$. The outer unit normal direction is denoted by $\vec{n}$. Initially, we assume that Ω is divided into two non-overlapping subdomains Ω_1 and Ω_2. The intersection between the boundaries of Ω_1 and Ω_2 is denoted by $\Gamma = \partial\Omega_1 \cap \partial\Omega_2$ (see Fig. 5.1.1). We also assume that Γ is a Lipschitz $(d-1)$-dimensional manifold.

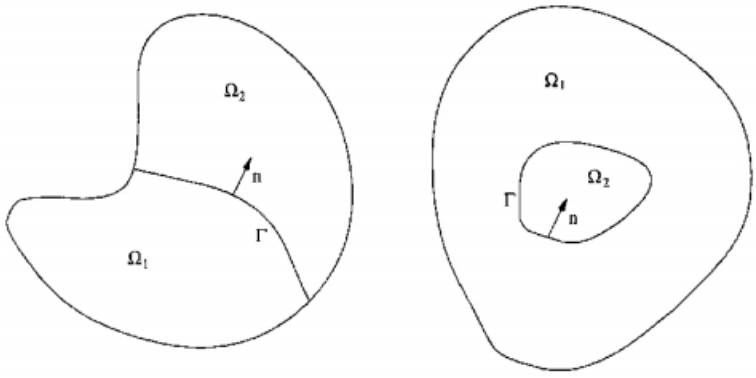


Figure 5.1.1: Non-overlapping partition of the domain Ω into two subdomains. Source: Quarteroni and Valli [1999].

5.1.2. Managing interfaces between subdomains

Consider a general differential problem given by

$$\mathcal{L}u = f \quad \text{in } \Omega, \tag{5.1.1}$$

where $\mathcal{L}$ is a partial differential operator, f is a given data, and u is the unknown solution. To address this problem Eq. (5.1.1), we partition the domain Ω into two separate problems by considering the disjoint subdomains Ω_1 and Ω_2 as shown in Fig. 5.1.1. Let's denote u_i as the part of the solution u that exists in Ω_i, where $i = 1, 2$. This leads to the following equations:

$$\begin{cases} \mathcal{L}u_1 = f, & \text{in } \Omega_1, \\ \mathcal{L}u_2 = f, & \text{in } \Omega_2. \end{cases} \tag{5.1.2}$$

For the above problem to be equivalent to the original problem Eq. (5.1.1), we need to ensure continuity across the interface Γ between u_1 and u_2. The specific continuity conditions depend on the nature of the problem, e.g.,

$$\Phi(u_1) = \Phi(u_2), \quad \text{on } \Gamma. \tag{5.1.3}$$

The function Φ is problem-specific and defines the continuity requirements at the interface. These conditions ensure that the solutions u_1 and u_2 smoothly connect across the interface and form a consistent overall solution. For example, in acoustic wave problems, we typically require continuity of the pressure field and the normal component of the particle velocity field across the interface (Cfr. Kaltenbacher and Floss [2018]).

5.2. Rectangular Domain Decomposition

In the majority of current Domain Decomposition methodologies, particularly for wave propagation, the primary objective remains the distribution and parallelization of workloads across multiple processors. Hence, the essential criterion in such scenarios is to achieve equal-sized subdomains with minimal interface areas, as this ensures computational balance and reduces communication costs.
Our approach to domain partitioning is motivated not only by parallelization, but mostly by a different goal: to obtain subdomains with rectangular shapes, even if it results in subdomains of varying sizes. This choice yields numerous algorithmic enhancements in terms of computational efficiency and numerical accuracy during simulations within the subdomains. Moreover, this approach demonstrates improvements even in sequential performance by employing the Fourier method within a rectangular domain (Raghuvanshi et al. [2011]).

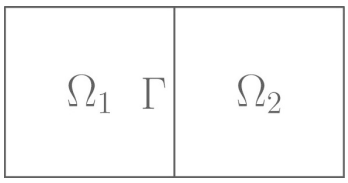


Figure 5.2.1: Non-overlapping partition of a 2D domain Ω into two rectangular subdomains with a shared boundary Γ (line).

Consider the wave propagation problem 2.1.7 on a domain $\Omega \subseteq \mathbb{R}^3$, omitting the initial conditions and the boundary condition on $\partial\Omega$ for simplicity:

$$\begin{cases} \frac{\partial^2 p(\underline{x},t)}{\partial t^2} - c^2 \Delta_G \, p(\underline{x},t) = f(\underline{x},t), & t \in \mathbb{R}^+, \quad \underline{x} \in \Omega, \\ \frac{\partial p(\underline{x},t)}{\partial \underline{n}} = 0, & t \in \mathbb{R}^+, \quad \underline{x} \in \partial\Omega, \end{cases} \tag{5.2.1}$$

where Δ_G is the global Laplacian operator. It is well known that p_1 (corresponding to Ω_1) and p_2 (corresponding to Ω_2) represent the solutions of Eq. (5.2.1) restricted to their

respective subdomains. This can be stated mathematically as follows:

$$\begin{cases} \dfrac{\partial^2 p_i(\underline{x},t)}{\partial t^2} - c^2 \Delta p_i(\underline{x},t) = f_i(\underline{x},t), & t \in \mathbb{R}^+, \quad \underline{x} \in \Omega_i, \quad i = 1,2, \\ p_1(\underline{x},t) = p_2(\underline{x},t), & t \in \mathbb{R}^+, \quad \underline{x} \in \Gamma, \\ \dfrac{\partial p_1(\underline{x},t)}{\partial \vec{n}} = \dfrac{\partial p_2(\underline{x},t)}{\partial \vec{n}}, & t \in \mathbb{R}^+, \quad \underline{x} \in \Gamma. \end{cases} \tag{5.2.2}$$

In simpler terms, the local functions p_1 and p_2 must satisfy the wave equation within their respective subdomains and must be consistent at the interface. Since the wave equation is a second-order partial differential equation, we have two continuity equations on the interface: one for the unknowns and another for their normal derivatives.
To understand how to impose continuity on the interface, we compute the difference between Eq. (5.2.1) and Eq. (5.2.2):

$$\left(\frac{\partial^2}{\partial t^2} - c^2 \Delta_G\right) \begin{bmatrix} p_1(\underline{x},t) \\ p_2(\underline{x},t) \end{bmatrix} - \left(\frac{\partial^2}{\partial t^2} - c^2 \Delta_L\right) \begin{bmatrix} p_1(\underline{x},t) \\ p_2(\underline{x},t) \end{bmatrix} = c^2 \Delta_R \begin{bmatrix} p_1(\underline{x},t) \\ p_2(\underline{x},t) \end{bmatrix},$$

where Δ_L is the local Laplacian operator:

$$\Delta_L = \begin{bmatrix} \Delta & 0 \\ 0 & \Delta \end{bmatrix},$$

and Δ_R is the residual operator. This means that the term

$$c^2 \Delta_R \begin{bmatrix} p_1(\underline{x},t) \\ p_2(\underline{x},t) \end{bmatrix}$$

can be seen as a *residual* term, which depends on the boundary conditions assumed for the interface Γ. We include it in Eq. (5.2.2) as follows:

$$\left(\frac{\partial^2}{\partial t^2} - c^2 \Delta_L\right) \begin{bmatrix} p_1(\underline{x},t) \\ p_2(\underline{x},t) \end{bmatrix} = \begin{bmatrix} f_1(\underline{x},t) \\ f_2(\underline{x},t) \end{bmatrix} + c^2 \Delta_R \begin{bmatrix} p_1(\underline{x},t) \\ p_2(\underline{x},t) \end{bmatrix}, \quad t \in \mathbb{R}^+, \quad \underline{x} \in \Omega_i, \quad i = 1,2.$$

In Mehra et al. [2012], it has been proven that, in case of an unbounded domain, it's possible to derive the exact residual operator employing an infinite width finite difference

stencil; however, this approach is unpractical and highly computationally intensive, meaning that it's desirable to derive compact residual operators. Hence, in the next sections we'll derive the residual term with a sixth order of accuracy centered finite difference stencil to approximate the Laplacian.

5.3. Derivation of RDD for the second order wave equation

Following Raghuvanshi et al. [2008], we consider a 1D domain $\Omega = (a, b)$, and define a set of $2I$ points $\{x_i\}_{i=1}^{2I}$ with grid spacing dh. As derived in Sec. 3.1, the finite difference scheme equation for the (second order) wave equation is given by Eq. (3.1.3), which we report below:

$$\underline{p}^{n+1} = \frac{2\underline{p}^n - (1 - dt\,\alpha)\underline{p}^{n-1} + \left(\frac{cdt}{dh}\right)^2 [K]\,\underline{p}^n + dt^2 \underline{f}^n}{1 + dt\,\alpha},$$

where $[K]$ is the $2I \times 2I$ stiffness matrix defined in Eq. (3.1.4) as

$$[K] = \begin{bmatrix} \ddots & & & & & & & & & \\ & A & B & C & D & C & B & A & & \\ & & A & B & C & D & C & B & A & \\ & & & A & B & C & D & C & B & A \\ & & & & & & & & & \ddots \end{bmatrix}.$$

This represents the FDTD equation for the solution in the entire domain Ω.
For the sake of presentation, we partition the domain into two non-overlapping subdomains of I points. We refer to the left half with $x_{1,i},\ i = 1, \ldots, I$, and to the right half with $x_{2,i},\ i = 1, \ldots, I$. This means that $[K]$ is decoupled into a block diagonal form matrix $[A]$ of dimension $2I \times 2I$, and the off-diagonal entries (residual terms) must be accounted for properly through a residual matrix $[C]$ of dimension $2I \times 2I$. Mathematically, this can be achieved as follows:

$$[K] = [A] + [C],$$

where

$$[A] = \begin{bmatrix} [A]_1 & [0] \\ [0] & [A]_2 \end{bmatrix},$$

$$[C] = \begin{bmatrix} [C]_{11} & [C]_{12} \\ [C]_{21} & [C]_{22} \end{bmatrix}.$$

The matrix $[A]$ assumes the homogeneous Neumann boundary conditions at the interface Γ between the two subdomains (refer to Sec. 3.3):

$$[A] = \left[\begin{array}{cccccccc|cccccccc}
\ddots & & & & & & & & & & & & & & & \\
 & A & B & C & D & C & B & A & & & & & & & & \\
 & & A & B & C & D & C & B+A & & & & & & & & \\
 & & & A & B & C & D+A & C+B & & & & & & & & \\
 & & & & A & B+A & C+B & D+C & & & & & & & & \\
\hline
 & & & & & & & & D+C & C+B & B+A & A & & & & \\
 & & & & & & & & C+B & D+A & C & B & A & & & \\
 & & & & & & & & B+A & C & D & C & B & A & & \\
 & & & & & & & & A & B & C & D & C & B & A & \\
 & & & & & & & & & & & & & & & \ddots
\end{array}\right],$$

where the upper-left block corresponds to the left subdomain $\underline{p}_1$, and the lower-right block corresponds to the right subdomain $\underline{p}_2$.
$[C]$ is computed by difference:

$$[C] = \left[\begin{array}{cccc|ccc|c}
0 & & & & & & & 0 \\
 & \ddots & & -A & A & & & \\
 & & -A & -B & B & A & & \\
 & -A & -B & -C & C & B & A & \\
\hline
 & A & B & C & -C & -B & -A & \\
 & & A & B & -B & -A & & \\
 & & & A & -A & & \ddots & \\
0 & & & & & & & 0
\end{array}\right]. \tag{5.3.1}$$

This represents the imposition of even symmetry about $x_{1,I+\frac{1}{2}}$.
Thus, the FDTD scheme can be rewritten as

$$\underline{p}^{n+1} = \frac{2\underline{p}^n - (1 - dt\,\alpha)\underline{p}^{n-1} + \left(\frac{cdt}{dh}\right)^2 [A]\,\underline{p}^n + dt^2 \underline{f}^n + \left\{\left(\frac{cdt}{dh}\right)^2 [C]\,\underline{p}^n\right\}}{1 + dt\,\alpha}, \tag{5.3.2}$$

where the term in curly brackets is responsible for the enforcement of the transmission conditions, which we define as the *residual vector* $\underline{r}^n$:

$$\underline{r}^n \triangleq \left(\frac{cdt}{dh}\right)^2 [C]\,\underline{p}^n. \tag{5.3.3}$$

To make the notation uniform, we define the *modified residual vector* $\underline{R}^n$ as

$$\underline{R}^n \triangleq \left(\frac{c}{dh}\right)^2 [C]\underline{p}^n. \tag{5.3.4}$$

To impose $\underline{r}^n$ we can proceed in two ways:

- *Pre-merge* (originally proposed in Raghuvanshi et al. [2008]) - The residual (correction) term is treated as a forcing term:

$$\underline{f}^n := \underline{f}^n + \frac{1}{dt^2}\underline{r}^n = \underline{f}^n + \underline{R}^n.$$

- *Post-merge* - After the computation of the solution at time t_{n+1} in each subdomain, we sum the residual term to obtain a solution which satisfies the wave equation in the whole domain Ω:

$$\underline{p}^{n+1} := \underline{p}^{n+1} + \frac{1}{1+dt\,\alpha}\underline{r}^n = \underline{p}^{n+1} + \frac{dt^2}{1+dt\,\alpha}\underline{R}^n.$$

5.4. Derivation of RDD for the first order wave equation

Consider again a 1D domain $\Omega = (a, b)$, and define a set of $2I$ points $\{x_i\}_{i=1}^{2I}$ with grid spacing dh. We partition the domain into two non-overlapping subdomains of I points. We refer to the left half with $x_{1,i}$, $i = 1, \ldots, I$, and to the right half with $x_{2,i}$, $i = 1, \ldots, I$; $\underline{p}_1$ and $\underline{v}_1$ represent the pressure and the pressure velocity of the first subdomain respectively, while $\underline{p}_2$ and $\underline{v}_2$ represent the pressure and the pressure velocity of the second subdomain respectively. Hence, the state vector $\underline{z}_1 = [\underline{p}_1^T\ \underline{v}_1^T]^T$, while $\underline{z}_2 = [\underline{p}_2^T\ \underline{v}_2^T]^T$.
In the case of the second order wave equation, the global pressure vector $\underline{p}$ was simply expressed as a concatenation of the local pressure vectors $\underline{p}_1, \underline{p}_2$. In this case, recalling Sec. 3.2, the solution is expressed as a global state vector $\underline{z}$ which cannot be expressed as a concatenation of the local state vectors $\underline{z}_1, \underline{z}_2$. Nevertheless, it's useful to define the

vector $\underset{\sim}{\widetilde{z}}$ obtained from the concatenation of $\underline{z}_1, \underline{z}_2$:

$$\underset{\sim}{\widetilde{z}} \triangleq \begin{bmatrix} \underline{z}_1 \\ \\ \underline{z}_2 \end{bmatrix},$$

and, analogously, we define the vector $\underline{\widetilde{\overline{f}}}$ obtained from the concatenation of the force vectors $\underline{\overline{f}_1}$ and $\underline{\overline{f}_2}$.
We define the *permutation matrix* $[G]$:

$$[G] \triangleq \begin{bmatrix} [I]_{I\times I} & [0]_{I\times I} & [0]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & [0]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} \end{bmatrix},$$

such that

$$\underline{z} = [G]\underset{\sim}{\widetilde{z}}, \quad \underset{\sim}{\widetilde{z}} = [G]\underline{z},$$

and

$$\underline{\overline{f}} = [G]\underline{\widetilde{\overline{f}}}, \quad \underline{\widetilde{\overline{f}}} = [G]\underline{\overline{f}}.$$

It's trivial to prove that $[G]$ is invertible, orthogonal and symmetric, meaning that $[G] = [G]^{-1} = [G]^T$ and that $[G]^T[G] = [G][G] = [G]^{-1}[G] = [I]$.
The FD scheme for the entire domain can be expressed, recalling Eq. (3.2.6), as

$$\underline{z}^{n+1} = \left[\overline{K}\right] \underline{z}^n + \underline{\overline{f}}^n, \tag{5.4.1}$$

while the FD schemes for the subdomains can be expressed as

$$\underline{z}_1^{n+1} = \left[\overline{A}\right]_1 \underline{z}_1^n + \underline{\overline{f}}_1^n,$$
$$\underline{z}_2^{n+1} = \left[\overline{A}\right]_2 \underline{z}_2^n + \underline{\overline{f}}_2^n,$$

which, recalling the definition of $\underset{\sim}{\widetilde{z}}$, can be rewritten as

$$\underset{\sim}{\widetilde{z}}^{n+1} = [\widetilde{A}]\, \underset{\sim}{\widetilde{z}}^n + \underline{\widetilde{\overline{f}}}^n,$$

where the block matrix $[\widetilde{A}]$ is defined as

$$[\widetilde{A}] \triangleq \begin{bmatrix} \left[\overline{A}\right]_1 & [0]_{2I\times 2I} \\ [0]_{2I\times 2I} & \left[\overline{A}\right]_2 \end{bmatrix}.$$

We now convert $\widetilde{\underline{z}}$ to $\underline{z}$ and $\widetilde{\overline{f}}$ to $\overline{f}$ in the following way:

$$\begin{aligned} [G]\underline{z}^{n+1} &= [\widetilde{A}][G]\underline{z}^n + [G]\overline{f}^n, \\ \underline{z}^{n+1} &= [G][\widetilde{A}][G]\underline{z}^n + \overline{f}^n, \\ \underline{z}^{n+1} &= \left[\overline{A}\right]\underline{z}^n + \overline{f}^n, \end{aligned}$$

where the matrix $[A]$ is defined as

$$\begin{aligned} \left[\overline{A}\right] \triangleq [G][\widetilde{A}][G] &= [G]\begin{bmatrix} \left[\overline{A}\right]_1 & [0]_{2I\times 2I} \\ [0]_{2I\times 2I} & \left[\overline{A}\right]_2 \end{bmatrix}[G] \\ &= [G]\begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[A]_1}{1+2dt\ \alpha} & \frac{c^2\frac{dt}{dh^2}[A]_1}{1+2dt\ \alpha} & [0]_{I\times I} & [0]_{I\times I} \\ dt[I]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & [0]_{I\times I} & \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[A]_2}{1+2dt\ \alpha} & \frac{c^2\frac{dt}{dh^2}[A]_1}{1+2dt\ \alpha} \\ [0]_{I\times I} & [0]_{I\times I} & dt[I]_{I\times I} & [I]_{I\times I} \end{bmatrix}[G] \\ &= \begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[A]_1}{1+2dt\ \alpha} & [0]_{I\times I} & \frac{c^2\frac{dt}{dh^2}[A]_1}{1+2dt\ \alpha} & [0]_{I\times I} \\ [0]_{I\times I} & \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[A]_2}{1+2dt\ \alpha} & [0]_{I\times I} & \frac{c^2\frac{dt}{dh^2}[A]_2}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} \end{bmatrix}. \end{aligned}$$

The global system matrix $\left[\overline{K}\right]$ in Eq. (5.4.1) can be expressed, similarly to the second order case, as

$$\left[\overline{K}\right] = \left[\overline{A}\right] + \left[\overline{C}\right],$$

so that Eq. (5.4.1) becomes

$$\underline{z}^{n+1} = \left[\overline{A}\right]\ \underline{z}^n + \overline{f}^n + \left\{\left[\overline{C}\right]\ \underline{z}^n\right\}, \tag{5.4.2}$$

where the term in curly brackets is responsible for the enforcement of the transmission conditions.
To simplify the calculation of $\left[\overline{C}\right]$, we express $\left[\overline{K}\right]$ as

$$\begin{aligned}\left[\overline{K}\right] &= \begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]}{1+2dt\ \alpha} & \frac{c^2\frac{dt}{dh^2}[K]}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & [I]_{I\times I} \end{bmatrix} \\ &= \begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]_{11}}{1+2dt\ \alpha} & \frac{c^2\frac{dt^2}{dh^2}[K]_{12}}{1+2dt\ \alpha} & c^2\frac{dt}{dh^2}[K]_{11} & c^2\frac{dt}{dh^2}[K]_{12} \\ \frac{c^2\frac{dt^2}{dh^2}[K]_{21}}{1+2dt\ \alpha} & \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]_{22}}{1+2dt\ \alpha} & c^2\frac{dt}{dh^2}[K]_{21} & c^2\frac{dt}{dh^2}[K]_{22} \\ dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} \end{bmatrix},\end{aligned}$$

We can now compute $\left[\overline{C}\right]$:

$$\begin{aligned}\left[\overline{C}\right] &= \left[\overline{K}\right] - \left[\overline{A}\right] \\ &= \begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]_{11}}{1+2dt\ \alpha} & \frac{c^2\frac{dt^2}{dh^2}[K]_{12}}{1+2dt\ \alpha} & c^2\frac{dt}{dh^2}[K]_{11} & c^2\frac{dt}{dh^2}[K]_{12} \\ \frac{c^2\frac{dt^2}{dh^2}[K]_{21}}{1+2dt\ \alpha} & \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[K]_{22}}{1+2dt\ \alpha} & c^2\frac{dt}{dh^2}[K]_{21} & c^2\frac{dt}{dh^2}[K]_{22} \\ dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} \end{bmatrix} \\ &- \begin{bmatrix} \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[A]_1}{1+2dt\ \alpha} & [0]_{I\times I} & \frac{c^2\frac{dt}{dh^2}[A]_1}{1+2dt\ \alpha} & [0]_{I\times I} \\ [0]_{I\times I} & \frac{[I]_{I\times I}+c^2\frac{dt^2}{dh^2}[A]_2}{1+2dt\ \alpha} & [0]_{I\times I} & \frac{c^2\frac{dt}{dh^2}[A]_2}{1+2dt\ \alpha} \\ dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} & [0]_{I\times I} \\ [0]_{I\times I} & dt[I]_{I\times I} & [0]_{I\times I} & [I]_{I\times I} \end{bmatrix} \\ &= \begin{bmatrix} \left[\overline{C}\right]_{11} & \left[\overline{C}\right]_{12} \\ [0]_{2I\times 2I} & [0]_{2I\times 2I} \end{bmatrix},\end{aligned} \tag{5.4.3}$$

where

$$[\overline{C}]_{11} = \frac{1}{1+2dt\,\alpha} c^2 \frac{dt^2}{dh^2}[C]$$

$$= \frac{1}{1+2dt\,\alpha} c^2 \frac{dt^2}{dh^2} \left[\begin{array}{cccc|ccc|c} 0 & & & & & & & 0 \\ & \ddots & & -A & A & & & \\ & & -A & -B & B & A & & \\ & -A & -B & -C & C & B & A & \\ \hline & A & B & C & -C & -B & -A & \\ & & A & B & -B & -A & & \\ & & & A & -A & & \ddots & \\ 0 & & & & & & & 0 \end{array}\right], \tag{5.4.4}$$

where the matrix $[C]$ is defined in Eq. (5.3.1), while

$$[\overline{C}]_{12} = c^2 \frac{dt}{dh^2}[C]$$

$$= c^2 \frac{dt}{dh^2} \left[\begin{array}{cccc|ccc|c} 0 & & & & & & & 0 \\ & \ddots & & -A & A & & & \\ & & -A & -B & B & A & & \\ & -A & -B & -C & C & B & A & \\ \hline & A & B & C & -C & -B & -A & \\ & & A & B & -B & -A & & \\ & & & A & -A & & \ddots & \\ 0 & & & & & & & 0 \end{array}\right]. \tag{5.4.5}$$

We now express the term in curly brackets in Eq. (5.4.2) as a *residual vector* $\underline{r}^n$. Recalling the definition of $[C]_{11}$ in Eq. (5.4.4), of $[C]_{12}$ in Eq. (5.4.5), and of $[C]$ in Eq. (5.4.3), we

can express it as

$$\underline{r}^n \triangleq [\overline{C}]\underline{z}^n = [C]_{11}\,\underline{v}^n + [C]_{12}\,\underline{p}^n = \frac{1}{1+2dt\,\alpha}\left(c^2\frac{dt^2}{dh^2}[C]\,\underline{v}^n + c^2\frac{dt}{dh^2}[C]\,\underline{p}^n\right)$$
$$= \frac{1}{1+2dt\,\alpha}c^2\frac{dt}{dh^2}[C]\left(dt\underline{v}^n + \underline{p}^n\right) = \frac{1}{1+2dt\,\alpha}c^2\frac{dt}{dh^2}[C]\underline{p}^{n+1}.$$

To make the notation uniform, we employ the *modified residual vector* $\underline{R}^n$ defined in Eq. (5.3.4):

$$\underline{R}^{n+1} = \left(\frac{c}{dh}\right)^2[C]\underline{p}^{n+1}.$$

Again, to enforce the transmission conditions we need to impose the residual vector, and we can proceed in two ways:

- *Pre-merge* - We treat the residual (correction) term as a forcing term:

$$\underline{f}^{n+1} := \underline{f}^{n+1} + \frac{1+2dt\,\alpha}{dt}\underline{r}^n = \underline{f}^{n+1} + \underline{R}^{n+1}.$$

- *Post-merge* - After the computation of v^{n+1} in each subdomain, we sum the residual term to the pressure velocity to obtain a solution which satisfies the wave equation in the whole domain Ω:

$$\underline{v}^{n+1} := \underline{v}^{n+1} + \underline{r}^n = \underline{v}^{n+1} + \frac{dt}{1+2dt\,\alpha}\underline{R}^{n+1}.$$

5.5. RDD algorithm

In this subsection, we present Alg. 5.1, an algorithm for the solution of the one dimensional wave equation in domains of arbitrary shapes exploiting rectangular domain decomposition. This algorithm, which is focused on the solution update and correction steps, is general: we can choose different interface handling methods (pre-merge or post-merge), different simulation methods for the subdomains (FDTD or the Fourier method), and different orders (first or second). The preprocessing steps (rectangular decomposition and interface inference) will be explained in more detail in Sec. 7.1.

Algorithm 5.1 RDD Algorithm

Input: Initial conditions of pressure and pressure velocity, and geometry of the room.
Rectangular Decomposition: Decompose the domain into rectangular subdomains Ω_k. The **subdomains** data structure stores, for each subdomain, the force, the pressure, the pressure velocity, the residual, and the corrected force, accessed through dot indexing (e.g., $\Omega_k.\underline{p}^n$).
Interfaces Inference: Determine the interfaces Γ_m between subdomains by examining their adjacency relationships. The **interfaces** data structure stores, for each interface, the reference to the two subdomains Ω_1 and Ω_2 associated with it, accessed through dot indexing.
Initialize: Set $n = 0$.
repeat
    for each subdomain Ω_k in subdomains **do**
        Update the pressure
        $\Omega_k.\underline{p}^{n+1} \leftarrow$ update_pressure()
    end for
    for each interface Γ_m in interfaces **do**
        if interface handling method is post-merge and simulation method is second order **then**
            Correct the pressure
            $\Gamma_m.\Omega_1.\underline{p}^{n+1}$ += $\frac{dt^2}{1+dt\alpha}\Gamma_m.\Omega_1.\underline{R}^n$
            $\Gamma_m.\Omega_2.\underline{p}^{n+1}$ += $\frac{dt^2}{1+dt\alpha}\Gamma_m.\Omega_2.\underline{R}^n$
        end if
    end for
    for each Ω_k in subdomains **do**
        Compute the residual
        $\Omega_k.\underline{R}^{n+1} \leftarrow \left(\frac{c}{dh}\right)^2 [C]\Omega_k.\underline{p}^{n+1}$
    end for
    for each Γ_m in interfaces **do**
        if interface handling method is pre-merge **then**
            Correct the force
            $\Gamma_m.\Omega_1.\underline{f}_R^{n+1} \leftarrow \Gamma_m.\Omega_1.\underline{f}^{n+1} + \Gamma_m.\Omega_1.\underline{R}^{n+1}$
            $\Gamma_m.\Omega_2.\underline{f}_R^{n+1} \leftarrow \Gamma_m.\Omega_2.\underline{f}^{n+1} + \Gamma_m.\Omega_2.\underline{R}^{n+1}$
        end if
    end for
    for each Ω_k in subdomains **do**

Update the velocity
$\Omega_k.\underline{v}^{n+1} \leftarrow$ update_pressure_velocity()
end for
for each Γ_m in interfaces **do**
if interface handling method is post-merge and simulation method is first order **then**
Correct the velocity
$\Gamma_m.\Omega_1.\underline{v}^{n+1} \mathrel{+}= \frac{dt}{1+2dt\alpha}\Gamma_m.\Omega_1.\underline{R}^{n+1}$
$\Gamma_m.\Omega_2.\underline{v}^{n+1} \mathrel{+}= \frac{dt}{1+2dt\alpha}\Gamma_m.\Omega_2.\underline{R}^{n+1}$
end if
end for
Increment n by 1.
until termination condition is met

Notice that, for the first step ($n = 0$), we must compute the solution at time t^1 using a *first order* simulation method (the first order Fourier method or first order FDTD), as the solution at time t^{-1} (needed for second order simulation methods) is not defined.
We now show the simulation methods we can use to update the solution in each subdomain. Again, we refer to the pressure in the left subdomain as $\underline{p}_1$ and to that of the right subdomain as $\underline{p}_2$, and analogously for the pressure velocity.

♣ ♣ ♣

Second order FDTD. Recalling Eq. (3.1.3) and Eq. (3.1.5), we obtain the pressure update algorithm

$$\left\{ \underline{p}_k^{n+1} = \frac{2\underline{p}_k^n - (1 - dt\,\alpha)\underline{p}_k^{n-1} + \left(\frac{cdt}{dh}\right)^2 [A]_k\, \underline{p}_k^n + dt^2 \underline{f}_k^n}{1 + dt\,\alpha}, \right\}$$

and the pressure velocity update algorithm

$$\left\{ \underline{v}_k^{n+1} = \frac{\underline{p}_k^{n+1} - \underline{p}_k^n}{dt} \right\}$$

in the subdomain Ω_k.

♣ ♣ ♣

Second order Fourier method. Recalling Eq. (4.2.3) we obtain the pressure update algo-

rithm

$$\left\{\begin{array}{l} \underline{P}_k^n = \mathrm{DCT}\{\underline{p}_k^n\}, \\ \underline{F}_k^n = \mathrm{DCT}\{\underline{f}_k^n\}, \\ P_{k,1}^{n+1} = 2P_{k,1}^n - P_{k,1}^{n-1} + dt^2 F_{k,1}^n, \\ P_{k,i}^{n+1} = 2P_{k,i}^n \cos(\omega_{k,i} dt) - P_{k,i}^{n-1} + \dfrac{2}{\omega_{k,i}^2} F_{k,i}^n (1 - \cos(\omega_{k,i} dt)), \quad i = 2, \ldots, I, \\ \underline{p}_k^{n+1} = \mathrm{iDCT}\{\underline{P}_k^{n+1}\}, \end{array}\right\}$$

and, recalling Eq. (4.2.4), we obtain the pressure velocity update algorithm

$$\left\{\begin{array}{l} \underline{P}_k^n = \mathrm{DCT}\{\underline{p}_k^n\}, \\ \underline{P}_k^{n+1} = \mathrm{DCT}\{\underline{p}_k^{n+1}\}, \\ \underline{V}_k^n = \mathrm{DCT}\{\underline{v}_k^n\}, \\ \underline{F}_k^n = \mathrm{DCT}\{\underline{f}_k^n\}, \\ V_{k,1}^n = \dfrac{P_{k,1}^{n+1} - P_{k,1}^n}{dt}, \\ V_{k,i}^n = \dfrac{\omega_{k,i}}{\sin(\omega_{k,i} dt)} (P_{k,i}^{n+1} - \cos(\omega_{k,i} dt) P_{k,i}^n) - \dfrac{1}{\omega_{k,i}} \tan\left(\dfrac{\omega dt}{2}\right) F_{k,i}^n, \quad i = 2, \ldots, I, \\ \underline{v}_k^{n+1} = \mathrm{iDCT}\{\underline{V}_k^{n+1}\} \end{array}\right\}$$

in the subdomain Ω_k, where i is used as an index for the DCT coefficients. The set of DCT coefficients has the same cardinality I as the set of spatial samples.

♣ ♣ ♣

First order FDTD. Recalling Eq. (3.2.6), we obtain the pressure and pressure velocity update algorithm

$$\left\{\underline{z}_k^{n+1} = [\overline{A}_k]\underline{z}_k^n + \overline{\underline{f}}_k^{n+1}\right\}$$

in the subdomain Ω_k, which can be analogously expressed as the pressure update algorithm

$$\left\{\underline{p}_k^{n+1} = dt\, \underline{v}_k^n + \underline{p}_k^n, \right\}$$

and the pressure velocity update algorithm

$$\left\{ \underline{v}_k^{n+1} = \frac{\underline{v}_k^n + c^2 \frac{dt}{dh^2} [A] \underline{p}_k^{n+1} + dt \underline{f}_k^{n+1}}{1 + 2dt\ \alpha} \right\}$$

in the subdomain Ω_k.

First order Fourier method. Recalling Eq. (4.3.3), we obtain the pressure update algorithm

$$\left\{ \begin{array}{ll} \underline{P}_k^n = \mathrm{DCT}\{\underline{p}_k^n\}, & \\ \underline{V}_k^n = \mathrm{DCT}\{\underline{v}_k^n\}, & \\ \underline{F}_k^n = \mathrm{DCT}\{\underline{f}_k^n\}, & \\ P_{k,1}^{n+1} = P_{k,1}^n + dt\ V_{k,1}^n, & \\ P_{k,i,e}^n = \dfrac{F_{k,i}^n}{\omega_{k,i}^2}, & i = 2, \ldots, I, \\ P_{k,i}^{n+1} = P_{k,i,e}^n + e^{-dt\ \alpha} \left((P_{k,i}^n - P_{k,i,e}) \left(\cos(\omega_{k,i} dt) + \dfrac{\alpha}{\omega_{k,i}} \sin(\omega_{k,i} dt) \right) + \dfrac{\sin(\omega_{k,i} dt)}{\omega_{k,i}} V_{k,i}^n \right), & i = 2, \ldots, I, \\ \underline{p}_k^{n+1} = \mathrm{iDCT}\{\underline{P}_k^{n+1}\}, & \end{array} \right\}$$

and, recalling Eq. (4.3.4), we obtain the pressure velocity update algorithm

$$\left\{ \begin{array}{ll} \underline{P}_k^n = \mathrm{DCT}\{\underline{p}_k^n\}, & \\ \underline{V}_k^n = \mathrm{DCT}\{\underline{v}_k^n\}, & \\ \underline{F}_k^{n+1} = \mathrm{DCT}\{\underline{f}_k^{n+1}\}, & \\ V_{k,1}^{n+1} = \dfrac{V_{k,1}^n + dt F_{k,1}^{n+1}}{1 + 2dt\ \alpha}, & \\ P_{k,i,e}^{n+1} = \dfrac{F_{k,i}^{n+1}}{\omega_{k,i}^2}, & i = 2, \ldots, I, \\ V_{k,i}^{n+1} = e^{-dt\ \alpha} \left(V_{k,i}^n \left(\cos(\omega_{k,i} dt) - \dfrac{\alpha}{\omega_{k,i}} \sin(\omega_{k,i} dt) \right) - \left(\omega_{k,i} + \dfrac{\alpha^2}{\omega_{k,i}} \right) (P_{k,i}^n - P_{k,i,e}^{n+1}) \sin(\omega_{k,i} dt) \right), & i = 2, \ldots, I, \\ \underline{v}_k^{n+1} = \mathrm{iDCT}\{\underline{V}_k^{n+1}\}, & \end{array} \right\}$$

in the subdomain Ω_k, where i is used as an index for the DCT coefficients.

5.5.1. Stability

It's trivial to prove that the stability condition of the handling algorithm corresponds to the stability condition of the FDTD method. In the 1D case this corresponds to $|\lambda| < 0.8135\ldots$ for both the second order wave equation (see Eq. 3.1.12) and the first order wave equation (see Eq. 3.2.18).

5.5.2. Numerical errors

The interface handling algorithm is built on the FDTD method, meaning that there's no additional numerical error if we employ this method also to update the local solutions: in particular, the coupling is perfect also if we mix second order FDTD and first order FDTD, or if we mix pre-merge and post-merge. However, we've extended the applicability of this algorithm to the Fourier method, which is able to eliminate numerical dispersion errors (see Subsec. 4.2.2 and Subsec. 4.3.2) inside each subdomain: this comes at a cost. The coupling, in this latter case, is not perfect, which leads to erroneous reflections at the interface.

When a wave packet encounters the interface between the subdomains, spurious reflections occur. To evaluate the quality (amount of spurious reflections) of RDD for these four cases, we consider the 1D propagating wave test case discussed in Subsec. 2.1.7, with parameters $dh = 0.01$, $\lambda = 0.8$, and $\mu = 2.5$. The domain $\Omega = [0, 10]$ is divided into two subdomains: $\Omega_1 = [0, 5]$ and $\Omega_2 = [5, 10]$. To emphasize the errors, the pressure values are expressed in decibels.

In practical cases, as we will see in Ch. 7, we use the four following combinations:

1. Undamped air-air interfaces. We employ the second order Fourier method for both subdomains for the simulation, and pre-merge as interface handling method. In Fig. 5.5.1 we present some snapshots of a numerical simulation using this configuration for different accuracy orders. The amplitude of these reflections remains relatively low, ranging from 50 dB (for an accuracy order of 2) to 80 dB (for an accuracy order of 8) below the amplitude of the incoming wave packet. This highlights a trade-off between computational efficiency and the attenuation of spurious reflections.

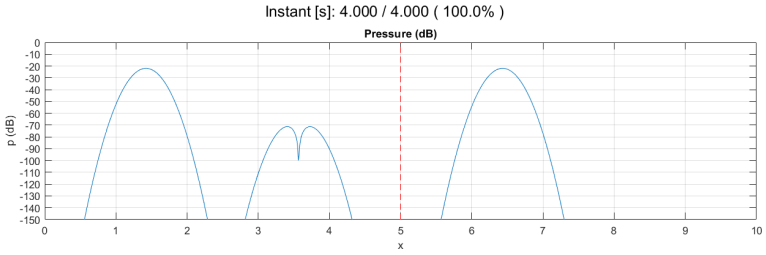
2. Damped air-air interfaces. We employ the first order Fourier method in both subdomains for the simulation, and post-merge as interface handling method. In Fig. 5.5.2 we present some snapshots of a numerical simulation using this configuration for accuracy order 6. The amplitude of these reflections remains relatively low, about 55 dB below the amplitude of the incoming wave packet.

3. PML[†]-undamped air interfaces. For simplicity, we consider an analogous case: we employ second order FDTD on the left and the second order Fourier method on the right for the simulation, and pre-merge as interface handling method. In Fig. 5.5.3 we present some snapshots of a numerical simulation using this configuration for accuracy order 6. The amplitude of these reflections remains relatively low, about 80 dB below the amplitude of the incoming wave packet.

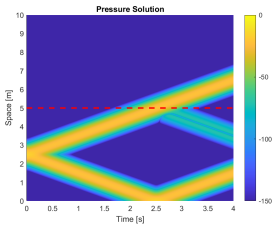
[†]Cfr. Sec. 5.6 and Ch. 6.

4. PML-damped air interfaces. For simplicity, we consider an analogous case: we employ second order FDTD on the left and the first order Fourier method on the right for the simulation, and post-merge as interface handling method. In Fig. 5.5.4 we present some snapshots of a numerical simulation using this configuration for accuracy order 6. The amplitude of these reflections remains relatively low, about 60 dB below the amplitude of the incoming wave packet.

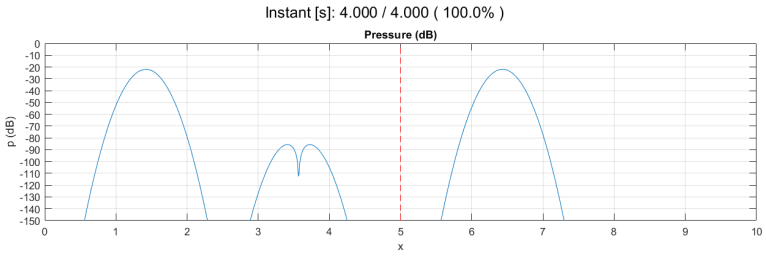
Although the errors in all four cases are not audible, it is important to note that in practical situations, where there are multiple subdomains, the errors can accumulate. Furthermore, we notice that, if at least one of the two subdomains is simulated with a first order method, the amplitude of the spurious reflections is a bit higher than the remaining cases: this means that, in these cases, it would be better to employ higher orders of accuracy.



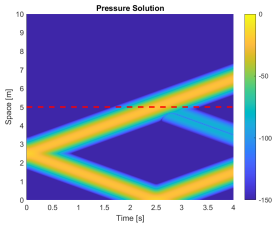
(a) Order of accuracy 2, snapshot at $t = 4$.



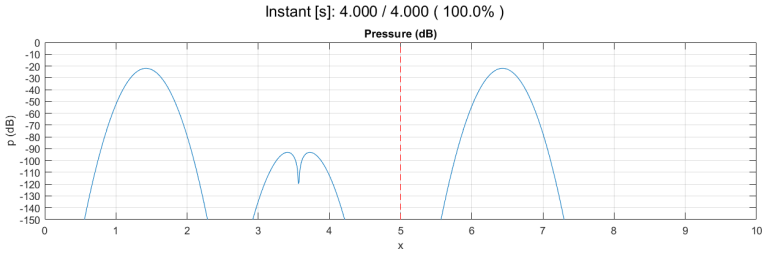
(b) Order of accuracy 2, space-time snapshot.



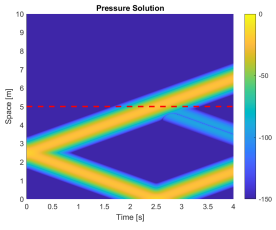
(c) Order of accuracy 4, snapshot at $t = 4$.



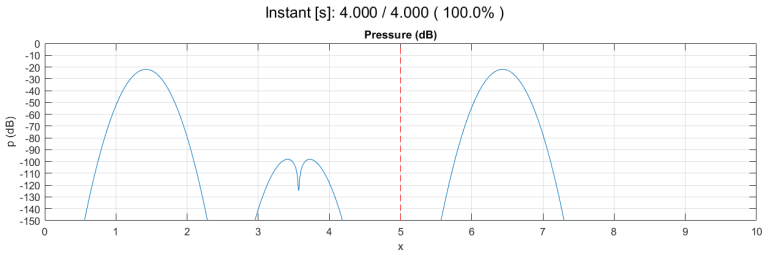
(d) Order of accuracy 4, space-time snapshot.



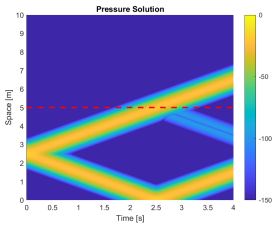
(e) Order of accuracy 6, snapshot at $t = 4$.



(f) Order of accuracy 6, space-time snapshot.

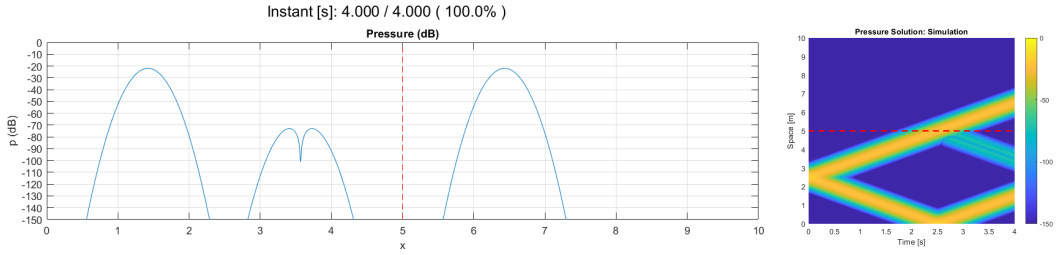


(g) Order of accuracy 8, snapshot at $t = 4$.



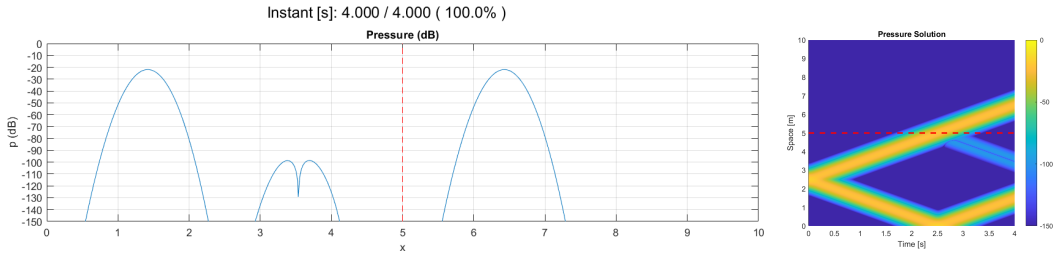
(h) Order of accuracy 8, space-time snapshot.

Figure 5.5.1: Snapshots of a numerical simulation in one dimension using Rectangular Domain Decomposition: the simulation domain Ω is divided into two subdomains, $\Omega_1 = [0, 5]$ and $\Omega_2 = [5, 10]$, and we use pre-merge with different orders of accuracy. The simulation methods used are the second order Fourier method for both subdomains, with a grid spacing of $dh = 0.01$, a Courant number of $\lambda = 0.8$. The specific test case used is the wave propagation scenario described in Subsec. 2.1.7, with $\mu = 2.5$. To emphasize the errors, the pressure values are expressed in decibels.



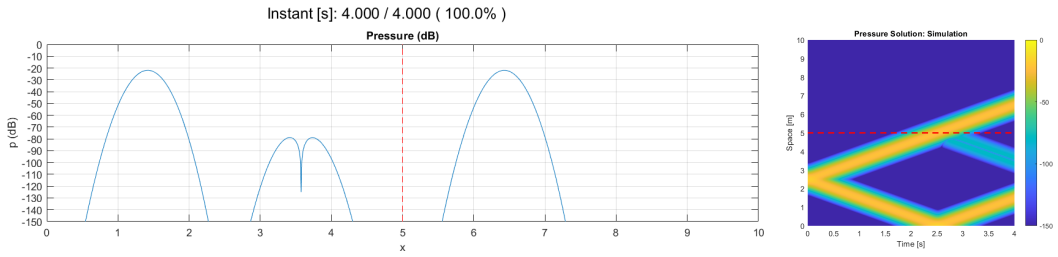
(a) Snapshot at $t = 4$. (b) Space-time snapshot.

Figure 5.5.2: Snapshots of a numerical simulation in one dimension using Rectangular Domain Decomposition: the simulation domain Ω is divided into two subdomains, $\Omega_1 = [0, 5]$ and $\Omega_2 = [5, 10]$, and we use post-merge with order of accuracy 6. The simulation methods used are the first order Fourier method for both subdomains, with a grid spacing of $dh = 0.01$, a Courant number of $\lambda = 0.8$. The specific test case used is the wave propagation scenario described in Subsec. 2.1.7, with $\mu = 2.5$. To emphasize the errors, the pressure values are expressed in decibels.



(a) Snapshot at $t = 4$. (b) Space-time snapshot.

Figure 5.5.3: Snapshots of a numerical simulation in one dimension using Rectangular Domain Decomposition: the simulation domain Ω is divided into two subdomains, $\Omega_1 = [0, 5]$ and $\Omega_2 = [5, 10]$, and we use pre-merge with order of accuracy 6. The simulation methods used are second order FDTD on the left and the second order Fourier method on the right, with a grid spacing of $dh = 0.01$, a Courant number of $\lambda = 0.8$. The specific test case used is the wave propagation scenario described in Subsec. 2.1.7, with $\mu = 2.5$. To emphasize the errors, the pressure values are expressed in decibels.



(a) Snapshot at $t = 4$. (b) Space-time snapshot.

Figure 5.5.4: Snapshots of a numerical simulation in one dimension using Rectangular Domain Decomposition: the simulation domain Ω is divided into two subdomains, $\Omega_1 = [0, 5]$ and $\Omega_2 = [5, 10]$, and we use post-merge with order of accuracy 6. The simulation methods used are second order FDTD on the left and the first order Fourier method on the right, with a grid spacing of $dh = 0.01$, a Courant number of $\lambda = 0.8$. The specific test case used is the wave propagation scenario described in Subsec. 2.1.7, with $\mu = 2.5$. To emphasize the errors, the pressure values are expressed in decibels.

5.6. Absorbing boundary conditions

The objective of this section is to provide insights on the implementation of partially or fully absorbing boundary conditions for the FDTD or the Fourier method using Rectangular Domain Decomposition.

To achieve absorption of outgoing waves within the computational domain, we introduce a specialized technique called the Perfectly Matched Layer (PML), which will be discussed in detail in Ch. 6. The computational domain, denoted as Ω, consists of two distinct subdomains. The first subdomain, Ω_1, represents the air region, while the second subdomain, Ω_2, serves as the PML layer. This PML subdomain acts as an absorbing layer, effectively attenuating the waves passing through it (see Fig. 6.1.1).

In Sec. 5.3, we have introduced the residual vector $\underline{r}^n$. As suggested in Raghuvanshi et al. [2011], to model partial absorption, we can multiply the residual term by a dimensionless coefficient, namely the *virtual boundary absorption coefficient* $0 \leqslant \beta_B \leqslant 1$.

Ideally, we would obtain full absorption by imposing $\beta_B = 1$, full reflection with $\beta_B = 0$ (which reduces to imposing the homogeneous Neumann boundary conditions), and partial absorption (partial reflection) in the range $]0, 1[$. In practice, neither the PML is perfect nor the handling algorithm, thus the achieved *physical boundary absorption coefficient* α_B is less than the imposed value of β_B; furthermore, the PML is generally more efficient in attenuating waves with perpendicular incidence compared to oblique incidence. Finally, we have that $\alpha_B \to \beta_B$ if we choose a finer grid and a higher value of the PML thickness.

6 | Perfectly matched layer conditions

The following disseration is based on Grote and Sim [2010] and Duru [2012].
The *perfectly matched layer* (PML) conditions provide a flexible and accurate way to simulate wave propagation on an unbounded domain. With the PML conditions, one has to consider a computational domain surrounded by an absorbing layer, which generates no reflections between them. Inside the absorbing layer, a damping term is added to the wave equation such that the pressure decays rapidly; the absorption effect acts only in the direction perpendicular to the layer. This approach, suggested in Raghuvanshi et al. [2011], is analogous to the physical treatment of the walls of an anechoic chamber and provides an alternative to absorbing or nonreflecting boundary conditions.
At the discrete level, once truncated at a finite thickness, the layer is no longer perfectly absorbing and the optimal damping parameters need to be determined via numerical experiments.

6.1. PML formulation

Let's examine a scenario where an acoustic wave field denoted by p propagates through an unbounded three-dimensional space over time. The propagation speed of the wave field is c. We make the assumption that all sources and initial disturbances are restricted to a rectangular domain Ω defined as

$$\Omega \triangleq \left\{\underline{x} = (x, y, z) \in \mathbb{R}^3 : |x| \leqslant a_x, |y| \leqslant a_y, |z| \leqslant a_z\right\},$$

where a_x, a_y, and a_z are positive real numbers. Consequently, all waves outside this region, in the unbounded space $\mathbb{R}^3 \backslash \Omega$, are purely outgoing.
Within Ω, the wave field $p(\underline{x}, t)$ obeys Eq. (2.1.7), which we rewrite neglecting the bound-

ary conditions:

$$\begin{cases} \frac{\partial^2 p(\underline{x},t)}{\partial t^2} - c^2 \Delta p(\underline{x},t) = f(\underline{x},t), & t \in \mathbb{R}^+, \ \underline{x} \in \mathbb{R}^3, \\ p(\underline{x},0) = p_0(\underline{x}), & \underline{x} \in \mathbb{R}^3, \\ \frac{\partial p}{\partial t}(\underline{x},0) = v_0(\underline{x}), & \underline{x} \in \mathbb{R}^3. \end{cases} \tag{6.1.1}$$

To solve equation Eq. (6.1.1) numerically within the domain Ω, we must consider a finite computational domain. Additionally, we need to ensure that waves propagating outward from Ω do not generate undesired reflections. Therefore, we enclose Ω with a perfectly matched layer (PML) that has a thickness denoted by L'_x, L'_y, and L'_z in each coordinate direction. The PML is specifically designed to absorb waves leaving Ω without any reflections. Refer to Fig. 6.1.1 for an illustration of this configuration. Within the absorbing layer, the wave field p obeys a modified wave equation, and its solutions decay exponentially as they move away from the computational domain.

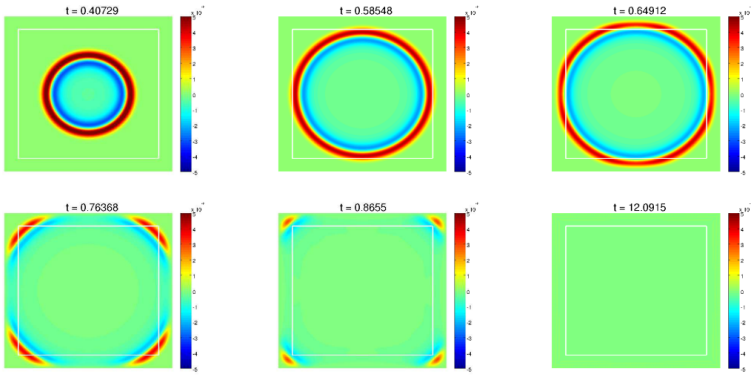


Figure 6.1.1: Temporal snapshots of the numerical solutions for a point source in two dimensions ($\Omega = [-0.5, 0.5]^2$) within a PML of width $L = 0.1$. Source: Grote and Sim [2010].

6.2. Derivation of PML equations

In this section we derive PML equations using the complex coordinate stretching technique, which was first presented in Chew and Weedon [1994] in the context of Maxwell's equations. In this technique, a modified set of equations is presented, which incorporates complex coordinate stretching along the three Cartesian coordinates. By introducing complex coordinates into the wave equation, we introduce additional degrees of freedom into the equations. These added degrees of freedom provide us with a greater level of control over the behavior of waves near boundaries. In particular, we apply this technique to the

Laplace transformed wave equation.
We let $\hat{p}$ denote the Laplace transform in time of p:

$$\hat{p} \triangleq \hat{p}(\underline{x}, s) = \int_0^\infty e^{st} p(\underline{x}, t)\, dt, \quad \mathrm{Re}\{s\} < 0.$$

In $\mathbb{R}^3 \backslash \Omega$, $\hat{p}$ satisfies the Helmholtz equation:

$$s^2 \hat{p} = \frac{\partial^2 \hat{p}}{\partial x^2} + \frac{\partial^2 \hat{p}}{\partial y^2} + \frac{\partial^2 \hat{p}}{\partial z^2}. \tag{6.2.1}$$

By considering the x direction, we introduce the coordinate transformation

$$x \mapsto \widetilde{x} = x + \frac{1}{s} \int_0^x \zeta_x(x)\, dx, \tag{6.2.2}$$

where the damping profile $\zeta_x(x)$ is positive inside the absorbing layer, $|x| > a_x$, but vanishes inside Ω. Analogous coordinate transformation are defined for the two remaining space coordinates: $y \mapsto \widetilde{y}, z \mapsto \widetilde{z}$.

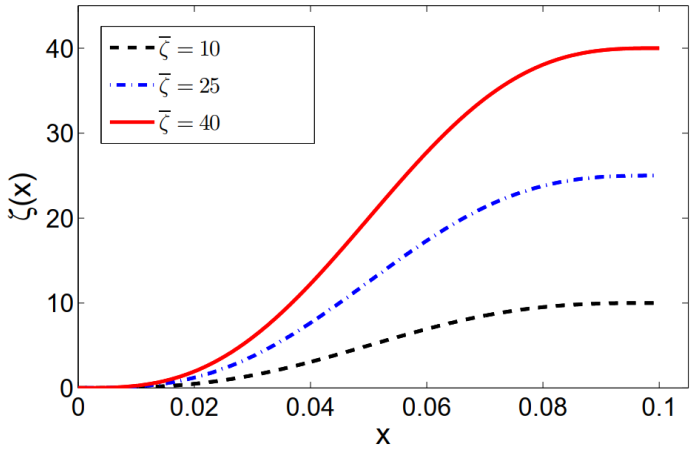


Figure 6.2.1: Example of the damping profile $\zeta_x(x)$ with varying $\overline{\zeta}_x$ values, where $c = 1$ and $L_x = 0.1$. The profile is depicted based on the equation Eq. (6.2.3). Source: Grote and Sim [2010].

The damping profile $\zeta_x(x) \geqslant 0$ can be chosen arbitrarily and can take various forms, such as constant, linear, or quadratic. An example of such a profile is given by

$$\zeta_x(x) = \begin{cases} 0, & |x| < a_x, \\ \overline{\zeta}_x \left(\frac{x - a_x}{L_x} - \frac{\sin\left(\frac{2\pi(x-a_x)}{L_x}\right)}{2\pi} \right), & a_x \leqslant |x| \leqslant a_x + L'_x. \end{cases} \tag{6.2.3}$$

Since the damping profile $\zeta_x(x)$ is twice continuously differentiable at the interface $x = a_x$, no special transmission conditions are required there. The constant $\overline{\zeta}_x$ depends on the discretization and the thickness of the layer. In practice, the layer is truncated using a homogeneous Dirichlet (or Neumann) boundary condition. Fig. 6.2.1 illustrates the damping profile obtained with different values of $\overline{\zeta}_x$.
The relative reflection R, which is a measure of the amount of reflection that occurs when a wave encounters an interface (ratio of the reflected wave intensity to the incident wave intensity), is related to $\overline{\zeta}_x$ through the equation:

$$\overline{\zeta}_x = \frac{c}{L_x} \log\left(\frac{1}{R}\right),$$

which holds for all the space coordinates.
We now require $\hat{p}$ to satisfy the modified Helmholtz equation in those stretched coordinates,

$$s^2\hat{p} = c^2\left(\frac{\partial^2\hat{p}}{\partial\widetilde{x}^2} + \frac{\partial^2\hat{p}}{\partial\widetilde{y}^2} + \frac{\partial^2\hat{p}}{\partial\widetilde{z}^2}\right). \tag{6.2.4}$$

The above equation ensures that the variable p remains unchanged within the domain Ω, while decaying exponentially fast within the layer. Consequently, the absorbing layer will be perfectly matched. Our objective is to convert equation Eq. (6.2.4) back to the time domain without introducing excessive high-order derivatives or auxiliary variables.
From equation Eq. (6.2.2), we observe that the partial derivative with respect to $\widetilde{x}$ can be related to the partial derivative with respect to the physical coordinate x using the following formula:

$$\frac{\partial}{\partial\widetilde{x}} = \frac{s}{s+\zeta_x}\frac{\partial}{\partial x}. \tag{6.2.5}$$

We not let $\gamma_x = \gamma_x(\zeta_x; s)$ denote

$$\gamma_x = 1 + \frac{\zeta_x}{s}, \tag{6.2.6}$$

and the same holds for the two remaining space coordinates.
Then, by replacing partial derivatives according to Eq. (6.2.5) and multiplying the resulting expression by $\gamma_x\gamma_y\gamma_z$, we rewrite Eq. (6.2.4) in physical coordinates as

$$s^2\gamma_x\gamma_y\gamma_z\hat{p} = c^2\left(\frac{\gamma_y\gamma_z}{\gamma_x}\frac{\partial^2\hat{p}}{\partial x^2} + \frac{\gamma_x\gamma_z}{\gamma_y}\frac{\partial^2\hat{p}}{\partial y^2} + \frac{\gamma_x\gamma_y}{\gamma_z}\frac{\partial^2\hat{p}}{\partial z^2}\right). \tag{6.2.7}$$

From Eq. (6.2.6) we derive after some algebra the following identities:

$$\begin{aligned}
\frac{\gamma_y\gamma_z}{s\gamma_x} &= 1 + \frac{(\zeta_y+\zeta_z-\zeta_x)s+\zeta_y\zeta_z}{(s+\zeta_x)s},\\
\frac{\gamma_x\gamma_z}{s\gamma_y} &= 1 + \frac{(\zeta_x+\zeta_z-\zeta_y)s+\zeta_x\zeta_z}{(s+\zeta_y)s},\\
\frac{\gamma_x\gamma_y}{s\gamma_z} &= 1 + \frac{(\zeta_x+\zeta_y-\zeta_z)s+\zeta_x\zeta_y}{(s+\zeta_z)s}.
\end{aligned} \tag{6.2.8}$$

By using Eq. (6.2.8) in Eq. (6.2.7) we find

$$\begin{aligned}
&\left(s^2 + s(\zeta_x+\zeta_y+\zeta_z) + (\zeta_y\zeta_z+\zeta_x\zeta_z+\zeta_x\zeta_y)\right)\hat{p}\\
&= c^2\left[\left(\frac{\partial^2\hat{p}}{\partial x^2}+\frac{\partial^2\hat{p}}{\partial y^2}+\frac{\partial^2\hat{p}}{\partial z^2}\right) + \frac{\partial}{\partial x}\left(\frac{(\zeta_y+\zeta_z-\zeta_x)s+\zeta_y\zeta_z}{(s+\zeta_x)s}\frac{\partial\hat{p}}{\partial x}\right)\right.\\
&\left. + \frac{\partial}{\partial y}\left(\frac{(\zeta_x+\zeta_z-\zeta_y)s+\zeta_x\zeta_z}{(s+\zeta_y)s}\frac{\partial\hat{p}}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{(\zeta_x+\zeta_y-\zeta_z)s+\zeta_x\zeta_y}{(s+\zeta_z)s}\frac{\partial\hat{p}}{\partial z}\right)\right].
\end{aligned} \tag{6.2.9}$$

Next, we introduce the auxiliary functions ψ and $\underline{\phi} = (\phi_x, \phi_y, \phi_z)^T$:

$$\begin{aligned}
\hat{\psi} &= \frac{1}{s}\hat{p}\\
\hat{\phi}_x &= c^2\left(\frac{(\zeta_y+\zeta_z-\zeta_x)s+\zeta_y\zeta_z}{(s+\zeta_x)s}\right)\frac{\partial\hat{p}}{\partial x},\\
\hat{\phi}_y &= c^2\left(\frac{(\zeta_x+\zeta_z-\zeta_y)s+\zeta_x\zeta_z}{(s+\zeta_y)s}\right)\frac{\partial\hat{p}}{\partial y},\\
\hat{\phi}_z &= c^2\left(\frac{(\zeta_x+\zeta_y-\zeta_z)s+\zeta_x\zeta_y}{(s+\zeta_z)s}\right)\frac{\partial\hat{p}}{\partial z},
\end{aligned}$$

or equivalently

$$\begin{aligned}
s\hat{\psi} &= \hat{p}\\
(s+\zeta_x)\hat{\phi}_x &= c^2\left(\frac{(\zeta_y+\zeta_z-\zeta_x)s+\zeta_y\zeta_z}{s}\right)\frac{\partial\hat{p}}{\partial x},\\
(s+\zeta_y)\hat{\phi}_y &= c^2\left(\frac{(\zeta_x+\zeta_z-\zeta_y)s+\zeta_x\zeta_z}{s}\right)\frac{\partial\hat{p}}{\partial y},\\
(s+\zeta_z)\hat{\phi}_z &= c^2\left(\frac{(\zeta_x+\zeta_y-\zeta_z)s+\zeta_x\zeta_y}{s}\right)\frac{\partial\hat{p}}{\partial z}.
\end{aligned}$$

Finally, using the above relations in Eq. (6.2.9) and transforming the resulting equations

back to the time domain, it yields the PML modified wave equation:

$$\begin{cases} p_{tt} + (\zeta_x + \zeta_y + \zeta_z)p_t + (\zeta_y\zeta_z + \zeta_x\zeta_z + \zeta_x\zeta_y)p = c^2\Delta p + \nabla \cdot \underline{\phi} - \zeta_x\zeta_y\zeta_z\psi, & \text{in } \mathbb{R}^3, \\ \underline{\phi}_t = \Gamma_1\underline{\phi} + c^2\Gamma_2\nabla p + c^2\Gamma_3\nabla\psi, & \text{in } \mathbb{R}^3, \\ \psi_t = p, & \text{in } \mathbb{R}^3, \end{cases} \tag{6.2.10}$$

where

$$\Gamma_1 = \begin{bmatrix} -\zeta_x & 0 & 0 \\ 0 & -\zeta_y & 0 \\ 0 & 0 & -\zeta_z \end{bmatrix},$$

$$\Gamma_2 = \begin{bmatrix} \zeta_y + \zeta_z - \zeta_x & 0 & 0 \\ 0 & \zeta_x + \zeta_z - \zeta_y & 0 \\ 0 & 0 & \zeta_x + \zeta_y - \zeta_z \end{bmatrix},$$

$$\Gamma_3 = \begin{bmatrix} \zeta_y\zeta_z & 0 & 0 \\ 0 & \zeta_x\zeta_z & 0 \\ 0 & 0 & \zeta_x\zeta_y \end{bmatrix}.$$

Within the domain Ω, the damping profiles ζ_x, ζ_y, and ζ_z, as well as the auxiliary variables $\underline{\phi}$ and ψ, all become zero. As a result, equation Eq. (6.2.10) simplifies to Eq. (6.1.1) within Ω. The advantage of this simplified form is that the PML formulation in Eq. (6.2.10) only requires the computation of four auxiliary scalar variables (ϕ_x, ϕ_y, ϕ_z, and ψ) within the layer. Consequently, the implementation of this formulation is not only straightforward but also inexpensive.

6.3. Simulation of the PML through the FDTD method

In Sec. 6.1, we derived the PML modified wave equation Eq. (6.2.10). In this section, we will focus on discretizing this equation using the Finite-Difference Time-Domain (FDTD) method. Accurately modeling various levels of reflection and absorption is often necessary, and we will explain how to achieve this. The approach described here is based on the work by Grote et al. Grote and Sim [2010].

The spatial and temporal discretization follows a similar approach as described in Sub-

sec. 2.3.2. However, within the absorbing layer, we introduce a staggered grid in both space and time at positions $x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}, z_{k+\frac{1}{2}}$, and times $t_{n+\frac{1}{2}}$. Consequently, the numerical solution $p^n_{i,j,k}$ satisfies

$$\begin{aligned}
&\frac{p^{n+1}_{i,j,k} - 2p^n_{i,j,k} + p^{n-1}_{i,j,k}}{dt^2} + (\zeta_{x,i} + \zeta_{y,j} + \zeta_{z,k})\frac{p^{n+1}_{i,j,k} - p^{n-1}_{i,j,k}}{2dt} \\
&\quad + (\zeta_{y,j}\zeta_{z,k} + \zeta_{x,i}\zeta_{z,k} + \zeta_{x,i}\zeta_{y,j})p^n_{i,j,k} = c^2\left[\frac{p^n_{i+1,j,k} - 2p^n_{i,j,k} + p^n_{i-1,j,k}}{dh^2} + \right. \\
&\quad \left. \frac{p^n_{i,j+1,k} - 2p^n_{i,j,k} + p^n_{i,j-1,k}}{dh^2} + \frac{p^n_{i,j,k+1} - 2p^n_{i,j,k} + p^n_{i,j,k-1}}{dh^2}\right] \\
&\quad + \frac{\widetilde{\phi}^n_{x,i+\frac{1}{2},j,k} - \widetilde{\phi}^n_{x,i-\frac{1}{2},j,k}}{dh} + \frac{\widetilde{\phi}^n_{y,i,j+\frac{1}{2},k} - \widetilde{\phi}^n_{y,i,j-\frac{1}{2},k}}{dh} \\
&\quad + \frac{\widetilde{\phi}^n_{z,i,j,k+\frac{1}{2}} - \widetilde{\phi}^n_{z,i,j,k-\frac{1}{2}}}{dh} - \zeta_{x,i}\zeta_{y,j}\zeta_{z,k}\frac{\psi^{n+\frac{1}{2}}_{i,j,k} + \psi^{n-\frac{1}{2}}_{i,j,k}}{2}, \quad (6.3.1)
\end{aligned}$$

where the cell averages of the auxiliary functions ϕ_x, ϕ_y and ϕ_z are defined as

$$\begin{aligned}
\widetilde{\phi}^n_{x,i+\frac{1}{2},j,k} &\triangleq \frac{1}{4}\left(\phi^n_{x,i+\frac{1}{2},j-\frac{1}{2},k-\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}\right), \\
\widetilde{\phi}^n_{x,i,j+\frac{1}{2},k} &\triangleq \frac{1}{4}\left(\phi^n_{x,i-\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + \phi^n_{x,i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k-\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}\right), \\
\widetilde{\phi}^n_{x,i,j,k+\frac{1}{2}} &\triangleq \frac{1}{4}\left(\phi^n_{x,i-\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + \phi^n_{x,i-\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j-\frac{1}{2},k+\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}\right).
\end{aligned}$$

Concurrently with the above discretized wave equation, we also advance the (scalar) auxiliary variables $\psi, \phi_x, \phi_y, \phi_z$ inside the absorbing layer by using standard finite differences. For ψ, we use

$$\frac{\psi^{n+\frac{1}{2}}_{i,j,k}}{dt} = p^n_{i,j,k},$$

whereas for ϕ_x we u

$$\begin{aligned}
\frac{\phi^{n+1}_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} - \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}}{dt} &= -\zeta_{x,i+\frac{1}{2}}\frac{\phi^{n+1}_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + \phi^n_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}}}{2}, \\
&+ \left(\zeta_{y,j+\frac{1}{2}} + \zeta_{z,k+\frac{1}{2}} - \zeta_{x,i+\frac{1}{2}}\right) D^h_x p^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} + \zeta_{y,j+\frac{1}{2}}\zeta_{z,k+\frac{1}{2}} D^h_x \psi^{n+\frac{1}{2}}_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}},
\end{aligned}$$

where

$$D_x^h p^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = \frac{1}{2}\left(\frac{\widetilde{p}^{n+1}_{i+1,j+\frac{1}{2},k+\frac{1}{2}} - \widetilde{p}^{n+1}_{i,j+\frac{1}{2},k+\frac{1}{2}}}{dh} + \frac{\widetilde{p}^{n}_{i+1,j+\frac{1}{2},k+\frac{1}{2}} - \widetilde{p}^{n}_{i,j+\frac{1}{2},k+\frac{1}{2}}}{dh}\right),$$

$$D_x^h \psi^{n+\frac{1}{2}}_{x,i+\frac{1}{2},j+\frac{1}{2},k+\frac{1}{2}} = \frac{\widetilde{\psi}^{n+\frac{1}{2}}_{i+1,j+\frac{1}{2},k+\frac{1}{2}} - \widetilde{\psi}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}}}{dh}.$$

Here, the cell averages of p and ψ are defined as

$$\widetilde{p}^{n}_{i,j+\frac{1}{2},k+\frac{1}{2}} \triangleq \frac{1}{4}\left(p^n_{i,j,k} + p^n_{i,j,k+1} + p^n_{i,j+1,k} + p^n_{i,j+1,k+1}\right),$$

$$\widetilde{p}^{n+\frac{1}{2}}_{i,j+\frac{1}{2},k+\frac{1}{2}} \triangleq \frac{1}{4}\left(p^{n+\frac{1}{2}}_{i,j,k} + p^{n+\frac{1}{2}}_{i,j,k+1} + p^{n+\frac{1}{2}}_{i,j+1,k} + p^{n+\frac{1}{2}}_{i,j+1,k+1}\right).$$

The finite difference approximations for ϕ_y and ϕ_z are analogous.
Eq. (6.3.1) can be rewritten making explicit $p^{n+1}_{i,j,k}$ as

$$\begin{aligned} p^{n+1}_{i,j,k} = {} & 2p^n_{i,j,k} - p^{n-1}_{i,j,k} - dt^2(\zeta_{x,i} + \zeta_{y,j} + \zeta_{z,k})\frac{p^{n+1}_{i,j,k} - p^{n-1}_{i,j,k}}{2dt} \\ & - dt^2(\zeta_{y,j}\zeta_{z,k} + \zeta_{x,i}\zeta_{z,k} + \zeta_{x,i}\zeta_{y,j})p^n_{i,j,k} + dt^2c^2\left[\frac{p^n_{i+1,j,k} - 2p^n_{i,j,k} + p^n_{i-1,j,k}}{dh^2} + \right. \\ & \left. \frac{p^n_{i,j+1,k} - 2p^n_{i,j,k} + p^n_{i,j-1,k}}{dh^2} + \frac{p^n_{i,j,k+1} - 2p^n_{i,j,k} + p^n_{i,j,k-1}}{dh^2}\right] \\ & + dt^2\frac{\widetilde{\phi}^n_{x,i+\frac{1}{2},j,k} - \widetilde{\phi}^n_{x,i-\frac{1}{2},j,k}}{dh} + dt^2\frac{\widetilde{\phi}^n_{y,i,j+\frac{1}{2},k} - \widetilde{\phi}^n_{y,i,j-\frac{1}{2},k}}{dh} \\ & + dt^2\frac{\widetilde{\phi}^n_{z,i,j,k+\frac{1}{2}} - \widetilde{\phi}^n_{z,i,j,k-\frac{1}{2}}}{dh} - dt^2\zeta_{x,i}\zeta_{y,j}\zeta_{z,k}\frac{\psi^{n+\frac{1}{2}}_{i,j,k} + \psi^{n-\frac{1}{2}}_{i,j,k}}{2}. \end{aligned}$$

7 | Adaptive Rectangular Decomposition

In this chapter we will present the *Adaptive Rectangular Decomposition* (ARD), a parallel algorithm designed for conducting acoustic simulations.
The following dissertation is based on Raghuvanshi et al. [2011], Mehra et al. [2012] and Savioja et al. [2010].

7.1. ARD algorithm

In this section we present the Adaptive Rectangular Decomposition algorithm, an evolution of the RDD algorithm presented in Sec. 5.5 which supports partially and fully absorbing boundary conditions. Furthermore, we describe in more details the *Preprocessing* stage, including the *rectangular decomposition* and *interface inference steps*.
We define two types of subdomains:

- *Air subdomains* - Subdomains where we model sound propagation in air employing the Fourier method (Ch. 4).
- *PML subdomains* - Subdomains where we model boundary (e.g., a wall) absorption. We attenuate incoming sound waves employing a Perfectly Matched Layer (Ch. 6).

As interface handling method, we can use either pre-merge or post-merge. As explained in Sec. 5.6, in the case of air-PML interfaces, we model partial absorption by multiplying the residual by a virtual boundary absorption coefficient β_B.
An ARD solver consists of two primary stages, *Preprocessing* and *Simulation*.

- *Preprocessing*.
    1. *Voxelization*. The input scene is voxelized [†] into grid cells at grid resolution dh determined by the relation

$$dh = \frac{\lambda_{min}}{s} = \frac{c}{f_{max}\, s},$$

[†]Voxelization is the process of discretizing an object into a 3D matrix, where each cell represents a voxel. A voxel is the equivalent three-dimensional unit element of a pixel (Bacciaglia et al. [2019]).

where λ_{min} is the minimum simulation wavelength, s is the number of samples per wavelength, c is the speed of sound, and f_{max} is the maximum usable simulation frequency (delimiting a bandwidth where dispersion is negligible).

2. *Rectangular decomposition.* This is followed by a rectangular decomposition step in which grid cells generated during voxelization are grouped into rectangles corresponding to air subdomains. PML subdomains are generated for each boundary. Both types of subdomain have the same grid resolution dh. Throughout this step, we perform any necessary precomputation for the DCTs to be performed at runtime to implement the Fourier method.

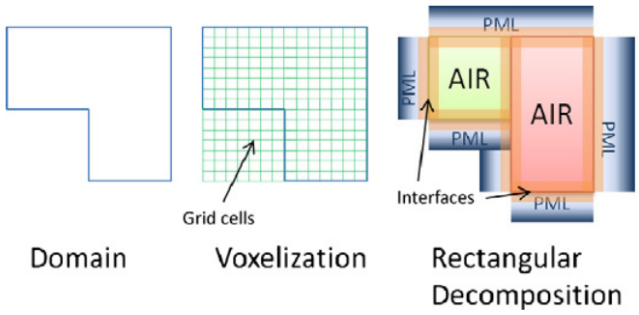
3. *Interface inference.* The interfaces between adjacent air-air and air-PML subdomains are created. This represents a one-time pre-computation.

- *Simulation.* We employ Alg. 5.1 (except the rectangular decomposition and interface inference steps) to perform the simulation. In air subdomains we employ the Fourier method, either the first ($\alpha > 0$) or the second order one ($\alpha = 0$). In PML subdomains, we employ the FDTD scheme derived in Sec. 6.3.

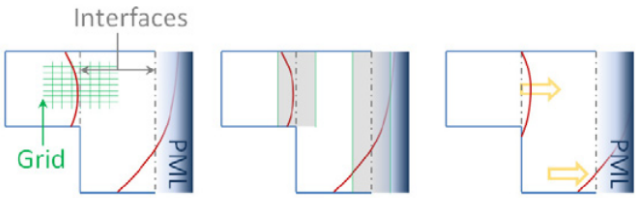
These steps are summarized in Fig. 7.1.1.

7.2. Parallelization

Mehra et al. [2012] presented a GPU-based acoustic solver which exploits parallelization based on the previously defined numerical simulation algorithm. In particular, the underlying algorithm exploits two levels of parallelism: *coarse grained* and *fine grained*.
Coarse grained parallelism is due to the fact that each of the subdomains (air or PML) solves the wave equation in an independent manner. Fine grained parallelism is achieved because, within each subdomain, all the grid cells are independent of each other with regards to solving the wave equation at a particular time-step. Therefore, within each subdomain, all the grid cells can run in parallel exhibiting fine grained parallelism.
The resulting GPU-based solver exploits both these levels of parallelism. It launches as many threads in parallel as there are subdomains. Each of these threads performs the computations to solve the wave equation for a particular subdomain. All these threads are grouped into blocks and grids, and scheduled by the runtime environment on the GPU. It is able to compute accurate impulse responses for complex scenes using single precision arithmetic.



(a) Preprocessing.



(b) Simulation.

Figure 7.1.1: (a) In the preprocessing stage, the input domain is voxelized into grid cells and adaptively decomposed into rectangular subdomains. Artificial interfaces and PML absorbing layers are created between neighboring subdomains and on the scene boundary respectively. (b) During the simulation stage, we update the pressure field in each subdomain and perform interface handling between neighboring subdomains. Source: Mehra et al. [2012].

7.3. Implementation

In this section, we present our implementation of the Adaptive Rectangular Decomposition (ARD) algorithm in C++, which is based on an existing GitHub project by the user *jinnsjj* called *ARD simulator*, available at `https://jinnsjj.github.io/ARD-simulator/`. This section presents the architecture of ARD simulator, outlines the modifications we made, and highlights the enhanced capabilities that we have introduced.

Libraries and dependencies

In our implementation of the ARD algorithm, we utilize several libraries and dependencies which provide various functionalities and support for different aspects of the program. Below, we introduce the key libraries used in our implementation and discuss their importance.

- **SDL**: SDL (Simple DirectMedia Layer) is used as the interface to visualize and

display the wave propagation in the ARD simulator. It provides a convenient and efficient way to create graphical user interfaces (GUI) and handle multimedia elements. The integration of SDL allows us to present the ARD algorithm's results in an intuitive and visually appealing manner.

- **SDL_ttf**: SDL_ttf is an extension library for SDL that enables the rendering of TrueType fonts. We utilize this library to display text and annotations within the ARD simulator. It provides us with the flexibility to present additional information and details to the user during the simulation.
- **OpenMP**: OpenMP is an API that supports parallel programming in shared-memory systems. We incorporate OpenMP into our implementation to leverage the power of multi-core processors and enhance the performance of the ARD algorithm. By parallelizing certain computations (see Sec. 7.2), we can speed up the execution and improve the overall efficiency of the algorithm.
- **FFTW**: FFTW (Fastest Fourier Transform in the West) is a library for computing discrete Fourier transforms (DFT) efficiently. We utilize FFTW to perform the necessary Discrete Cosine Transform (see Subsec. 2.2.2) within the ARD algorithm. Specifically, we use the provided functions for real-to-real (r2r) transforms, such as FFTW_REDFT10 and FFTW_REDFT01, to transform the input values and modes. FFTW's optimized algorithms help us achieve faster and more accurate computations.

7.3.1. Initialization and simulation loop

In this subsection, we provide description of the initialization and simulation phases of our implementation. We also include relevant snippets of the C++ code.
Initialization:

1. Initialize simulation parameters, propagation speed c_0, grid spacing dh, time step dt, air absorption coefficient α, virtual boundary absorption coefficient β_B, number of PML layers.
2. Build air subdomains reading their position, geometry, and air absorption coefficient from file.
3. Initialize sources and recorders reading their location from file.
4. Build PML subdomains to model the boundaries.
5. For each subdomain, initialize the variables to perform simulation. For air subdo-

mains, we employ the Fourier method: second order Fourier if there's no damping, first order Fourier if there's damping. For PML subdomains, we employ second order FDTD.

6. Infer the boundaries (interfaces, both Air-Air and Air-PML) from geometry.
7. Initialize rendering engine with SDL.

In particular, we show a code snippet used to perform the initialization of the variables needed for the Fourier method:

```
alpha_ = alpha_abs;
alpha2_ = alpha_ * alpha_;
eatm_ = exp(-alpha_ * dt_);

lx2_ = width_ * width_*dh_*dh_;
ly2_ = height_ * height_*dh_*dh_;
lz2_ = depth_ * depth_*dh_*dh_;

for (int i = 1; i <= depth_; i++)
{
  for (int j = 1; j <= height_; j++)
  {
    for (int k = 1; k <= width_; k++)
    {
      int idx = (i - 1) * height_ * width_ + (j - 1) * width_ +
          (k - 1);
      double w = c0_ * M_PI * sqrt((i - 1) * (i - 1) / lz2_ + (j
          - 1) * (j - 1) / ly2_ + (k - 1) * (k - 1) / lx2_);
      cwt_[idx] = cos(w * dt_);
      swt_[idx] = sin(w * dt_);
      w_omega_[idx] = w;
      w2_[idx] = w * w;
      inv_w_[idx] = 1 / w;
      inv_w2_[idx] = inv_w_[idx] * inv_w_[idx];
    }
  }
}
```

The following snippet represents the simulation loop (Cfr. Sec. 5.5):

```
// update pressure
#pragma omp parallel for
for (int i = 0; i < partitions_.size(); i++)
{
  partitions_[i]->Update_pressure();
  //std::cout << "update pressure partition " << partition->
     info_.id << " ";
}

// post-merge phase 1 (correct pressure)
if (!is_pre_merge) {
  #pragma omp parallel for
  for (int i = 0; i < partitions_.size(); i++)
  {
    partitions_[i]->PostMerge(1);
  }
  //std::cout << std::endl;
}

// reset residue
#pragma omp parallel for
for (int i = 0; i < partitions_.size(); i++)
{
  partitions_[i]->reset_residues();
}

// compute residue
#pragma omp parallel for
for (int i = 0; i < boundaries_.size(); i++)
{
  boundaries_[i]->ComputeResidues();
}
//std::cout << std::endl;

// reset force
#pragma omp parallel for
for (int i = 0; i < partitions_.size(); i++)
```

```
{
  partitions_[i]->reset_forces();
}

// compute force
#pragma omp parallel for
for (int i = 0; i < partitions_.size(); i++)
{
  partitions_[i]->ComputeSourceForcingTerms(time_step);
  //std::cout << "impose force partition " << partition->info_.
     id << " ";
}

// pre-merge (correct force)
if (is_pre_merge) {
  #pragma omp parallel for
  for (int i = 0; i < partitions_.size(); i++)
  {
    partitions_[i]->PreMerge(); // use corrected force
  }
  //std::cout << std::endl;
}
else {
  #pragma omp parallel for
  for (int i = 0; i < partitions_.size(); i++)
  {
    partitions_[i]->NoPreMerge(); // use not corrected force
  }
  //std::cout << std::endl;
}

// update pressure velocity
#pragma omp parallel for
for (int i = 0; i < partitions_.size(); i++)
{
  partitions_[i]->Update_velocity();
  //std::cout << "update pressure velocity partition " <<
```

```
      partition->info_.id << " ";
}

// post-merge phase 2 (correct pressure velocity)
if (!is_pre_merge) {
  #pragma omp parallel for
  for (int i = 0; i < partitions_.size(); i++)
  {
    partitions_[i]->PostMerge(2);
  }
  //std::cout << std::endl;
}
```

In particular, we show two code snippets respectively used to update the pressure and the pressure velocity with the Fourier method:

```
// prev_pressure_modes_ previous pressure
pressure_.ExecuteDct(); // current pressure
velocity_.ExecuteDct(); // current pressure velocity
force_.ExecuteDct(); // current force
force_r_.ExecuteDct(); // current corrected force

for (int i = 0; i < depth_; i++)
{
  for (int j = 0; j < height_; j++)
  {
    for (int k = 0; k < width_; k++)
    {
      int idx = i * height_ * width_ + j * width_ + k;

      if (idx == 0) {
        if (second_order_)
        next_pressure_modes_[idx] = (1 - 1e-10) * ( 2.0 *
            pressure_.modes_[idx] - prev_pressure_modes_[idx] +
            dt_ * dt_ * force_r_.modes_[idx]);
        else{
          next_pressure_modes_[idx] = (1 - 1e-10) * ( pressure_.
              modes_[idx] + dt_ * velocity_.modes_[idx] );
```

```
        }
      }else{
        if (second_order_)
        next_pressure_modes_[idx] = (1 - 1e-10) * ( 2.0 *
            pressure_.modes_[idx] * cwt_[idx] -
            prev_pressure_modes_[idx] + (2.0 * force_r_.modes_[
            idx] * inv_w2_[idx]) * (1.0 - cwt_[idx]) );
        else{
          double xe = force_.modes_[idx] * inv_w2_[idx];
          next_pressure_modes_[idx] = (1 - 1e-10) * (xe + eatm_
              * ((pressure_.modes_[idx] - xe) * (cwt_[idx] +
              alpha_ * inv_w_[idx] * swt_[idx]) + swt_[idx] *
              inv_w_[idx] * velocity_.modes_[idx]) );
        }
      }
    }
  }
}

memcpy((void *)prev_pressure_modes_, (void *)pressure_.modes_,
    depth_ * width_ * height_ * sizeof(double));
memcpy((void *)pressure_.modes_, (void *)next_pressure_modes_,
    depth_ * width_ * height_ * sizeof(double));

pressure_.ExecuteIdct();
```

```
if (second_order_) // pressure velocity is not considered in the
    second order case
  return;

// prev_pressure_modes_ current pressure
velocity_.ExecuteDct(); // current pressure velocity
force_.ExecuteDct(); // next force
force_r_.ExecuteDct(); // next corrected force

for (int i = 0; i < depth_; i++)
```

```
{
  for (int j = 0; j < height_; j++)
  {
    for (int k = 0; k < width_; k++)
    {
      int idx = i * height_ * width_ + j * width_ + k;

      if (idx == 0) {
        next_velocity_modes_[idx] = (velocity_.modes_[idx] + dt_
            * force_r_.modes_[idx]) / (1 + 2 * dt_ *
            air_absorption_);
      }
      else {
        double xe = force_r_.modes_[idx] * inv_w2_[idx];
        next_velocity_modes_[idx] = eatm_ * (velocity_.modes_[
            idx] * (cwt_[idx] - alpha_ * inv_w_[idx] * swt_[idx])
            - (w_omega_[idx] + alpha2_ * inv_w_[idx]) * (
            prev_pressure_modes_[idx] - xe) * swt_[idx]);
      }
    }
  }
}

memcpy((void*)prev_velocity_modes_, (void*)velocity_.modes_,
    depth_ * width_ * height_ * sizeof(double));
memcpy((void*)velocity_.modes_, (void*)next_velocity_modes_,
    depth_ * width_ * height_ * sizeof(double));

velocity_.ExecuteIdct();
```

7.3.2. Bugfixes and new features

We made several contributions to the original implementation, including:

- Improved code readability.
- Added support for air damping using the first order Fourier method.
- Fixed the normalization of the inverse Discrete Cosine Transform (iDCT) result,

ensuring that the solution is not doubled as a result of the concatenation of the Discrete Cosine Transform (DCT) and the inverse DCT (iDCT) operations.

- Implemented post-merge.

These contributions enhance the overall functionality and reliability of the implementation.

7.4. Numerical simulation of a test scenario

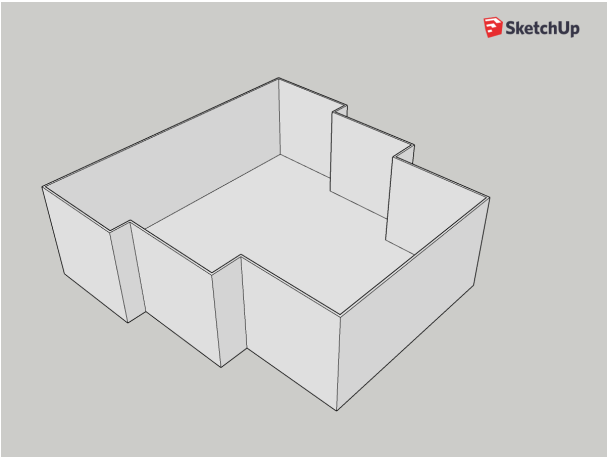
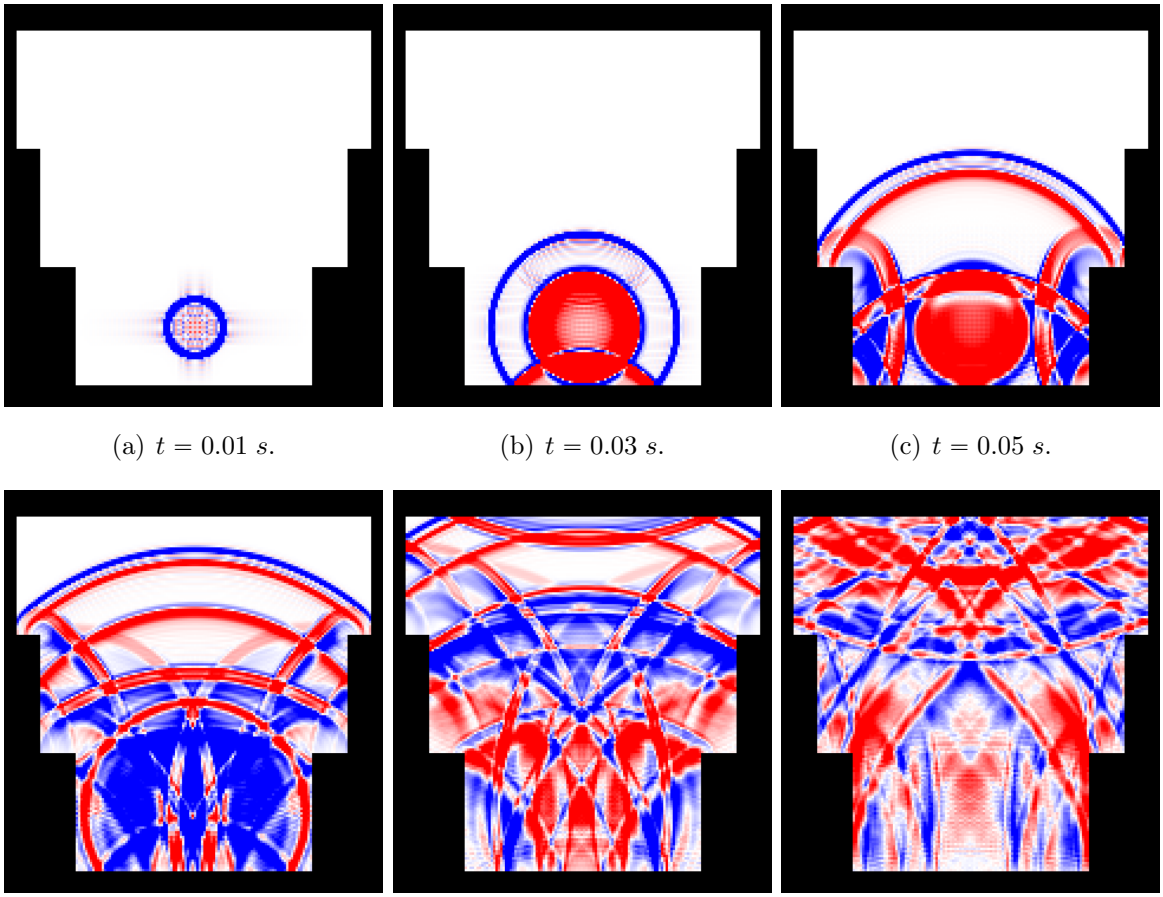


Figure 7.4.1: Test scenario for ARD simulator: minimalistic hall. Source: noa.

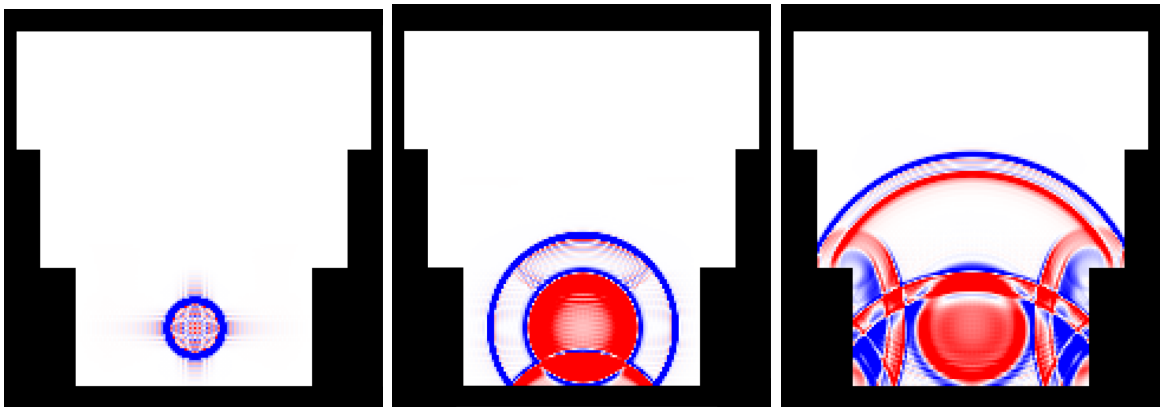
We have conducted experiments using the *ARD simulator* in a simplified hall environment depicted in Fig. 7.4.1. The hall has a volume of 7600 m^3 and a surface area of 760 m^2, partitioned into three rectangles. The simulation parameters we have chosen are as follows: the propagation speed c_0 is 343.5 m/s, the grid spacing dh is 0.2, and the time step dt is $2e-4$.
For the first case, where the air absorption coefficient is $\alpha = 0$ and the boundaries absorption coefficient is $\alpha_B = 0.5$, we present snapshots of the simulation at different time points in Fig. 7.4.2. Conversely, for the second case, where the air absorption coefficient is $\alpha = 10$ and the boundaries absorption coefficient is $\alpha_B = 0.5$, we display snapshots in Fig. 7.4.3. In both cases, the applied force is modeled as in Sec. 2.1.5, so the force envelope $f_{env}(\underline{x})$ is a Gaussian, and the time evolution term $f_{time}(t)$ is a Gaussian too (see Fig. 7.6.4). The interface handling method chosen is pre-merge for both cases, and the thickness of the perfectly matched layer (PML) is set to 5.
It is worth noting that the second solution is identical to the first one, except for the decaying pressure caused by air damping.



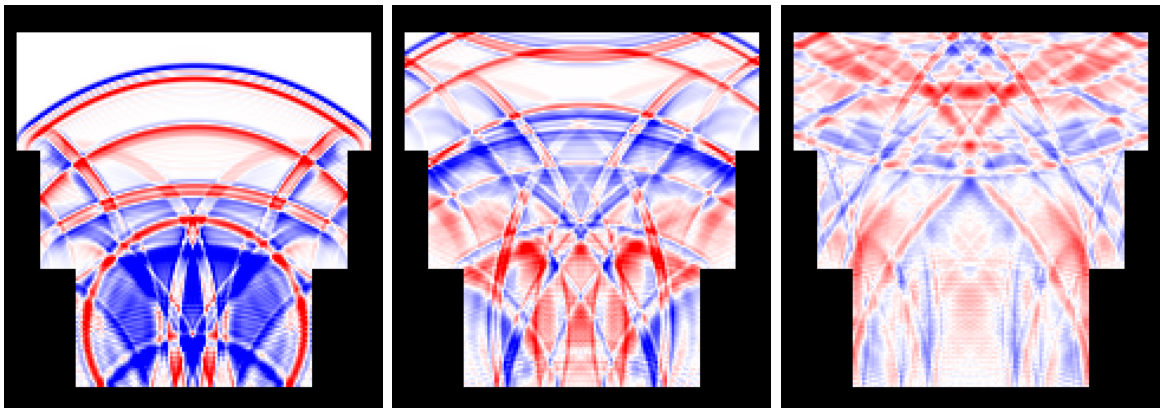
(a) $t = 0.01\ s$. (b) $t = 0.03\ s$. (c) $t = 0.05\ s$.

(d) $t = 0.07\ s$. (e) $t = 0.09\ s$. (f) $t = 0.11\ s$.

Figure 7.4.2: Snapshots of ARD simulator at different time instants of a test case with propagation speed $c_0 = 343.5\, m/s$, grid spacing $dh = 0.2$, time step $dt = 2e-4$, air absorption coefficient $\alpha = 0$, boundaries absorption coefficient $\alpha_B = 0.5$, pre-merge interface handling method, and PML thickness 5.



(a) $t = 0.01\ s$. (b) $t = 0.03\ s$. (c) $t = 0.05\ s$.



(d) $t = 0.07\ s$. (e) $t = 0.09\ s$. (f) $t = 0.11\ s$.

Figure 7.4.3: Snapshots of ARD simulator at different time instants of a test case with propagation speed $c_0 = 343.5\, m/s$, grid spacing $dh = 0.2$, time step $dt = 2e-4$, air absorption coefficient $\alpha = 10$, boundaries absorption coefficient $\alpha_B = 0.5$, pre-merge interface handling method, and PML thickness 5.

7.5. Numerical errors

Numerical errors in ARD are introduced mainly through two sources: boundary approximation and interface errors. The following dissertation is based on Raghuvanshi et al. [2011].

By employing voxelization to approximate the simulation domain, staircasing errors occur near the boundary (see Fig. 7.5.1). Voxelization, in fact, entails shape representation approximations, leading to a slight alteration of the external shape of a part based on the voxel dimensions (Bacciaglia et al. [2019]). Luckily, geometric features comparable or smaller than the wavelength of sound (34 cm at 1 kHz) lead to very small variations in the overall acoustics of the scene due to the presence of diffraction.

The net effect of staircasing error is that for frequencies with wavelengths comparable to the cell size (1 kHz), the walls act as diffuse instead of specular reflectors. For frequencies with large wavelengths (500 Hz and below), the roughness of the surface is effectively "invisible" to the wave, and the boundary errors are small with near-specular reflections. Therefore, the perceptual impact of boundary approximation is lesser in acoustic simulation.

However, if very high boundary accuracy is critical for a certain scene, this can be achieved by coupling our approach with a high-resolution grid near the boundary, running FDTD at a smaller time step. In Borrel-Jensen et al. [2023], instead, it's suggested to couple the Fourier method with the Spectral Element Method. Of course, this would create extra computational overhead, so its an efficiency-accuracy trade-off.

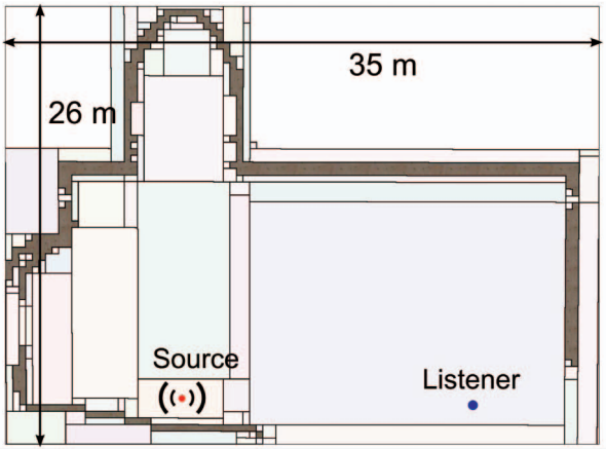


Figure 7.5.1: Voxelization and rectangular decomposition of a Cathedral. Source: Raghuvanshi et al. [2011].

As explained in Sec. 5.5.2, coupling the Fourier method with interface handling based

on $(2, 6)$ FDTD creates interface errors. These errors increase with increasing frequency, but it stays ≈ -40 dB for most of the spectrum. To sum up, ideal numerical dispersion is traded off for very small spurious reflections which are imperceptible.

7.6. Reverberation analysis

The following dissertation is based on Kuttruff and Everton [2010].
Within an enclosure, from the geometrical acoustics point of view, each sound reflection is characterized by the time of arrival, its amplitude, and its direction. We can plot these reflections as perpendicular dashes over a horizontal time axis, whose weight depends on the amplitude: we call this plot *reflection diagram* or *echogram*, and an example is shown in Fig. 7.6.1.

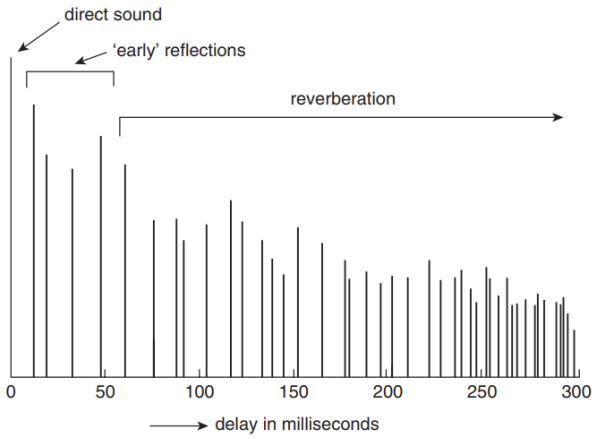


Figure 7.6.1: Schematic reflection diagram. Source: Kuttruff and Everton [2010].

Analyzing the figure, we notice the direct response (DR), the early response (ER), and the late response (LR). The DR is the direct sound from the source to the receiver, the ER is typically the sound reaching the receiver within the first $80 - 100$ ms, and the LR is the sound reaching the receiver after 100 ms of being emitted from the source (Chandak et al. [2011]). Usually, the early reflections are strong and distinguishable, while the later reflections (which constitute the *reverberation tail*) are denser and weaker.
In general, from the wave based acoustics point of view, the same information can be inferred from the *room's impulse response*. When a room is excited by a very short sound impulse emitted at time $t = 0$, we obtain, in the limit of vanishing pulse duration, the impulse response $g(t)$: an example is shown in Fig. 7.6.2. In general, we have a different impulse response for each combination of source and listener position, that's why it's also called *point-to-point impulse response*.

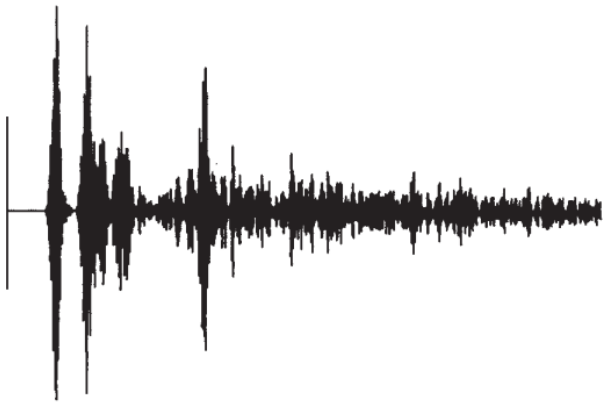


Figure 7.6.2: Impulse response of a room for a given source and listener position. Source: Kuttruff and Everton [2010].

Making a huge simplification, we can say that the echogram contains the peaks of the room's impulse response.
Typically, reverberation analysis is performed on a *decay level* curve L_r (i.e., the sound pressure level of the decaying sound field), defined as

$$L_r(t) \triangleq 10 \log_{10} \left(\frac{w(t)}{w(0)} \right) = 20 \log_{10} \left(\frac{p(t)}{p(0)} \right),$$

where w is the *energy density*, while p is the pressure.
This means that the decay level curve L_r does not only depend on the enclosure geometry and absorption characteristics, but also on the excitation provided to the room. In fact, an expression proportional to the energy density w (also known as *energy decay*) is obtained by squaring the room's response:

$$w(t) \propto h^2(t),$$

where $h(t)$ is given by the convolution between the room's impulse response $g(t)$ and a given stationary signal $s(t)$ which is switched off at $t = 0$, i.e.,

$$h(t) = g(t) * s(t) = \int_{-\infty}^{0} g(\tau) s(t - \tau) d\tau.$$

The best known and most important quantity in room acoustics is the *reverberation time* T_{60}, introduced by Sabine, and defined as the time interval in which the decay level drops down by 60 dB. Typical values of reverberation times range from $0.3s$ in living rooms to $10s$ in large churches and empty reverberation chambers. Usually, large halls have reverberation times in the range $[0.7, 2]s$.
The reverberation time can be either measured or estimated. The most general and simple

formula is the modified † *Sabine's reverberation time*:

$$T_{60} \approx 0.161 \frac{V}{A + 4mV}, \tag{7.6.1}$$

where V is the room's volume, $m \triangleq \alpha/c$ is the *dimensionless air absorption coefficient* (which can be neglected for small rooms), and A is the *equivalent absorption area*, defined as

$$A \triangleq \alpha_{av} S,$$

where α_{av} is the *average boundary absorption coefficient* of the boundaries, while S is the room's surface.
Eq. (7.6.1) fails for $\alpha_{av} \to 1$: it predicts a finite reverberation time, but an enclosure whose walls are perfectly reflecting cannot generate a reverberating field. A more accurate formula is the modified *Eyring's reverberation time*:

$$T_{60} \approx 0.161 \frac{V}{S\,|\ln(1 - \alpha_{av})| + 4mV}. \tag{7.6.2}$$

Let's now consider the measurement of reverberation. Usually, the energy decay curves have quasi-random fluctuations: one possibility to attenuate them would be to average over a great number of individual decay curves, each of which obtained by random noise excitation of the room. A more elegant approach was proposed by Schroeder, which invented a technique called *backward integration* used to compute the ensemble average of all possible energy decay curves:

$$\langle w(t) \rangle = \langle h^2(t) \rangle \triangleq \int_t^\infty g^2(\tau) d\tau = \int_0^\infty g^2(\tau) d\tau - \int_0^t g^2(\tau) d\tau. \tag{7.6.3}$$

Of course, the upper limit ∞ must be replaced with a finite value. If this limit is too long, we pick up too much noise (in our case, numerical error); instead, if the limit is too short, we notice an early downward bend of the curve.
The theoretical energy decay curve can be expressed as

$$w(t) = w(0) e^{-\left(\frac{cA}{4V} + 2\alpha\right)t}. \tag{7.6.4}$$

Notice that this is an approximation valid under the *Sabine's continuity hypothesis*; in practice, as Eyring first discovered, the energy decay is a step-wise process. In Fig. 7.6.3 we compare the two energy decay models.

†It's "modified" because it also depends on the dimensionless air absorption coefficient m.

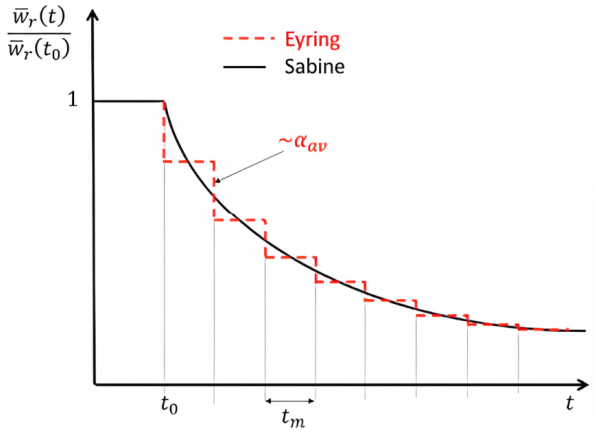


Figure 7.6.3: Sabine's versus Eyring's energy decay process. Source: Diffuse Field Model slide from the course "Room Acoustics" (2022-2023) by Maria Cairoli and Livio Mazzarella.

Expressing $\langle w(t) \rangle$ in decibel we obtain the ensemble average of all possible decay level curves:

$$\langle L_r(t) \rangle = 10 \log_{10} \left(\frac{\langle w(t) \rangle}{w(0)} \right).$$

Starting from the ensemble average of the decay level curves, we measure T_{60}, recalling the definition, as

$$T_{60} = t_2 - t_1,$$

where $t_2 = t \;\; s.t. \;\; \langle L_r(t) \rangle = -60$ dB, while t_1 is the time instant when the sound source is turned off.
We also define another related parameter, T_{30}:

$$T_{30} = 2(t_2 - t_1),$$

where $t_2 = t \;\; s.t. \;\; \langle L_r(t) \rangle = -30$ dB, while t_1 is the time instant when the sound source is turned off. This parameter is used to provide an estimation of the reverberation time over a shorter range.
Up to now we have only considered time domain analysis of reverberation. Useful insight is obtained by analyzing the reverberation in the frequency domain: we obtain the *room's frequency response* by taking the Fourier transform of the room's impulse response. It provides a way to detect the most prominent modes of oscillation within a room and, as we'll see in Sec. 7.7, it makes possible to efficiently simulate the acoustics of the environment through frequency domain filtering.

7.6.1. Algorithm to compute Room's Impulse Response

A commonly used method to measure the impulse response of a room, denoted as $h(t)$, involves applying a known input signal, denoted as $s(t)$, measuring the room's response, denoted as $g(t)$, and then performing deconvolution (Stan et al. [2002]). There are different approaches to obtain the room's response:

1. One approach is to provide the input signal to a real room using a loudspeaker and measure the response using a microphone.
2. Another approach is similar to the first one, but instead of a real room, the response is measured in a scale model of the room. This requires appropriate scaling of the response.
3. A numerical simulation can also be performed, where the receiver is modeled by the pressure measured at specific grid points, analogous to a microphone.

In our case, we choose the numerical simulation approach. This method has the advantage of only requiring a computer and does not involve measurement noise, unlike the other approaches. In this chapter, we have presented the ARD algorithm, which is well-suited for numerical acoustics simulations due to its characteristics, such as no dispersion and high efficiency. Therefore, it is a suitable choice for measuring the impulse response.

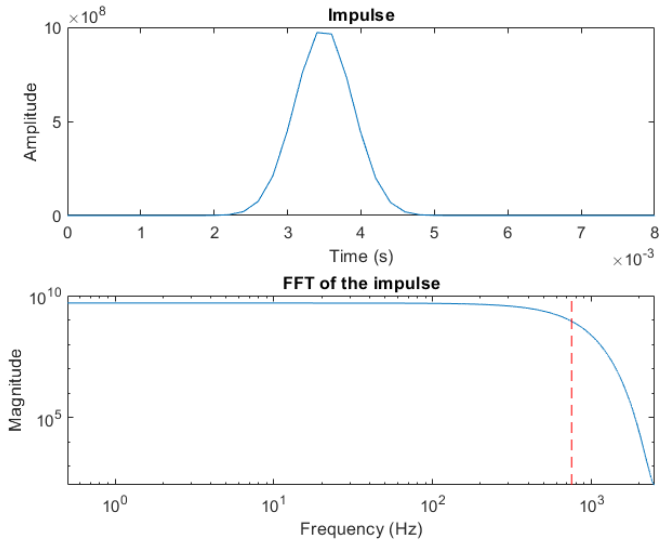


Figure 7.6.4: Excitation signal in the time domain and associated FFT for $dt = 2e - 4$. The dashed vertical red line represents the cutoff frequency 750 Hz, which is lower than the Nyquist frequency $1/dt = 5000$ Hz.

To ensure accurate measurements, it is important to use an omnidirectional sound source

with a flat frequency response within the desired range. Various source signals have been proposed in the literature (Murphy et al. [2014]). In our study, we employ a force whose envelope $f_{env}(\underline{x})$ is a Gaussian, and whose time evolution term $f_{time}(t)$ is a Gaussian too, as described in Sec. 7.4. Fig. 7.6.4 shows the excitation signal in the time domain and its associated FFT. The dashed vertical red line represents the cutoff frequency of 750 Hz, which is lower than the Nyquist frequency of 5000 Hz, determined by the simulation parameter dt.
After executing ARD, we employ the following algorithm to perform the measurement and the upsampling of the room's impulse response, which is based on the one proposed in Raghuvanshi et al. [2011]:

1. **Data loading and preprocessing**.
    - *Load Source and Response*: The code loads the source (excitation) signal $\underline{s}$ and the room's response signal $\underline{g}$ at listener location in the desired environment obtained through simulation. Based on the dh and dt parameters employed, it computes the sampling rate f_s of the signals and the cutoff frequency f_c for the impulse response.
    - *Truncation of Response*: The response $\underline{g}$ is truncated before it contains only noise. The truncation time corresponds to Sabine's reverberation time estimated with Eq. (7.6.1).
2. **Impulse Response Computation**: The impulse response $\underline{h}$ is computed through deconvolution. Following the procedure described in Havelock et al. [2008], which is valid in case of white excitation, the deconvolution is executed by performing the cross-correlation between the room's response $\underline{g}$ and the time-reversed excitation signal $\underline{s}$. The code performs cross-correlation between the truncated response and the source signal, followed by low pass filtering with cutoff frequency f_c.
3. **Impulse Response Upsampling**.
    - *Upsampling*: The computed impulse response $\underline{h}$ is upsampled to a desired higher sampling rate f_s^{new}, obtaining $\underline{h}^{new}$.
    - *Cleaning through Peak Detection*: The code performs peak detection on the upsampled impulse response $\underline{h}^{new}$ to identify the significant peaks. By retaining only the significant peaks, we obtain a cleaner version of the upsampled impulse response, denoted by $\underline{h}^{clean}$.
    - *High-pass Filtering*: The process of cleaning reduces the accuracy of the impulse response. As for the low frequencies we already have an accurate impulse

response, i.e., $\underline{h}$, the clean upsampled impulse response is high-pass filtered with cutoff frequency f_c, obtaining $\underline{h}^{high}$.

- *Summation of Impulse Responses*: The accurate low-frequency impulse response $\underline{h}$ and the estimated high-frequency impulse response $\underline{h}^{high}$ are summed to create the final impulse response $\underline{h}^{up}$ for auralization.

The **Impulse Response Upsampling** step is a signal enhancement strategy performed on the impulse response to extend its bandwidth while retaining the accuracy given by simulation in the low-frequency band. Directly simulating in a higher bandwidth would be too computationally expensive. A more accurate approach would be to combine our technique with a Geometrical Acoustic simulator for the higher frequency range.

7.6.2. Reverberation analysis on a test scenario

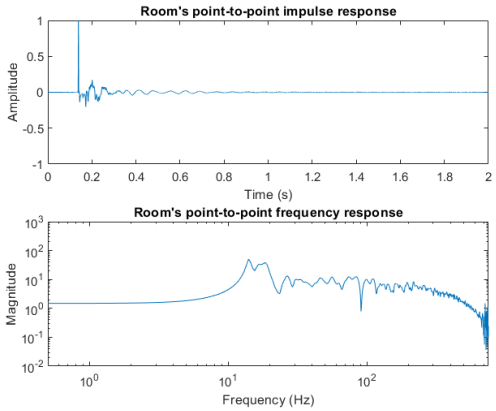
We now show the results of reverberation analysis (impulse response, frequency response, energy decay curve, measured T_{30} and T_{60}) for the test scenario in Sec. 7.4. We also plot the theoretical energy decay curve given by Eq. (7.6.4).
We consider the following cases:

- Air absorption coefficient $\alpha = 0$, PML thickness 25, and virtual boundary absorption coefficient $\beta_B = 1 \longrightarrow \alpha_B = 0.20247$ - The impulse response, frequency response, and energy decay curve are shown in Fig. 7.6.5, while the measured reverberation time parameters are $T_{30} = 2.0168s$ and $T_{60} = 1.9884s$.

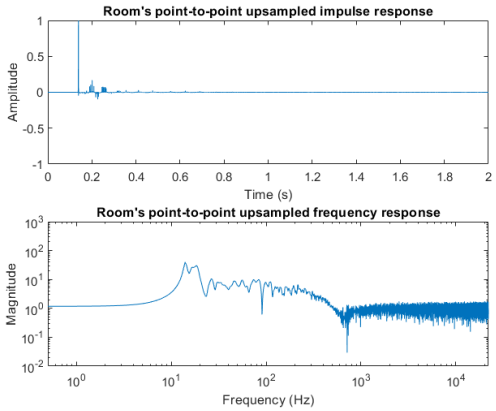
- Air absorption coefficient $\alpha = 0$, PML thickness 5, and virtual boundary absorption coefficient $\beta_B = 0.9 \longrightarrow \alpha_B = 0.08724$ - The impulse response, frequency response, and energy decay curve are shown in Fig. 7.6.6, while the measured reverberation time parameters are $T_{30} = 5.2192s$ and $T_{60} = 4.9282s$. Notice that, in this case, the measured value of T_{60} is not valid because the energy decay curve has a downward bend near $t = 5$.

- Air absorption coefficient $\alpha = 10$, PML thickness 5, and virtual boundary absorption coefficient $\beta_B = 0.9 \longrightarrow \alpha_B = 0.08724$ - The impulse response, frequency response, and energy decay curve are shown in Fig. 7.6.7, while the measured reverberation time parameters are $T_{30} = 0.7732s$ and $T_{60} = 0.6494s$.

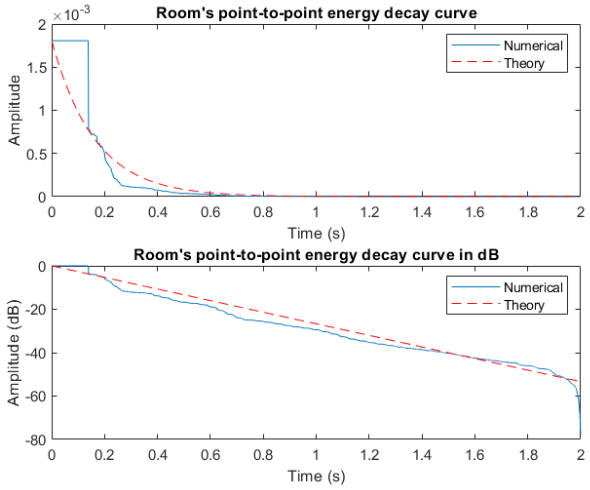
Notice that the values of α_B associated to the selected values of β_B are the values such that the expected value of reverberation time (computed with the modified Eyring's formula in Eq. (7.6.2)) most closely matches the measured value.



(a) Original.

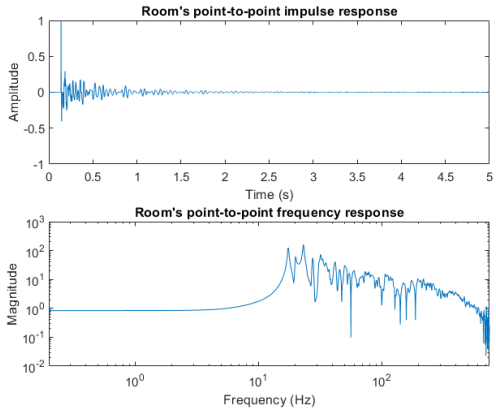


(b) Upsampled.

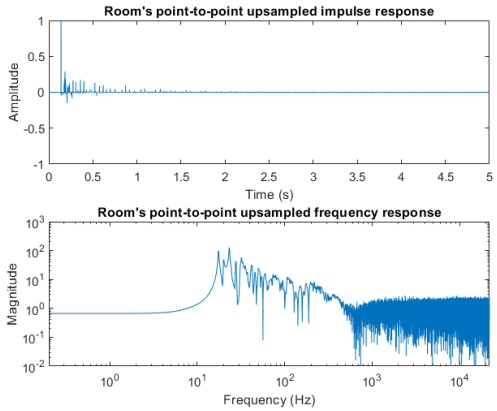


(c) Energy decay curve.

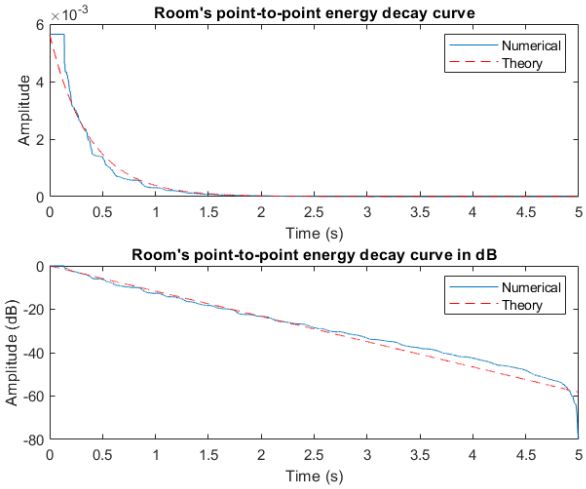
Figure 7.6.5: Point-to-point impulse response and frequency response before (a) and after upsampling (b) for the test scenario in Sec. 7.4 with air absorption coefficient $\alpha = 0$, virtual boundary absorption coefficient $\beta_B = 1$, and 25 PML layers. The frequency response magnitude is plotted in logarithmic scale. In (c) we plot the numerical and theoretical energy decay curve, both in linear scale and in decibels.



(a) Original.

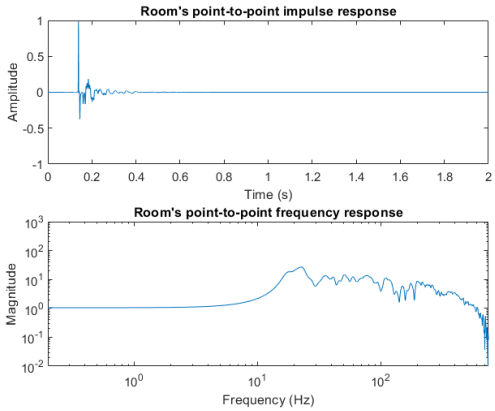


(b) Upsampled.

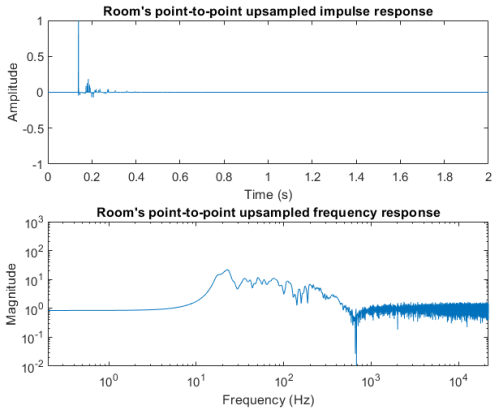


(c) Energy decay curve.

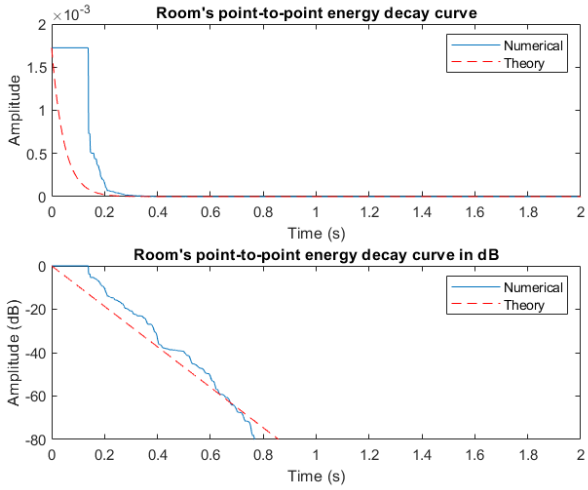
Figure 7.6.6: Point-to-point impulse response and frequency response before (a) and after upsampling (b) for the test scenario in Sec. 7.4 with air absorption coefficient $\alpha = 0$, virtual boundary absorption coefficient $\beta_B = 0.9$, and 5 PML layers. The frequency response magnitude is plotted in logarithmic scale. In (c) we plot the numerical and theoretical energy decay curve, both in linear scale and in decibels.



(a) Original.



(b) Upsampled.



(c) Energy decay curve.

Figure 7.6.7: Point-to-point impulse response and frequency response before (a) and after upsampling (b) for the test scenario in Sec. 7.4 with air absorption coefficient $\alpha = 10$, virtual boundary absorption coefficient $\beta_B = 0.9$, and 5 PML layers. The frequency response magnitude is plotted in logarithmic scale. In (c) we plot the numerical and theoretical energy decay curve, both in linear scale and in decibels.

7.7. Auralization

Auralization is the process of rendering audible, by physical or mathematical modeling, the sound field of a source in a space, in such a way as to simulate the listening experience at a given position in the modeled space. The aim is not primarily to recreate the sensation of the speech of music per se, but to recreate the aural impression of the acoustic characteristics of a space, be it outdoors or indoors (Kleiner et al. [1993]).
Another aspect which is strictly related to auralization, and which can be consider a subset of it, is that of *spatialization*. The purpose of this technique is to emulate the differences in sound heard at each ear, which is important for localization and immersion. If only auralization is performed, the user using headphones perceives the sound as being inside its head; through spatialization, the sound is externalized. This process involves filtering the monaural signal at the listener position with the HRTF (head-related transfer function), which models the scattering process from the user's body, head and ears (Kosikowski [2018], Zotkin et al. [2002]).
There are different types of auralization. The most important step is, indeed, the computation of the room's impulse response; thus, we can categorize auralization methods by the chosen simulation technique. We'll consider the algorithm presented in Sec. 7.6.1 to perform fully computed room's impulse measurement.
A second categorization is between *Real-time calculation* and *Pre-calculated* approaches. As the name suggests, in the former approach, the entire room acoustic simulation and the spatialization are computed in real-time; wave-based methods as Adaptive Rectangular Decomposition are ruled out because of their large computational overhead. This means that we need to go for a Pre-calculated approach: in a pre-calculation stage, we calculate the room impulse response for all possible positions of the source and the receiver. Of course, in the SS-SR case, this is trivial; however, expecially in the MS-MR case, it's clear that the computational effort to perform the pre-calculation step and the storage resources needed will be really demanding (Kosikowski [2018]).
Another categorization is given by the movement of the source and/or of the receiver:

- SS-SR - *Static Source - Static Receiver*;
- SS-MR - *Static Source - Moving Receiver*;
- MS-SR - *Moving Source - Static Receiver*;
- MS-MR - *Moving Source - Moving Receiver*.

We'll now present some techniques to perform auralization in these four scenarios which are based on Chandak et al. [2011].

In the SS-SR case, the propagation from the source to the receiver can be modeled by a *linear, time-invariant* (LTI) system. This means that the propagation effects are completely characterized by the source-to-listener impulse response. Hence, to perform auralization in the SS-SR case, we only need to compute the room's impulse response for the given source-receiver configuration and filter the dry (anechoic) signal with it. Filtering is achieved through convolution; to perform this operation efficiently, one may employ the *overlap-and-add* method: an extensive explanation of this technique can be found in Diniz et al. [2010].

Assuming that the propagation from the source to the receiver can be modeled as an LTI system is appropriate only for SS-SR scenarios. For the more general MS-MR scenarios, we need to consider the room's impulse response as *time-varying*. We'll work on short and constant duration frames of the dry signal: let $s_k(t)$ be the k-th frame of the signal $s(t)$. Furthermore, let S_k (R_k) be the position of the source (receiver) at frame k, and let $g_{k_1,k_2}(t)$ be the impulse response between R_{k_1} and S_{k_2}. The sound reaching R_k arrives from many previous source positions $\{S_k, S_{k-1}, \ldots, S_{k-L+1}\}$, as the sound emitted by previous source positions propagates through the scene before arriving at the receiver. For any previous source position S_{k-i}, the sound reaching the listener can be obtained by convolving $s_{k-i}(t)$, the sound emitted from the source in frame $k-i$, with $g_{k,k-i}(t)$, the room's impulse response between S_{k-i} and R_k. Hence, the signal received by receiver in frame k can be expressed as

$$r_k(t) = \sum_{i=0}^{L-1} g_{k,k-i}(t) * s_{k-i}(t). \tag{7.7.1}$$

Usually, to avoid discontinuities between frames, we multiply each frame by a window function $w(t)$ [†] before computing the output signal. The signal processing pipeline to perform MS-MR is illustrated in Fig. 7.7.1. The approach described for the MS-MR scenario can be particularized to the simpler scenarios of SS-MR and MS-SR. In particular, in the former case, we have that $S_1 = S_2 = \ldots = S_k$. Therefore, $g_{k,k-i}(t) = g_{k,k}(t)$, and Eq. (7.7.1) reduces to

$$r_k(t) = g_{k,k}(t) * \sum_{i=0}^{L-1} s_{k-i}(t).$$

Notice that, employing Adaptive Rectangular Decomposition for the simulation (or any other wave-based method), we directly obtain the room's response at all listener positions for a fixed source, meaning that the SS-MR scenario is clearly favored.

[†]A window function is a mathematical function that smoothly transitions from high values at the center to low values at the edges, typically symmetrically.

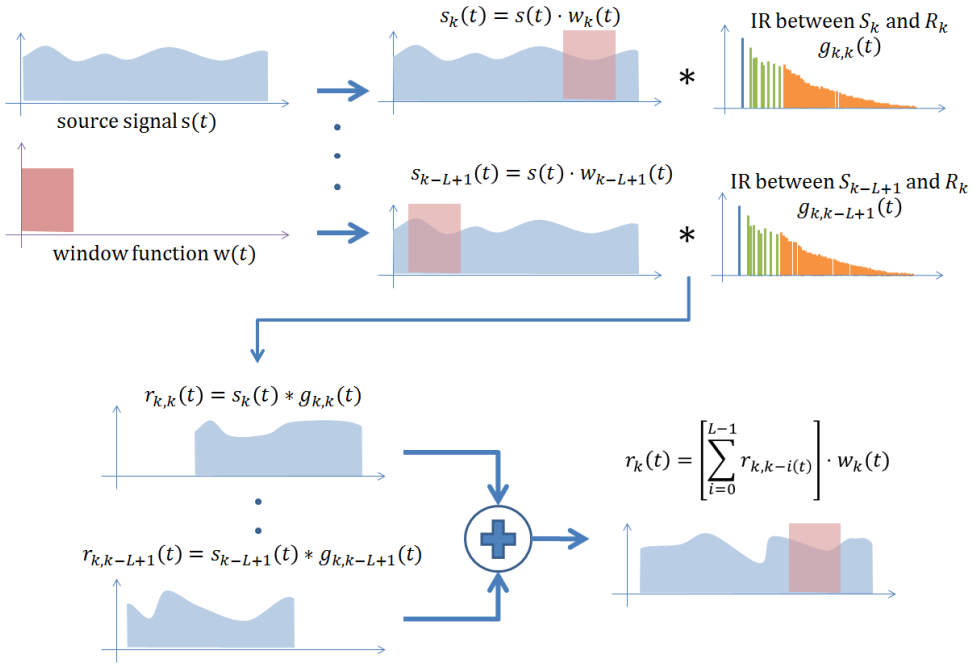


Figure 7.7.1: Signal processing pipeline for MS-MR auralization. Source: Chandak et al. [2011].

A simpler approach to perform SS-MR auralization was proposed in Raghuvanshi et al. [2011]. Current game and simulation engines, to perform an approximate auralization, employ manually configured reverb filters whose parameters depend on the listener location. The room's impulse response can be used to precompute high-quality reverb filters, removing the need of human intervention. Hence, we set physically motivated parameters to the reverb filters, improving the auralization quality and, at the same time, ensuring that the audio pipeline remains unchanged. This approach can be easily extended to the MS-MR case by performing numerical simulations of the acoustic response of a scene from several sampled source positions; to perform auralization, these responses will be interpolated according to the actual position of the source.
In the MS-SR case, instead, we have that $R_1 = R_2 = \ldots = R_k$. Therefore, $g_{k-i,k}(t) = g_{k,k}(t)$, and Eq. (7.7.1) reduces to

$$r_k(t) = \sum_{k=0}^{L-1} r_{k,k-i}(t),$$

$$r_{k,k-i}(t) = g_{k,k-i}(t) * s_{k-i}(t).$$

Finally, the quality of the auralization depends on how accurate the simulated room's impulse response is. The ARD algorithm, while being accurate in simulating sound prop-

agation, fails in realistically modeling the boundary conditions. Recent developments in frequency dependent absorbing and diffusive boundaries would offer a more accurate simulation. Furthermore, it could be possible to increase the auralization quality by considering source directivity and by employing a personalized HRTF function.

8 | Conclusion and future work

In this study, we have introduced Adaptive Rectangular Decomposition (Ch. 7) as a computation- and memory-efficient technique for accurate numerical acoustic simulations on large and complex domains. This approach surpasses traditional methods like the Finite-Difference Time-Domain (FDTD) method.

In fact, by employing the Fourier method (Ch. 4) for updating the solution in each subdomain, it does not introduce dispersion (as demonstrated in Subsec. 4.2.2 and Subsec. 4.3.2), making it well-suited for simulating acoustics in expansive environments and capturing high-order reflections. Furthermore, we achieve a remarkable enhancement in efficiency, with a minimum ten-fold reduction (Raghuvanshi et al. [2011]) in computational and memory requirements compared to the FDTD method (Ch. 3).

While our implementation showcases notable improvements, there are areas that warrant further refinement. One such area involves integrating a fine-grid simulation near the boundaries to mitigate errors caused by boundary reflections, as proposed in Borrel-Jensen et al. [2023]. The inclusion of a fine-grid simulation enables the reduction of artifacts resulting from reflections, leading to improved overall accuracy in the simulated acoustic environment.

Additionally, future research can explore the integration of geometric techniques to simulate wave (ray) propagation in the high-frequency range, going beyond the sole reliance on the upsampling technique utilized in computing the room's impulse response (Subsec. 7.6.1), which produces realistic but inexact results. By combining the ARD approach with geometric techniques, we could enhance the fidelity of the simulations in the high-frequency range.

Lastly, there is room for improvement in the modeling of boundaries: our ARD algorithm only supports non frequency dependent boundary absorption. Recent developments in frequency dependent absorbing and diffusive boundaries would offer a more complete simulation framework for room acoustics.

Bibliography

Lauri Savioja, Jyri Huopaniemi, Tapio Lokki, and Ritta Väänänen. Creating Interactive Virtual Acoustic Environments. *Journal of the Audio Engineering Society*, 47(9):675–705, September 1999. URL `https://www.aes.org/e-lib/browse.cfm?elib=12095`.

Jean-Marc Jot, Remi Audfray, Mark Hertensteiner, and Brian Schmidt. Rendering Spatial Sound for Interoperable Experiences in the Audio Metaverse. In *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, pages 1–15, Bologna, Italy, September 2021. IEEE. ISBN 9781665409988. doi: 10.1109/I3DA48870.2021.9610971. URL `https://ieeexplore.ieee.org/document/9610971/`.

F.G. Katz Brian, David Poirier-Quinot, and Jean-Marc Lyzwa. La Vierge 2020: Reconstructing a Virtual Concert Performance Through Historic Auralisation of Notre-Dame Cathedral. In *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, pages 1–9, September 2021. doi: 10.1109/I3DA48870.2021.9610849.

Kacper Kosikowski. Acoustic Virtual Reality – Methods and challenges. January 2018. URL `https://www.academia.edu/65330207/Acoustic_Virtual_Reality_Methods_and_challenges`.

Nikunj Raghuvanshi, Nikunj Raghuvanshi, Rahul Narain, Rahul Narain, Ming C. Lin, and Ming C. Lin. Efficient and accurate sound propagation using adaptive rectangular decomposition. *IEEE Transactions on Visualization and Computer Graphics*, page 4782956, 2011. ISSN 1077-2626. doi: 10.1109/TVCG.2009.27. URL `https://ieeexplore.ieee.org/document/4782956`.

Anish Chandak, Lakulish Antani, and Dinesh Manocha. Efficient Auralization for Moving Sources and Receiver. Technical Report, 2011. URL `https://techreports.cs.unc.edu/papers/11-008.pdf`.

Nakhlé H. Asmar. *Partial differential equations with Fourier series and boundary value problems*. Pearson Education, Upper Saddle River, NJ, 2nd ed., international ed edition, 2010. ISBN 9780132449007.

Stefan D. Bilbao. *Numerical sound synthesis: finite difference schemes and simulation in musical acoustics*. Wiley, Chichester, 2009. ISBN 9780470510469. OCLC: ocn319497904.

Heinrich Kuttruff and Pascal Everton. Room Acoustics, 5th Edition. *Noise Control Engineering Journal*, 58(4):464, 2010. ISSN 07362501. doi: 10.3397/1.3455049. URL http://www.ingentaconnect.com/content/ince/ncej/2010/00000058/00000004/art00011.

B. P. Lathi and R. A. Green. *Linear systems and signals*. The Oxford Series in Electrical and Computer Engineering. Oxford University Press, New York, third edition edition, 2018. ISBN 9780190200176.

D. R. Bull and Fan Zhang. *Intelligent image and video compression: communicating pictures*. Academic Press, London, second edition edition, 2021. ISBN 9780128203545. OCLC: 1245777824.

Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of Computation*, 51(184):699–706, 1988. ISSN 0025-5718, 1088-6842. doi: 10.1090/S0025-5718-1988-0935077-0. URL https://www.ams.org/mcom/1988-51-184/S0025-5718-1988-0935077-0/.

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical mathematics*. Number 37 in Texts in applied mathematics. Springer, Berlin ; New York, 2nd ed edition, 2007. ISBN 9783540346586.

G. D. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford applied mathematics and computing science series. Clarendon Press ; Oxford University Press, Oxford [Oxfordshire] : New York, 3rd ed edition, 1985. ISBN 9780198596417 9780198596509.

Jan L. Cieśliński. On the exact discretization of the classical harmonic oscillator equation. *Journal of Difference Equations and Applications*, 17(11):1673–1694, November 2011. ISSN 1023-6198, 1563-5120. doi: 10.1080/10236191003730563. URL http://www.tandfonline.com/doi/abs/10.1080/10236191003730563.

C. D. Meyer. *Matrix analysis and applied linear algebra*. Society for Industrial and Applied Mathematics, Philadelphia, 2000. ISBN 9780898714548.

Howard E. Bell. Gershgorin's Theorem and the Zeros of Polynomials. *The American Mathematical Monthly*, 72(3):292, March 1965. ISSN 00029890. doi: 10.2307/2313703. URL https://www.jstor.org/stable/2313703?origin=crossref.

Alfio Quarteroni and A. Valli. *Domain decomposition methods for partial differential equations*. Numerical mathematics and scientific computation. Clarendon Press, Oxford ; New York, 1999. ISBN 9780198501787.

Victorita Dolean, Pierre Jolivet, and Frederic Nataf. *An introduction to domain decomposition methods: algorithms, theory, and parallel implementation*. Society for Industrial and Applied Mathematics, Philadelphia, 2015. ISBN 9781611974058.

Manfred Kaltenbacher and Sebastian Floss. Nonconforming Finite Elements Based on Nitsche-Type Mortaring for Inhomogeneous Wave Equation. *Journal of Theoretical and Computational Acoustics*, 26(03):1850028, September 2018. ISSN 2591-7285, 2591-7811. doi: 10.1142/S2591728518500287. URL https://www.worldscientific.com/doi/abs/10.1142/S2591728518500287.

Ravish Mehra, Nikunj Raghuvanshi, Lauri Savioja, Ming C. Lin, and Dinesh Manocha. An efficient GPU-based time domain solver for the acoustic wave equation. *Applied Acoustics*, 73(2):83–94, February 2012. ISSN 0003682X. doi: 10.1016/j.apacoust.2011.05.012. URL https://linkinghub.elsevier.com/retrieve/pii/S0003682X11001605.

Nikunj Raghuvanshi, Nico Galoppo, and Ming C. Lin. Accelerated wave-based acoustics simulation. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 91–102, Stony Brook New York, June 2008. ACM. ISBN 9781605581064. doi: 10.1145/1364901.1364916. URL https://dl.acm.org/doi/10.1145/1364901.1364916.

Marcus J. Grote and Imbo Sim. Efficient PML for the wave equation, January 2010. URL http://arxiv.org/abs/1001.0319. arXiv:1001.0319 [math].

Kenneth Duru. *Perfectly matched layers and high order difference methods for wave equations*. Acta Universitatis Upsaliensis, Uppsala, 2012. ISBN 9789155483654. OCLC: 939792524.

Weng Cho Chew and William H. Weedon. A 3D perfectly matched medium from modified maxwell's equations with stretched coordinates. *Microwave and Optical Technology Letters*, 7(13):599–604, September 1994. ISSN 08952477, 10982760. doi: 10.1002/mop.4650071304. URL https://onlinelibrary.wiley.com/doi/10.1002/mop.4650071304.

L. Savioja, Dinesh Manocha, and Ming C. Lin. Use of GPUs in room acoustic modeling and auralization. 2010. URL https://www.semanticscholar.org/paper/Use-of-GPUs-in-room-acoustic-modeling-and-Savioja-Manocha/cf668f67f1de559efb76fbe1294b326e877a02da.

Antonio Bacciaglia, Alessandro Ceruti, and Alfredo Liverani. A systematic review of voxelization method in additive manufacturing. *Mechanics & Industry*, 20(6):630, 2019. ISSN 2257-7777, 2257-7750. doi: 10.1051/meca/2019058. URL https://www.mechanics-industry.org/10.1051/meca/2019058.

Adaptive rectangular decomposition simulator. URL https://jinnsjj.github.io/ARD-simulator/.

Nikolas Borrel-Jensen, Allan Peter Engsig-Karup, Maarten Hornikx, and Cheol-Ho Jeong. Accelerated sound propagation using an error-free Fourier method coupled with the spectral-element method. *INTER-NOISE and NOISE-CON Congress and Conference Proceedings*, 265(5):2731–2742, February 2023. ISSN 0736-2935. doi: 10.3397/IN_2022_0383. URL https://www.ingentaconnect.com/content/10.3397/IN_2022_0383.

Guy-Bart Stan, Jean-Jacques Embrechts, and Dominique Archambeau. Comparison of Different Impulse Response Measurement Techniques. *Journal of the Audio Engineering Society*, 50(4):249–262, April 2002. URL https://www.aes.org/e-lib/browse.cfm?elib=11083.

Damian T. Murphy, Alex Southern, and Lauri Savioja. Source excitation strategies for obtaining impulse responses in finite difference time domain room acoustics simulation. *Applied Acoustics*, 82:6–14, August 2014. ISSN 0003682X. doi: 10.1016/j.apacoust.2014.02.010. URL https://linkinghub.elsevier.com/retrieve/pii/S0003682X14000401.

David Ian Havelock, Sonoko Kuwano, and Michael Vorlander, editors. *Handbook of signal processing in acoustics*. Springer, New York, NY, 2008. ISBN 9780387776989 9780387304410.

Mendel Kleiner, Bengt-Inge Dalenback, and Peter Svensson. Auralization - an overview. *Journal of the Audio Engineering Society*, 41(11):861–875, November 1993. URL http://www.mattmontag.com/auralization/media/Auralization%20-%20An%20Overview.pdf.

Dmitry N. Zotkin, Ramani Duraiswami, and Larry S. Davis. Creation of virtual auditory spaces. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages II–2113–II–2116, May 2002. doi: 10.1109/ICASSP.2002.5745052. ISSN: 1520-6149.

Paulo Sergio Ramirez Diniz, Eduardo A. B. Da Silva, and Sergio L. Netto. *Digital signal processing: system analysis and design*. Cambridge University Press, New York, 2nd ed edition, 2010. ISBN 9780521887755.

List of Figures

List of Tables

List of Symbols

Variable	Description	SI unit
$\underline{x}$	space coordinates	m
t	time instant	s
L	domain length	m
T	simulation duration	s
p	pressure	Pa
v	pressure velocity	Pa/s
f	external force	Pa/s^2
z	state variable	
P	pressure DCT coefficients	Pa
V	pressure velocity DCT coefficients	Pa/s
F	external force DCT coefficients	Pa/s^2
c_0	propagation speed	m/s
ω	radian frequency	rad/s
k or β	wavenumber	rad/m
v_ϕ	phase velocity	m/s
v_g	group velocity	m/s
α	air absorption coefficient	m/s
m	dimensionless air absorption coefficient	m/s
β_B	virtual boundary absorption coefficient	m/s
α_B	physical boundary absorption coefficient	m/s
σ	standard deviation	m
μ	statistic mean	m
dh	space step	m
dt	time step	s
f_s	sampling frequency	Hz
λ	Courant number	
τ	truncation error	Pa/s^2

Variable	Description	SI unit
h	room's response	
g	room's impulse response	
S	room's boundaries surface	m^2
V	room's volume	m^3
α_{av}	average boundary absorption coefficient	
A	equivalent absorption area	m^2
w	energy density	$\mathrm{J/m}^3$
L_r	decay level	dB
T_{60}	reverberation time (measured over 60 dB)	s
T_{30}	reverberation time measured over 30 dB	s

Acknowledgements

I would like to express my heartfelt gratitude to all those who have contributed to the completion of this thesis. Their unwavering support, belief in my abilities, and presence throughout my academic journey have been invaluable.

Desy, thank you for the time we spent together, enjoying moments of relaxation and fun. Your belief in my potential to achieve my goals has been a constant source of encouragement.

To my parents, I am deeply grateful for your unwavering support, both emotionally and financially. Your belief in my capabilities and constant reassurance have been instrumental in my academic pursuits.

My brothers, Antonio Pio and Vittoria, have been pillars of love and affection throughout my academic journey. Although not directly involved in my studies, their unwavering support has been a source of strength and joy.

Nonno Vincenzo, your qualities of rationality, kindness, humor, and humility have inspired me. I strive to emulate these qualities in my own life.

Nonna Vittoria, thank you for your contributions to my upbringing and personal development. From our shared moments at the sea to your delicious cooking, you have enriched my life in countless ways.

Nonno Antonio, your love for your work, your team, and your unwavering belief in your skills have instilled in me ambition and self-esteem. I have learned the importance of acknowledging and appreciating my own abilities, knowing that they hold significant worth.

Nonna Pina, your love and support as my grandmother have been immeasurable. Your presence in my life is a constant reminder of the importance of family and unconditional love.

My lifelong friends, Giovanni, Alessandro, and Marco, have always been there for me. We have shared memorable moments of fun, laughter, and journeys together, providing unwavering support.

I cherish the memorable moments and experiences shared with Marco, Simone, and Guglielmo during my undergraduate years. Singing Neapolitan songs, going out, and enjoying each other's company have created lasting memories.

Jacopo, your presence as my academic rival has positively impacted my academic growth and competition. Together, we have pushed each other to strive for excellence.

Umberto, Giorgio, Vittoria, Lorenzo, and Riccardo, my fellow students during my master's degree, have shared significant study-related moments and joyful experiences. Our camaraderie has made the journey memorable.

To all my university colleagues, I extend my gratitude for enriching my academic experience. Your friendship and shared study sessions have created a supportive and stimulating environment.

My sincere gratitude goes to my thesis supervisor, Ilario Mazzieri. His passion and competence for numerical modeling, unwavering support, availability, and genuine kindness have been a constant source of inspiration. I am deeply grateful for his belief in my abilities and invaluable guidance during the thesis writing process. His insightful suggestions have significantly enhanced the quality of this work.

I would also like to acknowledge my co-advisor, Gabriele Ciaramella, for his exceptional availability, expertise, and valuable feedback in the field of domain decomposition methods. His constructive criticism has played a pivotal role in refining and improving the outcome of this research.

To everyone mentioned above, and to all those whose names I haven't explicitly stated, but whose contributions have been no less significant, I offer my heartfelt thanks. Your unwavering support, guidance, and companionship have made this academic journey a truly rewarding experience.