POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Efficient Techniques for Accurate Sound Propagation Modeling in Virtual Acoustics

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

Author: Gerardo Cicalese

Advisor: Prof. Ilario Mazzieri

Co-advisor: Prof. Gabriele Ciaramella

Academic year: 2022-2023

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* [5]. Throughout this work, our focus will be extensively on numerical acoustics. Ray-based methods are the most commonly used modeling techniques for their efficiency. However, 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, an efficient and accurate wave-based simulation method. ARD is used to precompute high-quality reverb filters, which are often part of the auralization pipeline in video game and simulation engines [4].

The ARD method lacks of support for air absorption, which is critical in larger spaces, such as cathedrals, where it becomes a significant factor. In our research, we will address this limitation.

2. Methodology

2.1. Simulation of the wave equation in the space-time domain

Following [1], we express the sound propagation in a fluid by the *viscous acoustic wave equation*:

$$\frac{\partial^2 p}{\partial t^2} + 2\alpha \frac{\partial p}{\partial t} - c^2 \Delta p = f, \tag{2.1}$$

where $p = p(\underline{x}, t)$ denotes the pressure at the point $\underline{x} \in \mathbb{R}^3$ at time $t \in \mathbb{R}^+$, α is the *absorption coefficient*, c is the *propagation speed*, and $f = f(\underline{x}, t)$ denotes an external force at the point $\underline{x}$ at time t. In order to derive the Fourier method, we'll consider the problem of wave propagation in a bounded domain Ω with arbitrary initial conditions and the homogeneous Neumann boundary conditions, namely:

$$\begin{cases} \frac{\partial^2 p}{\partial t^2} - c^2 \Delta p = f, & t \in \mathbb{R}^+, \ \underline{x} \in \Omega, \\ p(\underline{x}, 0) = p_0(\underline{x}), & \underline{x} \in \Omega, \\ \frac{\partial p}{\partial t}(\underline{x}, 0) = v_0(\underline{x}), & \underline{x} \in \Omega, \\ \frac{\partial p(\underline{x}, t)}{\partial \underline{n}} = 0, & t \in \mathbb{R}^+, \ \underline{x} \in \partial\Omega, \end{cases} \tag{2.2}$$

For the sake of presentation, we consider the wave propagation problem in one dimension.

We define a space-time grid with constant step size in both space (dh) and time (dt), i.e.,

$$\underline{x}_i = i\ dh, \quad t^n = n\ dt,$$

with $i = \{0, \dots, N_x - 1\}$ and $n = \{0, \dots, N_t - 1\}$ where $N_x, N_t \geqslant 2$, and we introduce the notation $p(\underline{x}_i, t^n) = p_i^n$ for simplicity.
Employing a second order accurate centered stencil to approximate the first and second time derivative, and using a sixth order accurate one to approximate the Laplacian operator, we obtain the following second order FDTD $(2, 6)$ scheme:

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

where $\underline{p}$ is the vector storing the pressure values at each space step, $\underline{f}$ is the vector storing the force values at each space step, $[K]$ is the *stiffness matrix* defined as

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

and $A = \frac{1}{90}$, $B = -\frac{3}{20}$, $C = \frac{3}{2}$, $D = -\frac{49}{18}$.
It can be proven that, employing a first order representation of Eq. (2.2), we can obtain the following first order FDTD $(1, 6)$ scheme:

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

where $\underline{v}$ is the vector storing the pressure velocity (first time derivative of the pressure) values at each space step.
The CFL condition of both FDTD schemes is

$$|\lambda| < \frac{3\sqrt{85}}{34} = 0.8135\dots, \qquad (2.3)$$

where $\lambda \triangleq c\ dt/dh$ is the *Courant number*.
These two FDTD schemes are dispersive, as shown in Fig. 2.1. The amount of numerical dispersion is reduced for low values of λ, trading off efficiency.

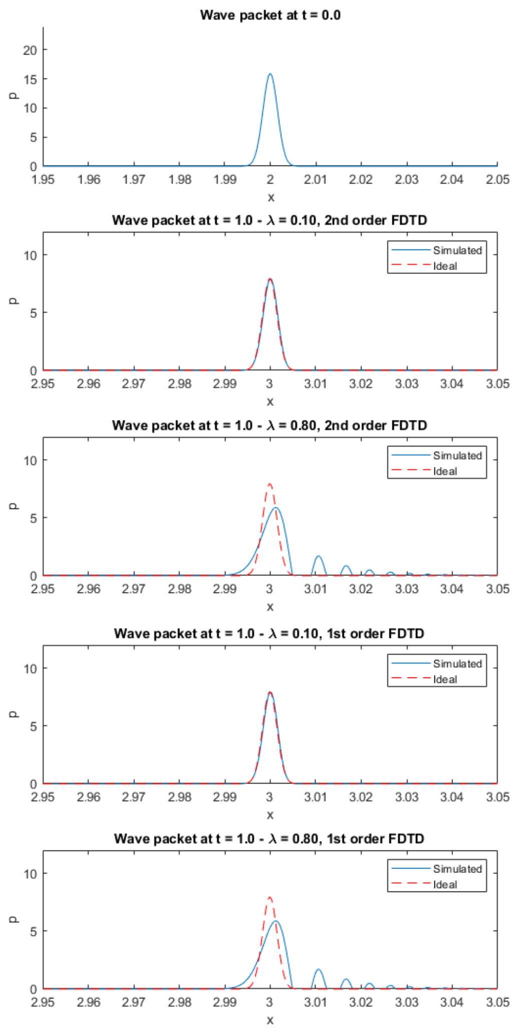


Figure 2.1: The snapshot on top depicts the wave packet at $t = 0$. The remaining snapshots depict the output of second and first order FDTD at $t = 1$ for different values of λ. The wave speed c is set to 1, $\alpha = 0$, and the spatial step size is $dh = 4e-4$. The ground truth solution is displayed using dashed lines.

2.2. Simulation of the wave equation in the Fourier-time domain

Consider the wave propagation problem (2.2) in a 3D rectangular domain Ω of dimensions L_x, L_y, L_z. Its *mode shapes* are

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

meaning that we can express $p(\underline{x}, t)$ as a *triple Cosine Series expansion* in $\underline{x}$ as follows:

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

where $P_{mnp}(t)$ are the triple Fourier cosine series coefficients or the *modal coefficients*. Treating the force analogously, we derive the viscous wave equation in the Fourier-time domain:

$$\frac{d^2 P_{mnp}}{dt^2} + 2\alpha \frac{dP_{mnp}}{dt} + \omega_{mnp}^2 P_{mnp} = F_{mnp},$$

which, for each tuple (m, n, p), with $m \in \mathbb{N}^+$, $n \in \mathbb{N}^+$, $p \in \mathbb{N}^+$, represents the equation of motion for a single degree-of-freedom system (forced damped harmonic oscillator).
We want to select a discrete transform which let us express the solution $p(\underset{\sim}{x}, t)$ in the Fourier-time domain. With the DFT, a windowed finite-length data sequence is naturally extended by periodic extension prior to transformation. 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 produces a smearing effect worse than in the case of rectangular window which has a narrower main lobe. Instead, the real valued DCT of a signal x_n corresponds to the DFT Y_k of the even extension of x_n. Furthermore, the DCT basis functions correspond to the mode shapes of the wave equation with the homogeneous Neumann boundary conditions, so the DCT spectrum of $p(\underset{\sim}{x}, t)$ decays to zero much faster than the DFT spectrum [2].

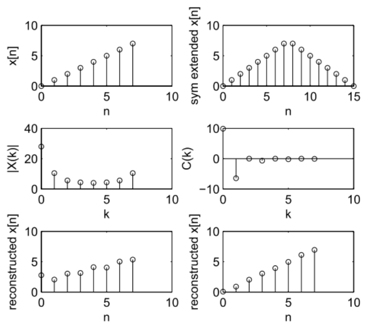


Figure 2.2: Comparison of DFT and DCT. Top: Input signal $x[n]$ and the symmetrically extended sequence $x_1[n]$. Middle: Eight-point DFT (magnitude only) and the eight-point DCT. Bottom: inverse DFT (magnitude only) and inverse DCT. Source: [2].

To represent the DCT coefficients $P_{mnp}(t)$ and $F_{mnp}(t)$, 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 $i = 1, 2, \dots, I$ with $\omega_I \geqslant \omega_{max}$, where ω_{max} is the highest radian frequency we're interested in. Hence, we obtain the wave equation in the Fourier-time (DCT-time) domain:

$$\frac{d^2 P_i}{dt^2} + 2\alpha \frac{dP_i}{dt} + \omega_i^2 P_i = F_i, \;\; i \geqslant 1, \qquad (2.4)$$

where ω_i can be expressed as

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

Eq. (2.2) and Eq. (2.4) are equivalent in the frequency band $[0, \omega_{max}]$; we can go back and forth between the two representation employing the DCT and the inverse DCT.
It has been proven in [3] that, for $\alpha = 0$, Eq. (2.4) is discretized in time by the following scheme, which represents *second order Fourier method*:

$$\begin{cases} P_1^{n+1} = 2P_1^n - P_1^{n-1} + dt^2 F_1^n, \\ P_i^{n+1} = 2P_i^n \cos(\omega_i dt) - P_i^{n-1} \\ \qquad + \dfrac{2}{\omega_i^2} F_i^n (1 - \cos(\omega_i dt)), \;\; i > 1, \end{cases}$$

for $n = 1, 2, \dots$. Instead, for $\alpha \neq 0$, we must use the following scheme to update the pressure:

$$\begin{cases} P_1^{n+1} = P_1^n + dt\, V_1^n, \\ P_i^{n+1} = P_{i,e} + e^{-\alpha dt} \\ \qquad \cdot \left[(P_i^n - P_{i,e}) \left(\cos(\omega_i dt) + \dfrac{\alpha}{\omega_i} \sin(\omega_i dt) \right) \right. \\ \qquad \left. + \dfrac{\sin(\omega_i dt)}{\omega_i} V_i^n \right], \;\; i \geqslant 1, \end{cases}$$

for $n = 1, 2, \dots$, where $P_{i,e} = F_i^n / \omega_i^2$, and V_i represent the DCT coefficients of the pressure velocity, which is updated through the following scheme:

$$\begin{cases} V_1^{n+1} = \dfrac{V_1^n + dt F_1^n}{1 + 2\alpha dt}, \\ V_i^{n+1} = e^{-\alpha dt} \left[V_i^n \left(\cos(\omega_i dt) - \dfrac{\alpha}{\omega_i} \sin(\omega_i dt) \right) \right. \\ \qquad \left. - \left(\omega_i + \dfrac{\alpha^2}{\omega_i} \right) (P_i^n - P_{i,e}) \sin(\omega_i dt) \right], \;\; i \geqslant 1, \end{cases}$$

for $n = 1, 2, \dots$, where $P_{i,e} = F_i^{n+1} / \omega_i^2$. These two schemes represent the *first order Fourier*

method.
The Fourier method, both in the second order and in the first order case, is non dispersive, as shown in Fig. 2.3 for the 1D case.

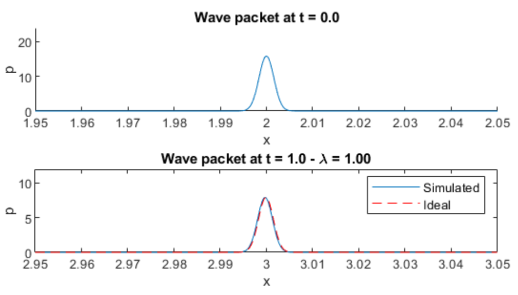


Figure 2.3: The snapshot on top depicts the wave packet at $t = 0$. The remaining snapshots depict the output of the first order Fourier method at $t = 1$ for $\lambda = 1$. The wave speed c is set to 1, $\alpha = 0$, and the spatial step size is $dh = 4e-4$. The ground truth solution is displayed using dashed lines.

2.3. Rectangular Domain Decomposition

In the majority of Domain Decomposition methodologies, the primary objective remains the distribution and parallelization of workloads across multiple processors. Our approach to domain partitioning, which we call *Rectangular Domain Decomposition* (RDD), is motivated not only by parallelization, but mostly by obtaining partitions with rectangular shapes, so that we can exploit the efficiency and lack of numerical dispersion of the Fourier method [4].

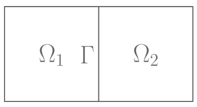


Figure 2.4: Non-overlapping partition of a 2D domain Ω into two rectangular subdomains with a shared boundary Γ.

Consider the wave propagation problem 2.2 on a domain $\Omega \subseteq \mathbb{R}^3$, omitting the viscous dissipation term for simplicity. It is well known that p_1 (corresponding to Ω_1) and p_2 (corresponding to Ω_2) must satisfy the wave equation within their respective subdomains:

$$\frac{\partial^2 p_i}{\partial t^2} - c^2 \Delta p_i = f_i, \tag{2.5}$$

where $p_i = p_i(\underline{x}, t)$ and $f_i = f_i(\underline{x}, t)$, with $t \in \mathbb{R}^+$, $\underline{x} \in \Omega_i$, $i = 1, 2$.
To understand how to merge the functions p_1 and p_2 to satisfy the global problem (2.2), we compute the difference between Eq. (2.2) and Eq. (2.5):

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

where Δ_G is the global Laplacian operator, Δ_L is the local Laplacian operator:

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

and Δ_R is the residual operator. We include the residual term in Eq. (2.5) as follows:

$$\left(\frac{\partial^2}{\partial t^2} - c^2\Delta_L\right)\begin{bmatrix} p_1 \\ p_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} + c^2\Delta_R \begin{bmatrix} p_1 \\ p_2 \end{bmatrix},$$

with $\quad t \in \mathbb{R}^+,\ \underline{x} \in \Omega_i,\ i = 1, 2.$
If we discretize the residual operator Δ_R using a second order centered stencil (analogously to the discretization of the Laplacian operator in Subsec. 2.1), we obtain the *residual matrix* $[C]$, defined as

$$[C] \triangleq \left[\begin{array}{cccc|cccc}
0 & & & & & & & 0 \\
 & \ddots & & -A & A & & & \\
 & & -A & -B & B & A & & \\
 & -A & -B & -C & C & B & A & \\
\hline
 & A & B & C & -C & -B & -A & \\
 & & A & B & -B & -A & & \\
 & & & A & -A & & \ddots & \\
0 & & & & & & & 0
\end{array}\right].$$

The residual term $c^2\Delta_R\, p$, then, becomes the *modified residual vector* $\underline{R}^n$, defined as

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

The residual term can be enforced by embedding it into the force [4], resulting in the *pre-merge* interface handling method, or by correcting the pressure p (or, in the first order case, the pressure velocity v) after updating it with FDTD or the Fourier method. In particular, we present the RDD simulation algorithm (1). The interface handling steps are conditionally stable: we must satisfy Eq. (2.3).

Algorithm 1 RDD Algorithm

```
 1: Input: Initial conditions of pressure and pressure velocity, and geometry of the room.
 2: Rectangular Decomposition: Decompose the domain into rectangular subdomains $\Omega_k$. 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., $\Omega_k.\underline{p}^n$).
 3: Interfaces Inference: Determine the interfaces $\Gamma_m$ between subdomains by examining their adjacency relationships. The interfaces data structure stores, for each interface, the reference to the two subdomains $\Omega_1$ and $\Omega_2$ associated with it, accessed through dot indexing.
 4: Initialize: Set $n = 0$.
 5: repeat
 6:     for each subdomain $\Omega_k$ in subdomains do
 7:         Update the pressure
 8:         $\Omega_k.\underline{p}^{n+1} \leftarrow$ update_pressure()
 9:     end for
10:     for each interface $\Gamma_m$ in interfaces do
11:         if interface handling method is post-merge and simulation method is second order then
12:             Correct the pressure
13:             $\Gamma_m.\Omega_1.\underline{p}^{n+1} \mathrel{+}= \frac{dt^2}{1+dt\alpha}\Gamma_m.\Omega_1.\underline{R}^n$
14:             $\Gamma_m.\Omega_2.\underline{p}^{n+1} \mathrel{+}= \frac{dt^2}{1+dt\alpha}\Gamma_m.\Omega_2.\underline{R}^n$
15:         end if
16:     end for
17:     for each $\Omega_k$ in subdomains do
18:         Compute the residual
19:         $\Omega_k.\underline{R}^{n+1} \leftarrow \left(\frac{c}{dh}\right)^2 [C]\Omega_k.\underline{p}^{n+1}$
20:     end for
21:     for each $\Gamma_m$ in interfaces do
22:         if interface handling method is pre-merge then
23:             Correct the force
24:             $\Gamma_m.\Omega_1.\underline{f}_R^{n+1} \leftarrow \Gamma_m.\Omega_1.\underline{f}^{n+1} + \Gamma_m.\Omega_1.\underline{R}^{n+1}$
25:             $\Gamma_m.\Omega_2.\underline{f}_R^{n+1} \leftarrow \Gamma_m.\Omega_2.\underline{f}^{n+1} + \Gamma_m.\Omega_2.\underline{R}^{n+1}$
26:         end if
27:     end for
28:     for each $\Omega_k$ in subdomains do
29:         Update the velocity
30:         $\Omega_k.\underline{v}^{n+1} \leftarrow$ update_pressure_velocity()
31:     end for
32:     for each $\Gamma_m$ in interfaces do
33:         if interface handling method is post-merge and simulation method is first order then
34:             Correct the velocity
35:             $\Gamma_m.\Omega_1.\underline{v}^{n+1} \mathrel{+}= \frac{dt}{1+2dt\alpha}\Gamma_m.\Omega_1.\underline{R}^{n+1}$
36:             $\Gamma_m.\Omega_2.\underline{v}^{n+1} \mathrel{+}= \frac{dt}{1+2dt\alpha}\Gamma_m.\Omega_2.\underline{R}^{n+1}$
37:         end if
38:     end for
39:     Increment $n$ by 1.
40: until termination condition is met
```

2.4. Adaptive Rectangular Decomposition

The *Adaptive Rectangular Decomposition* (ARD) is an algorithm to solve the wave propagation problem (2.2). It supports domains of arbitrary shape thanks to RDD, and it simulates both partially and fully absorbing boundary conditions. In fact, we have two types of partitions:

- *Air partitions* - Partitions where we model sound propagation in air employing the Fourier method (Subsec. 2.2).
- *PML partitions* - Partitions where we model boundary absorption. We attenuate incoming sound waves employing a Perfectly Matched Layer.

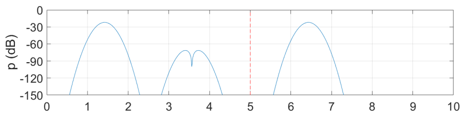
ARD consists of two primary stages:

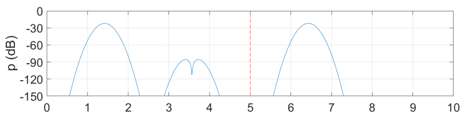
- *Preprocessing.*
  1. The input scene is voxelized into grid cells at grid resolution dh.
  2. The grid cells are grouped into rectangles corresponding to air partitions. PML partitions are generated for each boundary.
  3. The interfaces between adjacent air-air and air-PML partitions are created.
- *Simulation.* We employ the RDD algorithm 1 to perform the simulation. In air partitions we employ the Fourier method, either the first ($\alpha > 0$) or the second order one ($\alpha = 0$). For PML partitions, we can employ a FDTD scheme.

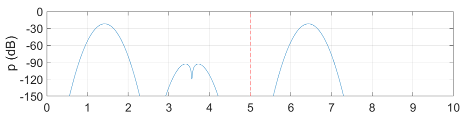
Our implementation of the ARD algorithm is based on *ARD simulator*, a project in C++ maintained by the user *jinnsjj* and available at `https://jinnsjj.github.io/ARD-simulator/`. We have made multiple notable contributions, namely, we have introduced support for air damping by employing the first-order Fourier method, and we have implemented post-merge as an interface handling method.

3. Results

The interface handling algorithm is built on the FDTD method, meaning that there's no additional numerical error if we employ this simulation method. Instead, employing the Fourier method, the coupling is not perfect, which leads to erroneous reflections at the interface, as shown in Fig. 3.1 for different orders of accuracy. The amplitude of these reflections ranges 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.
We have conducted experiments using the *ARD simulator* in a minimal hall of volume 7600 m^3, surface area of 760 m^2, and partitioned into three rectangles. We present snapshots of the


(a) Order of accuracy 2.


(b) Order of accuracy 4.


(c) Order of accuracy 6.

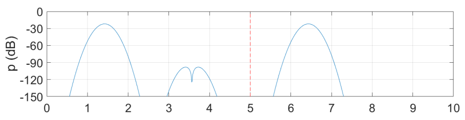
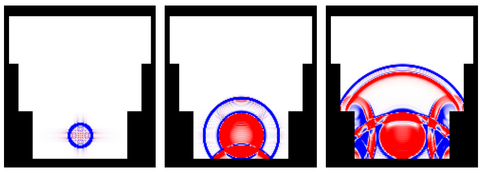

(d) Order of accuracy 8.

Figure 3.1: Snapshots of a numerical simulation in one dimension using Rectangular Domain Decomposition: the simulation domain Ω is divided into two partitions, $\Omega_1 = [0, 5]$ 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 partitions, with a grid spacing of $dh = 0.01$, a Courant number of $\lambda = 0.8$. To emphasize the errors, we show the pressure values at $t = 4$ s expressed in decibels.

simulation at different time points in Fig. 3.2. The applied force is a Gaussian pulse with a Gaussian spatial envelope.


(a) $t = 0.01$ s. (b) $t = 0.03$ s. (c) $t = 0.05$ s.

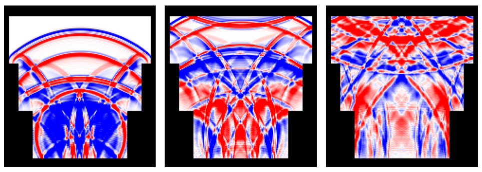

(d) $t = 0.07$ s. (e) $t = 0.09$ s. (f) $t = 0.11$ s.

Figure 3.2: Snapshots of ARD simulator at different time instants of a test case. The propagation speed c_0 is 343.5 m/s, the grid spacing dh is 0.2, and the time step dt is $2e-4$. There is no air damping, the boundaries are partially absorbing, and the interface handling method chosen is pre-merge.

4. Conclusions

In this study, we have introduced and improved the Adaptive Rectangular Decomposition algorithm. This approach surpasses traditional methods for its efficiency and lack of numerical dispersion, making it well-suited for simulating acoustics in expansive environments and capturing high-order reflections.
Future research can explore the integration of geometric techniques with ARD to enhance the efficiency of the simulations in the high-frequency range.
Lastly, there is room for improvement in the modeling of boundaries. Recent developments in frequency dependent absorbing and diffusive boundaries would offer a more complete simulation framework.

References

[1] 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.

[2] D. R. Bull and Fan Zhang. *Intelligent image and video compression: communicating pictures.* Academic Press, London, second edition edition, 2021. OCLC: 1245777824.

[3] 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.

[4] 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.

[5] 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.

Acknowledgements

I express deep gratitude to all who have contributed to my thesis for their support and presence throughout my academic journey. From family members and friends to thesis advisors and colleagues, their invaluable contributions have made this academic journey rewarding and memorable.