LIÈGE université
Sciences Appliquées

Collège de doctorat en
Électricité, électronique
et informatique

December 2022

Efficient finite element methods for solving high-frequency time-harmonic acoustic wave problems in heterogeneous media

Thesis submitted by

Anthony Royer

in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy (Ph.D.)
in Engineering Science.

Thesis committee

Prof. Christophe Geuzaine (Université de Liège), Advisor
Prof. Eric Béchet (Université de Liège), Co-advisor
Dr. Nicole Spillane (Institut Polytechnique de Paris, Ecole Polytechnique)
Dr. Axel Modave (Institut Polytechnique de Paris, ENSTA Paris)
Dr. Hadrien Bériot (Siemens Industry Software, Simulation and Test Solutions)
Prof. Koen Hillewaert (Université de Liège), President

Author contact information

Anthony Royer

ACE - Applied and Computational Electromagnetics,
Dept. of Electrical Engineering and Computer Science,
University of Liège

Montefiore Institute B28,
Quartier Polytech 1,
Allée de la Découverte 10,
4000 Liège, Belgium

Email: anthony.royer@uliege.be

Funding

This thesis was funded through the ARC grant for Concerted Research Actions (ARC WAVES 15/19-03), financed by the Wallonia-Brussels Federation of Belgium, the PRACE project funded by the EU's Horizon 2020 Research and Innovation programme (2014-2020) under grant agreement 823767, and the Win2WAL Projet EXPANSION (No. 2010161) funded by Wallonia.

Computational resources have been provided by the Consortium des Équipements de Calcul Intensif (CÉCI), funded by the Fonds de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under Grant No. 2.5020.11, as well as the Belgian share of the EuroHPC LUMI supercomputer `www.lumi-supercomputer.eu`.

Abstract

This thesis focuses on the efficient numerical solution of frequency-domain wave propagation problems using finite element methods. In the first part of the manuscript, the development of domain decomposition methods is addressed, with the aim of overcoming the limitations of state-of-the art direct and iterative solvers. To this end, a non-overlapping substructured domain decomposition method with high-order absorbing conditions used as transmission conditions (HABC DDM) is first extended to deal with cross-points, where more than two subdomains meet. The handling of cross-points is a well-known issue for non-overlapping HABC DDMs. Our methodology proposes an efficient solution for lattice-type domain partitions, where the domains meet at right angles. The method is based on the introduction of suitable relations and additional transmission variables at the cross-points, and its effectiveness is demonstrated on several test cases. A similar non-overlapping substructured DDM is then proposed with Perfectly Matched Layers instead of HABCs used as transmission conditions (PML DDM). The proposed approach naturally considers cross-points for two-dimensional checkerboard domain partitions through Lagrange multipliers used for the weak coupling between subproblems defined on rectangular subdomains and the surrounding PMLs. Two discretizations for the Lagrange multipliers and several stabilization strategies are proposed and compared. The performance of the HABC and PML DDM is then compared on test cases of increasing complexity, from two-dimensional wave scattering in homogeneous media to three-dimensional wave propagation in highly heterogeneous media. While the theoretical developments are carried out for the scalar Helmholtz equation for acoustic wave propagation, the extension to elastic wave problems is also considered, highlighting the potential for further generalizations to other physical contexts. The second part of the manuscript is devoted to the presentation of the computational tools developed during the thesis and which were used to produce all the numerical results: GmshFEM, a new `C++` finite element library based on the application programming interface of the open-source finite element mesh generator Gmsh; and GmshDDM, a distributed domain decomposition library based on GmshFEM.

Résumé

Cette thèse porte sur la résolution numérique efficace de problèmes de propagation d'ondes dans le domaine fréquentiel avec la méthode des éléments finis. Dans la première partie du manuscrit, le développement de méthodes de décomposition de domaine est abordé, dans le but de surmonter les limitations des solveurs directs et itératifs de l'état de l'art. À cette fin, une méthode de décomposition de domaine sous-structurée sans recouvrement avec des conditions absorbante d'ordre élevé utilisées comme conditions de transmission (HABC DDM) est d'abord étendue pour traiter les points de jonction, où plus de deux sous-domaines se rencontrent. Le traitement des points de jonction est un problème bien connu pour les HABC DDM sans recouvrement. La méthodologie proposée mène à une solution efficace pour les partitions en damier, où les domaines se rencontrent à angle droit. La méthode est basée sur l'introduction de variables de transmission supplémentaires aux points de jonction, et son efficacité est démontrée sur plusieurs cas-tests. Une DDM sans recouvrement similaire est ensuite proposée avec des couches parfaitement adaptées au lieu des HABC (DDM PML). L'approche proposée prend naturellement en compte les points de jonction des partitions de domaine en damier par le biais de multiplicateurs de Lagrange couplant les sous-domaines et les couches PML adjacentes. Deux discrétisations pour les multiplicateurs de Lagrange et plusieurs stratégies de stabilisation sont proposées et comparées. Les performances des DDM HABC et PML sont ensuite comparées sur des cas-tests de complexité croissante, allant de la diffraction d'ondes dans des milieux homogènes bidimensionnelles à la propagation d'ondes tridimensionnelles dans des milieux hautement hétérogènes. Alors que les développements théoriques sont effectués pour l'équation scalaire de Helmholtz pour la simulation d'ondes acoustiques, l'extension aux problèmes d'ondes élastiques est également considérée, mettant en évidence le potentiel de généralisation des méthodes développées à d'autres contextes physiques. La deuxième partie du manuscrit est consacrée à la présentation des outils de calcul développés au cours de la thèse et qui ont été utilisés pour produire tous les résultats numériques : GmshFEM, une nouvelle bibliothèque d'éléments finis `C++` basée sur le générateur de maillage open-source Gmsh ; et GmshDDM, une bibliothèque de décomposition de domaine distribuée basée sur GmshFEM.

Remerciements

Je voudrais exprimer ma gratitude envers mes promoteurs, Christophe Geuzaine et Éric Béchet pour m'avoir fait confiance et donné l'opportunité de mener ce travail avec une grande liberté, dans d'excellentes conditions, au sein des équipes de recherche ACE de l'Université de Liège. Je vous suis reconnaissant de m'avoir fait découvrir le monde de la recherche scientifique, et de m'avoir guidé par vos conseils et votre soutien. Cela a été un plaisir de travailler avec vous durant ces cinq années.

En outre, je remercie également les membres de mon comité de thèse, Christophe Geuzaine, Éric Béchet, Nicole Spillane, Axel Modave, Hadrien Bériot, et Koen Hillewaert, pour leur participation, l'intérêt porté sur mon travail et leurs conseils prodigués pour la révision du présent manuscrit. Plus particulièrement, je tiens à remercier Axel Modave et Hadrien Bériot. Axel a été un collaborateur précieux avec qui j'ai eu la chance de travailler étroitement, notamment lorsqu'il m'a accueilli dans son équipe du laboratoire POems à Paris. Hadrien m'a plus récemment aidé en me mettant en contact avec l'équipe de développement de StarCCM+ de Siemens, où je travaille maintenant.

Je tiens à souligner l'excellente atmosphère de travail qui a régné au sein du groupe ACE depuis le départ et à remercier chaleureusement tous les collègues actuels et anciens que j'ai eu l'opportunité de côtoyer, à savoir Alexandre Halbach, Ariel Lozano, Boris Martin, Christophe Geuzaine, David Colignon, Erin Kuci, Fabrice Frebel, Florent Purnode, François Henrotte, Ismaïl Badia, Jean Arban, Jonathan Velasco, Julien Dular, Kaoutar Hazim, Kevin Jacques, Matteo Cicuttin, Maxime Spirlet, Nicolas Davister, Orian Louant, Philippe Marchner, Pierre Beerten, Sophie Cimino, Valera Biangani, Vanessa Mattesi, Véronique Beauvois, Xavier Adriaens, et Yannick Paquay. Sans eux, la vie en thèse n'aurait vraiment pas été ce qu'elle fût entre les "restos du jeudi", les "vendredis post-newsletter" et les repas de service.

Je souhaite également adresser une pensée à toutes les personnes qui m'ont entouré et soutenu à un moment où un autre au cours de cette aventure. Merci à mes amis et ma famille pour tous les bons moments passés ensemble. Je remercie du fond du cœur mes parents qui ont toujours cru en moi et sur qui j'ai pu compter en toutes

circonstances. Merci pour vos encouragements, votre soutien et votre détermination sans lesquels je n'aurais certainement pas pu réaliser ce travail. J'espère être à la hauteur de vos sacrifices et vous rendre fiers. Enfin, je souhaite remercier mon frère, Dr. Amaury Royer, avec qui j'ai cohabité durant toute la durée de cette thèse.

Anthony Royer, Mars 2023.

Contents

List of Figures

List of Tables

Listings

Introduction

Large-scale wave propagation problems are common in many application areas, such as acoustics, seismology, medical imaging, ground characterization, or electromagnetic compatibility, just to name a few. In many practical applications, the complexity of the propagation media induces reflections that delay the establishment of a steady-state. In this context, time-domain modeling can require very long simulations to reach the steady-state, especially when small time steps are needed. Frequency-domain modeling is thus advantageous for such applications, and the associated computational cost often becomes unbearable, especially if the range of working frequencies is quite restricted and dominated by a few known frequencies of interest. Accurate and fast resolutions of time-harmonic wave propagation problems remain a very challenging issue in engineering though, especially in the high-frequency regime, when the wavelength is much smaller than the geometrical dimensions of the domain of study [80, 183] while leads to computationally large problems. Building accurate and fast solvers for such problems requires both adequate mathematical methods and efficient numerical implementations on modern computer architectures.

The present thesis is focused on the numerical solution of the most fundamental equation encountered in frequency-domain wave propagation problems: the Helmholtz equation. The Helmholtz equation is a second-order elliptic partial differential equation encountered in many domains of physics like acoustics, elasticity, fluid dynamics, or electromagnetics. Indeed, in certain situations, the more complex time-harmonic Maxwell equations for electromagnetics or the time-harmonic Navier equation for elastodynamics can be reduced to scalar or vector Helmholtz equations, which are thus at the heart of linear time-harmonic wave propagation research.

Helmholtz equation and finite elements

Among the various approaches proposed to solve large-scale time-harmonic wave problems, the Finite Difference Method (FDM), based on the discretization of the strong

form of the Helmholtz equation on a regular Cartesian grid, is not well-suited for geometries with complex boundaries. On the other hand, the Boundary Element Method (BEM), based on the discretization of a boundary integral equation [156, 49, 126] is a popular method to solve problems with complex boundaries but is limited to piecewise constant material properties. The Finite Element Method (FEM) is then a natural choice for its ability to handle both complex geometrical configurations and materials with heterogeneous properties.

The brute-force application of the FEM however faces three significant difficulties [80]. First, to establish a numerical finite element model, the unbounded domain must be truncated to become a bounded computational domain, using for example a fictitious boundary with an Absorbing Boundary Condition (ABC) [192, 24, 75, 12, 121] or a fictitious boundary layer around the computational domain with a Perfectly Matched Layer (PML) [29, 186, 35]. The definition of these accurate boundary operators is crucial to minimize nonphysical wave reflections on the fictitious exterior boundaries. Second, the pollution effect forces to use high-order interpolations, especially in the high-frequency regime [114, 115]. In addition to the large number of unknowns needed to capture the highly oscillatory nature of the fields, high-order discretizations produce a sizeable finite element matrix with poorer sparsity than with low-order interpolations, which is an issue for the scalability of direct linear solvers. Finally, the indefinite nature of the Helmholtz operator leads to a poorly-conditioned complex-valued linear system [149, 11, 77, 80] for which Krylov subspace iterative solvers exhibit slow convergence or can even diverge, while efficiently preconditioning proves very difficult [80].

Domain decomposition methods

To tackle the aforementioned difficulties, Domain Decomposition Methods (DDMs) provide a promising alternative. These methods rely on a partition of the computational domain into subdomains and on an iterative procedure linking the resulting subproblems, of smaller sizes, amenable to direct solvers (see *e.g.* [185]). Among all DDMs, we can highlight Schwarz methods with overlap [48, 124, 87] or without overlap [27, 58, 85], FETI algorithms [65, 83, 83, 82] and the method of polarized traces [190, 191], which are eventually combined with preconditioning techniques (see *e.g.* [61, 89, 96, 177, 178, 188, 63]).

The underlying mathematical concepts for overlapping DDMs were first proposed by Schwarz [174] in 1870, and gained a lot of interest for their inherent parallelism in the 1980s, when coupled with the finite element method. In the 1990s, Lions proposed non-overlapping DDMs for the Laplace equation [131], and Després subsequently proved the convergence of non-overlapping DDMs with Robin-type transmission conditions for the Helmholtz equation in 1991 [66]. Significant improvements have been made since these early works, especially regarding transmission conditions, giving birth to the class of methods referred to as optimized Schwarz methods [85]. For Helmholtz problems, a recent overview of DDMs can be found in [88, 158, 159].

In this thesis, we focus on Schwarz-type domain decomposition algorithms without overlap, which have shown to be robust and efficient for high-frequency acoustic scattering problems [45], provided that accurate high-order transmission conditions are used on the interfaces between subdomains. Treating the cases where more than

two subdomains meet (leading to so-called "cross-points") is still an open problem when high-order transmission conditions are used, tough, and is a very active research area [157, 67, 84, 54, 55, 158]. This thesis aims at developing a fast and efficient DDM that tackles this problem, by combining high-order finite elements and high-order transmission conditions with a dedicated, scalable cross-point treatment strategy.

Outline of the thesis and scientific contributions

This manuscript is organized as follows. This introduction is followed in Chapter 1 by an exposition of the state of the art of domain decomposition methods applied to finite element discretizations of the Helmholtz equation. The first part of the thesis (Chapters 2, 3 and 4) is then devoted to developing cross-point treatments for high-order transmission conditions for checkerboard partitions. Chapters 2 and 3 present two numerical strategies while Chapter 4 compares their performance. The last part (Chapters 5 and 6) presents the open-source FEM and DDM solvers (GmshFEM and GmshDDM) developed during the thesis, and that were used to produce all the numerical results.

The thesis has led to the publication of the following articles:

- Modave, Axel, Royer, Anthony, Antoine, Xavier and Geuzaine, Christophe. A non-overlapping domain decomposition method with high-order transmission conditions and cross-point treatment for Helmholtz problems. *Computer Methods in Applied Mechanics and Engineering*, 368:113162, 2020 [148]. This article constitutes the basis for Chapter 2.

- Anthony Royer, Eric Béchet, and Christophe Geuzaine. GmshFEM: An Efficient Finite Element Library Based On Gmsh. In *14th World Congress on Computational Mechanics (WCCM), ECCOMAS Congress 2020*, 2021 [167]. This article constitutes the basis for Chapter 5.

- Royer, Anthony, Geuzaine, Christophe, Béchet, Eric and Modave, Axel. A non-overlapping domain decomposition method with perfectly matched layer transmission conditions for the Helmholtz equation. *Computer Methods in Applied Mechanics and Engineering*, 395:115006, 2022 [168]. This article constitutes the basis for Chapter 3.

It has also led to the following presentations at conferences or workshops:

- Optimized Multithreaded Assembly Using Hilbert Curves: In the Context of Domain Decomposition Methods Applied to Time-Harmonic Elastodynamic Problems. *13th World Congress on Computational Mechanics (WCCM) and 2nd PANACM Congress 2018.*

- An optimized non-overlapping Schwarz method with cross-point treatment for high-frequency acoustic scattering problems. *Groupe de Travail des Thésard·e·s du LJLL (GTT)*, 2019.

- A non-overlapping DDM with PML transmission conditions for Helmholtz problems. *Rencontre jeunes chercheuses et jeunes chercheurs (JCJC 2020)*, 2020 (virtual workshop).

- A non-overlapping DDM with PML transmission conditions for Helmholtz problems. *26th International Domain Decomposition Conference, DD XXVI*, 2020 (virtual conference).

- GmshFEM: An Efficient Finite Element Library Based on Gmsh. *14th World Congress on Computational Mechanics (WCCM), ECCOMAS Congress 2020*, 2021 (virtual conference).

- GetDP Workshop: GmshFEM - future direction, High Performance Computing. *GetDP Workshop, CERN*, 2021 (virtual workshop).

- Comparison of optimized high-order domain decomposition methods with cross-point treatment for Helmholtz problems. *ICOSAHOM 2020*, 2021 (virtual conference).

- GmshFEM & GmshDDM: Efficient finite element and domain decomposition libraries based on Gmsh, *Rencontre jeunes chercheuses et jeunes chercheurs (JCJC 2021)*, 2021.

Below is a list of the main original contributions of the thesis:

- The development, implementation and study of several variants of PML-based non-overlapping DDMs with cross-points, as published in [168] and presented in Chapter 3.

- The implementation and study of the HABC-based DDM proposed in [148] with high-order finite elements, in a unified setting including both HABC- and PML-based DDMs, allowing the comparison of the methods and the generation of the numerical results presented in Chapter 2.

- The performance comparison of HABC- and PML-based DDMs on both homogeneous and heterogeneous acoustic problems with checkerboard partitions, as well as the extension of PML-based DDMs to elastodynamics, as presented in Chapter 4.

- The development of the open source GmshFEM and GmshDDM libraries:

  - GmshFEM (`https://gitlab.onelab.info/gmsh/fem.git`) is a `C++` finite element library (about 40k lines of code) based on the application programming interface of Gmsh, presented in [167] and in Chapter 5;

  - GmshDDM (`https://gitlab.onelab.info/gmsh/ddm.git`) is a small `C++` domain decomposition library (about 2k lines of code) based on GmshFEM, presented in Chapter 6.

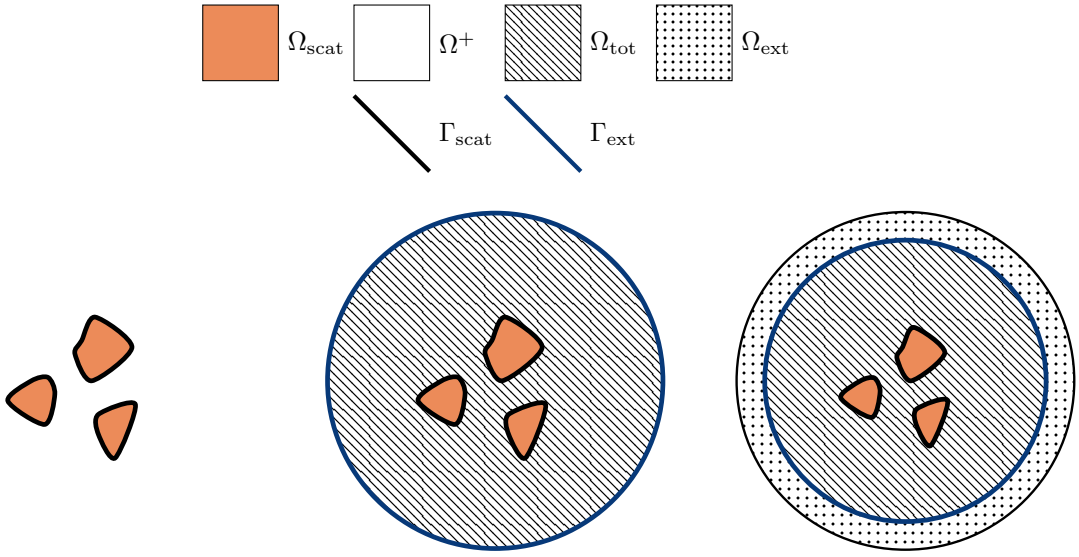
The numerical results presented in the manuscript were obtained with GmshFEM version 1.0.0 (`https://gitlab.onelab.info/gmsh/fem/-/releases/gmshfem_1_0_0`) and GmshDDM version 1.0.0 (`https://gitlab.onelab.info/gmsh/ddm/-/releases/gmshddm_1_0_0`), linked to Gmsh version 4.11.1 (`https://gitlab.onelab.info/gmsh/gmsh/-/releases/gmsh_4_11_1`). Besides this thesis, the GmshFEM and GmshDDM libraries have been extensively used in three other PhD theses [16, 134, 1], as well as in several journal articles [168, 135, 2, 138, 4, 17, 3, 53].

1 State of the Art

This chapter is dedicated to presenting the state of the art for solving the Helmholtz equation with finite element methods. The first section presents the Helmholtz problem considered in this thesis. Then in the second section, a numerical finite element model for the Helmholtz problem is introduced. The following three sections present the three main difficulties linked to finite element modeling of Helmholtz problems, as mentioned in the Introduction, namely the definition of accurate boundary conditions, the pollution effect, and the indefinite nature of the Helmholtz operator. Finally, the last section is devoted to an overview of non-overlapping Domain Decomposition Methods (DDMs) for Helmholtz problems.

1 The Helmholtz problem

To introduce the Helmholtz problem of interest, let us consider an arbitrary set of open subsets Ω_{scat} of $\mathbb{R}^d$ where d is the dimension, with Γ_{scat} denoting the union of their boundary, such that the complementary of the union of their closed subset, Ω^+ is connected (see Figure 1.1a). The boundary Γ_{scat} is divided into two subsets $\Gamma^{\text{D}}_{\text{scat}}$ and $\Gamma^{\text{N}}_{\text{scat}}$ such that $\Gamma_{\text{scat}} = \bar{\Gamma}^{\text{D}}_{\text{scat}} \cup \bar{\Gamma}^{\text{N}}_{\text{scat}}$. A Dirichlet boundary condition is imposed on $\Gamma^{\text{D}}_{\text{scat}}$, while a Neumann boundary condition is imposed on $\Gamma^{\text{N}}_{\text{scat}}$. The open domain Ω^+ is made of non-dissipative media assumed to be at rest and characterized by a phase velocity c, assumed to be strictly positive. Under these assumptions, the behavior of a time-harmonic complex-valued scalar wave field u is described by the following



(a) The unbounded domain.

(b) The truncated domain with absorbing boundary.

(c) The truncated domain with absorbing boundary layer.

Figure 1.1: Open and truncated computational domains.

Helmholtz problem [113, 133, 166]:

$$\begin{cases} \Delta u + k^2 u = -f & \text{in } \Omega^+, \\ u = u_{\text{D}} & \text{on } \Gamma^{\text{D}}_{\text{scat}}, \\ \partial_{\mathbf{n}} u = u_{\text{N}} & \text{on } \Gamma^{\text{N}}_{\text{scat}}, \\ \lim\limits_{\|\mathbf{x}\| \to \infty} \|\mathbf{x}\|^{\frac{d-1}{2}} \left(\partial_{\|\mathbf{x}\|} - \imath\, k\right) u = 0, \end{cases} \tag{1.1}$$

where $k = \omega/c$ is the strictly positive wavenumber with ω the angular frequency, f is a source term and $\mathbf{n}$ is the outgoing normal to Ω_{scat}, and where $\partial_{\mathbf{n}}$ denotes the normal derivative.

A *Sommerfeld radiation condition* [192] is enforced by the fourth equation of (1.1) so that the system has a unique solution by selecting only the outgoing waves (radiating solution); it enforces a null incoming energy flux at infinity. Indeed, solutions of (1.1) can be sorted into three types of waves: an ingoing wave, an outgoing wave, and standing waves that are built by the superposition of the two previous ones. Further, the linearity of the Helmholtz equation implies that every linear combination of these waves is also a solution. Therefore, selecting only the purely outgoing solution makes (1.1) well-posed.

This Helmholtz equation models the steady-state solutions $\tilde{u}(\mathbf{x}, t) = \mathfrak{Re}[u(\mathbf{x})e^{-\imath\,\omega t}]$, with t the time, of the following time-domain acoustic wave equation [113]:

$$\Delta\, \tilde{u}(\mathbf{x}, t) - \frac{1}{c^2} \frac{\partial^2 \tilde{u}(\mathbf{x}, t)}{\partial t^2} = 0. \tag{1.2}$$

2 The truncated Helmholtz problem

To establish a numerical finite element model, the unbounded domain Ω^+ must be truncated to become a bounded computational domain Ω_{tot}, assumed to be Lipschitz continuous. The truncation can be performed by introducing a fictitious boundary Γ_{ext} with an Absorbing Boundary Condition (ABC) [192, 24, 75, 12, 121] (see Figure 1.1b) or a fictitious boundary layer Ω_{ext} around the computational domain where $\Gamma_{\text{ext}} = \bar{\Omega}_{\text{tot}} \cap \bar{\Omega}_{\text{ext}}$ with a Perfectly Matched Layer (PML) [29, 186, 35] (see Figure 1.1c). Both methods are based on an approximation of the exact Dirichlet-to-Neumann map (DtN), Λ, related to the complementary of Ω_{tot} and defined by [122, 95]:

$$\begin{aligned} \Lambda : H^{1/2}(\Gamma_{\text{ext}}) &\to H^{-1/2}(\Gamma_{\text{ext}}), \\ u|_{\Gamma_{\text{ext}}} &\mapsto (\partial_{\mathbf{n}} u)|_{\Gamma_{\text{ext}}} = \Lambda(u|_{\Gamma_{\text{ext}}}), \end{aligned} \tag{1.3}$$

where $H^{1/2}(\cdot)$ and $H^{-1/2}(\cdot)$ are the Sobolev spaces of the Dirichlet and Neumann traces, respectively [78, 37]. In a nutshell, this operator links the Dirichlet trace on the fictitious boundary Γ_{ext} to the Neumann trace that would correspond if the domain was not truncated. System (1.1) is reduced to the bounded approximation system

$$\begin{cases} \Delta u + k^2 u = -f & \text{in } \Omega_{\text{tot}}, \\ u = u_{\text{D}} & \text{on } \Gamma^{\text{D}}_{\text{scat}}, \\ \partial_{\mathbf{n}} u = u_{\text{N}} & \text{on } \Gamma^{\text{N}}_{\text{scat}}, \\ \partial_{\mathbf{n}} u - \mathcal{B} u = 0 & \text{on } \Gamma_{\text{ext}}, \end{cases} \tag{1.4}$$

where $\mathcal{B}$ is a boundary operator that approximates of the DtN operator. The definition of $\mathcal{B}$ depends on the ABC or PML enforced on the fictitious boundary.

In the following chapters of this thesis, three acoustic problems will be investigated: the scattering of an incident plane wave by a soft object or by a hard object and the propagation of point source excitations. The scattering problems model the scattered field resulting from the scattering of an incident plane wave $u_{\text{inc}} = e^{\imath \mathbf{k}\cdot\mathbf{x}}$ with $k = \|\mathbf{k}\|$ by the objects Ω_{scat}. The total acoustic field is defined by the sum of the computed scattered field and the incident plane wave. These scattered fields are modeled by System (1.4) without any source term, *i.e.* $f = 0$. For the soft scattering problem, a Dirichlet condition $u_{\text{D}} = -u_{\text{inc}}$ is enforced on the whole object boundary, *i.e.* $\Gamma^{\text{N}}_{\text{scat}}$ is empty, while for the hard scattering problem, a Neumann condition $u_{\text{N}} = -\partial_{\mathbf{n}} u_{\text{inc}}$ is enforced on the whole object boundary, *i.e.* $\Gamma^{\text{D}}_{\text{scat}}$ is empty. The propagation of point source excitations is also modeled by Equation (1.4) by assuming source terms defined by the sum of Dirac impulses on every source s located at $\mathbf{x}_s$ inside Ω_{tot}, *i.e.* $f(\mathbf{x}) = \sum_s \delta(\mathbf{x}_s - \mathbf{x})$.

3 Accurate boundary conditions

Historically, low-order Absorbing Boundary Conditions (ABCs) [192, 24] are amongst the first methods that were used to deal with unbounded problems. They are the least expensive in terms of computing cost and memory usage but are also the least accurate, especially in high-frequency regimes. On the opposite, non-local non-reflecting

boundary conditions [94, 43] are the most accurate but with significant additional computational cost. In between, local High-order Absorbing Boundary Conditions (HABCs) [75, 12, 121], where the accuracy and the cost can be controlled by choosing the order of the HABC, provide a good compromise.

The second family, the layer techniques, are another common way to truncate an unbounded problem. Where ABCs truncate the domain by enforcing an equation at the boundary to approximate the DtN, layer techniques add a layer around the computational domain where fields are treated to be dissipated over the thickness of the layer. This approach can be compared with techniques employed to insulate a room acoustically or construct an anechoic chamber where dissipating materials are applied on the walls. Perfectly matched layers (PMLs) were proposed in 1994 by Bérenger [29, 186, 35]. One great advantage of this family of layers is that they are perfectly matched, which means that every outgoing mode is perfectly transmitted from the computational domain to the layer. In other words, there is no reflection on the interface between the computational domain and the layer. Initially introduced for 2D temporal electromagnetic problems [29], they were quickly extended to 3D electromagnetic problems [31], then to acoustic problems [112] as well, elastodynamics [107]. The design of PMLs for time-harmonic problems is presented in numerous references [35, 33, 36, 22, 20, 106, 50].

3.1 The exact half-space Dirichlet-to-Neumann map

Non-reflective high-order boundary conditions are obtained by approximating the exact DtN as boundary operator. However, an analytic expression of the DtN can only be found in simple configurations. For instance let us consider the Problem (1.1) with an interior region $\Omega_{\text{int}} := \{(x, y, z) \in \mathbb{R}^3, x < 0\}$ and a free-of-source and homogeneous exterior region $\Omega_{\text{ext}} := \{(x, y, z) \in \mathbb{R}^3, x > 0\}$ such that the unbounded domain Ω^+ is made by the union of the interior and exterior region. The interface between the interior of exterior region defines the fictitious boundary $\Gamma_{\text{ext}} := \{(x, y, z) \in \mathbb{R}^3, x = 0\}$. To seek the exact DtN related to the exterior domain, let us solve the exterior Helmholtz problem with a Dirichlet source imposed on the fictitious boundary $u|_{\Gamma_{\text{ext}}} = u_{\text{D}}(y, z)$. In order to deal with the unbounded nature of Ω_{ext}, let us define the following multidimensional Fourier transform in the tangential direction $\mathbf{t} : (y, z)$

$$\mathcal{F}_{yz}[f] = \int_{\mathbb{R}^2} f(x, y, z) e^{-\iota \mathbf{t}\cdot\boldsymbol{\xi}} \, \mathrm{d}\mathbf{t}, \tag{1.5}$$

and the inverse Fourier transform by

$$\mathcal{F}_{yz}^{-1}[\hat{f}] = \frac{1}{2\pi} \int_{\mathbb{R}^2} \hat{f}(x, \xi_y, \xi_z) e^{\iota \mathbf{t}\cdot\boldsymbol{\xi}} \, \mathrm{d}\boldsymbol{\xi}, \tag{1.6}$$

where $\boldsymbol{\xi} : (\xi_y, \xi_z)$ is the dual variable of $\mathbf{t}$ in the Fourier space. Let us apply this Fourier transform $\mathcal{F}_{yz}$ to the Helmholtz equation to obtain

$$\left(\partial_x^2 + k^2 - \xi_y^2 - \xi_z^2\right) \mathcal{F}_{yz}[u](x, \xi_y, \xi_z) = 0, \qquad \text{for } x > 0, \xi_y \in \mathbb{R} \text{ and } \xi_z \in \mathbb{R}. \tag{1.7}$$

The homogeneous solution of this ordinary differential equation for $\mathcal{F}_{yz}[u](x, \xi_y, \xi_z)$ is

$$\mathcal{F}_{yz}[u](x, \xi_y, \xi_z) = A e^{x\sigma_+(\xi_y,\xi_z)} + B e^{x\sigma_-(\xi_y,\xi_z)}, \tag{1.8}$$

where σ_+ and σ_- are defined as

$$\sigma_{+,-} = \pm\,\iota\,\sqrt{k^2 - \xi_y^2 - \xi_z^2} = \pm\,\iota\,k^2\sqrt{1 - \frac{\xi_y^2 + \xi_z^2}{k^2}}, \tag{1.9}$$

and A and B are constants to be determinated through boundary conditions. Namely, A is fixed to $\mathcal{F}_{yz}[u_{\mathrm{D}}](\xi_y, \xi_z)$ thanks the Dirichlet condition on Γ_{ext} and B is fixed to zero because only outgoing traveling waves are considered. In the context of this half-space problem, the Neumann trace is retrieved by the derivative of (1.8) given by

$$\partial_x \mathcal{F}_{yz}[u](x, \xi_y, \xi_z) = \sigma_+(\xi_y, \xi_z)\mathcal{F}_{yz}[u](x, \xi_y, \xi_z). \tag{1.10}$$

Applying the inverse Fourier transform $\mathcal{F}_{yz}^{-1}$ leads to

$$\partial_x u(x, u, z) = \mathcal{F}_{yz}^{-1}\left[\sigma_+(\xi_y, \xi_z)\mathcal{F}_{yz}[u](x, \xi_y, \xi_z)\right](x, y, z). \tag{1.11}$$

The exact DtN $\Lambda^{\text{half-space}}$ is obtained by taking the restriction on Γ_{ext} [94, 181]:

$$\partial_x u|_{\Gamma_{\mathrm{ext}}} = \mathrm{Op}\left(\iota\,k^2\sqrt{1 - \frac{\xi_y^2 + \xi_z^2}{k^2}}\right) u|_{\Gamma_{\mathrm{ext}}} = \iota\,k^2\sqrt{1 + \frac{\Delta_{\Gamma_{\mathrm{ext}}}}{k^2}}u|_{\Gamma_{\mathrm{ext}}} := \Lambda^{\text{half-space}}u|_{\Gamma_{\mathrm{ext}}}, \tag{1.12}$$

where $\Delta_{\Gamma_{\mathrm{ext}}} = \partial_y^2 + \partial_z^2$ is the Laplace-Beltrami operator over Γ_{ext}.

Note that, if one considers a smooth surface Γ_{ext} instead of the half-space domain studied above, the half-space DtN is not exact, and two methods can be used to derive an approximated boundary operator $\mathcal{B}$ of the DtN. A formal one approximates the exact DtN (1.12) on the tangent plane, and a rigorous one applies the technique of microlocal diagonalization for hyperbolic systems. However, these approaches are not accurate for gazing modes ($k \approx \|\boldsymbol{\xi}\|$), namely modes in the transition zone from the propagating modes ($k > \|\boldsymbol{\xi}\|$) to the evanescent one ($k < \|\boldsymbol{\xi}\|$), due to the singularity of the symbol at $k = \|\boldsymbol{\xi}\|$. To model the behavior in the transition zone, one can introduce a regularization by adding a small damping parameter ϵ to the wavenumber $k_\epsilon = k + \iota\,\epsilon$, leading to the regularized non-local operator [12, 45]

$$\Lambda_\epsilon^{\text{half-space}} := \iota\,k_\epsilon\sqrt{1 + \frac{\Delta_{\Gamma_{\mathrm{ext}}}}{k_\epsilon^2}}. \tag{1.13}$$

3.2 High-order absorbing boundary conditions

The exact half-space DtN 1.12 is a non-local operator. In order to construct a local approximation, one can consider zeroth or first order Taylor approximation of the function $f(X) = \sqrt{1+X} = 1 + \frac{1}{2}X + \mathcal{O}(X^2)$, which lead to following low-order ABCs [192, 24]:

$$\mathcal{B}_{o0} = \iota\,k \tag{1.14}$$

$$\mathcal{B}_{o2} = -\frac{1}{2\,\iota\,k}\,\Delta_{\Gamma_{\mathrm{ext}}} + \iota\,k. \tag{1.15}$$

Many previous works have been devoted to extensions of these low-order ABCs, first to the case of a circular boundary [75, 25, 145, 150], and eventually for more general surfaces [125, 119, 10] by considering additional terms that depend on the local curvature of the boundary.

Instead of the Taylor approximation, a Padé approximation was proposed by Engquist and Majda [75] to obtain a local representation of the function $f(X) = (1+X)^{1/2}$. The classical $(2N_{\text{HABC}}+1)$th-order Padé approximation of the function f is written as the rational series [19, 98]

$$f^{\text{Padé}}_{2N_{\text{HABC}}+1}(X) = 1 + \frac{2}{M} \sum_{i=1}^{N_{\text{HABC}}} \frac{a_i X}{1 + b_i X}, \tag{1.16}$$

or alternatively,

$$f^{\text{Padé}}_{2N_{\text{HABC}}+1}(X) = 1 + \frac{2}{M} \sum_{i=1}^{N_{\text{HABC}}} c_i \frac{c_i + 1}{c_i + 1 + X}, \tag{1.17}$$

with $M = 2N_{\text{HABC}} + 1$, $a_i = \sin^2(i\pi/M)$, $b_i = \cos^2(i\pi/M)$ and $c_i = \tan^2(i\pi/M)$. Unfortunately, a direct implementation of the Padé approximation leads to a boundary operator that is inaccurate for evanescence modes [143]. Milinazzo et al. [143] proposed to rotate the branch cut of the square root by some angle ϕ to obtain the high-order absorbing boundary operator:

$$\mathcal{B}^{\text{Padé}}_{\text{HABC}} = \iota\, k e^{\iota\,\phi/2} \left[1 + \frac{2}{M} \sum_{i=1}^{N_{\text{HABC}}} c_i \frac{e^{\iota\,\phi}(c_i + 1)}{e^{\iota\,\phi}(c_i + 1) + \Delta_{\Gamma_{\text{ext}}}/k^2} \right]. \tag{1.18}$$

Generally speaking, the approximation of the square root function $f(X)$ as a sum of prime fractions (1.17) can also be rewritten as a rational fraction of X defined by the following recursive relation [19]:

$$\begin{aligned} f^{\text{Padé}}_m(X) &= 1 + \frac{X}{1 + f^{\text{Padé}}_{m-1}(X)} \quad \text{for } m = 2 \dots M, \\ f^{\text{Padé}}_1(X) &= 1. \end{aligned} \tag{1.19}$$

As a result, the HABC operator (1.18) written as a sum of prime fraction is a particular case of more general boundary conditions, called Complete Radiation Boundary Conditions (CRBCs), that approximate the square root function $f(X)$ with the following rational fraction of X:

$$\begin{aligned} f_m(X) &= \alpha_m + \frac{1 - \alpha_m^2 + X}{\alpha_m + f_{m-1}(X)} \quad \text{for } m = 2 \dots M, \\ f_1(X) &= \alpha_1, \end{aligned} \tag{1.20}$$

where $\{\alpha_m\}_{m=1\dots M}$ are complex coefficients. The equivalence between the CRBC representation (1.20) and the Padé representation as rational fraction (1.19) is obtained when $\alpha_m = 1$ for all m. Compared to the Padé approximation, CRBCs can deal with both propagative and evanescent modes but require M parameter optimizations, *i.e.* all α_m, while the Padé approximation has only the branch cut rotation parameter, *i.e.* the angle ϕ, to tune. Such approximations of the square root function with continued fractions are called continuous fraction ABCs (CFABCs).

In practice, the high-order boundary operator (1.18) is written as a system of wave-like equations that define N_{HABC} auxiliary fields φ_i with $i = 1 \dots N_{\text{HABC}}$ defined on the boundary Γ_{ext} [56, 57, 130]:

$$\Delta_{\Gamma_{\text{ext}}} \varphi_i + k^2 \left[(e^{\iota\,\phi} c_i + 1)\varphi_i + e^{\iota\,\phi}(c_i + 1)u \right] = 0, \quad \text{on } \Gamma_{\text{ext}}, \tag{1.21}$$

that are coupled with the Dirichlet trace of u on the boundary to define

$$\mathcal{B}_{\text{HABC}}^{\text{Padé}} u = \iota\, k e^{\iota\,\phi/2} \left[u + \frac{2}{M} \sum_{i=1}^{N_{\text{HABC}}} c_i (u + \varphi_i) \right] \quad \text{on } \Gamma_{\text{ext}}. \tag{1.22}$$

While Equation (1.22) is nothing more than a more sophisticated boundary condition than (1.14) and (1.15), Equation (1.22) is a wave-like equation applied to the boundary. For this reason, when HABC are applied to a polyhedron domain with corners, special boundary conditions must be used to act as a boundary condition for these auxiliary fields [147].

In Chapter 2, HABCs will be used as transmission condition for acoustic DDM, and a strategy to deal with cross-points in the presence of HABCs will be presented. Furthermore, numerical applications that implement HABCs with the proposed cross-point treatment will be studied in Chapter 4.

3.3 Perfectly matched layers

3.3.1 The Bérenger PML model

To introduce the Bérenger PML model, let us start with a homogeneous time-domain scalar wave equation defined on a square domain $\Omega_{\text{tot}} : \{\mathbf{x} : \mathbf{x} \in]-1, 1[^3\}$

$$\frac{1}{c^2} \frac{\partial^2 \tilde{p}}{\partial t^2} - \Delta\, \tilde{p} = 0, \quad \text{in } \Omega_{\text{tot}}, \tag{1.23}$$

where $\tilde{p}$ is a real unknown field and c is the phase velocity. This equation can be decomposed into the following system of first-order equations by defining a vector field $\tilde{\mathbf{v}}$

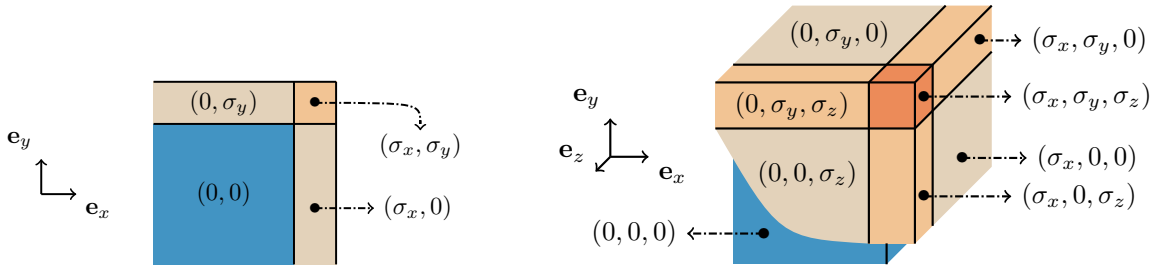
$$\begin{cases} \dfrac{\partial \tilde{p}}{\partial t} + \rho_0 c^2\, \operatorname{div} \tilde{\mathbf{v}} = 0 & \text{in } \Omega_{\text{tot}}, \\ \dfrac{\partial \tilde{\mathbf{v}}}{\partial t} + \dfrac{1}{\rho_0}\, \mathbf{grad}\, \tilde{p} = 0 & \text{in } \Omega_{\text{tot}}, \end{cases} \tag{1.24}$$

where ρ_0 is a positive constant. In the case of acoustic waves, $\tilde{p}$ is the acoustic pressure, *i.e.* the local variation of the ambient pressure, $\tilde{\mathbf{v}}$ is the local particle velocity, and ρ_0 is the ambient density, *i.e.* the density at rest when there is no wave perturbing it. Knowing that $\tilde{p} = c^2 \rho$ with ρ the local variation of the ambient density, the first equation of (1.24) is a linearization of the continuity condition while the second equation is the linearized Euler equation. Assuming a time dependency of $e^{-\iota\,\omega t}$ for fields $\tilde{p}$ and $\tilde{\mathbf{v}}$, one can deduce from (1.24) the following time-harmonic wave system:

$$\begin{cases} -\iota\, k p + \rho_0 c\, \operatorname{div} \mathbf{v} = 0 & \text{in } \Omega_L, \\ -\iota\, k \mathbf{v} + \dfrac{1}{\rho_0 c}\, \mathbf{grad}\, p = 0 & \text{in } \Omega_L, \end{cases} \tag{1.25}$$

where p and $\mathbf{v}$ are defined through the relation $\tilde{p}(\mathbf{x}, t) = \mathfrak{Re}[p(\mathbf{x}) e^{-\iota\,\omega t}]$ and $\tilde{\mathbf{v}}(\mathbf{x}, t) = \mathfrak{Re}[\mathbf{v}(\mathbf{x}) e^{-\iota\,\omega t}]$.

The idea behind the Bérenger PML is to extend the domain Ω_{tot} by a dissipating a layer of thickness δ, building the domain $\Omega_{\text{all}} : \mathbf{x} \in [-1-\delta, 1+\delta]^3\}$ such that the



(a) A two-dimensional PML configuration in a Cartesian coordinate system.

(b) A three-dimensional PML configuration in a Cartesian coordinate system.

Figure 1.2: Absorption functions $\boldsymbol{\sigma}$ in a Cartesian coordinate system. The blue region is the computational domain, while the orange regions are the PML domain.

PML region Ω_{PML} is defined by $\bar{\Omega}_{\text{all}} = \bar{\Omega}_{\text{tot}} \cup \bar{\Omega}_{\text{PML}}$. Then (1.25) is written over Ω_{all}, split over the three spatial coordinates and a damping term is added leading to

$$\begin{cases} -\iota\, k p_x + \rho_0 c \partial_x v_x = -\sigma_x p_x, & -\iota\, k v_x + \dfrac{1}{\rho_0 c}\partial_x p = -\sigma_x v_x, \\ -\iota\, k p_y + \rho_0 c \partial_y v_y = -\sigma_y p_y, & -\iota\, k v_y + \dfrac{1}{\rho_0 c}\partial_y p = -\sigma_y v_y, \\ -\iota\, k p_z + \rho_0 c \partial_z v_z = -\sigma_z p_z, & -\iota\, k v_z + \dfrac{1}{\rho_0 c}\partial_z p = -\sigma_z v_z, \end{cases} \tag{1.26}$$

where $\mathbf{v} : (v_x, v_y, v_z)$, p_x, p_y and p_z are formal variables such that $p = p_x + p_y + p_z$ and $\sigma_x(x)$, $\sigma_y(y)$ and $\sigma_z(z)$ are the *absorption functions*. These absorption functions are non-zero only in the PML regions that are extruded in the corresponding direction (see Figure 1.2).

Taking the spatial derivative of the Euler equations in (1.26) and combining with the continuity equations leads to [29]

$$\begin{cases} (\sigma_x - \iota\, k)\, p_x - \partial_x \left[\dfrac{1}{\sigma_x - \iota\, k} \partial_x p \right] = 0 \quad \text{in } \Omega_{\text{all}}, \\ (\sigma_y - \iota\, k)\, p_y - \partial_y \left[\dfrac{1}{\sigma_y - \iota\, k} \partial_y p \right] = 0 \quad \text{in } \Omega_{\text{all}}, \\ (\sigma_z - \iota\, k)\, p_z - \partial_z \left[\dfrac{1}{\sigma_z - \iota\, k} \partial_z p \right] = 0 \quad \text{in } \Omega_{\text{all}}. \end{cases} \tag{1.27}$$

Finally, using $p = p_x + p_y + p_z$, Equations of (1.27) can be combined into [29, 30, 31]

$$\operatorname{div}\left(\mathbb{D}\, \mathbf{grad}\, p\right) + D k^2 p = 0, \tag{1.28}$$

where $\mathbb{D}(\mathbf{x})$ is the *dissipation tensor* and $D(\mathbf{x})$ is the *dissipation scalar* defined as

$$\mathbb{D}(\mathbf{x}) = \operatorname{diag}\left(\frac{\gamma_y(y)\gamma_z(z)}{\gamma_x(x)}, \frac{\gamma_x(x)\gamma_z(z)}{\gamma_y(y)}, \frac{\gamma_x(x)\gamma_y(y)}{\gamma_z(z)} \right), \tag{1.29}$$

$$D(\mathbf{x}) = \gamma_x(x)\gamma_y(y)\gamma_z(z). \tag{1.30}$$

The *stretching functions* $\gamma_x(x)$, $\gamma_y(y)$ and $\gamma_z(z)$ are defined as

$$\gamma_x(x) = 1 + \frac{\iota\, \sigma_x(x)}{k}, \quad \gamma_y(y) = 1 + \frac{\iota\, \sigma_y(y)}{k}, \quad \text{and} \quad \gamma_z(z) = 1 + \frac{\iota\, \sigma_z(z)}{k}. \tag{1.31}$$

3.3.2 The reflection coefficient and some well-known absorbing function families

To study some properties of the PML and motivate the choice of absorption function, let us consider the previous half-space domain Ω_{int} with an PML region of thickness δ, $\Omega_{\text{PML}} := \{(x, y, z) : x \in]0, \delta[\}$. The Helmholtz equation with PML (1.28) and with a homogeneous Neumann condition on the exterior boundary of the PML, *i.e.* $\{\mathbf{x} : x = \delta\}$, is considered on this half-space domain with the PML, $\Omega_{\text{int}} \cup \Omega_{\text{PML}}$. Furthermore, let us also follow the change of coordinate inside the PML region proposed in [51],

$$x^\star(x) = \int_0^x \gamma_x(s)\, \mathsf{d}s = x + \frac{\iota}{k}\int_0^x \sigma_x(s)\, \mathsf{d}s, \tag{1.32}$$

such that,

$$\frac{\partial}{\partial x^\star} = \frac{1}{\gamma_x}\frac{\partial}{\partial x}. \tag{1.33}$$

Hence, the general solution of Equation (1.28) of this half-space problem is

$$p(\mathbf{x}) = \left(P_i + P_{r_1} e^{-2\,\iota\, k_x x}\right) e^{\iota\, \mathbf{k}\cdot\mathbf{x}} \qquad \text{in } \Omega_{\text{int}}, \tag{1.34}$$

$$p^\star(\mathbf{x}^\star) = \left(P_t + P_{r_2} e^{-2\,\iota\, k_x x^\star}\right) e^{\iota\, \mathbf{k}\cdot\mathbf{x}^\star} \qquad \text{in } \Omega_{\text{PML}}, \tag{1.35}$$

where $\mathbf{x}^\star : (x^\star, y, z)$, $k_x = \mathbf{k}\cdot\mathbf{e}_x$, $p^\star$ is the pressure field into de PML expressed in coordinate $(x^\star, y, z)$ and P_i, P_{r_1}, P_t and P_{r_2} are the incident wave amplitude, the amplitude of reflected wave by the PML, the amplitude of the transmitted wave inside the PML and the amplitude of the reflected wave by the end of the PML, respectively. By defining the *attenuation factor* by

$$\alpha(x) = \cos\theta \int_0^x \sigma_x(s)\, \mathsf{d}s, \tag{1.36}$$

where θ is the angle between $\mathbf{e}_x$ and the incident, the integration of (1.32) inside (1.35) leads to

$$p(\mathbf{x}) = \left(P_t e^{-\alpha(x)} + P_{r_2} e^{-2\,\iota\, k_x x} e^{\alpha(x)}\right) e^{\iota\, \mathbf{k}\cdot\mathbf{x}}. \tag{1.37}$$

Considering the continuity of the acoustic pressure and the local particle velocity at the interface between the PML and the interior domain, one can obtain that $P_i = P_t$ and $P_{r_1} = P_{r_2}$, which means that for every incident angle, the incident wave is perfectly transmitted to the PML layer, *i.e.* without reflections on the interface.

Moreover, considering the homogeneous Neumann condition imposed at the end of the PML, the following reflection coefficient can be obtained,

$$\Gamma = \left|\frac{P_{r_1}}{P_i}\right| = \exp\left[-2\alpha(\delta)\right], \tag{1.38}$$

which gives an indication on the choice of absorption function. Indeed, the reflection coefficient is zero if

$$\alpha(\delta) = \cos\theta \int_0^\delta \sigma_x(s)\, \mathsf{d}s = +\infty. \tag{1.39}$$

In practice, classical techniques proposed to take a bounded absorbing function such that the integral of (1.38) is large enough. Basically, the following linear and parabolic

absorbing functions varying form zero at the beginning to $\sigma^\star$ have been proposed [30, 60],

$$\sigma_l(X) = \sigma^\star \frac{X}{\delta} \qquad \text{and} \qquad \sigma_q(X) = \sigma^\star \left(\frac{X}{\delta}\right)^2, \tag{1.40}$$

where X is the coordinate starting to zero at the beginning of the PML and increasing inside the thickness of the PML until its end at $X = \delta$. Note that this family of PML requires to perform an optimization of the parameter $\sigma^\star$. Another popular technique proposes to take an unbounded function such as the following hyperbolic or shifted-hyperbolic function,

$$\sigma_h(X) = \frac{1}{\delta - X} \qquad \text{or} \qquad \sigma_{sh}(X) = \frac{1}{\delta - X} - \frac{1}{\delta}. \tag{1.41}$$

These choices are compared with linear or quadratic PML in [35, 32, 53]. Note that compared to linear or quadratic PML there is no parameter to optimize for this family of PMLs.

Finally, note that the continuous fraction expansions of the exact DtN presented in Section 3.2 can be compared with the PML introduced in this section. Indeed, finite continuous fraction expansions of the exact DtN like (1.20) lead to the development of so called *continuous fraction absorbing boundary conditions* (CFABCs) [101, 99, 100, 15]. These expansions of the exact DtN like (1.20) can be seen as an absorbing finite mesh built around the domain that is comparable with PMLs. These PMLs built using a continuous fraction expansions are called *perfectly matched discrete layer* (PMDLs) [101].

In Chapter 3, strategies to deal with cross-points in the presence of PML conditions will be presented. In addition, numerical applications that implement PMLs with cross-point treatment will be studied in Chapter 4.

4 The pollution effect

Application of low-order FEM, *e.g.* with classical piece-wise linear shape functions, to the Helmholtz equation leads to numerical errors that grow with the wavenumber k if a simple "rule of the thumb" of the form $kh = \text{const}$, where h is the characteristic element size, is used [114, 115]. More precisely, the relative FE error in H^1-seminorm $|\epsilon|_{H^1(\Omega)} := |u_{\text{ex}} - u|_{H^1(\Omega)}/|u_{\text{ex}}|_{H^1(\Omega)}$ with u_{ex} the exact solution follows

$$|\epsilon|_{H^1(\Omega)} \leq C_1(p)\left(\frac{kh}{2p}\right)^p + C_2(p)k\left(\frac{kh}{2p}\right)^{2p}, \tag{1.42}$$

where $C_1(p)$ and $C_2(p)$ are constant independent of h and k but not of p and $|\cdot|_{H^1(\Omega)} = \|\mathbf{grad}\,(\cdot)\|_{L^2(\Omega)}$ is the H^1-seminorm such that $\|\cdot\|_{H^1(\Omega)} = \|\cdot\|_{L^2(\Omega)} + |\cdot|_{H^1(\Omega)}$. The first term of the right-hand side is called the *approximation error* while the second one is called the *pollution effect*. The approximation error is a local property than can be studied for each element independently. On the opposite, the pollution error is a global property of the Helmholtz equation's FE simulation that induces a dispersion on the numerical solution, *i.e.* a difference of phase between the exact solution and the FE one.

In practice, (1.42) implies that high-frequency FE Helmholtz simulations require high-order FE basis functions. Indeed the approximation term decreases as the power of p while the pollution term decreases as the power of $2p$. The high-order basis functions tend to increase the FE assembly time and reduce the sparsity pattern of the FE matrix. The increase in FE assembly time directly results from two leading causes: first, an increase of the number of quadrature points is needed to integrate the high-order basis functions over the elements; second, increasing the elementary matrix size over each element increases the computational intensity.

In this thesis, we will always use at least second-order polynomial basis functions, and at most polynomial order ten. Implementing an algorithmically efficient finite element matrix assembly for high orders drove the development of GmshFEM, which will be presented in Chapter 5 of Part II of this manuscript. The huge memory footprint of matrix factorizations at high order further plead for a sub-structured DDM, where the size of subdomains can be chosen small enough for the factorization to hold in memory.

5 The indefinite nature of the Helmholtz operator

The resolution of Helmholtz problems with classical iterative methods such as Krylov methods is not effective, due to the indefinite nature of the Helmholtz operator [80]. Below we briefly review this property, as well as two major theorems that we be needed in later chapters to assess of the well-posedness of our proposed formulations for handling cross-points.

The finite element method does not discretize the Helmholtz equation in strong form, as written in (1.4), but rather discretizes its *weak formulation*, also called *variational formulation*. Multiplying the first equation of (1.4) by a test function $v \in H_0^1(\Omega)$, integrating over Ω and using Green's first identity, leads to

$$\int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, \overline{v} - k^2 u \overline{v} \,\mathrm{d}\Omega - \int_{\Gamma_{\text{ext}}} \mathcal{B} u \overline{v} \,\mathrm{d}\Gamma_{\text{ext}} = \int_\Omega f(\mathbf{x}) \overline{v} \,\mathrm{d}\Omega + \int_{\Gamma^N_{\text{scat}}} u_{\text{N}} \overline{v} \,\mathrm{d}\Gamma^N_{\text{scat}} \quad (1.43)$$

with the overline $\overline{\cdot}$ denoting the complex conjugate. Equation (1.43) is of the form

$$h(u, v) = f(v), \quad (1.44)$$

with

$$h(u, v) = \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, \overline{v} - k^2 u \overline{v} \,\mathrm{d}\Omega - \int_{\Gamma_{\text{ext}}} \mathcal{B} u \overline{v} \,\mathrm{d}\Gamma_{\text{ext}} \quad (1.45)$$

a sesquilinear form, and

$$f(v) = \int_\Omega f(\mathbf{x}) \overline{v} \,\mathrm{d}\Omega + \int_{\Gamma^N_{\text{scat}}} u_{\text{N}} \overline{v} \,\mathrm{d}\Gamma^N_{\text{scat}} \quad (1.46)$$

a continuous antilinear form.

Equation (1.43) follows the general form of a weak formulation for complex scalar field given by:

$$\text{find } u \in \mathcal{V} \text{ such that } a(u, v) = f(v) \text{ holds for all } v \in \mathcal{V}, \quad (1.47)$$

with $\mathcal{V}$ a complex Hilbert space, $a(\cdot,\cdot) : \mathcal{V}\times\mathcal{V} \to \mathbb{C}$ a sesquilinear form, and $f : \mathcal{V} \to \mathbb{C}$ an antilinear form.

A general problem such as (1.47) is well-posed, *i.e.* it exists an unique solution that depends continuously upon the specified datum f, if the following Lax–Milgram theorem is applicable [79]:

Theorem 1. *Let $\mathcal{V}$ be a Hilbert space and $a(\cdot,\cdot)$ a sesquilinear form on $\mathcal{V}\times\mathcal{V}$, which is*

$$\textit{bounded:} \quad |a(u,v)| \leq C\|u\|_{\mathcal{V}}\|v\|_{\mathcal{V}} \qquad \textit{for all } u,v \in \mathcal{V} \tag{1.48}$$

$$\textit{coercive:} \quad |a(v,v)| \geq \alpha\|v\|^2_{\mathcal{V}} \qquad \textit{for all } v \in \mathcal{V}. \tag{1.49}$$

with $C > 0$ and $\alpha > 0$. Then, for any f in $\mathcal{V}'$, i.e. the dual of $\mathcal{V}$, there is a unique solution $u \in \mathcal{V}$ to (1.47) and it holds

$$\|u\|_{\mathcal{V}} \leq \frac{1}{\alpha}\|f\|_{\mathcal{V}'}. \tag{1.50}$$

Note that (1.48) is called the *continuity* condition and (1.49) is called either *coercivity* or $\mathcal{V}$*-ellipticity* depending on the authors. The indefinite nature of the Helmholtz operator is another way the tell that the Helmholtz operator is not coercive.

Let us consider the Helmholtz problem (1.4) with a homogeneous wavenumber and with a 0^{th} order impedance ABC on the exterior boundary, *i.e.* $\mathcal{B} = \iota\, k$. The varitional formulation of this problem reads

$$h_{\text{imp}}(u,v) = f_{\text{imp}}(v), \tag{1.51}$$

with

$$h_{\text{imp}}(u,v) = \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, \overline{v} - k^2 u\overline{v}\, \mathsf{d}\Omega - \int_{\Gamma_{\text{ext}}} \iota\, k u\overline{v}\, \mathsf{d}\Gamma_{\text{ext}}, \tag{1.52}$$

$$f_{\text{imp}}(v) = \int_\Omega f(\mathbf{x})\overline{v}\, \mathsf{d}\Omega + \int_{\Gamma^N_{\text{scat}}} u_{\text{N}}\overline{v}\, \mathsf{d}\Gamma^N_{\text{scat}}. \tag{1.53}$$

The continuity of (1.52) follows by the Cauchy–Schwarz inequality and the continuity of the trace map from $H^1(\Omega)$ to $L^2(\partial\Omega)$. Then it is clear that (1.52) cannot be bounded below by $\|v\|^2_{\mathcal{V}}$ for all k and all $v \in \mathcal{V}$. Indeed, let us consider that the squared wavenumber k^2 is equal to one eigenvalue of the negative Laplacian with homogeneous Dirichlet boundary conditions on Γ_{ext}. Under this condition,

$$h_{\text{imp}}(v,v) = \int_\Omega \mathbf{grad}\, v \cdot \mathbf{grad}\, \overline{v} - k^2 v\overline{v}\, \mathsf{d}\Omega - \int_{\Gamma_{\text{ext}}} \iota\, k v\overline{v}\, \mathsf{d}\Gamma_{\text{ext}}, \tag{1.54}$$

is null for v corresponding to the eigenfunction since the two first terms cancel each other out and the last one vanishes as $v = 0$ on Γ_{ext}. Therefore, $h_{\text{imp}}(u,v)$ is not coercive, and the resulting algebraic system will be sign-indefinite. Furthermore, the algebraic system is also non-Hermitian because of the ABC, $h_{\text{imp}}(u,v) \neq \overline{h}_{\text{imp}}(v,u)$, thus the complex eigenvalues are spread on both side of the imaginary axis. This fact

strongly contributes to the solving difficulty of the Helmholtz equation with iterative solvers; see reviews [11, 77, 80] for more details. Note that it is possible to have a sign-definite formulation of the Helmholtz equation under impedance boundary conditions with some modification of the space H^1 on star-shaped domains [149].

Although the Lax-Milgram theorem is not fulfilled by the Helmholtz problem with a homogeneous wavenumber and a 0^{th} order impedance ABC on the exterior boundary, it is well-posed because the Lax-Milgram theorem is a sufficient condition of well-posedness. The following Banach–Nečas–Babuška (BNB) theorem [79] gives the necessary and sufficient condition of well-posedness.

Theorem 2. *Let $\mathcal{U}$ be a Banach space and let $\mathcal{V}$ be a reflexive Banach space. Let $a(\cdot,\cdot)$ be a bounded sesquilinear form on $\mathcal{U}\times\mathcal{V}$ and let $f\in\mathcal{V}'$. Then the problem (1.4) is well-posed iff:*

$$\inf_{u\in\mathcal{U}}\sup_{v\in\mathcal{V}}\frac{|a(u,v)|}{\|u\|_{\mathcal{U}}\|v\|_{\mathcal{V}}}:=\alpha>0, \tag{1.55}$$

$$\forall v\in\mathcal{V},\left[\forall u\in\mathcal{U}, a(u,v)=0\right]\Rightarrow\left[v=0\right]. \tag{1.56}$$

Moreover, the error estimate $\|u\|_{\mathcal{U}}\leq\frac{1}{\alpha}\|f\|_{\mathcal{V}'}$ holds. The first condition (1.55) is often called the inf-sup condition.

In Chapter 3, the BNB theorem will be used to analyze the stability of proposed strategies to efficiently deal with cross-points when interface conditions based on PMLs are used. In particular, a numerical test will be employed to evaluate the parameter α of Equation (1.55), also called the *inf-sup constant*.

6 The non-overlapping domain decomposition method

6.1 The sub-structuring algorithm

To describe the standard DDM, let us consider the simple Helmholtz problem (1.4). The total domain Ω_{tot} is decomposed into N non-overlapping subdomains Ω_n, with $n=1\cdots N$. Let us define $\Gamma^D_{n,\text{scat}}$ and $\Gamma^N_{n,\text{scat}}$ such that $\Gamma^D_{\text{scat}}=\bigcup_{n=0}^{N-1}\Gamma^D_{n,\text{scat}}$ and $\Gamma^N_{\text{scat}}=\bigcup_{n=0}^{N-1}\Gamma^N_{n,\text{scat}}$. Furthermore, let us define $\Gamma_{n,\text{ext}}$ as the boundary of subdomain Ω_n. The boundary $\Gamma_{n,\text{ext}}$ can be decomposed into the union of a *exterior boundary* that belongs to the boundary of Ω_{tot} (*i.e.* $\Gamma_{n,\text{ext}}\cap\partial\Omega_{\text{tot}}$), and an *interior boundary* that is shared by two subdomains. Let us further decompose the interior boundary of each subdomain Ω_n as the union of *interfaces*, $\bigcup_{n\in\mathfrak{N}_n}\Sigma_{n,m}$ where $\mathfrak{N}_n$ is the set of neighbor subdomains of subdomain Ω_n. For each subdomain Ω_n, let us consider the local solution u_n of the *subproblem*

$$\begin{cases}\Delta u_n+k^2u_n=-f & \text{in } \Omega_n,\\ u_n=u_{\text{D}} & \text{on } \Gamma^D_{n,\text{ scat}},\\ \partial_{\mathbf{n}}u_n=u_{\text{N}} & \text{on } \Gamma^N_{n,\text{scat}},\\ \partial_{\mathbf{n}_n}u_n-\mathcal{T}_nu_n=g_{n,m} & \text{on } \Gamma_{n,\text{ext}},\end{cases} \tag{1.57}$$

where $\mathbf{n}_n$ is the outgoing normal of subdomain Ω_n, $\mathcal{T}_n$ is a transmission operator defined as

$$\mathcal{T}_n : H^{1/2}(\partial\Omega_n) \to H^{-1/2}(\partial\Omega_n) : u_n|_{\partial\Omega_n} \mapsto (\partial_{\mathbf{n}_n} u_n)|_{\partial\Omega_n} \quad (= \mathcal{T}_n u_n|_{\partial\Omega_n}). \tag{1.58}$$

and $g_{n,m}$ is a transmission variable defined as

$$g_{n,m} := \begin{cases} 0 & \text{on } \Gamma_{n,\text{ext}} \cap \partial\Omega_{\text{tot}}, \\ \partial_{\mathbf{n}_n} u_m - \mathcal{T}_n u_m & \text{on each } \Sigma_{n,m}, \end{cases} \tag{1.59}$$

where u_m is the local solution on Ω_m, the neighboring subdomain of Ω_n.

At every step of an iterative procedure, a subproblem similar to Problem (1.57) is solved for every subdomain Ω_n. Then, the transmission variables are updated and exchanged between the subdomains. Since System (1.59) is defined for every transmission variable, the update condition

$$g_{m,n} := \partial_{\mathbf{n}_m} u_n - \mathcal{T}_m u_n \qquad \text{on } \Sigma_{n,m} \text{ for each } n \in \mathfrak{N}_m, \tag{1.60}$$

is prescribed on the interface. Assuming that the transmission operator is symmetric, *i.e.* $\mathcal{T}_n = \mathcal{T}_m$, Combining the interface equation of (1.57) and the update condition (1.60) gives:

$$g_{m,n} = -g_{n,m} - 2\mathcal{T}_n u_n \qquad \text{on } \Sigma_{m,n} \text{ for each } n \in \mathfrak{N}_m. \tag{1.61}$$

The sub-structuring DDM algorithm at iteration $\ell + 1$ can be described as follow:

a. For all $n = 1 \cdots N$, compute the solution $u_n^{(\ell+1)}$ by solving Subproblem (1.57), that can be rewritten as

$$u_n^{(\ell+1)} = \mathcal{S}_n(u_{\text{D}}, u_{\text{N}}, \mathsf{g}^{(\ell)}), \tag{1.62}$$

where $\mathsf{g}^{(\ell)}$ is the vector collecting all $g_{n,m}$. In the following u_{D} and u_{N} will be called the *physical sources*, as opposed to g which will be called the *artificial source*.

b. For all $n = 1 \cdots N$ and $m \in \mathfrak{N}_n$, update the interface unknowns through interface problem (1.61), which can be rewritten as

$$g_{m,n}^{(\ell+1)} = \mathcal{I}_{m,n}(g_{n,m}^{(\ell)}, u_n^{(\ell+1)}). \tag{1.63}$$

Note that by linearity and for every $\ell \in \mathbb{N}$, the field $u_n^{(\ell+1)}$ can be decomposed into a sum of a physical source contribution and an artificial source contribution, $u_n^{(\ell+1)} = u_{n,P}^{(\ell+1)} + u_{n,A}^{(\ell+1)}$ such that

$$u_{n,P}^{(\ell+1)} = \mathcal{S}_n(u_{\text{D}}, u_{\text{N}}, 0) \qquad \text{and} \qquad u_{n,A}^{(\ell+1)} = \mathcal{S}_n(0, 0, \mathsf{g}^{(\ell)}). \tag{1.64}$$

As the physical source contribution is independent of the iteration, it will be denoted $u_{n,P}$ hereafter. Equation (1.63) then becomes

$$g_{m,n}^{(\ell+1)} = \mathcal{I}_{m,n}(g_{n,m}^{(\ell)}, u_{n,A}^{(\ell+1)}) - 2\mathcal{T}_n u_{n,P}. \tag{1.65}$$

Let us define the vector b that collects all $b_{m,n} = -2\mathcal{T}_n u_{n,P}|_{\Sigma_{n,m}}$ and the operator $\mathcal{A} : \mathsf{g}^{(\ell)} \mapsto \mathcal{A}\mathsf{g}^{(\ell)}$

$$\begin{cases} u_{n,A}^{(\ell+1)} = \mathcal{S}_n(0, 0, \mathsf{g}^{(\ell)}), \\ (\mathcal{A}\mathsf{g}^{(\ell+1)})_{m,n} = \mathcal{I}_{m,n}(g_{n,m}^{(\ell)}, u_{n,A}^{(\ell+1)}), \end{cases} \tag{1.66}$$

such that (1.65) can be rewritten as

$$\mathsf{g}^{(\ell+1)} = \mathcal{A}\mathsf{g}^{(\ell)} + \mathsf{b}. \tag{1.67}$$

The interface problem written in this way can be interpreted as one iteration of a Jacobi algorithm applied to the linear system

$$(\mathcal{I}_d - \mathcal{A})\mathsf{g} = \mathsf{b}, \tag{1.68}$$

where $\mathcal{I}_d$ is the identity operator. While the Jacobi algorithm will only converge if the spectral radius of $\mathcal{I}_d - \mathcal{A}$ is smaller than one, any iterative linear solver can be applied to (1.68), for instance Krylov subspace solvers such as GMRES [169], BiCGSTAB [187], or ORTHODIR [170]. The convergence rate of these algorithms strongly depends on the spectrum of $\mathcal{I}_d - \mathcal{A}$, which itself depends on the transmission operator $\mathcal{T}_n$ enforced on the interface between the subdomains. One can easily see that the optimal convergence is obtained by imposing the DtN map (1.12) introduced in Section 3.1, related to the complementary of each subdomain [152, 151], which leads to $\mathcal{A} \equiv 0$. Since the cost of computing the exact DtN is prohibitive, following the same idea as for the boundary conditions detailed in Section 3.2, operators based on low-order absorbing boundary conditions to approximate the DtN have been developed since the late '80s and early '90s [103, 66, 153], followed in the late '90s and early '00s by (optimized) second-order transmission conditions [85, 161]. More recently, domain decomposition strategies were developed with high-order transmission conditions [45, 46, 123, 136] based on HABCs presented in Section 3.2, transmission conditions based on PMLs [177, 188] as introduced in Section 3.3 and non-local transmission operators [179, 128, 127, 59]. In general, high-order and PML-based conditions accelerate the convergence of DDMs compared to low-order conditions, with an extra cost per iteration that can be controlled thanks to the order of the conditions or the thickness of the PMLs. Non-local approaches are more expensive per iteration in terms of computational cost, but they have the best convergence rate, and a solid theoretical background is available [58, 127, 59].

It is important to note that in the sub-structured DDM, the iteration unknowns are surface quantities, *i.e.* fields $g_{n,m}$, and not the volume ones. The whole sub-structuring DDM algorithm is summed up in pseudo-code in Figure 1.3.

6.2 Cross-points

Most of the DDMs with high-order, PML-based, and non-local transmission operators have initially been tested and studied for configurations with one-dimensional domain partitions, as depicted on Figure 1.4a (*e.g.* layered partitions, or partitions of spherical shells into onion peels). Such partitions do not exhibit (interior) cross-points (*i.e.* points where more than two subdomains meet), which simplifies the implementation and avoids technical difficulties with the transmission conditions. However, for large-scale

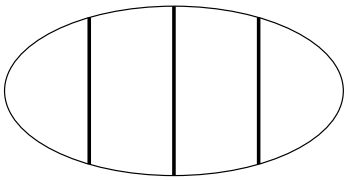
Data: N subdomains
Data: All $\mathfrak{N}_n$ connectivity data

`1. Compute the right-hand size b:`
for *n=0;* $n < N$ **do**
  Compute $u_{n,P} = \mathcal{S}_n(u_\mathrm{D}, u_\mathrm{N}, 0)$;
  foreach m **in** $\mathfrak{N}_n$ **do**
    Compute $b_{m,n} = \mathcal{I}_{m,n}(0, u_{n,P})$;
  end
end
Gather all $b_{m,n}$ into the vector $\mathtt{b}$;
Initialize $\mathtt{g}$ with zeros;

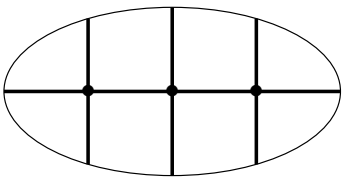
`2. Solve the interface problem` $(\mathcal{I}_d - \mathcal{A})\mathtt{g} = \mathtt{b}$ `iteratively using a Krylov subspace solver. At each iteration:`
for *n=0;* $n < N$ **do**
  Compute $u_{n,A}^{\ell+1} = \mathcal{S}_n(0, 0, \mathtt{g}^\ell)$;
  foreach m **in** $\mathfrak{N}_n$ **do**
    Evaluate $(\mathcal{A}\mathtt{g}^{(\ell+1)})_{m,n} = \mathcal{I}_{m,n}(g_{n,m}^{(\ell)}, u_{n,A}^{(\ell+1)})$;
  end
end

`3. Compute the final solution:`
for *n=0;* $n < N$ **do**
  Compute $u_n = \mathcal{S}_n(u_\mathrm{D}, u_\mathrm{N}, \mathtt{g}^\ell)$;
end

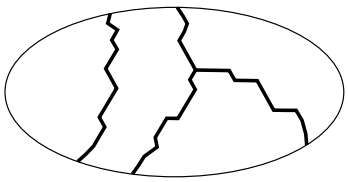
Figure 1.3: Pseudo-code of the sub-structuring DDM algorithm.



(a) The layer partition.



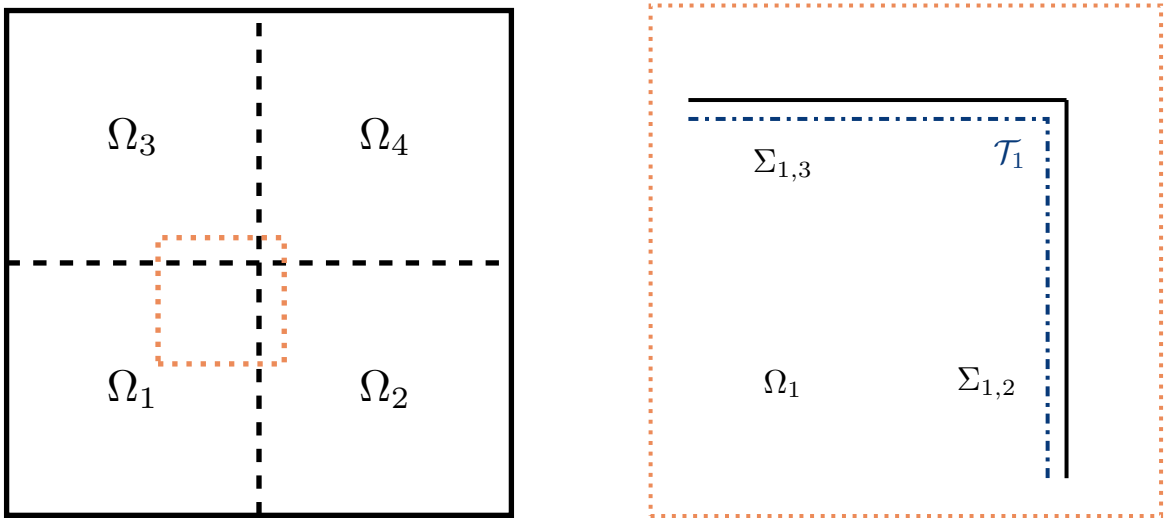
(b) The checkerboard partition. Note the three cross-points appearing between the three vertical interfaces and the horizontal one.



(c) The automatic mesh partition.

Figure 1.4: Three kind of partitions of a computational domain.

two- and three-dimensional applications, one-dimensional partitions are not optimal, as the amount of interface data can be minimized with more general multi-dimensional partitions, *e.g.* Cartesian checkerboard partitions leading to right-angle cross-points (as in Figure 1.4b), Cartesian lattice-type with arbitrary-angle cross-points, or general partitions generated with automatic mesh partitioners (as in Figure 1.4c). The cross-



(a) A square domain is partitioned into four subdomains, building a two-times-two grid with an interior cross-point at the intersection between the four subdomains.

(b) Zoom inside the subdomain Ω_1 and around the cross-point. A DtN operator $\mathcal{T}_1$ is prescribed on the whole boundary of Ω_1.

Figure 1.5: A cross-point between four subdomains.

points require some care with such partitions, especially when transmission conditions with differential or integral operators are used. Specific strategies to deal with cross-points have been proposed for DDMs with low-order transmission conditions and nodal finite element discretizations (*e.g.* [86, 28, 82]). Recently, cross-point treatments have been proposed for DDMs with second-order conditions [157, 67, 84] and non-local approaches [54, 55, 158] in order to address general domain partitions.

In this thesis, we propose an alternative local treatment of cross-points that can deal with high-order transmission operators based on high order ABCs or PMLs and that does not lead to a global system coupling all subdomains. This is crucial for the scalability of the methods on massively parallel architectures, as will be demonstrated in Chapter 4.

To set the stage for the cross-point treatments introduced in the next two chapters, let us give an operator view of the sub-structuring DDM with a single cross-point. We consider a two-by-two grid partition (i.e. with four subdomains) with a single interior cross-point, as shown in Figure 1.5a. Then let us focus on the expression of the transmission operator in the first subdomain Ω_1 with interior boundary called $\Sigma_{1,2}$ and $\Sigma_{1,3}$. In order to keep things simple, let us assume that on the exterior boundary, a boundary condition is enforced, such that $\mathcal{T}_1$ is only defined on $\Sigma_{1,2} \cup \Sigma_{1,3}$. Moreover, let us reformulate the transmisson operator around the interior cross-point as shown in Figure 1.5b as

$$\mathcal{T}_1 u_1|_{\Sigma_{1,2}\cup\Sigma_{1,3}} = \begin{bmatrix} \mathcal{D}_1^{1,2} & \mathcal{C}_1^{1,2} \\ \mathcal{C}_1^{1,3} & \mathcal{D}_1^{1,3} \end{bmatrix} \begin{bmatrix} u_1|_{\Sigma_{1,2}} \\ u_1|_{\Sigma_{1,3}} \end{bmatrix} \quad \left(= \begin{bmatrix} (\partial u_1)|_{\Sigma_{1,2}} \\ (\partial u_1)|_{\Sigma_{1,3}} \end{bmatrix} \right), \tag{1.69}$$

where the operators are defined as

$$\mathcal{D}_1^{1,2} : H^{1/2}(\Sigma_{1,2}) \to H^{-1/2}(\Sigma_{1,2}) : u_1|_{\Sigma_{1,2}} \mapsto (\mathcal{D}_1^{1,2} u_1)|_{\Sigma_{1,2}}, \tag{1.70}$$

$$\mathcal{D}_1^{1,3} : H^{1/2}(\Sigma_{1,3}) \to H^{-1/2}(\Sigma_{1,3}) : u_1|_{\Sigma_{1,3}} \mapsto (\mathcal{D}_1^{1,3} u_1)|_{\Sigma_{1,3}}, \tag{1.71}$$

$$\mathcal{C}_1^{1,2} : H^{1/2}(\Sigma_{1,3}) \to H^{-1/2}(\Sigma_{1,2}) : u_1|_{\Sigma_{1,3}} \mapsto (\mathcal{C}_1^{1,2} u_1)|_{\Sigma_{1,2}}, \tag{1.72}$$

$$\mathcal{C}_1^{1,3} : H^{1/2}(\Sigma_{1,2}) \to H^{-1/2}(\Sigma_{1,3}) : u_1|_{\Sigma_{1,2}} \mapsto (\mathcal{C}_1^{1,3} u_1)|_{\Sigma_{1,3}}. \tag{1.73}$$

In a nutshell, $\mathcal{D}_1^{1,2}$ and $\mathcal{D}_1^{1,3}$ are "direct" operators that link a Dirichlet to a Neumann trace defined on the same interface. In contrast, $\mathcal{C}_1^{1,2}$ and $\mathcal{C}_1^{1,2}$ are "coupling" operators that link a Dirichlet trace defined on an interface to the Neumann trace defined on the other one. Note that in the case of a local ABC, such as a simple impedance condition, $\mathcal{C}_1^{1,2} = \mathcal{C}_1^{1,3} = 0$ since the Neumann trace on one point depends only on the Dirichlet trace on this same point and not on the others.

The transmission condition on the neighbor subdomain Ω_2 can also be rewritten as

$$\mathcal{T}_2 u_2|_{\Sigma_{2,1}\cup\Sigma_{2,4}} = \begin{bmatrix} \mathcal{D}_2^{2,1} & \mathcal{C}_2^{2,1} \\ \mathcal{C}_2^{2,4} & \mathcal{D}_2^{2,4} \end{bmatrix} \begin{bmatrix} u_2|_{\Sigma_{2,1}} \\ u_2|_{\Sigma_{2,4}} \end{bmatrix}. \tag{1.74}$$

Furthermore, if the same transmission condition is used on subdomains Ω_1 and Ω_2 then $\mathcal{D}_1^{1,2} = \mathcal{D}_2^{2,1}$ but $\mathcal{C}_1^{1,2} \neq \mathcal{C}_2^{2,1}$. Following the same reasoning as done to derive Equation (1.61), one can find

$$g_{2,1} = -g_{1,2} - \begin{bmatrix} 2\mathcal{D}_1^{1,2} & \mathcal{C}_1^{1,2} + \mathcal{C}_2^{1,2} \end{bmatrix} \begin{bmatrix} u_1|_{\Sigma_{1,2}} \\ u_1|_{\Sigma_{1,3}} \end{bmatrix}, \tag{1.75}$$

which is almost the same as (1.61), excepted that the transmission operator $\mathcal{T}_i$ cannot be considered as symmetric due to coupling operators $\mathcal{C}_1^{1,2}$ and $\mathcal{C}_2^{2,1}$ that do not have the same domain of definition. Both HABC-based and PML-based DDMs presented in Chapters 2 and 3 propose to define two operators $\mathcal{P}_1^{1,2}$ and $\mathcal{P}_2^{2,1}$ such that $\mathcal{C}_1^{1,2}\mathcal{P}_1^{1,2} = \mathcal{C}_2^{2,1}\mathcal{P}_2^{2,1}$. The exact definition of these operators depends on the considered method and will be detailed later on. In short, they give a mapping between the Dirichlet trace of one or more fields defined on a geometric entity shared by subdomain Ω_1 and Ω_2 with the Dirichlet trace of u_1 and u_2 on $\Sigma_{1,3}$ and $\Sigma_{2,4}$, respectively.

Let us go a bit further in the decomposition of the cross-point problem by defining the interface fields g as the sum of *direct interface fields* g^{D} and *coupling interface fields* g^{C}

$$g_{2,1} = g_{2,1}^{\mathrm{D}} + g_{2,1}^{\mathrm{C}}, \tag{1.76}$$

$$g_{1,2} = g_{1,2}^{\mathrm{D}} + g_{1,2}^{\mathrm{C}}, \tag{1.77}$$

where direct and coupling interface fields are defined as

$$g_{2,1}^{\mathrm{D}} := \left(\partial_{\mathbf{n}_2} u_1 - n_1^C\right) - \mathcal{D}_2^{2,1} u_1, \tag{1.78}$$

$$g_{1,2}^{\mathrm{D}} := \left(\partial_{\mathbf{n}_1} u_2 - n_2^C\right) - \mathcal{D}_1^{1,2} u_2, \tag{1.79}$$

and

$$g_{2,1}^{\mathrm{C}} := n_1^C - \left(\mathcal{C}_1^{2,1} + \mathcal{C}_2^{2,1}\right) u_1, \tag{1.80}$$

$$g_{1,2}^{\mathrm{C}} := n_2^C - \left(\mathcal{C}_1^{1,2} + \mathcal{C}_2^{1,2}\right) u_2, \tag{1.81}$$

with n_1^C and n_2^C are auxiliary fields used to decouple the direct and coupling interface fields. Using this definitions, Equation (1.75) becomes:

$$\begin{bmatrix} g_{2,1}^{\mathrm{D}} \\ g_{2,1}^{\mathrm{C}} \end{bmatrix} = - \begin{bmatrix} g_{1,2}^{\mathrm{D}} \\ g_{1,2}^{\mathrm{C}} \end{bmatrix} - \begin{bmatrix} 2\mathcal{D}_1^{1,2} & 0 \\ 0 & \mathcal{C}_1^{1,2} + \mathcal{C}_2^{2,1} \end{bmatrix} \begin{bmatrix} u_1|_{\Sigma_{1,2}} \\ u_1|_{\Sigma_{1,3}} \end{bmatrix}, \tag{1.82}$$

and the coupling equation of System (1.57) becomes:

$$\begin{bmatrix} (\partial_{\mathbf{n}_1} u_1)|_{\Sigma_{1,2}} - n_1^{\mathrm{C}} \\ n_1^{\mathrm{C}} \end{bmatrix} - \begin{bmatrix} \mathcal{D}_1^{1,2} & 0 \\ 0 & \mathcal{C}_1^{1,2} \end{bmatrix} \begin{bmatrix} u_1|_{\Sigma_{1,2}} \\ u_1|_{\Sigma_{1,3}} \end{bmatrix} = \begin{bmatrix} g_{1,2}^{\mathrm{D}} \\ g_{1,3}^{\mathrm{C}} \end{bmatrix}. \tag{1.83}$$

Compared with the global interface transmission operator (1.69), which fully couples the update on interfaces $\Sigma_{1,2}$ and $\Sigma_{1,3}$, the introduction of auxiliary corner variables allows to "localize" the updates on each interface plus the cross-point. These interface fields are the cornerstone of the strategies that will be presented in Part I to deal with cross-points when HABC (Chapter 2) or PML (Chapter 3) are used as transmission conditions for the DDM. Their local nature will allow for an efficient parallelization on distributed HPC clusters, as will be demonstrated in Chapter 4 for large scale acoustic and elastic simulations.

Part I

Numerical methods

2 Cross-points for high-order transmission conditions

In this chapter, a non-overlapping DDM with HABC-based transmission conditions is extended to efficiently deal with cross-points for lattice-type partitionings. The cross-point treatments when the HABC operator is used in the transmission conditions and when it is used in the exterior boundary condition are both addressed. The proposed cross-point treatments rely on corner conditions developed for Padé-type HABCs. Two-dimensional numerical results with a nodal finite-element discretization are presented to validate the approach, including convergence studies with respect to the frequency, the mesh size and the number of subdomains. These results demonstrate the efficiency of the cross-point treatment for settings with regular partitions and homogeneous media. Numerical experiments with distorted partitions and smoothly varying heterogeneous media show the robustness of the approach for partitionings exhibiting less regularity.

1 Introduction

In this chapter, a domain decomposition approach with non-overlapping subdomains, which minimizes the data transfer between subdomains, is investigated. In the perspective of large-scale applications, the DDMs must be applicable with domain partitions having interior cross-points (where more than two subdomains meet) and exterior cross-points (that belong to both the exterior boundary and at least two subdomains). For non-overlapping DDMs, the cross-points require special care at both continuous and discrete levels.

In the present chapter and following [45], we consider an optimized non-overlapping DDM with a transmission condition based on a Padé-type HABC operator (Equation 1.18). Here, we address the question of the cross-point treatment when the HABC operator is used in the transmission condition, or when it is used in the exterior boundary condition, or both. For a complete definition of the local problems defined on

the subdomains, additional conditions are required at the interior corners of the subdomains. Following the recent contribution [147] on the treatment of corners with HABCs, we introduce suited corner conditions into the variational formulation of the subproblems and additional transmission variables at the cross-points. The obtained cross-point treatment accelerates the convergence of the method with a very limited overcost. When a HABC is used as an exterior condition, the cross-point treatment is actually necessary, since the method cannot converge without it. While the approach is designed for regular lattice-type domain partition (*i.e.* with only parallel and perpendicular interfaces), we will show that it actually also gives good results with distorted partitions. The main results of this chapter have been published in [148].

The chapter is organized as follows. In Section 2, we present the Helmholtz boundary-value problem with a HABC and its suitable corner treatment based on adding suitable boundary conditions. The nodal FEM formulation is given next. Section 3 introduces the optimized Schwarz DDM with high-order transmission boundary conditions. The cross-point treatment is detailed for two subdomains and then for the multi-subdomain decomposition. The FEM formulation is next stated and some technical aspects about the algorithmic procedure are discussed. In Section 4, we propose some numerical examples to analyze the behavior of the proposed method. Two model configurations with lattice-type partitions are considered for the convergence study. The sensitivity of the method to the tuning parameters of the HABC operator is studied, as well as the influence of the frequency, the mesh refinement and the number of subdomains. After, a numerical investigation with distorted partitions is proposed.

2 Helmholtz problem with HABC and corner treatment

To describe the method, we consider a two-dimensional bounded Helmholtz problem (1.4) introduced in Chapter 1 where Ω_{tot} is rectangular computational domain. Furthermore, we assume a constant positive wavenumber and a source term $f(\mathbf{x})$ such that in the numerical simulations, the source term is replaced with a scattering object to model scattering problems introduced in Section 2 of Chapter 1. The boundary of Γ_{ext} is split into four edges Γ_f with $f = 1 \dots 4$ such that the problem is written as

$$\begin{cases} \Delta u + k^2 u = -f & \text{in } \Omega_{\text{tot}}, \\ \partial_{\mathbf{n}} u - \mathcal{B}_f u = 0 & \text{on } \Gamma_f, \end{cases} \tag{2.1}$$

For each edge Γ_f, $\partial_{\mathbf{n}_f}$ is the (exterior) normal derivative and $\mathcal{B}_f$ is the boundary operator on f which takes into account the behavior of waves outside the computational domain, that we suppose to be the free-space here.

To simulate wave propagation in free-space, the simplest boundary condition is the Sommerfeld Absorbing Boundary Condition (ABC), which corresponds to using the impedance operator $\mathcal{B}_f = -\,\imath\, k$ on the edges. This condition is cheap and easy to use, but the accuracy is known to be poor. In this work, we consider HABCs [75, 121, 12, 147], which provide a better accuracy. To preserve the accuracy at the corners of the rectangle, a specific treatment based on compatibility relations derived in [147] is

used leading to very low spurious reflections at the boundary. For the HABC, a finite element implementation of the problem is described later. Let us remark that other alternative solutions could be considered for truncating the free-space, like for example by using the well-known PML. This approach will be studied in Chapter 3.

2.1 High-order absorbing boundary condition (HABC)

The Padé-type HABC is obtained by approximating an exact non-reflecting boundary condition derived for planar boundaries as detailed in Section 3.1 of Chapter 1. In Section 3.2 of Chapter 1, this operator is localized by using a Padé approximation of the square-root after a rotation of the branch-cut. For each face Γ_f, this leads to the HABC impedance operator

$$\mathcal{B}^{\text{Padé}}_{\text{HABC},f} = \iota\, k\alpha_f \left[1 + \frac{2}{M_f} \sum_{i=1}^{N_{\text{HABC},f}} c_{f,i} \frac{\alpha_f^2(c_{f,i}+1)}{\alpha_f^2(c_{f,i}+1) + \Delta_{\Gamma_f}/k^2}\right], \tag{2.2}$$

with $\alpha_f = e^{\iota\,\phi_f/2}$, $c_{f,i} = \tan^2(i\pi/M_f)$ and $M_f = 2N_{\text{HABC},f} + 1$. The accuracy of the Padé-type HABC depends on the number of terms $N_{\text{HABC},f}$ and the angle of rotation ϕ_f (see [147, 121] for further details). In particular, the parameters $N_{\text{HABC},f} = 0$ and $\phi_f = 0$ yield $\mathcal{B}_f = \iota\, k$, which corresponds to the basic ABC.

Let us remember that for the effective implementation of the HABC, $N_{\text{HABC},f}$ auxiliary fields $\{\varphi_{f,i}\}_{i=1\ldots N_{\text{HABC},f}}$ are defined on Γ_f, and the boundary condition is rewritten as

$$\partial_{\mathbf{n}_f} u - \mathcal{B}^{\text{Padé}}_{\text{HABC},f}\left(u, \{\varphi_{f,i}\}_{i=1\ldots N_{\text{HABC},f}}\right) = 0, \quad \text{on } \Gamma_f, \tag{2.3}$$

with the operator $\mathcal{B}^{\text{Padé}}_{\text{HABC},f}$ defined as

$$\mathcal{B}^{\text{Padé}}_{\text{HABC},f} u = \iota\, k\alpha_f \left[u + \frac{2}{M_f} \sum_{i=1}^{N_{\text{HABC},f}} c_{f,i}(u + \varphi_{f,i})\right]. \tag{2.4}$$

The additional fields are governed by the auxiliary equations

$$\Delta_{\Gamma_f}\, \varphi_i + k^2 \left[(\alpha_f^2 c_{f,i} + 1)\varphi_{f,i} + \alpha_f^2(c_{f,i}+1)u\right] = 0, \quad \text{on } \Gamma_f, \tag{2.5}$$

The operator $\mathcal{B}^{\text{Padé}}_{\text{HABC},f}$ will be simply refeered as $\mathcal{B}_f$ to simplify the expressions in the next of the chapter.

2.2 Corner treatment

When the HABC is prescribed on a boundary with corners, a specific treatment must be used at the corners. Because of the second-order spatial derivative in (2.5), boundary conditions must be added on the auxiliary fields at the extremities of each edge, which are at the corners of the domain. In a previous work [147], several strategies have been analyzed to preserve the accuracy of the solution at the corners. For configurations with right angles, the best approach consists in using a different set of auxiliary fields for each edge, with compatibility relations to couple the auxiliary fields of adjacent edges at the common corner.

Let us consider two adjacent edges Γ_f and $\Gamma_{f'}$ meeting at the corner $P_{f,f'} = \bar{\Gamma}_f \cap \bar{\Gamma}_{f'}$. Two sets of surface fields $\{\varphi_{f,i}\}_{i=1\ldots N_{\text{HABC},f}}$ and $\{\varphi_{f',i'}\}_{i'=1\ldots N_{\text{HABC},f'}}$ are defined on Γ_f and $\Gamma_{f'}$, respectively. Globally, a total of $N_{\text{HABC},f} + N_{\text{HABC},f'}$ boundary conditions must be written on these auxiliary fields at the corner $P_{f,f'}$. Following the approach detailed in [147], well-suited conditions are such that

$$\partial_{\mathbf{n}_{f'}} \varphi_{f,i} - \mathcal{B}_{f'}\left(\varphi_{f,i}, \{\psi_{ff',ii'}\}_{i'=1\ldots N_{\text{HABC},f'}}\right) = 0 \quad \text{on } P_{ff'} \quad (i = 1 \ldots N_{\text{HABC},f}) \tag{2.6}$$

$$\partial_{\mathbf{n}_f} \varphi_{f',i'} - \mathcal{B}_f\left(\varphi_{f',i'}, \{\psi_{f'f,i'i}\}_{i=1\ldots N_{\text{HABC},f}}\right) = 0 \quad \text{on } P_{ff'} \quad (i' = 1 \ldots N_{\text{HABC},f'}), \tag{2.7}$$

with $N_{\text{HABC},f} \times N_{\text{HABC},f'}$ corner auxiliary variables $\{\psi_{ff',ii'}\}_{i=1\ldots N_{\text{HABC},f}, i'=1\ldots N_{\text{HABC},f'}}$ defined as

$$\psi_{ff',ii'} = -\frac{\alpha_{f'}^2(c_{f',i'}+1)\varphi_{f,i} + \alpha_f^2(c_{f,i}+1)\varphi_{f',i'}}{\alpha_f^2 c_{f,i} + \alpha_{f'}^2 c_{f',i'} + 1} \quad \text{on } P_{ff'} \quad (i = 1 \ldots N_{\text{HABC},f}, i' = 1 \ldots N_{\text{HABC},f'}) \tag{2.8}$$

Let us remark that $\psi_{f'f,i'i} = \psi_{ff',ii'}$. In a nutshell, the HABC defined on the field u on one edge is also imposed on the auxiliary fields living on the adjacent edge at the common corner [147], with new auxiliary variables defined at the corner. For instance, the HABC set on $\Gamma_{f'}$ is also forced on the fields $\{\varphi_{f,i}\}_{i=1\ldots N_{\text{HABC},f}}$ at $P_{f,f'}$ (Equation (2.6)).

As a particular case, let us consider a configuration with a HABC given on Γ_f and the basic ABC set on the adjacent edge $\Gamma_{f'}$, *i.e.*

$$\partial_{\mathbf{n}_f} u - \mathcal{B}_f\left(u, \{\varphi_{f,i}\}_{i=1\ldots N_{\text{HABC},f}}\right) = 0 \quad \text{on } \Gamma_f, \tag{2.9}$$

$$\partial_{\mathbf{n}_{f'}} u - \iota\, k u = 0 \quad \text{on } \Gamma_{f'}. \tag{2.10}$$

At the corner $P_{f,f'}$, $N_{\text{HABC},f}$ boundary conditions must be imposed on the auxiliary fields living on Γ_f. Following the approach, the basic ABC must be prescribed

$$\partial_{\mathbf{n}_{f'}} \varphi_{f,i} - \iota\, k \varphi_{f,i} = 0 \quad \text{on } P_{f,f'} \quad (i = 1 \ldots N_{\text{HABC},f}) \tag{2.11}$$

which corresponds to equation (2.6) with $N_{\text{HABC},f'} = 0$ and $\phi_{f'} = 0$.

2.3 Finite element formulation

The problem finally consists in solving the main field u on the rectangular domain with a HABC on each edge by (2.3). Auxiliary fields defined on the edges are governed by 1D Helmholtz equations through (2.5) and are coupled at the corners by auxiliary relations (2.6)-(2.7) and auxiliary variables using (2.8). If the basic ABC is given for the main field on an edge, there is no auxiliary field on that edge, and the basic ABC is prescribed on the auxiliary variables living on the adjacent edges at the common corners.

In order to solve the problem with a finite element scheme, we straightforwardly adapt the bilinear form of the Helmholtz equation. The variational formulation of

the problem reads: find $u \in H^1(\Omega_{\text{tot}})$ and $\varphi_{f,i} \in H^1(\Gamma_f)$, for $f = 1 \dots 4$ and $i = 1 \dots N_{\text{HABC},f}$, such that

$$\int_{\Omega_{\text{tot}}} \mathbf{grad}\, u \cdot \mathbf{grad}\, \overline{v} - k^2 u \overline{v}\, \mathrm{d}\Omega_{\text{tot}} - \sum_{f=1}^{4} \int_{\Gamma_f} \mathcal{B}_f \left(u, \{\varphi_{f,i}\}_{i=1\dots N_{\text{HABC},f}}\right) \overline{v}\, \mathrm{d}\Gamma_f = \int_{\Omega_{\text{tot}}} f \overline{v}\, \mathrm{d}\Omega_{\text{tot}}, \quad (2.12)$$

holds for all v in $H^1(\Omega_{\text{tot}})$ and

$$\int_{\Gamma_f} \mathbf{grad}\, \varphi_{f,i} \cdot \mathbf{grad}\, \overline{\rho}_{f,i} - k^2 \left((\alpha_f^2 c_{f,i} + 1)\varphi_{f,i} + \alpha_f^2 (c_{f,i} + 1) u\right) \overline{\rho}_{f,i}\, \mathrm{d}\Gamma_f - \sum_{f'} \left[\mathcal{B}_{f'} \left(\varphi_{f,i}, \{\psi_{ff',ii'}\}_{i'=1\dots N_{\text{HABC},f'}}\right) \overline{\rho}_{f,i}\right]_{P_{ff'}} = 0, \quad (2.13)$$

holds for all $\overline{\rho}_{f,i}$ in $H^1(\Gamma_f)$. In the last equation, the index f' corresponds to any edge $\Gamma_{f'}$ adjacent to Γ_f, and the variables $\{\psi_{ff',ii'}\}_{i'=1\dots N_{\text{HABC},f'}}$ are defined on $P_{ff'}$ by (2.8). Standard Lagrange or arbitrary high-order hierarchical basis functions can then be used to discretize the problem.

3 Domain decomposition method with HABC and cross-point treatment

In this section, we present a non-overlapping DDM for lattice-type partitions of the domain. The convergence of the method is accelerated by using a Padé-type HABC as a transmission condition with a novel strategy to deal with cross-points. This strategy relies on the corner treatment derived for the HABCs in the previous section. The DDM and the cross-point strategy are presented in Sections 3 and 3.2, respectively.

Let us consider a partition of the rectangular domain Ω_{tot} into a grid of N rectangular non-overlapping subdomains Ω_n. The edges of each subdomain Ω_n are denoted by $\Gamma_{n,f}$ $(f = 1 \dots 4)$. Let us recall that each edge is either an *exterior boundary* if it belongs to the boundary of the global domain $(\Gamma_{n,f} \subset \partial\Omega_{\text{tot}})$, or an *interior boundary* if there is a neighboring subdomain beyond the edge $(\Sigma_{n,m} = \{\Gamma_{n,f} \not\subset \partial\Omega_{\text{tot}}\}$. In this decomposition, two kinds of points deserve attention: the *exterior cross-points* that belong to two subdomains and that touch the boundary of the global domain, and the *interior cross-points* belonging to four subdomains and that do not touch the boundary of the global domain. These edges and points are illustrated in Figure 2.1 for a 2×2 partition.

3.1 Optimized sub-structuring domain decomposition method

According to Equation 1.57 of Chapter 1, the global problem (2.1) is decomposed into local subproblems defined on the subdomains. The solution u_n for the subdomain Ω_n

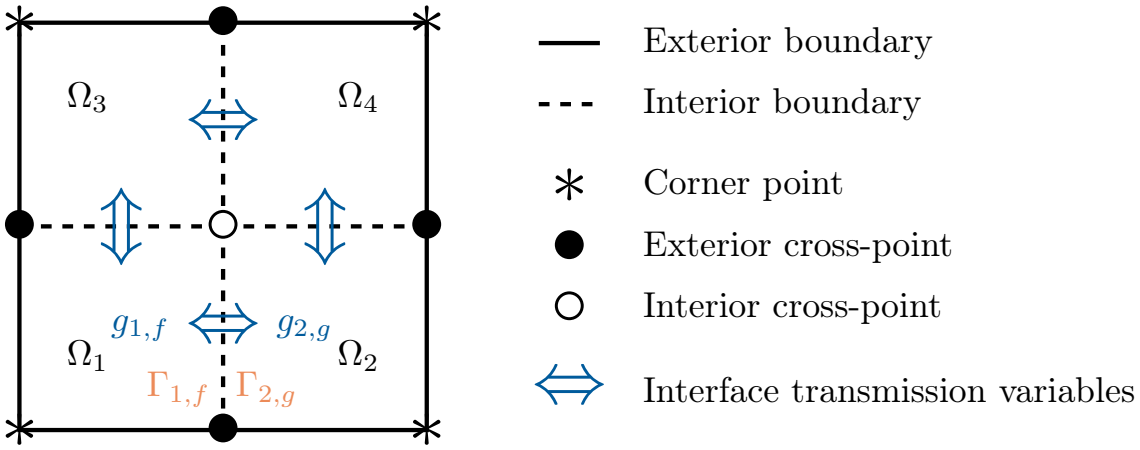


Figure 2.1: Terminology and transmission variables across the interface edges. In this example, the continuity of the local solution u_1 and u_2 on the interface edge $\Gamma_{1,f} = \Gamma_{2,g}$ is ensured thanks to the transmission variables $g_{1,f}$ and $g_{2,g}$.

is obtained by solving

$$\begin{cases} \Delta u_n + k^2 u_n = -f & \text{in } \Omega_n, \\ \partial_{\mathbf{n}_n} u_n - \mathcal{T}_{n,f} u_n = g_{n,f} & \text{on } \Gamma_{n,f}, \end{cases} \tag{2.14}$$

where $\mathcal{T}_{n,f}$ is an impedance operator, $g_{n,f}$ is a transmission variable which is set to zero if $\Gamma_{n,f}$ is an exterior boundary, while it depends on the local solution belonging to the neighboring subdomain if $\Gamma_{n,f}$ is an interior boundary and $\mathcal{T}_{n,f}$ is the transmission operator. Following the derivation of the transmission update equation of Section 6 of Chapter 1, the following interface update is obtained

$$g_{n,f} = -g_{m,g} + 2\mathcal{B}_{m,g} u_m. \tag{2.15}$$

Following [45], the transmission operators $\mathcal{T}_{n,f}$ for the transmission conditions are based on Padé-type HABCs, *i.e.* it is chosen to use the same transmission operator as the boundary operator defined in (2.4): $\mathcal{T}_{n,f} = \mathcal{B}_f$ such that in the following no distinction between the boundary operator $\mathcal{B}_f$ and the $\mathcal{T}_{n,f}$ (*i.e.* $\mathcal{B}_{n,f}$) is made. For each subdomain Ω_n, the local solution u_n verifies

$$\begin{cases} \Delta u_n + k^2 u_n = -f & \text{in } \Omega_n, \\ \partial_{\mathbf{n}_n} u_n - \mathcal{B}_{n,f}\left(u_n, \{\varphi_{n,f,i}\}_{i=1\ldots N_{\mathrm{HABC},n,f}}\right) = g_{n,f} & \text{on } \Gamma_{n,f}, \end{cases} \tag{2.16}$$

with the transmission variable $g_{n,f}$ that verifies

$$g_{n,m} := \begin{cases} 0 & \text{on } \Gamma_{n,\mathrm{ext}} \cap \partial\Omega_{\mathrm{tot}}, \\ -g_{m,g} + 2\mathcal{B}_{m,g}\left(u_m, \{\varphi_{m,g,j}\}_{j=1\ldots N_{\mathrm{HABC},m,g}}\right) & \text{on each } \Sigma_{n,m}, \end{cases} \tag{2.17}$$

The second equation of system (2.16) is a boundary condition if $\Gamma_{n,f}$ is an exterior boundary, or a transmission condition if $\Gamma_{n,f}$ is an interior boundary. In both cases,

if $N_{\mathrm{HABC},n,f} > 0$, auxiliary fields $\{\varphi_{n,f,i}\}_{i=1\ldots N_{\mathrm{HABC},n,f}}$ are defined on the edge, and are governed by

$$\Delta\,\varphi_{n,f,i} + k^2\left[(\alpha_{n,f}^2 c_{n,f,i}+1)\varphi_{n,f,i} + \alpha_{n,f}^2(c_{n,f,i}+1)u_n\right] = 0 \quad \text{on } \Gamma_{n,f}, \tag{2.18}$$

with $i = 1\ldots N_{\mathrm{HABC},n,f}$. The parameters of the transmission conditions used on both sides of an interface edge must be the same (*i.e.* $N_{\mathrm{HABC},n,f} = N_{\mathrm{HABC},m,g}$ and $\phi_{n,f} = \phi_{m,g}$, with $\Gamma_{n,f} = \Gamma_{m,g}$), since we assumed that the transmission operators are the same on a shared interface. For consistency, the same boundary condition must be prescribed on the boundary edges of the subdomains and on the corresponding edges of the global domain.

Boundary conditions must be set on the auxiliary fields at the extremities of the edges because of the second-order partial derivative in the governing equation (2.18). The extremities of an edge are at corners of a subdomain, and correspond to interior cross-points, exterior cross-points or corners of the global domain. The cross-point treatment, described in the next section, actually provides the missing boundary conditions at the cross-points.

3.2 Dealing with cross-points

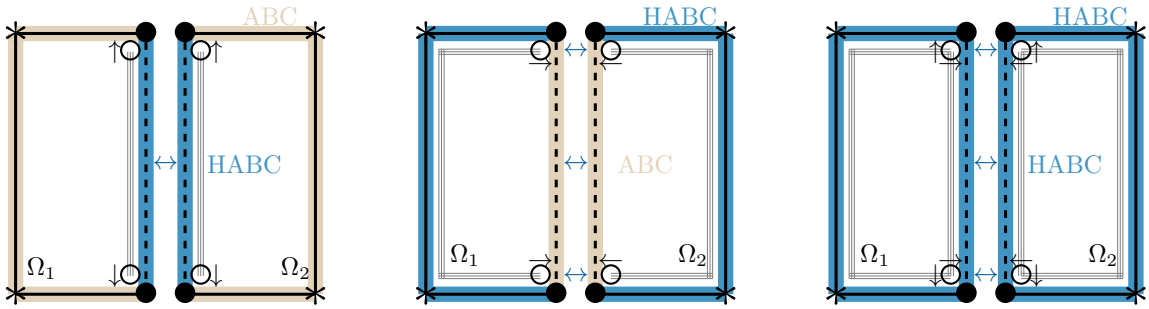
The cross-point treatment relies on the corner treatment described in Section 2. It is applied at the corners of the subdomains. Depending on the configuration, it provides boundary conditions or transmission conditions for the auxiliary fields at the cross-points. In the latter case, new transmission variables are defined at the cross-points.

3.2.1 Two-subdomain case

To describe the approach, we first consider a partition of the rectangular domain Ω_{tot} into two rectangular subdomains with an interface Γ and two exterior cross-points. Three configurations, represented on Figure 2.2, are studied: the basic ABC prescribed on $\partial\Omega$ with a HABC-based transmission condition on Γ (Configuration 1), a HABC on $\partial\Omega$ with a transmission condition based on the basic ABC on Γ (Configuration 2), and the HABC operator used both for $\partial\Omega$ and Γ (Configuration 3). Because the HABC is used on the exterior boundary and/or the interface, a specific treatment must be used at the exterior cross-points.

In the first configuration (Figure 2.2a), auxiliary fields are defined on both sides of the interface. These fields require boundary conditions at the extremities of the interface, which are corners of the subdomains. The basic ABC is set on the adjacent edges (*i.e.* the upper and lower boundary edges). Following the strategy of Section 2, the basic ABC is also given on the auxiliary fields at the boundary cross-points.

In the second configuration (Figure 2.2b), a HABC is given on each global edge Γ_f in the global problem, auxiliary fields are defined on each edge and the corner treatment is used. After the domain partition, a HABC is imposed on each boundary edge $\Gamma_{n,f}$ of each subdomain Ω_n, and a set of auxiliary fields is defined on each of these edges. For the consistency of the global problem, the parameters of the HABC on $\Gamma_{n,f}$ must be the same as the parameters of the HABC given on the global edge $\Gamma_n \supset \Gamma_{n,f}$. For a global edge Γ_f that has been divided by the partitioning (upper and



(a) Configuration 1 (b) Configuration 2 (c) Configuration 3

Figure 2.2: Three configurations for two-subdomain case. The exterior boundary condition is a basic ABC or a HABC, and the transmission condition is based on the HABC operator or the basic ABC operator. The thin gray lines illustrate the position of auxiliary fields. The black arrows indicate where boundary conditions are required for auxiliary fields. The blue arrows indicate transmission conditions on the edge or at the cross-points.

lower edges in Figure 2.2b), the continuity of the auxiliary fields must be enforced at the cross-points. As the ABC-based transmission condition is used on the main field on the interface, this transmission condition is also used on each auxiliary field at the boundary cross-points and auxiliary transmission variables are defined at these points.

The last configuration combines the difficulties. The exterior boundary condition and the transmission condition are based on HABCs (Figure 2.2c). The auxiliary fields living on every edge require boundary conditions at the boundary cross-points. To deal with this case, we recall that the operators used on the edges of each subdomain should approximate the DtN map of the free-space if the exterior medium relative to each subdomain is homogeneous. If the transmission variables are canceled, it corresponds to forcing a HABC on every edge. Therefore, we apply the corner treatment described in Section 2 to all the corners of the subdomains, which gives boundary conditions for the auxiliary fields. If the transmission variables are not canceled, the continuity of the fields $\{u_n\}_{n=1,2}$ is enforced at the interface thanks to the right-hand side of the second equation of system (2.16). For the auxiliary fields living on the boundary edges, the boundary conditions at the cross-point become transmission conditions by adding transmission variables in the right-hand sides, as for the second configuration. These transmission variables verify relations similar to equation (2.18) at the cross-points.

3.2.2 Multi-subdomains case

In the general case, the rectangular domain Ω_{tot} is partitioned into a grid of rectangular subdomains, with interior and exterior cross-points. The strategy relies on the following principles, which generalize the approaches used for the three previous configurations:

- The same transmission condition is used on both sides of each interface. The boundary condition used on each boundary is the same as the one prescribed on the corresponding edge of the global domain. In the domain partition, the HABC operators used on edges that are on a same line have the same parameters (*e.g.*

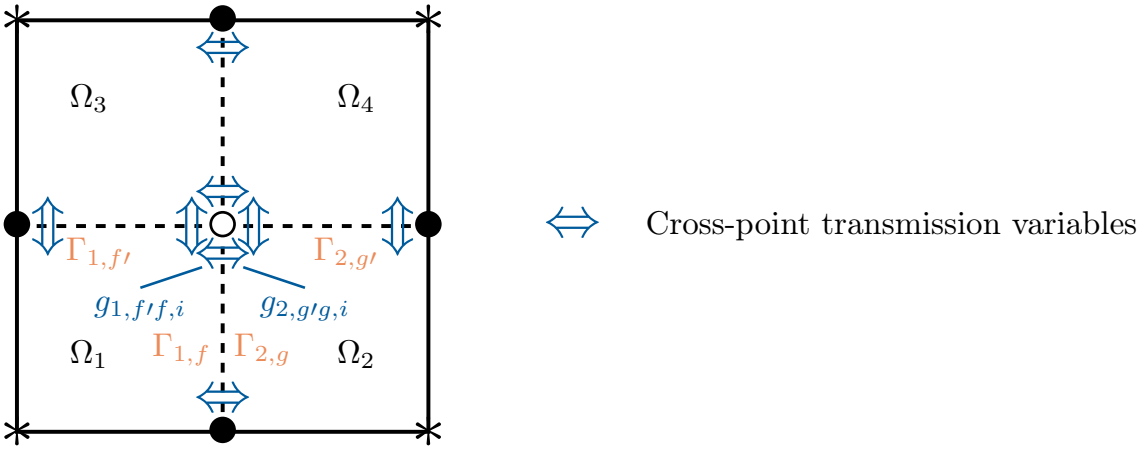


Figure 2.3: Transmission variables across the exterior and interior cross-points, if the HABC operator is used both in the exterior boundary condition and in the interface conditions. In the example, the continuity of the auxiliary fields $\varphi_{1,f',i}$ and $\varphi_{2,g',i}$ (defined on the aligned edges $\Gamma_{1,f'}$ and $\Gamma_{2,g'}$) at the interior cross-point $P_{1,ff'} = P_{2,gg'}$ is ensured thanks to the transmission variables $g_{1,f'f,i}$ and $g_{4,g'g,i}$. These variables verify equation (2.20).

in Figure 2.3: the top edges of Ω_1 and Ω_2, the top edges of Ω_3 and Ω_4, the left edges of Ω_1 and Ω_4, ...).

- If auxiliary fields are defined on an edge $\Gamma_{n,f}$ of a subdomain Ω_n, boundary conditions or transmission conditions must be set on these fields at the extremities of this edge (which can be interior cross-points, exterior cross-points, or corners of Ω). These conditions are given by the condition already used for u_n on the adjacent edges. If a transmission condition is used on u_n on an adjacent edge, transmission conditions are considered on the auxiliary fields at the cross-point, and new transmission variables are introduced.

- The corner treatment described in Section 2 is used at the corners of each subdomain, which gives boundary conditions to the auxiliary fields living on the edges. At the cross-points, these conditions can become transmission conditions by adding transmission variables in the right-hand sides, which are similar to Equation (2.17).

Following these principles, the description of the problem with domain decomposition can be completed.

For each subdomain Ω_n, the local solution u_n verifies Equations (2.16). For each edge $\Gamma_{n,f}$, the transmission variable $g_{n,f}$ satisfies Equation (2.17). Each auxiliary field $\varphi_{n,f,i}$ $(i = 1 \dots N_{\mathrm{HABC},n,f})$ defined on a exterior or interior boundary $\Gamma_{n,f}$ is such that

$$\begin{cases} \Delta\,\varphi_{n,f,i} + k^2\left((\alpha_{n,f}^2 c_{n,f,i}+1)\varphi_{n,f,i} + \alpha_{n,f}^2(c_{n,f,i}+1)u_n\right) = 0 & \text{on } \Gamma_{n,f}, \\ \quad \partial_{\mathbf{n}_{n,f'}}\varphi_{n,f,i} - \mathcal{B}_{n,f'}\left(\varphi_{n,f,i}, \{\psi_{n,ff',ii'}\}_{i'=1\dots N_{\mathrm{HABC},n,f'}}\right) = g_{n,ff',i} & \text{on each } P_{n,ff'}, \end{cases} \tag{2.19}$$

with the transmission variable $g_{n,ff',i}$

$$g_{n,ff',i} = \begin{cases} 0 & \text{on } \Gamma_{n,f'} \subset \partial\Omega_{\text{tot}}, \\ -g_{n,gg',i} + 2\mathcal{B}_{m,g}\left(\varphi_{m,g,i}, \{\psi_{m,gg',ii'}\}_{i'=1\dots N_{\text{HABC},m,g'}}\right) & \text{on } \Sigma_{n,m}. \end{cases} \tag{2.20}$$

In these relations, $\Gamma_{n,f'}$ is any edge that is adjacent to $\Gamma_{n,f}$, and $P_{n,ff'} = \Gamma_{n,f} \cap \Gamma_{n,f'}$ is the corner that is shared by these edges. The second equation of system (2.19) is a boundary condition if $\Gamma_{n,f'}$ is an exterior boundary, or a transmission condition if $\Gamma_{n,f'}$ is an interface boundary, also called an interface $\Sigma_{n,m}$ between Ω_n and Ω_m. The transmission variable is set to zero in the former case, and it depends on the solution of the other side of $\Gamma_{n,f'}$ in the latter case. The variables $\psi_{n,ff',ii'}$ are defined using Equation (2.8).

In Equation (2.20), the indices are chosen in such a way that Ω_m is the neighboring subdomain on the other side of $\Gamma_{n,f'}$, the edge $\Gamma_{m,g'}$ is shared by the subdomains (*i.e.* $\Gamma_{n,f'} = \Gamma_{m,g'}$), and the edge $\Gamma_{m,g}$ is aligned with $\Gamma_{n,f}$ (*i.e.* $f = g$), as illustrated in Figure 2.3. The variable $g_{m,gg',i}$ is used in a transmission condition for an auxiliary field $\varphi_{m,g,i}$ living on $\Gamma_{m,g}$. Therefore, the transmission conditions enforce the continuity of the auxiliary fields $\varphi_{n,f,i}$ and $\varphi_{m,g,i}$, which live on edges that are on the same line. Let us note that, since the HABC parameters are the same for edges that are aligned, $N_{\text{HABC},n,f} = N_{\text{HABC},m,g}$ and $\phi_{n,f} = \phi_{m,g}$.

3.3 Finite element scheme and algorithmic procedure

Each step of the DDM iterative procedure consists in solving a local subproblem on each subdomain, and updating the transmission variables both on the interface edges and at the cross-points. The numerical solution of the subproblems is performed with a standard nodal finite element scheme built on a conformal mesh made of triangles or quadrangles. For each subdomain Ω_n, the variational formulation of the subproblem reads: Find $u_n \in H^1(\Omega_n)$ and $\varphi_{n,f,i} \in H^1(\Gamma_{n,f})$, with $i = 1 \dots N_{\text{HABC},n,f}$ and $f = 1 \dots 4$, such that

$$\begin{aligned} \int_{\Omega_n} \mathbf{grad}\, u_n \cdot \mathbf{grad}\, \overline{v}_n - k^2 u_n \overline{v}_n \, \mathrm{d}\Omega_n - \sum_{f=1}^{4} \int_{\Gamma_{n,f}} \mathcal{B}_{n,f}\left(u_n, \{\varphi_{n,f,i}\}_{i=1\dots N_{\text{HABC},n,f}}\right) \overline{v}_n \, \mathrm{d}\Gamma_{n,f} \\ = \int_{\Omega_n} f \overline{v}_n \, \mathrm{d}\Omega_n + \sum_{f=1}^{4} \int_{\Gamma_{n,f}} g_{n,f} \overline{v}_n \, \mathrm{d}\Gamma_{n,f} \end{aligned} \tag{2.21}$$

holds for all v_n in $H^1(\Omega_n)$, and

$$\begin{aligned} \int_{\Gamma_{n,f}} \mathbf{grad}\, \varphi_{n,f,i} \cdot \mathbf{grad}\, \overline{\rho}_{n,f} - k^2 \left((\alpha_{n,f}^2 c_{n,f,i} + 1)\varphi_{n,f,i} + \alpha_{n,f}^2 (c_{n,f,i} + 1) u_n\right) \overline{\rho}_{n,f} \, \mathrm{d}\Gamma_{n,f} \\ - \sum_{f'} \left[\mathcal{B}_{n,f'}\left(\varphi_{n,f,i}, \{\psi_{n,ff',ii'}\}_{i'=1\dots N_{\text{HABC},n,f'}}\right) \overline{\rho}_{n,f}\right]_{P_{n,ff'}} = \sum_{f'} \left[g_{n,ff',i} \overline{\rho}_{n,f}\right]_{P_{n,ff'}}, \end{aligned} \tag{2.22}$$

holds for all $\rho_{n,f}$ in $H^1(\Gamma_{n,f})$. In the last equation, the index f' corresponds to any edge $\Gamma_{n,f'}$ adjacent to $\Gamma_{n,f}$, and $P_{n,ff'} = \Gamma_{n,f} \cap \Gamma_{n,f'}$ is the shared corner. The variables $\psi_{n,ff',ii'}$ are defined using (2.8). This variational formulation is an extension of the one used in [45] (see Equation (62) in that reference). In that work, there is only one set of auxiliary fields and equations on the subdomain boundary $\partial\Omega_n$. Here, there is one set for each edge $\Gamma_{n,f}$ of the subdomain, and new terms appear in (2.22) to deal with the corners of the subdomain.

In the DDM iterative procedure, the transmission variables computed at an iteration ℓ are used in the right-hand side of Equations (2.21)-(2.22) to compute the local fields of the iteration $\ell+1$. The transmission variables are then updated using Equations (2.17)-(2.20). Therefore, at each iteration, the interface transmission variables are computed using

$$g_{n,f}^{(\ell+1)} = -g_{m,g}^{(\ell)} + 2\mathcal{B}_{m,g}\left(u_m^{(\ell+1)}, \{\varphi_{m,g,j}^{(\ell+1)}\}_{j=1\ldots N_{\mathrm{HABC},m,g}}\right), \tag{2.23}$$

for each interface $\Gamma_{m,f} \not\subset \partial\Omega_{\mathrm{tot}}$. Similarly, the cross-point transmission variables are updated through

$$g_{n,ff',i}^{(\ell+1)} = -g_{m,gg',i}^{(\ell)} + 2\mathcal{B}_{m,g}\left(\varphi_{m,g,i}^{(\ell+1)}, \{\psi_{m,gg',ii'}^{(\ell+1)}\}_{i'=1\ldots N_{\mathrm{HABC},m,g'}}\right), \tag{2.24}$$

at each cross-point $P_{n,ff'}$, with $\Gamma_{n,f'} \not\subset \partial\Omega_{\mathrm{tot}}$.

4 Numerical results

This section reports some finite element simulations to study the HABC-based domain decomposition method with cross-point treatment. After a description of three benchmarks in Section 4.1, we analyze the convergence history (Section 4.2), the sensitivity to the HABC parameters (Section 4.3) and the influence of the wavenumber, the mesh density and the number of subdomains on the convergence rate (Section 4.4). Finally, configurations with distorted domains partitions is investigated in Section 4.5.

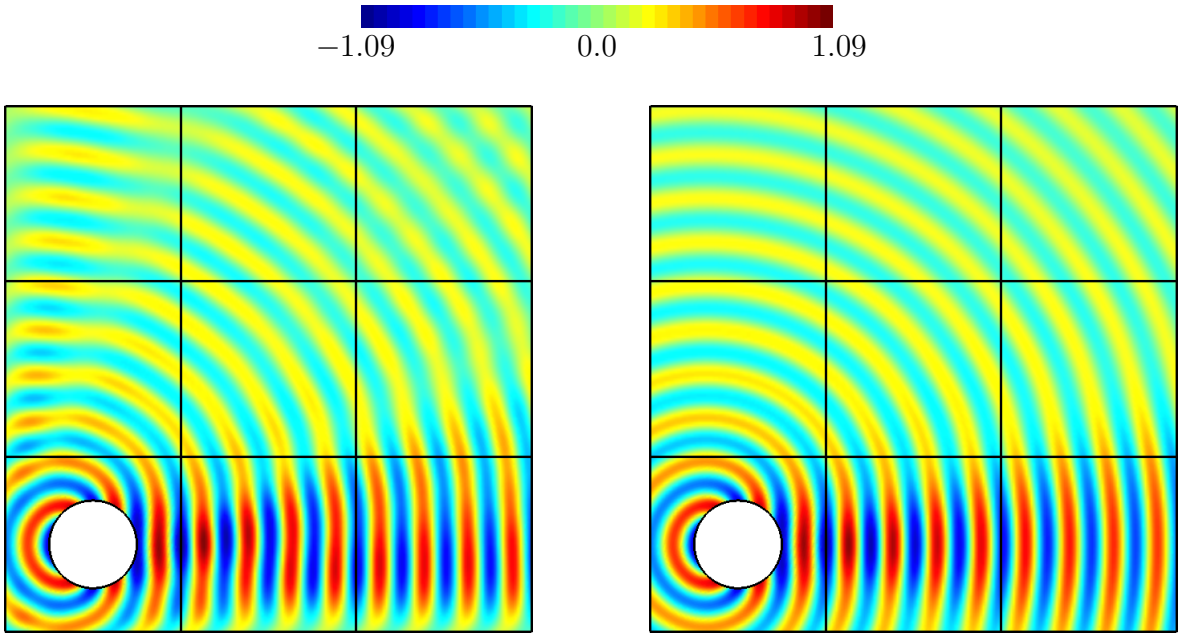
4.1 Description of the benchmarks

The reference benchmark used through this section is the scattering of an incident plane wave $u_{\mathrm{inc}}(\mathbf{x}) = e^{\iota\, kx}$ by a sound-soft circular scatterer. For a circle of radius R centered at the origin, the scattered field is given by

$$u_{\mathrm{ref}}(r,\theta) = -\sum_{i=0}^{\infty} \epsilon_i\, \iota^i\, \frac{J_i(kR)}{H_i^{(1)}(kR)} H_i^{(1)}(kr)\cos(i\theta) \qquad r \geq R, \tag{2.25}$$

where (r,θ) are the polar coordinates, J_i is the i^{th}-order Bessel's function, $H_1^{(1)}$ is the 1^{th}-order first-kind Hankel function, and ϵ_i is the Neumann function which is equal to 1 for $m = 0$ and 2 otherwise.

Two configurations are considered. For the first configuration (Figure 2.4a), the finite element simulations are performed on the square computational domain $[0,6] \times [0,6]$ with checkerboard partitions. The scatterer is the circle of radius $R_1 = 0.5$ centered at the origin. The basic ABC, *i.e.* $\partial_n u - \iota\, ku = 0$, is set on the exterior



(a) Configuration 1 (b) Configuration 2

Figure 2.4: Scattering benchmarks: real part of the scattered field of the reference numerical solution for the three configurations with $k = 4\pi$. The basic ABC and a HABC are set on the exterior border, respectively.

boundary of the domain. Because this boundary condition is a rather inaccurate non-reflecting boundary treatment, the numerical solution contains both the scattered field and spurious waves reflected on the exterior boundary. For the second configuration (Figure 2.4b), the HABC is used on the edges of the square domain with the suited treatment at the corners. The HABC parameters $N_{\text{HABC}} = 6$ and $\phi = 0.3\pi$ have been selected to avoid any visible modeling error in the numerical solution (*i.e.* the numerical error due to the finite element scheme is significantly larger than the modeling error due to the approximate boundary condition, see [147]).

For all the configurations, the finite element scheme is based on meshes made of straight triangular elements and second-order hierarchical polynomial basis functions. The Dirichlet boundary condition $u = -u_{\text{inc}}$ is set at the boundary of the (sound-soft) scatterer. By default, the wavenumber is $k = 4\pi$ and the characteristic number of vertices per wavelength is $\eta_h = 15$. The meshes of the square domain is made of 74 370 triangles. For the two configurations, the relative L^2-errors of the finite element solutions compared to the reference solution (2.25) are 1.89×10^{-1} and 4.61×10^{-4}, respectively.

In contrast to [148], where the numerical results were obtained with GetDDM [182], the numerical results presented hereafter were obtained with GmshDDM (Chapter 6). The related implementation of the test cases is available at the following address: `https://gitlab.onelab.info/gmsh/ddm/-/tree/master/examples/helmholtz/crossPoints`.

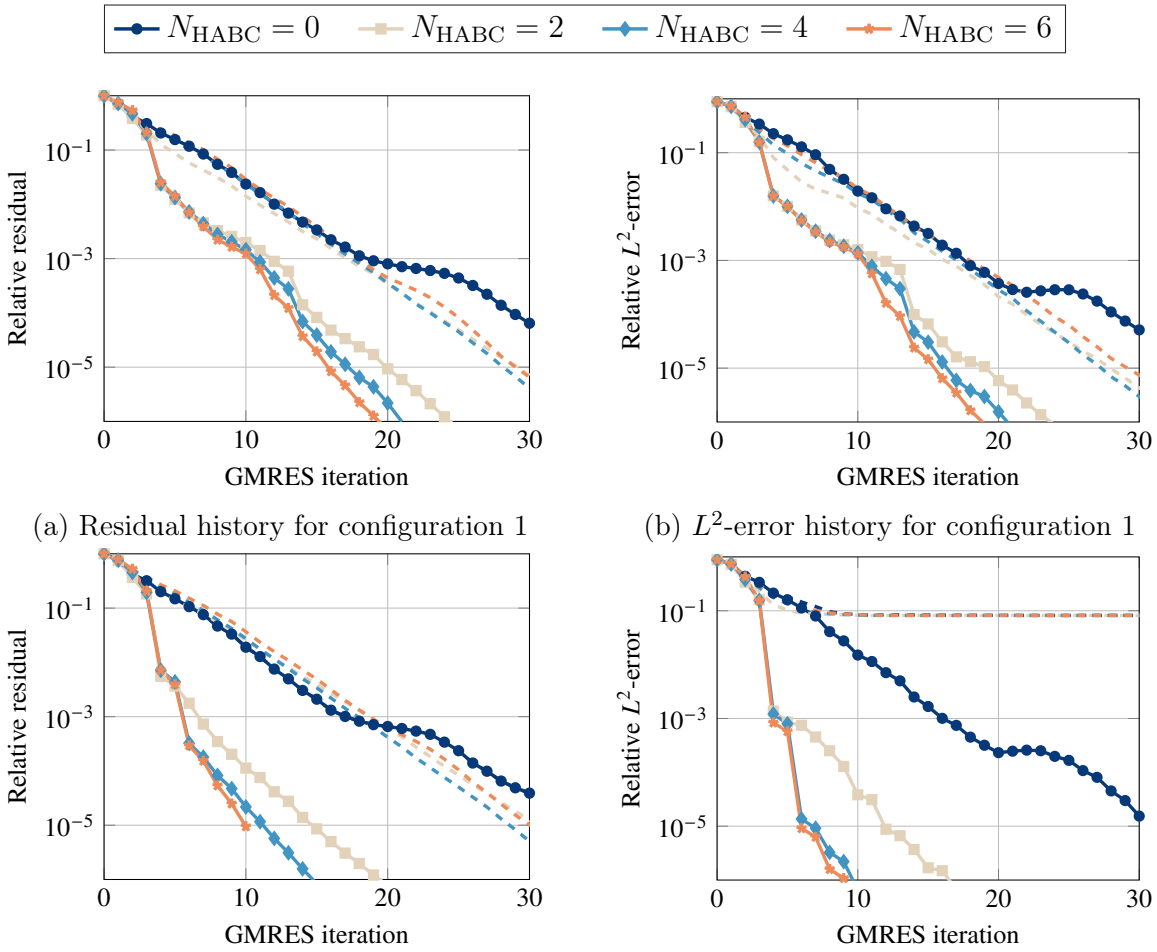
4.2 Convergence analysis

We begin by analyzing the convergence of the DDM procedure with cross-point treatment for both configurations. The relative L^2-errors and the relative residuals are plotted as functions of the number of the GMRES iterations in Figure 2.5 for both configurations. The L^2-error is calculated by comparing the solution obtained in each subdomain to the reference numerical solution computed on the same mesh without domain decomposition. In every case, HABC-based transmission conditions with different numbers of auxiliary fields are tested ($N_{\mathrm{HABC}} = 0$, 2, 4 and 6 with $\phi = 0.3\pi$). The effect of the cross-point treatment is analyzed by keeping or removing the corresponding terms in the finite element scheme. The latter case consists in setting a homogeneous Neumann boundary condition on the auxiliary fields at the cross-points. On all the figures, the dotted lines are associated to results without the cross-point treatment.

For the first configuration (*i.e.* square domain with basic ABC), the relative residual and the relative error have the same order of magnitude in all the cases (Figures 2.5a-2.5b) and decrease during the iterations. Using the cross-point treatment clearly accelerates the convergence, especially for transmission conditions with large values of N_{HABC}. The number of iterations to reach a relative error of magnitude 10^{-6} is reduced by 20% to 40% thanks to the treatment. When the cross-point treatment is enabled, the decay of residual and error can be accelerated further, up to a certain point, by taking a number of auxiliary fields N_{HABC} sufficiently large. Taking higher values for N_{HABC} does not change the results, while, without the cross-point treatment, increasing N_{HABC} slightly slows down the decays.

The good results obtained with the cross-point treatment and N_{HABC} sufficiently large can be interpreted by looking at the numerical solution after each iteration (Figure 2.6). At the initialization, the right-hand side term of the iteration system is computed by solving each subproblem with source terms only (see Section 3). Here, only the subdomain containing the scattering disk has a source, and then non-zero solution (Figure 2.6a). The numerical solution in this subdomain is already rather accurate since the transmission condition acts as a HABC, and the cross-point treatment behaves as the suited corner treatment. Since there is neither source nor very significant reflected waves generated outside the subdomain, the HABC and the corner treatment constitute a very good boundary treatment for the subdomain. During the iterations, the signal is propagated from subdomain to subdomain. At the fourth iteration, the signal reaches the last subdomain. This coincides with a sharp reduction of both the residual and error by an order one in magnitude.

For the second configuration (*i.e.* square domain with HABC), the impact of the cross-point treatment is more important. When the cross-point treatment is not enabled, the residuals decrease with the iterations (Figure 2.5c), but the relative errors reach a plateau and stagnate at 10^{-1} (Figure 2.5d). This can be explained by noting the only difference with the previous configuration: a HABC is prescribed on the exterior boundary instead of a basic ABC, and auxiliary fields defined on the edges of the domain Ω_{tot}. Without the cross-point treatment, the derivative of these auxiliary fields is set to zero at the boundary cross-points. Then, the problem with domain decomposition is not compatible with the original problem, and the iterative schemes converge towards a wrong solution. To fix this, the continuity of the auxiliary fields living on the



(a) Residual history for configuration 1

(b) L^2-error history for configuration 1

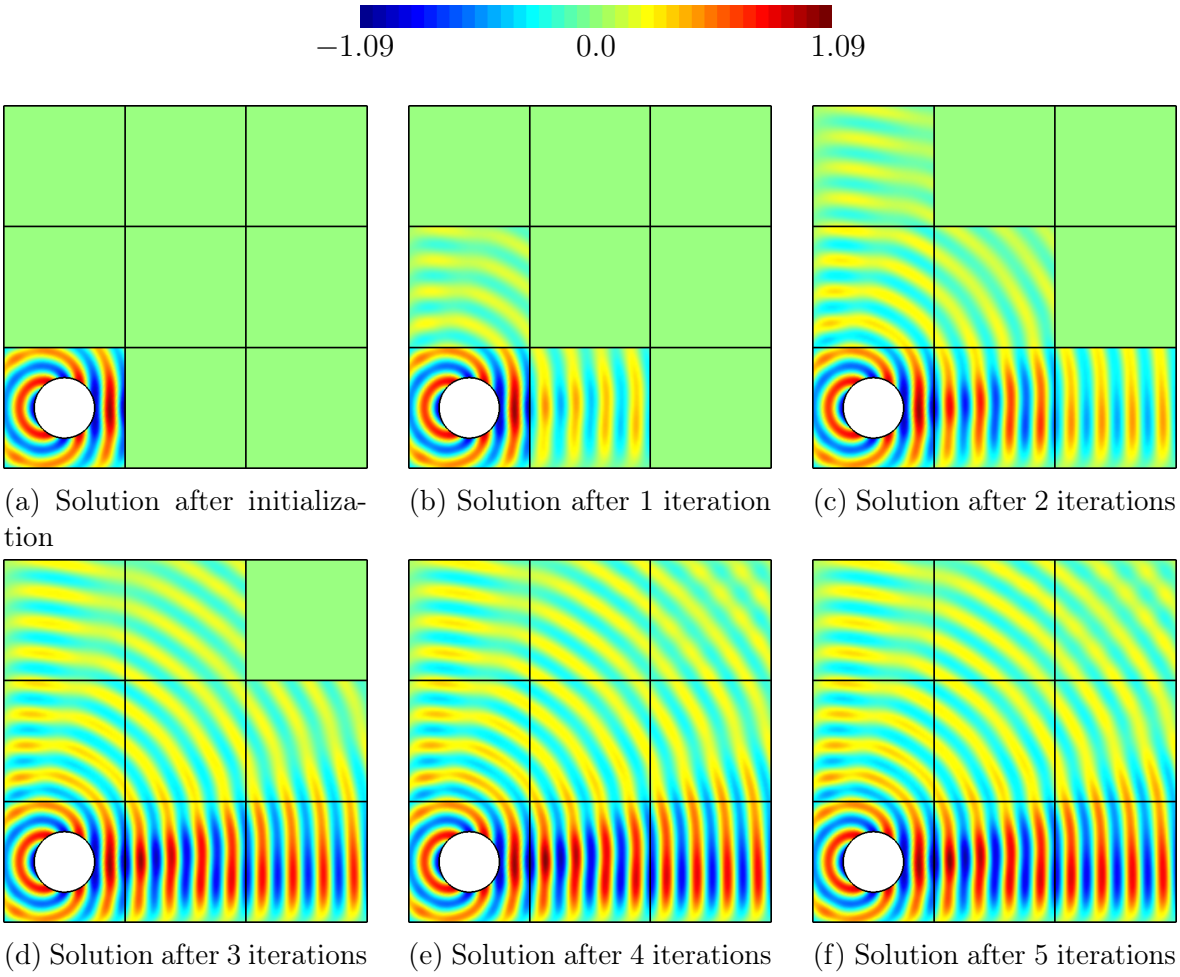
(c) Residual history for configuration 2

(d) L^2-error history for configuration 2

Figure 2.5: Evolution of relative residual (left) and relative L^2-error (right) in the course of the GMRES iterations for the two configurations represented in Figure 2.4. HABC-based transmission conditions with $N_{\text{HABC}} = 0, 2, 4, 6$ auxiliary fields and $\phi = 0.3\pi$ are used. The dotted lines correspond to the results obtained when the cross-point treatment is not used. Handling the cross-point procedure is represented by continuous lines.

exterior boundaries must be enforced at the exterior cross-points. With the cross-point treatment, transmission conditions are set at the exterior cross-points, and the error decays together with the residual, as it should be. It is worth to note that the absence of cross-point treatment in the first configuration does not break the convergence of the error because no auxiliary fields are defined on the boundary edges. In that case, the problem with domain partition is compatible with the original problem.

When comparing the results of the two first configurations for high values of N_{HABC}, we observe that the error decays faster for the second configuration, especially between the iterations 3 and 4, where the error drops by at least 3 orders of magnitudes. This tremendous result is likely due to the specificity of the benchmark: the exact scattering solution verifies the exact free-space boundary condition on the boundary and the interfaces. Since the HABC is used both as exterior boundary condition



(a) Solution after initialization (b) Solution after 1 iteration (c) Solution after 2 iterations

(d) Solution after 3 iterations (e) Solution after 4 iterations (f) Solution after 5 iterations

Figure 2.6: Evolution of the solution during the GMRES iterations for configuration 1 and the HABC-based transmission condition with $N_{\text{HABC}} = 4$ and $\phi = 0.3\pi$. The first picture is obtained after initialization of the right-hand side of the transmission system.

and transmission condition, the exact behavior of the solution is captured with a few iterations. By contrast, when the basic ABC is used as exterior BC, small waves reflected on the ABC must travel towards the subdomains.

4.3 Sensitivity to the HABC parameters

The efficiency of the transmission condition depends on the number N_{HABC} of auxiliary fields and the rotating angle ϕ. To study the sensitivity of the convergence to these parameters, we perform the DDM procedure with several values of N_{HABC} and ϕ for the three configurations. The number of GMRES iterations to reach the relative residual 10^{-6} is reported in Figure 2.1 for the first configuration.

For any given ϕ, increasing the number of auxiliary fields N_{HABC} accelerates the convergence, up to a certain limit, as already mentioned in the previous section. The only exception is for $\phi = 0$. Nevertheless, increasing N_{HABC} leads to a higher computational cost and the amount of data to exchange at the cross-points. It is then

Rotating angle (ϕ)	Number of auxiliary fields (N_{HABC})										
	0	1	2	3	4	5	6	7	8	9	10
0π	83	101	204	254	315	374	537	415	501	556	559
	74	47	40	36	34	32	30	30	29	27	26
0.1π	67	37	32	28	26	25	23	22	22	21	20
	62	33	29	26	24	23	22	21	21	20	20
0.2π	57	31	27	25	23	22	21	20	20	20	20
	54	30	26	23	22	21	20	20	20	19	19
0.3π	53	29	25	23	21	21	20	19	19	19	19
	54	28	24	22	21	20	19	19	19	19	19
0.4π	56	28	24	22	21	20	19	19	19	19	19
	58	28	23	22	21	20	19	19	19	19	19
0.5π	61	28	23	22	20	20	19	19	19	19	19

Table 2.1: Number of GMRES iterations to reach the relative residual 10^{-6} in configuration 1 for different values of the number of auxiliary fields N_{HABC} and rotating angle ϕ. For each column (*i.e.* each value of N_{HABC}), cells in blue correspond to the minimal number of iterations, while cells in orange are up to 10% from the minimal number of iterations.

advantageous to take the smallest N_{HABC} yielding the best convergence. For practical applications, the optimal N_{HABC} would likely depend on the configuration.

The selection of the parameter ϕ is an important matter, because it accelerates the convergence of the iterative process at no additional cost. We observe first that the Padé case ($\phi = 0$) gives the worst result in all the cases, and it should be avoided. The optimal value for ϕ, represented for each N_{HABC} by blue cells in Figure 2.1, depends on the number N_{HABC} of auxiliary fields. This can make the parameter selection rather tricky. Fortunately, the number of iterations is not very sensitive to ϕ as soon as ϕ is sufficiently large (*i.e.* larger than $\pi/4$ here). The range of the nearly-optimal values of ϕ, represented by the orange zone in Figure 2.1, is indeed rather wide.

The results for the other configurations lead to similar conclusions. They are not reported here for the sake of conciseness. In the remainder of the chapter, we always use $\phi = 0.3\pi$, which is a nearly-optimal value for all the configurations.

4.4 Influence of the wavenumber, the mesh density and the number of subdomains

In this section, we study the sensitivity of the method with respect to the wavenumber k, the characteristic number of vertices per wavelength η_h and the number of subdomains. High frequency simulations are challenging because they require fine meshes with high mesh densities to avoid the pollution effect. The efficiency of the method for large values of k and η_h is therefore an important issue.

Figure 2.7 shows the number of iterations to reach the relative residual 10^{-6} with respect to k and η_h for the various configurations and several values of N_{HABC}. For configuration 1, the dotted lines correspond to cases where the cross-point treatment is not used. As discussed in Section 4.2, the compatibility is not ensured for configuration

2 if the cross-point treatment is not used.

We first analyze the influence of k on the convergence. For $N_{\text{HABC}} = 0$, the number of iterations increases with respect to k in all the cases (Figures 2.7a and 2.7c). The increase is very slow for the second configuration. For higher values of N_{HABC}, the convergence does not change significantly with k when the cross-point treatment is used. As already observed, higher values of N_{HABC} accelerate the convergence, and the convergence is slower if the cross-point treatment is not used.

For the first configuration, the number of iterations increases with the characteristic number of vertices per wavelength η_h for all the values of N_{HABC} (Figure 2.7b). Fortunately, the number of iterations can be kept constant when increasing η_h by taking N_{HABC} larger: the number of iterations then remains approximately 20 for the first configuration. Therefore, a convergence independent of the mesh density can be achieved provided that N_{HABC} is sufficiently large. This was already observed in [45] on benchmarks without cross-points treatment. The results are slightly different for the second configuration (Figure 2.7d): the number of iterations increases very slowly for $N_{\text{HABC}} = 0$ and 2, while it decreases until a plateau for $N_{\text{HABC}} = 6$. This is likely due to the fact that the numerical solution is closer to the exact free-space scattering solution, and that the HABC-based transmission condition is perfectly suited to this specific case.

These results then indicate that the method is well-adapted to high-frequency problems with high density meshes, provided that N_{HABC} is sufficiently large.

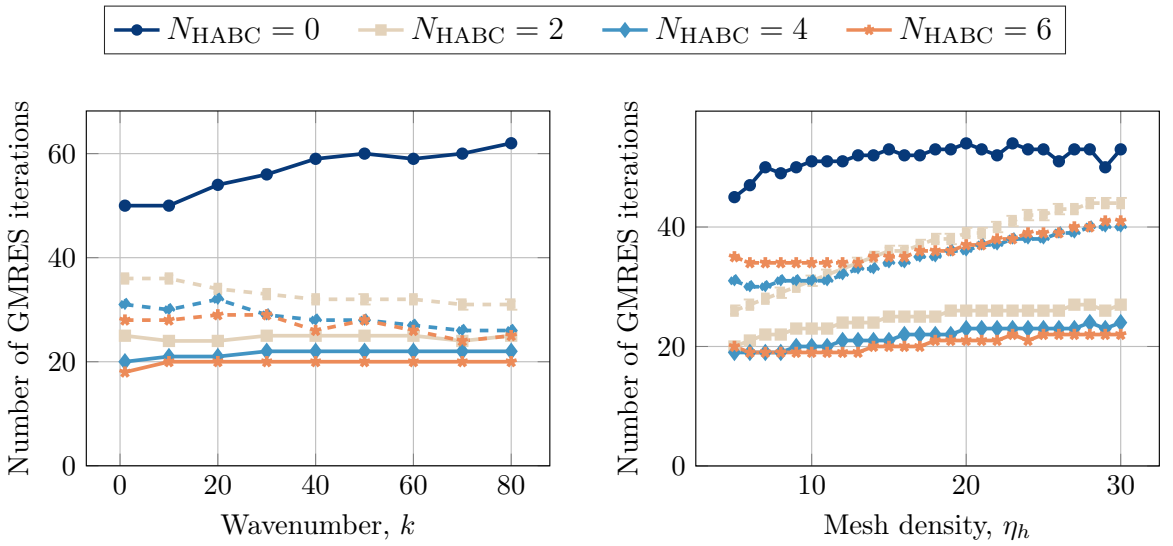
Figure 2.8 shows the evolution of the number of GMRES iterations with respect to the number of subdomains for both configuration. The simulations have been performed with increased numbers of subdomains in the x- and y-directions for the square domain (resp. $N_{\text{dom},x}$ and $N_{\text{dom},y}$). The size of the domains increases with the number of subdomains: the square domain is $[0, 2N_{\text{dom},x}] \times [0, 2N_{\text{dom},y}]$.

The scaling behavior of the method is as expected: the number of iterations increases linearly with the number of subdomains in each direction (Figures 2.8a and 2.8b). Indeed, since the transmission of propagating waves from subdomain to subdomain is local with the transmission conditions, a larger number of iterations is required to allow the propagation of waves across a larger number of subdomains. Preconditioning techniques based on sweeps (*e.g.* [76, 177, 180, 188, 189]) and coarse spaces (*e.g.* [39, 61, 83, 14]) allow for global transmissions of information between the subdomains with improved convergences. The combination of our approach with preconditioning techniques is currently under investigation.

4.5 Experiments with non-right angles

The proposed DDM is *a priori* suited only to wave propagation in lattice-type domain partitions with right angles. Indeed, the compatibility relations used in the cross-point treatment are derived for corners with right angles (see Section 2). Nevertheless, the HABC can be used as a good approximation with smoothly-varying heterogeneous media, since it can represent locally the transmission of waves at the interface (see *e.g.* [146]). The compatibility relations derived for right-angle corners can be used as an approximate treatment with non-right angles [147].

To analyze the method for partitions with non-right angles, we consider the scatter-



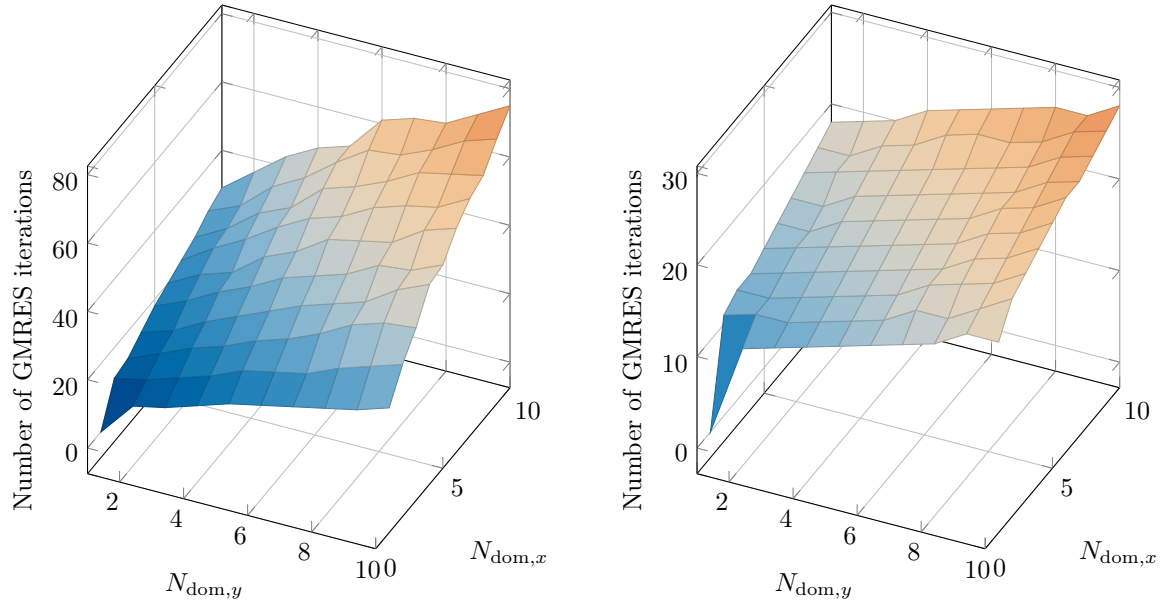
(a) Number of iterations *vs* k for configuration 1

(b) Number of iterations *vs* η_h for configuration 1

(c) Number of iterations *vs* k for configuration 2

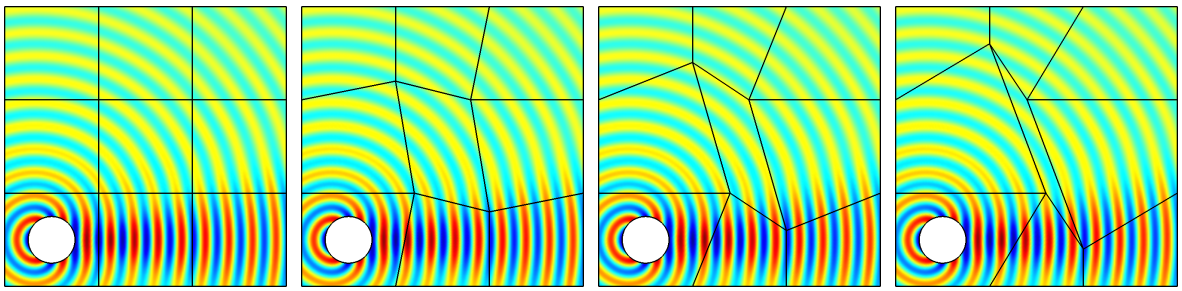
(d) Number of iterations *vs* η_h for configuration 2

Figure 2.7: Number of GMRES iterations to reach the relative residual 10^{-6} as a function of the wavenumber k with a fixed number of vertices per wavelength $\eta_h = 15$ (left) or as a function of η_h with a fixed wavenumber $k = 4\pi$ (right) to assert the scaling of the solution with k and η_h.



(a) Scaling for configuration 1 (b) Scaling for configuration 2

Figure 2.8: Number of GMRES iterations to reach the relative residual 10^{-6} for different number of subdomains to assert the scaling of the procedure. The size of the main domain increases with the number of subdomains in the x- and y-directions. Both configuration are run with $N_{\text{HABC}} = 6$.



(a) Shift of 0% (b) Shift of 20% (c) Shift of 40% (d) Shift of 60%

Figure 2.9: Snapshot of the distorted partitions for the square domain.

ing benchmark and the two configurations described in Section 4.1. The partitions are deformed by moving the cross-points, which create acute and obtuse angles, as shown in Figure 2.9. The points are shifted for both configurations (by a factor of 20%, 40% and 60%). In every case, HABC-based transmission conditions with different numbers of auxiliary fields are tested ($N_{\text{HABC}} = 0$, 2, 4 and 6 with $\phi = 0.3\pi$). The effect of the cross-point treatment is analyzed by keeping or removing the corresponding terms in the finite element scheme. The terms implemented for the right-angle case are used without modification for non-right angles.

Table 2.2 shows the number of GMRES iterations to reach the relative residual 10^{-6} for each case. The relative L^2-error (not shown for the sake of shortness) is always close to 10^{-6}, except for the second configuration (*i.e.* square domain with a HABC on the exterior border) without cross-point treatment. As discussed in Section 4.2, the

		Configuration 1				Configuration 2			
		0%	20%	40%	60%	0%	20%	40%	60%
No cross-point treatment	$N_{\text{HABC}} = 0$	53	63	68	80	75	78	78	85
	$N_{\text{HABC}} = 2$	35	38	39	42	37	39	40	42
	$N_{\text{HABC}} = 4$	34	33	36	37	34	33	36	37
	$N_{\text{HABC}} = 6$	35	30	32	33	36	30	32	33
With cross-point treatment	$N_{\text{HABC}} = 0$	53	63	68	80	52	64	69	80
	$N_{\text{HABC}} = 2$	25	28	30	34	20	27	30	33
	$N_{\text{HABC}} = 4$	21	23	24	28	15	22	24	27
	$N_{\text{HABC}} = 6$	20	20	22	25	13	19	21	24

Table 2.2: Number of GMRES iterations to reach the relative residual 10^{-6} for the different configurations with distorted domain partitions. The final relative L^2-error is also approximately 10^{-6} for every case, except for the second configuration without cross-point treatment (results not shown) where the method is not consistent.

compatibility is not ensured at the boundary cross-points for that case. It has been observed that, when using the cross-point treatment, the method converges towards the correct solution, even with an important distortion of the partition. In that case, several interfaces starting from boundary cross-points are not perpendicular to the exterior border.

In nearly all the cases, the number of GMRES iterations increases when the distortion of the partitions is amplified. For the first configuration, the increase is rather small, with and without cross-point treatment. For the second one, the number of iterations increases more rapidly when the cross-point treatment is used. Nevertheless, in all the cases, using the cross-point treatment accelerates the convergence. The speedup is smaller for the third configuration, but it is still significant. For the most distorted configurations (*i.e.* shift with 1.5 and twist with 0.3π), the smallest numbers of iterations always correspond to the cases with both the largest N_{HABC} and the cross-point treatment. These results show the robustness of the approach with non-right angles.

5 Conclusion

In this chapter, a non-overlapping DDM with HABC-based transmission operators was considered for the parallel finite-element solution of scattering and wave propagation problems. We presented a suitable way to tackle the cross-point problem for settings with lattice-type domain partitions. In particular, we addressed cases where a Padé-type HABC operator is used for the transmission condition (to accelerate the convergence of the procedure), for the exterior boundary condition (to improve the accuracy of the solution) or for both conditions.

To handle the cross-points, suitable relations and additional transmission variables were introduced at the points. Numerical results have shown that the convergence rate of the obtained DDM is improved. Systematic studies on the way the convergence depends on the tuning parameters of the method as well as the frequency, the mesh refinement and the number of subdomains were presented. Configurations with distorted

partitions were tested. While the method was conceived for lattice-type partitions with right angles, it also performed very well with partitions having non-right angles.

In Chapter 4, the HABC-based DDM will be applied to more challenging heterogeneous and three-dimensional test cases, and its performance compared with the PML-based DDM that we will introduce in the next chapter.

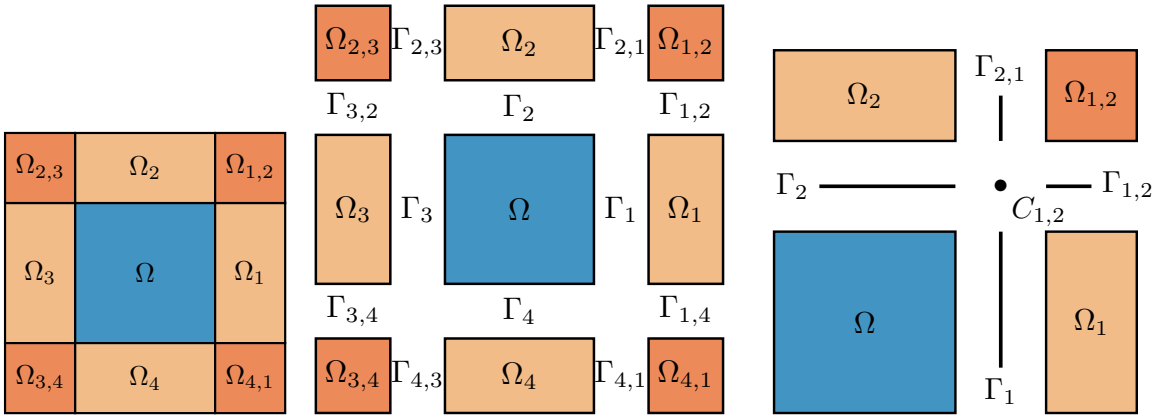
3 Cross-points for perfectly matched layer transmission conditions

In this chapter, a non-overlapping substructured DDM with PML transmission conditions for checkerboard (Cartesian) decompositions that takes cross-points into account is presented. In such decompositions, each subdomain is surrounded by PMLs associated to edges and corners. The continuity of Dirichlet traces at the interfaces between a subdomain and PMLs is enforced with Lagrange multipliers. This coupling strategy offers the benefit of naturally computing Neumann traces, which allows to use the PMLs as discrete operators approximating the exact Dirichlet-to-Neumann maps. Two possible Lagrange multiplier finite element spaces are presented, and the behavior of the corresponding DDM is analyzed on several numerical examples.

1 Introduction

In this chapter, we present a non-overlapping DDM with PML-based transmission conditions for two-dimensional Helmholtz problems. The method, designed for checkerboard domain partitions, takes naturally into account interior and exterior cross-points, with PMLs considered as operators imposed on interfaces through Lagrange multipliers. Two different discretization strategies for the Lagrange multipliers are studied. The main results of this chapter have been published in [168].

The chapter is organized as follows. In Section 2 the Helmholtz problem is introduced on a rectangular domain, and the coupling with surrounding PMLs (*i.e.* four PMLs associated to the edges of the rectangle, and four PMLs associated to the corners) using Lagrange multipliers is presented. Two discretizations for the Lagrange multipliers are introduced, and the solvability and the stability of the resulting finite element problems are analyzed. Then, in Section 3, the DDM with PML-based transmission operators is introduced. The cross-point treatments are naturally taken into account through the PMLs used at the corners of the rectangular subdomains. In Section 4, some numerical examples are presented to analyze the behavior of the proposed meth-



(a) Square domain surrounded by edge and corner PML regions.

(b) Coupling interfaces.

(c) Zoom on the upper right corner $C_{1,2}$.

Figure 3.1: A square domain Ω in blue with PML regions in orange.

ods. After an analysis of the influence of the PML parameters, the convergence of the domain decomposition algorithm is studied on representative finite element problems with homogeneous media.

2 Weak-coupling of PMLs with Lagrange multipliers

2.1 Definition of the problem

We consider the two-dimensional Helmholtz problem (1.4) of Chapter 1 defined on a square domain Ω, which is surrounded with PML regions associated to the edges and the corners of Ω. These families of PML, respectively called edge PMLs and corner PMLs, are illustrated in Figure 3.1. The edges and the corners of Ω are denoted Γ_i (with $i = 1, \ldots, 4$) and $C_{i,j}$ (with $i, j = 1, \ldots, 4$, such that Γ_i and Γ_j are adjacent), respectively. The edge PML and corner PML associated to Γ_i and $C_{i,j}$ are denoted Ω_i and $\Omega_{i,j}$, respectively. The interface between a corner PML $\Omega_{i,j}$ and an edge PML Ω_i is denoted $\Gamma_{i,j}$. Let us note that $\Omega_{i,j} = \Omega_{j,i}$ but $\Gamma_{i,j} \neq \Gamma_{j,i}$. The exterior boundaries of the edge PMLs and the corner PMLs are denoted with Γ_i^{ext} and $\Gamma_{i,j}^{\text{ext}}$, respectively.

Denoting the union of the domain Ω, the edge PMLs and the corner PMLs as Ω_{all}, the global problem reads

$$\begin{cases} \operatorname{div}(\mathbb{D}\,\mathbf{grad}\, w) + Dk^2 w = -f, & \text{in } \Omega_{\text{all}}, \\ \mathbf{n}_{\text{tot}} \cdot (\mathbb{D}\ \mathbf{grad}\, w) = 0, & \text{on } \partial\Omega_{\text{all}}, \end{cases} \tag{3.1}$$

where k is the (positive) wavenumber, the tensor field $\mathbb{D}(\mathbf{x})$ and the scalar field $D(\mathbf{x})$ are material properties as defined in Section 3.3 of Chapter 1, $f(\mathbf{x})$ is a source term, and $\mathbf{n}_{\text{all}}$ is the outgoing normal. Inside the domain Ω, the material properties are $\mathbb{D} = \operatorname{diag}(1, 1)$ and $D = 1$, and they depend on absorption functions in the PML regions. The definition of $\mathbb{D}(\mathbf{x})$ and $D(\mathbf{x})$ in the PML regions is discussed in Section 4. The natural functional space for the global solution w is $H^1(\Omega_{\text{all}})$.

The global problem (3.1) can be rewritten as the coupling of nine subproblems associated to the square domain Ω and the edge/corner PML regions. The corresponding local solutions are denoted u, u_i and $u_{i,j}$, for Ω, Ω_i and $\Omega_{i,j}$, respectively. Let us note that $u_{i,j} = u_{j,i}$. The global solution w is simply obtained by combining these local solutions. This approach is quite non-standard for such problems. Indeed, one usually directly solves Problem 3.1 with the field w defined in $H^1(\Omega_{\text{all}})$. Nevertheless, this classical approach would require additional treatments to compute the Neumann traces needed to build a sub-structuring DDM on the interfaces as explained in Section 2.2 and further in Section 3. Choosing the Lagrange multipliers formulation will allow us to gain direct access to the Neumann traces. The nine coupled subproblems read as follows:

- Subproblem associated to the square domain Ω:

$$\begin{cases} \Delta u + k^2 u = -f & \text{in } \Omega, \\ \partial_{\mathbf{n}} u = \mathbf{n} \cdot (\mathbb{D}_i \, \mathbf{grad}\, u_i) & \text{on each } \Gamma_i, \end{cases} \tag{3.2}$$

  where $\mathbf{n}$ is the outgoing normal with respect to Ω;

- Subproblems associated to each edge PML Ω_i (with $i = 1, \ldots, 4$):

$$\begin{cases} \operatorname{div}(\mathbb{D}_i \, \mathbf{grad}\, u_i) + D_i k^2 u_i = 0 & \text{in } \Omega_i, \\ \mathbf{n}_i \cdot (\mathbb{D}_i \, \mathbf{grad}\, u_i) = 0 & \text{on } \Gamma_i^{\text{ext}}, \\ u_i = u & \text{on } \Gamma_i, \\ \mathbf{n}_i \cdot (\mathbb{D}_i \, \mathbf{grad}\, u_i) = \mathbf{n}_i \cdot (\mathbb{D}_{i,j} \, \mathbf{grad}\, u_{i,j}) & \text{on each } \Gamma_{i,j}, \end{cases} \tag{3.3}$$

  where $\mathbf{n}_i$ is the outgoing normal with respect to Ω_i;

- Subproblems associated to each corner PML $\Omega_{i,j}$ (with $i, j = 1, \ldots, 4$ such that Γ_i and Γ_j are adjacent):

$$\begin{cases} \operatorname{div}(\mathbb{D}_{i,j} \, \mathbf{grad}\, u_{i,j}) + D_{i,j} k^2 u_{i,j} = 0 & \text{in } \Omega_{i,j}, \\ \mathbf{n}_{i,j} \cdot (\mathbb{D}_{i,j} \, \mathbf{grad}\, u_{i,j}) = 0 & \text{on } \Gamma_{i,j}^{\text{ext}}, \\ u_{i,j} = u_i & \text{on } \Gamma_{i,j}, \\ u_{i,j} = u_j & \text{on } \Gamma_{j,i}, \end{cases} \tag{3.4}$$

  where $\mathbf{n}_{i,j}$ is the outgoing normal with respect to $\Omega_{i,j}$.

In a nutshell, each local solution verifies the Helmholtz equation and a homogeneous Neumann condition on the exterior boundary (if any), and the coupling is performed by enforcing the continuity of the Dirichlet and Neumann traces at the interfaces.

2.2 Variational formulation with Lagrange multipliers

In the domain decomposition procedure, every rectangular subdomain of the checkerboard partition will be surrounded with edge and corner PMLs. The standard variational formulation based on System (3.1) could be used for each subproblem. However, the drawback of this approach lies in the lack of direct availability of the Neumann

traces at the interfaces between each subdomain and the corresponding surrounding PMLs, while the domain decomposition procedure requires the knowledge of both Dirichlet and Neumann traces at these interfaces. In this work, we consider an alternative variational formulation based on the coupled systems (3.2), (3.3) and (3.4), where the continuity conditions at the interfaces are enforced by using Lagrange multipliers. This approach offers the benefit to naturally give access to the Neumann traces thanks to the Lagrange multipliers.

Let us introduce four *edge* Lagrange multipliers λ_i on the interfaces Γ_i (with $i = 1, \ldots, 4$), and eight *corner* Lagrange multipliers $\lambda_{i,j}$ on $\Gamma_{i,j}$ (with $i, j = 1, \ldots, 4$ such that Γ_i and Γ_j are adjacent). These multipliers will be used to enforce the following continuity conditions:

$$\begin{aligned} u - u_i &= 0 \quad \text{on each } \Gamma_i, \\ u_i - u_{i,j} &= 0 \quad \text{on each } \Gamma_{i,j}. \end{aligned} \tag{3.5}$$

The dualization of these continuity conditions leads to Lagrange multipliers that weakly verify

$$\begin{aligned} \lambda_i &= \mathbf{n}_i \cdot (\mathbb{D}_i \, \mathbf{grad}\, u_i) \qquad \text{on each } \Gamma_i, \\ \lambda_{i,j} &= \mathbf{n}_{i,j} \cdot (\mathbb{D}_{i,j} \, \mathbf{grad}\, u_{i,j}) \quad \text{on each } \Gamma_{i,j}, \end{aligned} \tag{3.6}$$

which corresponds to the required Neumann traces that appear in the definition of the subproblems. Let us note that $\lambda_{i,j} \neq \lambda_{j,i}$.

In order to write the variational formulation in a concise form, we introduce the set of u-fields, denoted u_{all}, defined such that the restriction of u_{all} on Ω, Ω_i and $\Omega_{i,j}$ is respectively u, u_i and $u_{i,j}$. Similarly, the set of λ-fields, denoted λ_{all}, is defined such that the restriction of λ_{all} on Γ_i and $\Gamma_{i,j}$ is respectively λ_i and $\lambda_{i,j}$. The sets of fields u_{all} and λ_{all} belong to the following functional spaces:

$$\mathcal{U} := H^1(\Omega) \oplus \left[\bigoplus_i H^1(\Omega_i) \right] \oplus \left[\bigoplus_{i,j} H^1(\Omega_{i,j}) \right], \tag{3.7}$$

$$\mathcal{L} := \left[\bigoplus_i H^{-1/2}(\Gamma_i) \right] \oplus \left[\bigoplus_{i,j} H^{-1/2}(\Gamma_{i,j}) \right], \tag{3.8}$$

where $H^{-1/2}(\cdot)$ is the dual space of the Dirichlet trace space $H^{1/2}(\cdot)$. The variational formulation of the problem then reads: Find $(u_{\text{all}}, \lambda_{\text{all}}) \in \mathcal{U} \times \mathcal{L}$ such that

$$\begin{cases} h(u_{\text{all}}, v_{\text{all}}) + \overline{c(\lambda_{\text{all}}, v_{\text{all}})} = l(v_{\text{all}}), \\ c(u_{\text{all}}, \mu_{\text{all}}) = 0, \end{cases} \tag{3.9}$$

holds for all test functions $(v_{\text{all}}, \mu_{\text{all}}) \in \mathcal{U} \times \mathcal{L}$, where the sesquilinear forms and the

antilinear form are defined as

$$h(u_{\text{all}}, v_{\text{all}}) := \int_\Omega -\mathbf{grad}\, u \cdot \mathbf{grad}\, \bar{v} + k^2 u \bar{v}\ \mathsf{d}\Omega + \sum_i \int_{\Omega_i} -\mathbb{D}_i\, \mathbf{grad}\, u_i \cdot \mathbf{grad}\, \bar{v}_i + D_i k^2 u_i \bar{v}_i\ \mathsf{d}\Omega_i + \sum_{i,j} \int_{\Omega_{i,j}} -\mathbb{D}_{i,j}\, \mathbf{grad}\, u_{i,j} \cdot \mathbf{grad}\, \bar{v}_{i,j} + D_{i,j} k^2 u_{i,j} \bar{v}_{i,j}\ \mathsf{d}\Omega_{i,j}, \tag{3.10}$$

$$c(u_{\text{all}}, \mu_{\text{all}}) := \sum_i \int_{\Gamma_i} (u - u_i)\bar{\mu}_i\ \mathsf{d}\Gamma_i + \sum_{i,j} \int_{\Gamma_{i,j}} (u_i - u_{i,j})\bar{\mu}_{i,j}\ \mathsf{d}\Gamma_{i,j}, \tag{3.11}$$

$$l(v_{\text{all}}) := -\int_\Omega f\bar{v}\ \mathsf{d}\Omega. \tag{3.12}$$

The form $h(\cdot,\cdot)$ is the standard sesquilinear form for the Helmholtz equation with the PML material parameters, and the form $c(\cdot,\cdot)$ corresponds to the coupling with the Lagrange multipliers. The overline $\bar{\cdot}$ denotes the complex conjugate of a field.

Note that in the case of the classical approach for handling PMLs mentioned above, one could define a field on each interface, namely λ_i on each Γ_i and $\lambda_{i,j}$ on each $\Gamma_{i,j}$. However, a singularity would remain for each corner formulation since there will be two unknowns associated to the corner node, i.e. one for the fields $\lambda_{i,j}$ and one for the field $\lambda_{j,i}$.

2.3 Finite element discretizations

For the finite element discretization of Problem (3.9), we consider two conformal approximation spaces, $\mathcal{U}^h \subset \mathcal{U}$ and $\mathcal{L}^h \subset \mathcal{L}$, and the discrete fields $u^h_{\text{all}} \in \mathcal{U}^h$ and $\lambda^h_{\text{all}} \in \mathcal{L}^h$. The approximate variational formulation then reads: Find $\left(u^h_{\text{all}}, \lambda^h_{\text{all}}\right) \in \mathcal{U}^h \times \mathcal{L}^h$ such that

$$\begin{cases} h(u^h_{\text{all}}, v^h_{\text{all}}) + \overline{c(\lambda^h_{\text{all}}, v^h_{\text{all}})} = l(v^h_{\text{all}}), \\ c(u^h_{\text{all}}, \mu^h_{\text{all}}) = 0, \end{cases} \tag{3.13}$$

holds for every test function $\left(v^h_{\text{all}}, \mu^h_{\text{all}}\right) \in \mathcal{U}^h \times \mathcal{L}^h$. This formulation leads to the following linear system:

$$\begin{pmatrix} \mathsf{U} & \mathsf{L}^T \\ \mathsf{L} & 0 \end{pmatrix} \begin{pmatrix} \mathsf{u}^h_{\text{all}} \\ \mathsf{l}^h_{\text{all}} \end{pmatrix} = \begin{pmatrix} \mathsf{f} \\ 0 \end{pmatrix}, \tag{3.14}$$

where U is a block matrix derived form the sesquilinear form $h(\cdot,\cdot)$, L is a block matrix coming form the sesquilinear form $c(\cdot,\cdot)$ and f is the source vector. The vectors $\mathsf{u}^h_{\text{all}}$ and $\mathsf{l}^h_{\text{all}}$ contain the degrees of freedom associated to the discrete solution and the Lagrange multipliers, respectively. Let us note that U does not depend on the approximation space $\mathcal{L}^h$, while L does.

In this work, the space $\mathcal{U}^h$ is built by using standard hierarchical H^1-conforming basis functions. The approximation space for the Lagrange multipliers, $\mathcal{L}^h$, is based either on hierarchical H^1-conforming basis functions (choice called *"continuous discretization"*) or on the projection of hierarchical $\mathbf{H}(\text{div})$-conforming basis functions on the normal of the each interface (choice called *"discontinuous discretization"*). As we will see, this choice influences the well-posedness of Problem (3.13): the algebraic

system may not be solvable and stability issues may arise. Both discretizations are described and strategies to address these issues are discussed in Sections 2.3.1 and 2.3.2.

We recall herafter the conditions for the well-posedness of Problem (3.13) (see e.g. [37, 79]).

Theorem 3. (Well-posedness) *Problem (3.13) is well-posed if and only if there exist positive constants β and γ, independent of the mesh-size h, such that*

$$\inf_{u^h_{all}\in\mathcal{K}} \sup_{v^h_{all}\in\mathcal{K}} \frac{|h(u^h_{all}, v^h_{all})|}{\|u^h_{all}\|_{\mathcal{U}^h}\|u^h_{all}\|_{\mathcal{U}^h}} \geq \beta \tag{3.15}$$

$$\inf_{\mu^h_{all}\in\mathcal{L}^h} \sup_{u^h_{all}\in\mathcal{U}^h} \frac{|c(u^h_{all}, \mu^h_{all},)|}{\|u^h_{all}\|_{\mathcal{U}^h}\|\mu^h_{all}\|_{\mathcal{L}^h}} \geq \gamma \tag{3.16}$$

where $\|\cdot\|_{\mathcal{U}^h}$ and $\|\cdot\|_{\mathcal{L}^h}$ are norms associated to the approximation spaces $\mathcal{U}^h$ and $\mathcal{L}^h$, and $\mathcal{K}$ is the kernel of the operator associated to the sesquilinear form $c(\cdot,\cdot)$ in $\mathcal{U}^h$.

This theorem can be derived from Theorems 1 and 2 of Chapter 1 applied to problems written as (3.13). It also implies the existence and the uniqueness of the solution for any given mesh, and the stability of the problem with stability constants independent of the mesh. The following theorem gives the solvability conditions for System (3.14).

Theorem 4. (Solvability) *For a given mesh, the matrix of problem (3.14) is non singular if and only if the following two conditions are both satisfied:*

a. $\mathsf{U}_{KK} : K \to K$ *is surjective (or, equivalently, is injective),*

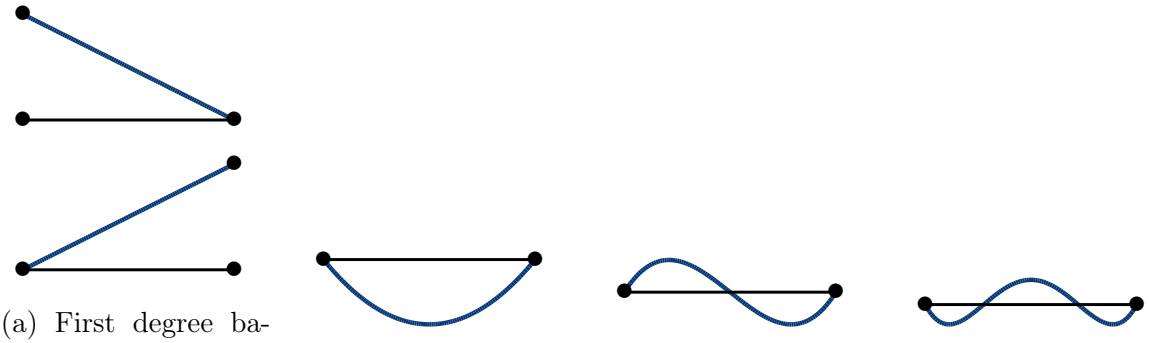
b. $\mathsf{L} : \mathbb{C}^n \to \mathbb{C}^m$ *is surjective (or, equivalently,* L^T *is injective),*

where U_{KK} *is the projection of* U *into the kernel* K *of* L.

If the solution belongs to the kernel of L, the second line of System (3.14) is verified. This line corresponds to relations that are enforced with the Lagrange multipliers. Assuming that the approximate solution is continuous at the interfaces thanks to the Lagrange multipliers, the resulting problem corresponds to the discretization of the standard Helmholtz problem with PML over Ω_{all}, which is well-posed [34, 35, 36].

2.3.1 Strategies with continuous Lagrange multipliers

In the first approach, the approximation space $\mathcal{L}^h$ is built by using hierarchical H^1-conforming basis functions on each interface. Then, the Lagrange multipliers are continuous over each interface. We assume that the same polynomial degree is used for both $\mathcal{L}^h$ and $\mathcal{U}^h$. The basis functions used for $\mathcal{L}^h$ then correspond to the restriction on the interfaces of the basis functions used for $\mathcal{U}^h$. The very first basis functions correspond to standard P1 finite elements, and the high-order basis functions are built by using the approach described *e.g.* in [176]. The basis functions for the one-dimensional case are represented on Figure 3.2.



(a) First degree basis functions (classical P1).

(b) Second degree basis function.

(c) Third degree basis function.

(d) Fourth degree basis function.

Figure 3.2: Hierarchical H^1-conforming basis functions

Unfortunately, the direct implementation of Problem (3.13) with continuous Lagrange multipliers leads to algebraic systems that are not solvable. Indeed, L is not surjective (or, equivalently, L^T is not injective), and the second solvability condition of Theorem 4 is not met. It can be shown by considering the relations verified on the interfaces around the cross-point $C_{1,2}$, represented on Figure 3.1c. The continuity of the discrete solution is enforced weakly by using the following relations:

$$\int_{\Gamma_1} (u^h - u_1^h)\bar{\mu}_1^h \, \mathrm{d}\Gamma_1 = 0, \tag{3.17}$$

$$\int_{\Gamma_2} (u^h - u_2^h)\bar{\mu}_2^h \, \mathrm{d}\Gamma_2 = 0, \tag{3.18}$$

$$\int_{\Gamma_{1,2}} (u_1^h - u_{1,2}^h)\bar{\mu}_{1,2}^h \, \mathrm{d}\Gamma_{1,2} = 0, \tag{3.19}$$

$$\int_{\Gamma_{2,1}} (u_2^h - u_{1,2}^h)\bar{\mu}_{2,1}^h \, \mathrm{d}\Gamma_{2,1} = 0, \tag{3.20}$$

where the test functions belong to the approximation spaces associated to the Lagrange multipliers. In these relations, the discrete solutions, u^h, u_1^h, u_2^h and $u_{1,2}^h$, can be replaced by their representations with the basis functions. Only the degrees of freedom associated to the interfaces are involved in these relations. The discrete solutions appearing in Equation (3.17) can be written as

$$u^h|_{\Gamma_1} = \sum_j u^{h,j}\phi_j|_{\Gamma_1} \qquad \text{and} \qquad u_1^h|_{\Gamma_1} = \sum_j u_1^{h,j}\phi_{1,j}|_{\Gamma_1}, \tag{3.21}$$

where ϕ_j and $\phi_{1,j}$ denote basis functions of u^h and u_1^h associated to the interface Γ_1. The sums are preformed over the J degrees of freedom of u^h associated to Γ_1 and the J_1 degrees of freedom of u_1^h associated to Γ_1. Finally, substituting these expressions into Equation (3.17), and using the basis functions of the Lagrange multipliers, $\{\psi_{1,i}\}_{i=1,\dots,I}$, as test functions leads to

$$\int_{\Gamma_1} \left(\sum_j u^{h,j}\phi_j - \sum_j u_1^{h,j}\phi_{1,j} \right) \psi_{1,i} \, \mathrm{d}\Gamma_1 = 0 \qquad \text{for } i = 1, \dots, I. \tag{3.22}$$

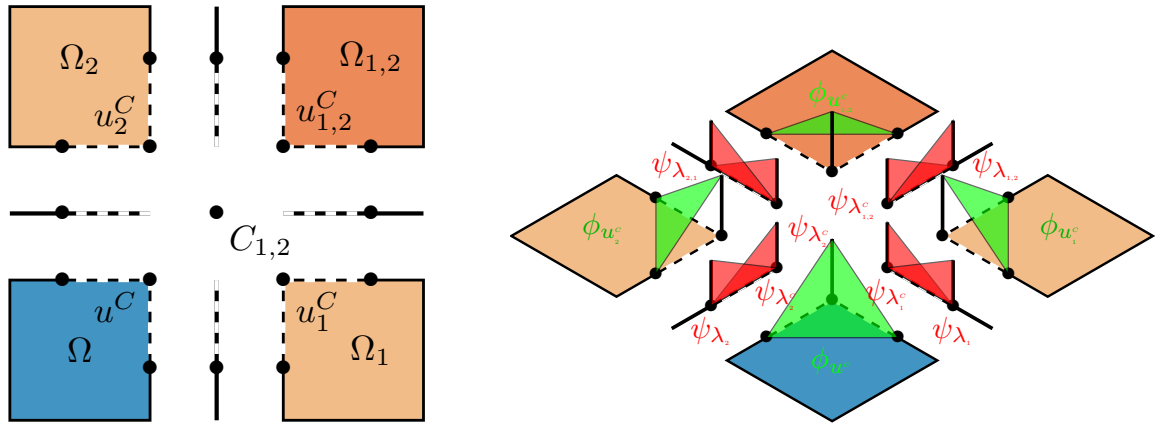


Figure 3.3: Degrees of freedom and P1 basis functions involved around the cross-point $C_{1,2}$

Denoting u^h and u_1^h the vectors of degrees of freedom associated to the interface Γ_1, we obtain

$$\mathsf{M}_{\Gamma_1}\mathsf{u}^h - \mathsf{M}_{1,\Gamma_1}\mathsf{u}_1^h = 0, \tag{3.23}$$

where

$$(\mathsf{M}_{\Gamma_1})_{ij} = \int_{\Gamma_1} \phi_j \, \psi_{1,i} \, \mathrm{d}\Gamma \qquad \text{and} \qquad (\mathsf{M}_{1,\Gamma_1})_{ij} = \int_{\Gamma_1} \phi_{1,j} \, \psi_{1,i} \, \mathrm{d}\Gamma \tag{3.24}$$

are mass matrices of size $I \times J$ and $I \times J_1$, respectively. If the same polynomial degree is used for the finite element approximation of the fields u^h, u_1^h and μ_1^h, the corresponding basis functions are identical on the interface Γ_1. Then, Equation (3.23) can be rewritten as

$$\mathsf{M}_{\Gamma_1}^{\text{cont}} \left(\mathsf{u}^h - \mathsf{u}_1^h\right) = 0, \tag{3.25}$$

and $\mathsf{M}_{\Gamma_1}^{\text{cont}}$ is a standard square mass matrix. Since this matrix is non-singular, we have $\mathsf{u}^h = \mathsf{u}_1^h$, and then $u^h|_{\Gamma_1} = u_1^h|_{\Gamma_1}$. In particular, the degrees of freedom of u^h and u_1^h associated to the cross-point $C_{1,2}$ are equal, *i.e.* $u^C = u_1^C$. The basis functions associated to the cross-point $C_{1,2}$ are represented on Figure 3.3. The same reasoning can be carried out for the other interfaces around the cross-point $C_{1,2}$, leading to the following relations,

$$u^C = u_1^C, \tag{3.26}$$
$$u^C = u_2^C, \tag{3.27}$$
$$u_1^C = u_{1,2}^C, \tag{3.28}$$
$$u_2^C = u_{1,2}^C, \tag{3.29}$$

where u_2^C and $u_{1,2}^C$ are the degrees of freedom at the cross-point associated to u_2^h and $u_{1,2}^h$. Because these relations are linear dependent, there is also a linear dependency between the relations resulting from the discretization of (3.17)-(3.20). Therefore, L is not surjective, and Problem (3.14) is not solvable.

Several approaches can be used to recover the surjectivity of matrix L, and then to make Problem (3.14) solvable:

a. An additional constraint on the unknowns can be added in order to avoid the linear dependency between Equations (3.26)-(3.29). This can be done by introducing an additional Lagrange multiplier, denoted λ^C, associated to the cross-point. In this work, the following constraint is used:

$$\lambda_1^C - \lambda_2^C + \lambda_{1,2}^C - \lambda_{2,1}^C = 0, \quad \text{at } C_{1,2}, \tag{3.30}$$

where λ_1^C, λ_2^C, $\lambda_{1,2}^C$ and $\lambda_{2,1}^C$ are the Lagrange multipliers associated to the interfaces around the cross-point $C_{1,2}$. Then, terms involving λ^C are introduced in the relations associated to these multipliers, in such a way that Equations (3.26)-(3.29) become

$$u^C = u_1^C + \lambda^C, \tag{3.31}$$
$$u^C = u_2^C - \lambda^C, \tag{3.32}$$
$$u_1^C = u_{1,2}^C + \lambda^C, \tag{3.33}$$
$$u_2^C = u_{1,2}^C - \lambda^C. \tag{3.34}$$

These relations are not longer linearly dependent, and Problem (3.14) with the supplementary equation becomes solvable. A similar strategy was used by Peng and Lee [160] to improve a domain decomposition method for time-harmonic electromagnetic problems.

b. A penalization strategy (see *e.g.* [37]), where a mass matrix is added in Problem (3.14), can be used to obtain the following modified system:

$$\begin{pmatrix} \mathsf{U} & \mathsf{L}^T \\ \mathsf{L} & \tau\mathsf{M} \end{pmatrix} \begin{pmatrix} \mathsf{u}_{\text{all}}^h \\ \mathsf{l}_{\text{all}}^h \end{pmatrix} = \begin{pmatrix} \mathsf{f} \\ 0 \end{pmatrix}, \tag{3.35}$$

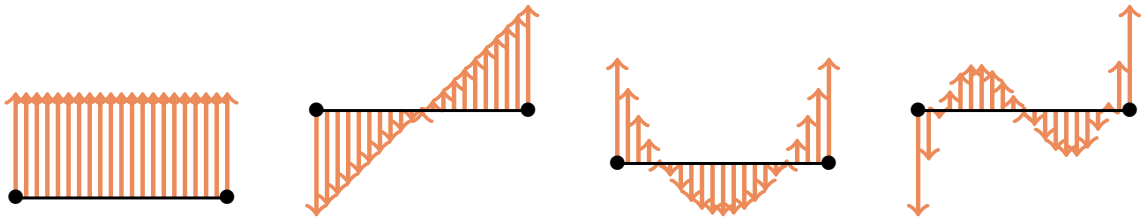
where τ is a penalization parameter to be tuned and M is the standard mass matrix associated to the Lagrange multiplier space $\mathcal{L}^h$. Because of the new block, the continuity conditions (3.5) are not exactly verified. They become

$$\begin{aligned} u - u_i &= \tau\lambda_i, \quad \text{on each } \Gamma_i, \\ u_i - u_{i,j} &= \tau\lambda_{i,j}, \quad \text{on each } \Gamma_{i,j}. \end{aligned} \tag{3.36}$$

Thanks to the penalty, a right-hand-side term is added to Equation (3.25), and the linear dependency between the relations at the corner is avoided.

c. A last strategy consists in taking approximate fields with different polynomial degrees in the domain and the PML regions. In preliminary tests (not shown), we have observed that, if the polynomial degree in the edge PMLs and the corner PMLs is larger by one and two, respectively, than in the domain, then the system is solvable. This strategy, which involves much more degrees of freedom than the others, will not be investigated further in this work.

Let us highlight that continuity of the Dirichlet traces at the interfaces is preserved exactly with the first strategy, but it is relaxed with the two last ones.



(a) First degree basis function. (b) Second degree basis function. (c) Third degree basis function. (d) Fourth degree basis function.

Figure 3.4: Hierarchical $\mathbf{H}(\mathrm{div})$-conforming basis functions

2.3.2 Strategies with discontinuous Lagrange multipliers

For the second approach, the basis functions of the approximation space $\mathcal{L}^h$ correspond to the projection of hierarchical $\mathbf{H}(\mathrm{div})$-conforming basis functions (defined on the domain and the PML regions) on the normal to the interfaces. In two dimensions, the hierarchical $\mathbf{H}(\mathrm{div})$-conforming basis functions correspond to the 90-degree rotation of $\mathbf{H}(\mathbf{curl})$-conforming basis functions. The first-order $\mathbf{H}(\mathbf{curl})$-conforming basis functions are the Nédélec basis functions [155, 154], and the higher-order functions are associated to the edges and the face of the elements (see *e.g.* [176]). The first basis functions for the Lagrange multipliers are represented on Figure 3.4. By contrast with the previous discretization, the basis functions are discontinuous between the elements, and the continuity of the Lagrange multipliers is not ensured. This approach is called the *discontinuous discretization*.

Depending on the polynomial degree used for the discontinuous Lagrange multipliers, the solvability issue discussed in the previous section can be naturally avoided. Indeed, with the discontinuous discretization, System (3.22) can be rewritten as

$$\mathsf{M}_{\Gamma_1}^{\mathrm{disc}} \left(\mathsf{u}^h - \mathsf{u}_1^h \right) = 0, \tag{3.37}$$

where $\mathsf{M}_{\Gamma_1}^{\mathrm{disc}}$ is a rectangular mass matrix associated to the discontinuous basis functions $\{\psi_{1,i}\}_i$ used for the Lagrange multipliers and the continuous basis functions $\{\phi_j\}_j$ used for the discrete solutions (see Equation (3.24)). This matrix cannot be square. If the polynomial degree of the Lagrange multipliers is lower than or equal to the degree of the discrete solution, then the system is underdetermined and $\mathsf{u}^h = \mathsf{u}_1^h$ does not hold. In this case, the continuity of the discrete solution at the interfaces is not exactly verified, but the system is solvable since L is full-rank. In the opposite case, Equation (3.37) is a homogeneous overdetermined system, and $\mathsf{u}^h = \mathsf{u}_1^h$ is the trivial solution. The problem in not solvable because L is not surjective, but the penalization strategy developed in the previous section can be used to make it solvable.

Unfortunately, we have observed that the problem is not stable if the same polynomial degree is used for both the discontinuous Lagrange multipliers and the discrete solutions. In that case, the first inf-sup condition of Theorem 3 is *a priori* not met, and the problem is not well-posed. For many practical situations, it is difficult to prove the first inf-sup condition. This is why, a numerical Chapelle-Bath test [23, 26] is applied to evaluate the inf-sup constant β for the different discretizations. In a nutshell, a

modified version of the system is considered,

$$\begin{pmatrix} \mathsf{U} & \mathsf{L}^T \\ \mathsf{L} & -\frac{1}{s}\mathsf{M} \end{pmatrix} \begin{pmatrix} \mathsf{u}_{\text{all}}^h \\ \mathsf{l}_{\text{all}}^h \end{pmatrix} = \begin{pmatrix} \mathsf{f} \\ 0 \end{pmatrix}, \tag{3.38}$$

where s is a positive constant and M is the mass matrix associated to the Lagrange multipliers. Taking $s \to \infty$ brings back the original system (3.14). The inf–sup test consists in checking that, using a sequence of meshes with decreasing mesh size h, the first non-vanishing eigenvalue α_h of the problem

$$\frac{1}{h}(\mathsf{L}^T\mathsf{M}^{-1}\mathsf{L})\mathsf{v}_h = \alpha_h \mathsf{U}^T\mathsf{v}_h, \tag{3.39}$$

does not depend too much on the mesh size h. Further, the inf-sup constant is given by $\beta = \min_h \beta_h$, with $\beta_h := \alpha_h^2$.

In Figure 3.5a, the value β_h is plotted for the continuous and the discontinuous discretizations of the Lagrange multipliers with several stabilization strategies. These results are obtained for a benchmark with the square domain $[-1,1]\times[-1,1]$, meshes with cell sizes from $h = 0.03$ to 0.32, the polynomial degree $p = 2$ for the discrete solution, and the penalization parameter $\tau = 0.002\ h^2$. When the discontinuous discretization is used with the same polynomial degree for both the Lagrange multipliers and the discrete solution, the value β_h decreases significantly with $1/h$, which indicates that the formulation is not stable, as mentionned earlier. By contrast, we observe that the other combinations are stable.

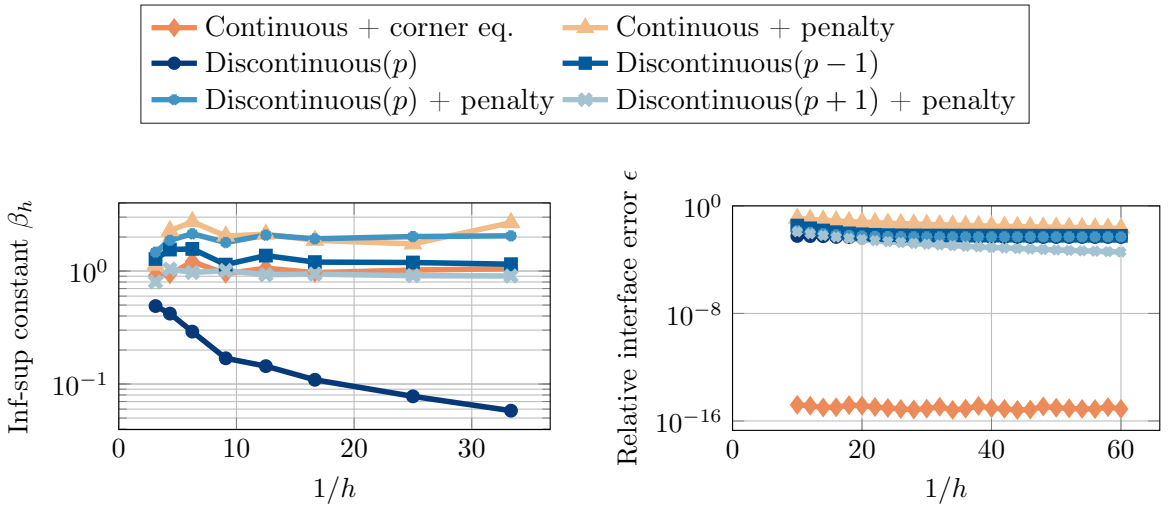
To study the effect of the discretization of the Lagrange multipliers on the discrete solution, we consider the relative continuity error at the interfaces, defined as

$$\epsilon = \frac{\sum_i \int_{\Gamma_i} \|u^h - u_i^h\|^2 \,\mathrm{d}\Gamma_i + \sum_{i,j} \int_{\Gamma_{i,j}} \|u_i^h - u_{i,j}^h\|^2 \,\mathrm{d}\Gamma_{i,j}}{\sum_i \int_{\Gamma_i} \|u^h\|^2 \,\mathrm{d}\Gamma_i + \sum_{i,j} \int_{\Gamma_{i,j}} \|u_i^h\|^2 \,\mathrm{d}\Gamma_{i,j}}. \tag{3.40}$$

This error is plotted according to the mesh size on Figure 3.5b for the different approaches. As expected, only the continuous discretization with the additional constraint leads to the perfect continuity of the Dirichlet trace (*i.e.* the relative error ϵ is close to the machine epsilon). Only that approach enforces exactly the continuity of the discrete field, while the continuity is relaxed with the other ones. The Neumann trace computed on the upper interface of Figure 3.1, namely on $\Gamma_{3,2} \cup \Gamma_2 \cup \Gamma_{1,2}$, is shown on Figure 3.5c for the different approaches. The trace obtained with the unstable discontinuous discretization clearly oscillates, while the others are smooth.

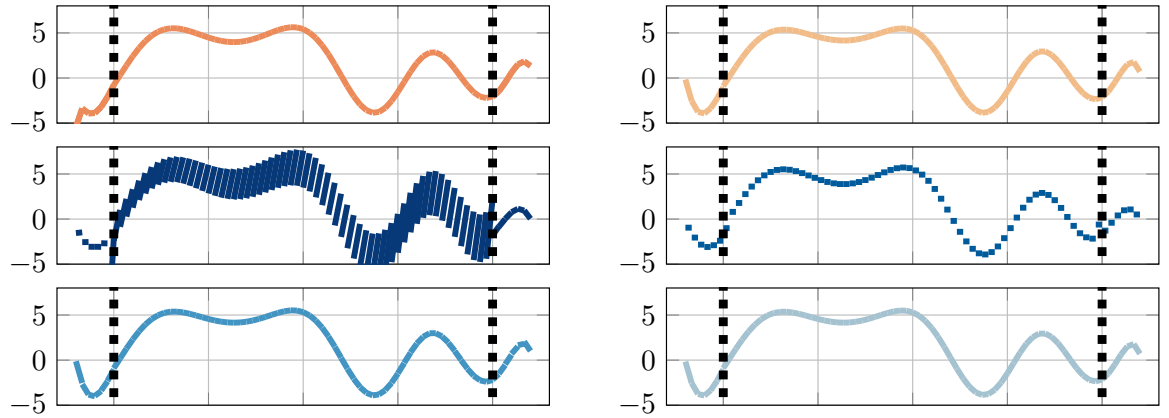
3 DDM with PML transmission conditions for checkerboard domain partitions

In this section, the non-overlapping domain decomposition method (DDM) presented in Section 6 of Chapter 1 is applied with the PML ransmission conditions for checkerboard (Cartesian) domain partitions. Transmission operators based on PMLs have been used to accelerate domain decomposition solvers [177, 188] and preconditioning techniques [184, 172, 14, 129]. However, for domain decomposition solvers with



(a) Numerical evaluation of the inf-sup constant of Theorem 3 obtained for u-field discretized at polynomial degree $p = 2$.

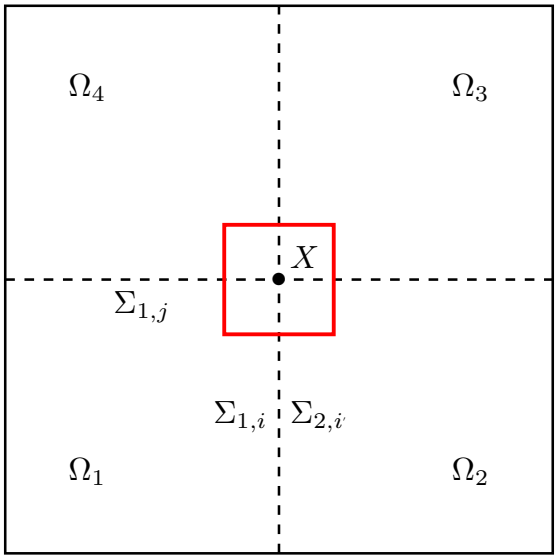
(b) The relative interface error shows that discontinuous discretizations (stabilized or not) introduce a lack of continuity at interfaces compared to the continuous discretization.



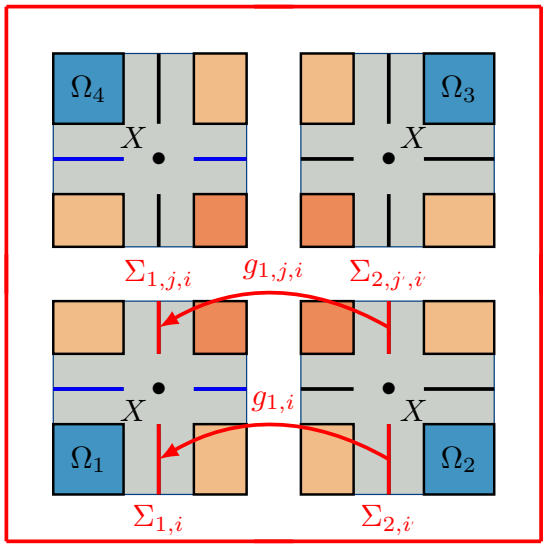
(c) Neumann traces along interface $\Gamma_{3,2} \cup \Gamma_2 \cup \Gamma_{1,2}$ (u-field discretized at polynomial degree $p = 2$).

Figure 3.5: Interface issues and Neumann traces.

checkerboard domain partitions, an inappropriate treatment of the interior cross-points (*i.e.* points where more than two subdomains meet) and the exterior cross-points (*i.e.* points where an interface meet an exterior border) may degrade convergence or even lead to incorrect results. In this work, the PML is used as a DtN operator thanks to the strategies developed in the previous section, leading to a domain decomposition method that naturally takes into account cross-points. Based on the standard non-overlapping DDM is presented in Section 6 of Chapter 1, modifications to use PMLs as transmission operators and to address cross-points are explained in Sections 3.1 and 3.2.



(a) Decomposition of a square domain Ω_{tot} into four subdomains Ω_n leading to a cross-point X at the intersection between the four subdomains.



(b) Zoom on the cross-point of Figure 3.6a (the red box) to see how data are exchanged in the case of a PML-based DDM. (For clarity the unique cross-point X is drawn four times and the domains are represented with a shift.)

Figure 3.6: Four subdomains and a cross-point X between them.

3.1 The PML-based transmission operator

The PML can be used rather naturally as a DtN operator thanks to the weak coupling introduced in the previous section. With that approach, the continuity of the solution at the interface between the PML and a given domain is enforced weakly thanks to a Lagrange multiplier, and the value of this multiplier can be interpreted as the Neumann trace of the solution. Therefore, the PML can be seen as an operator taking a Dirichlet trace and returning the corresponding Neumann trace. For the interface edge $\Sigma_{n,i}$ of subdomain Ω_n, the transmission operator $\mathcal{T}_{n,i}$ is formally defined as

$$\mathcal{T}_{n,i} \ : \ H^{1/2}(\Sigma_{n,i}) \ \mapsto \ H^{-1/2}(\Sigma_{n,i}) \ : \ u_n|_{\Sigma_{n,i}} \ \mapsto \ \mathcal{T}_{n,i}u_n := \lambda_{n,i}, \quad (= \partial_{\mathbf{n}_{n,i}} u_n|_{\Sigma_{n,i}}) \tag{3.41}$$

where $\lambda_{n,i}$ is the Lagrange multiplier used to prescribed the continuity of the solution at the interface between the subdomain and the PML.

In order to clarify the use of the PML-based DtN operator in the DDM, let us consider a rectangular waveguide partitioned into successive layers, with a homogeneous Neumann boundary condition on the lateral borders. In this configuration, every subdomain Ω_n has two neighbors, expect both subdomains that are at the extremities of the partition. At both interface edges, denoted $\Sigma_{n,1}$ and $\Sigma_{n,2}$, the subdomain is extended with two PMLs, denoted $\Omega_{n,1}$ and $\Omega_{n,2}$, respectively. The variational formulation associated to Ω_n can be written as: Find $(u_n, u_{n,1}, u_{n,2}) \in H^1(\Omega_n) \times H^1(\Omega_{n,1}) \times H^1(\Omega_{n,2})$

and $(\lambda_{n,1}, \lambda_{n,2}) \in H^{-1/2}(\Sigma_{n,1}) \times H^{-1/2}(\Sigma_{n,2})$ such that

$$\begin{aligned}\int_{\Omega_n} \big(- \mathbf{grad}\, u_n \cdot \mathbf{grad}\, \bar{v}_n + k^2 u_n \bar{v}_n \big)\, \mathsf{d}\Omega_n \\ + \sum_i \int_{\Omega_{n,i}} \big(- \mathbb{D}_{n,i}\, \mathbf{grad}\, u_{n,i} \cdot \mathbf{grad}\, \bar{v}_{n,i} + D_{n,i}\, k^2 u_{n,i} \bar{v}_{n,i} \big)\, \mathsf{d}\Omega_{n,i} \\ + \sum_i \int_{\Sigma_{n,i}} \lambda_{n,i} (\bar{v}_n - \bar{v}_{n,i})\, \mathsf{d}\Sigma_{n,i} = - \int_{\Omega_n} f \bar{v}_n\, \mathsf{d}\Omega_n - \sum_i \int_{\Sigma_{n,i}} g_{n,i} \bar{v}_n\, \mathsf{d}\Sigma_{n,i}, \end{aligned} \tag{3.42}$$

holds for all test functions $(v_n, v_{n,1}, v_{n,2}) \in H^1(\Omega_n) \times H^1(\Omega_{n,1}) \times H^1(\Omega_{n,2})$, and

$$\sum_i \int_{\Sigma_{n,i}} (u_n - u_{n,i}) \bar{\mu}_{n,i}\, \mathsf{d}\Sigma_{n,i} = 0 \tag{3.43}$$

holds for all test functions $(\mu_{n,1}, \mu_{n,2}) \in H^{-1/2}(\Sigma_{n,1}) \times H^{-1/2}(\Sigma_{n,2})$. The transmission variables are updated using the Lagrange multipliers. The update formula (1.61) of Chapter 1 becomes

$$g_{n,i}^{(\ell+1)} = -g_{m,i'}^{(\ell)} + 2\lambda_{m,i'}^{(\ell)}. \tag{3.44}$$

To summarize, every subdomain is extended with PMLs (second term in Equation (3.42)) and Lagrange multipliers are used as Neumann traces in the subdomain (third term in Equation (3.42)) and the update formula (Equation (3.44)). Note that, compared to an overlapping DDM, no communication between subdomains is needed in the PMLs, which can be generated ("grown") completely independently.

3.2 DDM with PML transmission conditions and cross-point treatment

The approach can be applied to the Helmholtz problem (1.4) of Chapter 1 with a checkerboard domain partition. If there is only one subdomain (*i.e.* $N = 1$), the variational formulation (3.42) should correspond to the original problem with PMLs at the boundaries, leading to the variational formulation (3.9). If there are more than one subdomain, the same variational formulation can be used for every subdomain, but terms with transmission variables must be added in the right-hand side of the first equation in order to enforce the coupling between the subproblems.

To write the subproblem associated to subdomain Ω_n, we introduce the sets of u_n-fields and λ_n-fields, denoted $u_{n,\mathrm{all}}$ and $\lambda_{n,\mathrm{all}}$, which contain the solutions and the Lagrange multipliers associated to the domain Ω_n, the surrounding PML regions and the interfaces. The corresponding functional spaces are denoted $\mathcal{U}_n$ and $\mathcal{L}_n$. More precise definitions of these objects are provived in Section 2.2 for the problem associated to Ω. The discretization strategies discussed in Section 2.3 will be used.

In the general case, the subproblem associated to Ω_n reads: Find $(u_{n,\mathrm{all}}, \lambda_{n,\mathrm{all}}) \in \mathcal{U}_n \times \mathcal{L}_n$ such that

$$\begin{cases} h_n(u_{n,\mathrm{all}}, v_{n,\mathrm{all}}) + \overline{c_n(\lambda_{n,\mathrm{all}}, v_{n,\mathrm{all}})} = l_n(v_{n,\mathrm{all}}), \\ c_n(u_{n,\mathrm{all}}, \mu_{n,\mathrm{all}}) = 0, \end{cases} \tag{3.45}$$

where the sesquilinear forms $h_n(\cdot,\cdot)$ and $c_n(\cdot,\cdot)$ are defined similarly to $h(\cdot,\cdot)$ and $c(\cdot,\cdot)$ (see Equations (3.10) and (3.11)), and the antilinear form $l_n(\cdot)$ is defined as

$$l_n(v_{n,\text{all}}) := -\int_{\Omega_n} f\bar{v}_n \,\mathsf{d}\Omega_n - \sum_i \int_{\Sigma_{n,i}} g_{n,i}\bar{v}_n \,\mathsf{d}\Sigma_{n,i} - \sum_{i,j} \int_{\Sigma_{n,i,j}} g_{n,i,j}\bar{v}_{n,i} \,\mathsf{d}\Sigma_{n,i,j}. \quad (3.46)$$

The second term in the right-hand side member of the previous equation introduces a coupling at the interface subdomain-PML for subproblems corresponding to neighboring subdomains. The last term corresponds to a coupling at interfaces PML-PML. These couplings are illustrated on Figure 3.6b. Let us not that these terms appear only if $\Sigma_{n,i}$ is an interface edge of the subdomain. In that case, the transmission variables are updated using the update relations

$$g_{n,i}^{(\ell+1)} = -g_{m,i'}^{(\ell)} + 2\lambda_{m,i'}^{(\ell)} \qquad \text{on } \Sigma_{n,i} \quad (3.47)$$

$$g_{n,i,j}^{(\ell+1)} = -g_{m,i',j'}^{(\ell)} + 2\lambda_{m,i',j'}^{(\ell)} \qquad \text{on } \Sigma_{n,i,j}, \quad (3.48)$$

where the variables $g_{n,i}$ and $g_{n,i,j}$ can be considered as *edge* and *corner* transmission variables, respectively, and the Lagrange multipliers $\lambda_{m,i'}$ and $\lambda_{m,i',j'}$ are computed in the subproblem associated to the neighboring subdomain Ω_m. The overscript ℓ corresponds to quantities computed at step ℓ of the iterative procedure. This version of the DDM naturally takes into account cross-points through the definition of the corner transmission variables.

In what follows, the subproblems are solved by using the finite element schemes described in the Section 2. The same discretizations are used for both the Lagrange multipliers and the transmission variables.

4 Numerical results

The performance of the proposed DDM and the discretization strategies for the Lagrange multipliers and the transmission variables are studied by using a reference two-dimensional benchmark described in Section 4.1. First, the continuous and discontinuous discretizations and the different stabilization techniques are compared in Section 4.2, and two approaches are selected. For the selected approaches, the parameters of the PML transmission conditions, namely the absorbing function and the layer thickness, are discussed in Sections 4.3 and 4.4, respectively. Finally, the influence of the wavenumber and the mesh density on the convergence of the DDM is analyzed in Section 4.5. The method will be tested with a smoothly varying heterogeneous medium in Chapter 4.

4.1 Description of the reference benchmark and PML parameters

Let us consider the same scattering benchmark as the one introduce in Section 4.1 of Chapter 2. The simulations are performed with a square computational domain, $\Omega_{\text{tot}} = [L_{-x}, L_x] \times [L_{-y}, L_y]$, surrounded with PMLs of thickness δ_{PML}. The material

parameters $\mathbb{D}$ and D are defined as

$$\mathbb{D}(x,y) = \operatorname{diag}\left(\frac{\gamma_y(y)}{\gamma_x(x)}, \frac{\gamma_x(x)}{\gamma_y(y)}\right) \qquad \text{and} \qquad D(x,y) = \gamma_x(x)\gamma_y(y), \tag{3.49}$$

where $\gamma_x(x)$ and $\gamma_y(y)$ defined in Equations 1.31 of Chapter 1. The absorption functions functions $\sigma_x(x)$ and $\sigma_y(y)$ are set to zero inside the domain, and they increase along the associated Cartesian directions inside the PMLs. Shifted hyperbolic absorbing functions are used inside the PMLs. Then, the function $\sigma_x(x)$ is defined as

$$\sigma_x(x) = \begin{cases} 1/(\delta_{\text{PML}} - (L_{-x} - x)) - 1/\delta_{\text{PML}} & \text{if } x \in [L_{-x} - \delta_{\text{PML}}, L_{-x}], \\ 0 & \text{if } x \in [L_{-x}, L_x], \\ 1/(\delta_{\text{PML}} - (x - L_x)) - 1/\delta_{\text{PML}} & \text{if } x \in [L_x, L_x + \delta_{\text{PML}}]. \end{cases} \tag{3.50}$$

The definition is similar for $\sigma_y(y)$. These functions form a couple such that a PML extruded in the x-direction (*i.e.* Ω_1 and Ω_3 on Figure 3.1a) is associated to $(\sigma_x(x), 0)$, a PML extruded in the y-direction (*i.e.* Ω_2 and Ω_4) is associated to $(0, \sigma_y(y))$, and corner PMLs are associated to $(\sigma_x(x), \sigma_y(y))$.

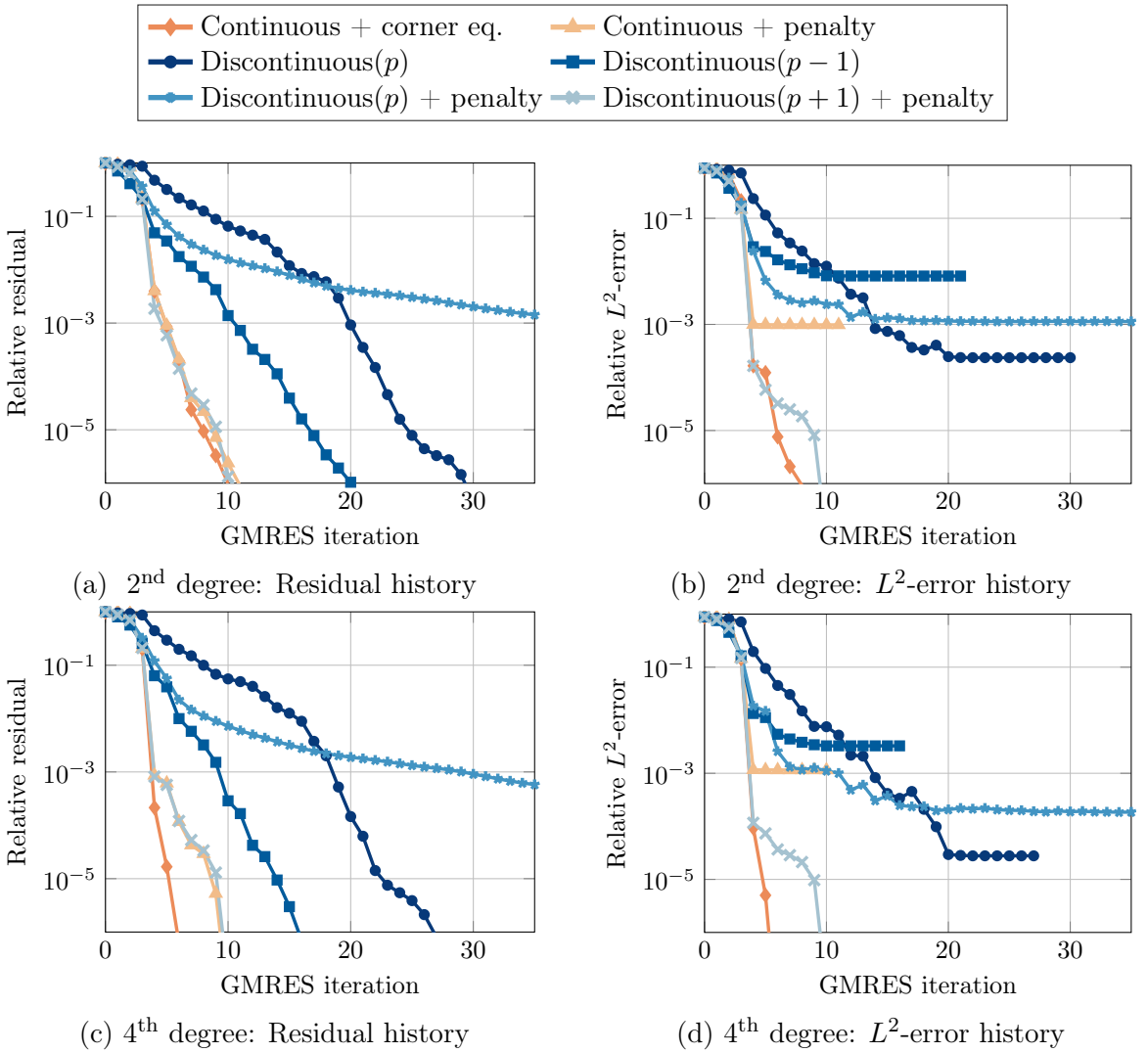
For the PML transmission conditions, the material parameters are defined similarly. Let us note that the absorbing functions and the layer thicknesses can be different for the interface edges (*i.e.* in the transmission conditions) and the boundary edges (*i.e.* for the exterior boundary condition) of the domain partition. Different combinations are tested in Sections 4.3 and 4.4.

In the following sections, the DDM is tested with the same checkerboard partition of Ω_{tot} into a 3×3 grid as presented in Chapter 2. The disk of radius $R = 0.5$ is placed in the middle of the lower left subdomain, and the borders of the square subdomains are of length 2. Every subdomain is meshed with triangular elements having straight edges, and the surrounding PMLs are generated with extruded square elements. The wavenumber is $k = 4\pi$ and the characteristic mesh size is $h \approx 1/30$. The numerical results were obtained with GmshDDM (Chapter 6) and the related implementation of the test cases is available at the following address: `https://gitlab.onelab.info/gmsh/ddm/-/tree/master/examples/helmholtz/crossPoints`.

4.2 Comparison of the discretization strategies for the Lagrange multipliers and the transmission variables

The convergence histories for both continuous and discontinuous discretizations and the different stabilization strategies are presented in Figure 3.7. For each case, the relative GMRES residual is plotted as a function of the number of GMRES iterations (on the left). The relative L_2-error between the numerical solution and a reference numerical solution is also shown (on the right). This error is computed by comparing the DDM solution u_n in each subdomain with the reference numerical solution u_{mono} computed on Ω_{glo} with the same mesh without domain decomposition procedure,

$$\text{error} = \sqrt{\frac{\sum_{n=1}^{N} \int_{\Omega_n} |u_n - u_{\text{mono}}|^2 \, \mathrm{d}\Omega_n}{\int_{\Omega_{\text{glo}}} |u_{\text{mono}}|^2 \, \mathrm{d}\Omega_{\text{glo}}}}. \tag{3.51}$$



(a) 2^{nd} degree: Residual history

(b) 2^{nd} degree: L^2-error history

(c) 4^{th} degree: Residual history

(d) 4^{th} degree: L^2-error history

Figure 3.7: Convergence of the relative residual and the relative error for each discretization and stabilization strategy. The computations are performed for $N_{\text{PML}} = 6$, and polynomial degrees $p = 2$ and 4.

The u-fields are discretized with hierarchical H^1-conforming basis functions of polynomial degree p equal to 2 and 4. For both boundary and transmission conditions, the PML thickness corresponds to $N_{\text{PML}} = 6$ mesh cells (*i.e.* $\delta_{\text{PML}} = 6h$) and shifted hyperbolic absorbing functions are used.

Let us note that the reference numerical solution u_{mono} is not exactly the same in all the cases. Then, the comparison is not carried out with the same reference problem. Indeed, because the discretization of the Lagrange multipliers is the same for both the exterior boundary condition and the transmission conditions, the reference problem depends on the considered discretization. This approach is chosen because in practice, when a discretization is chosen, it will be used everywhere.

For $p = 2$, the best convergence rate is provided by the continuous discretization (both versions) and the discontinuous discretization with both the polynomial degree $p + 1$ and the penalty term (Figure 3.7a). The decay of residual is slower with the

	$p = 2$	$p = 4$
Continuous + corner eq.	60	160
Continuous + penalty	60	160
Discontinuous(p)	56	154
Discontinuous($p-1$)	26	100
Discontinuous(p) + penalty	56	154
Discontinuous($p+1$) + penalty	98	220

Table 3.1: The number of degrees of freedom by element thickness for each strategy. The computations are performed for $N_{\text{PML}} = 6$, and polynomial degrees $p = 2$ and 4.

other approaches based on the discontinuous discretization. However, in nearly all the cases, the relative error decreases until a plateau, which the level depends on the case (Figure 3.7b). This behavior can be explained because, if the discontinuous discretization and/or the penalty strategy is used, the equivalence between the reference problem and the coupled subproblems is not exactly ensured, which introduces an error. The discontinuous discretization with both the higher polynomial degree and the penalty is the notable exception. Similar results are obtained with $p = 4$, except that both the relative residual and the relative error reach 10^{-6} more rapidly with the continuous discretization and the additional corner equation than with all the other approaches.

In order to quantify the relative cost of each strategy, we report in Table 3.1 the number of degrees of freedom required for the PML-based DtN, per element on the interfaces $\Sigma_{n,i}$ or $\Sigma_{n,i,j}$. Among the methods exhibiting the best convergence rate, the continuous approach are cheaper than the discontinuous discretization with polynomial degree $p + 1$. While in this 2D setting, the resulting difference in computational cost is not significant, it should be investigated further for 3D problems.

In the next sections, only the continuous discretization with the additional corner equation (called *selected continuous discretization*) and the discontinuous discretization with the higher polynomial degree and the penalty (called *selected discontinuous discretization*) are considered for the analyses.

For both selected discretizations, there is a sharp decay of the residual and the L_2-error between the third and the fourth GMRES iterations. This can be interpreted by considering that, at each iteration, information can be transferred only between neighboring subdomains. Given the considered domain partition and the position of the source in the lower left subdomain, four iterations are required to propagate the source accross all the subdomains. Because PML transmission conditions are particularly well-suited for this benchmark, the DDM solution is very close to the physical solution after only four iterations. Let us mention, that the relative error between the reference numerical solution (with any discretization) and the analytic solution (2.25) is equal to 6.8×10^{-3} for $p = 2$ and 6.3×10^{-3} for $p = 4$. These errors are higher that the relative errors observed between the DDM solutions and the reference solution after the four iterations.

4.3 Influence of the absorbing function in the transmission condition

Different absorbing functions as introduced by Equations (1.40) and (1.41) of Chapter 1 can be used in the PMLs. Smoothly increasing functions such as polynomial and hyperbolic functions are frequently chosen. In this work, we consider quadratic, hyperbolic and shifted hyperbolic functions defined as

$$\sigma_q(x) = \sigma^\star (x - L_x)^2 / \delta_{\text{PML}}^2, \tag{3.52}$$

$$\sigma_h(x) = 1/(\delta_{\text{PML}} - (x - L_x)), \tag{3.53}$$

$$\sigma_{hs}(x) = 1/(\delta_{\text{PML}} - (x - L_x)) - 1/\delta_{\text{PML}}, \tag{3.54}$$

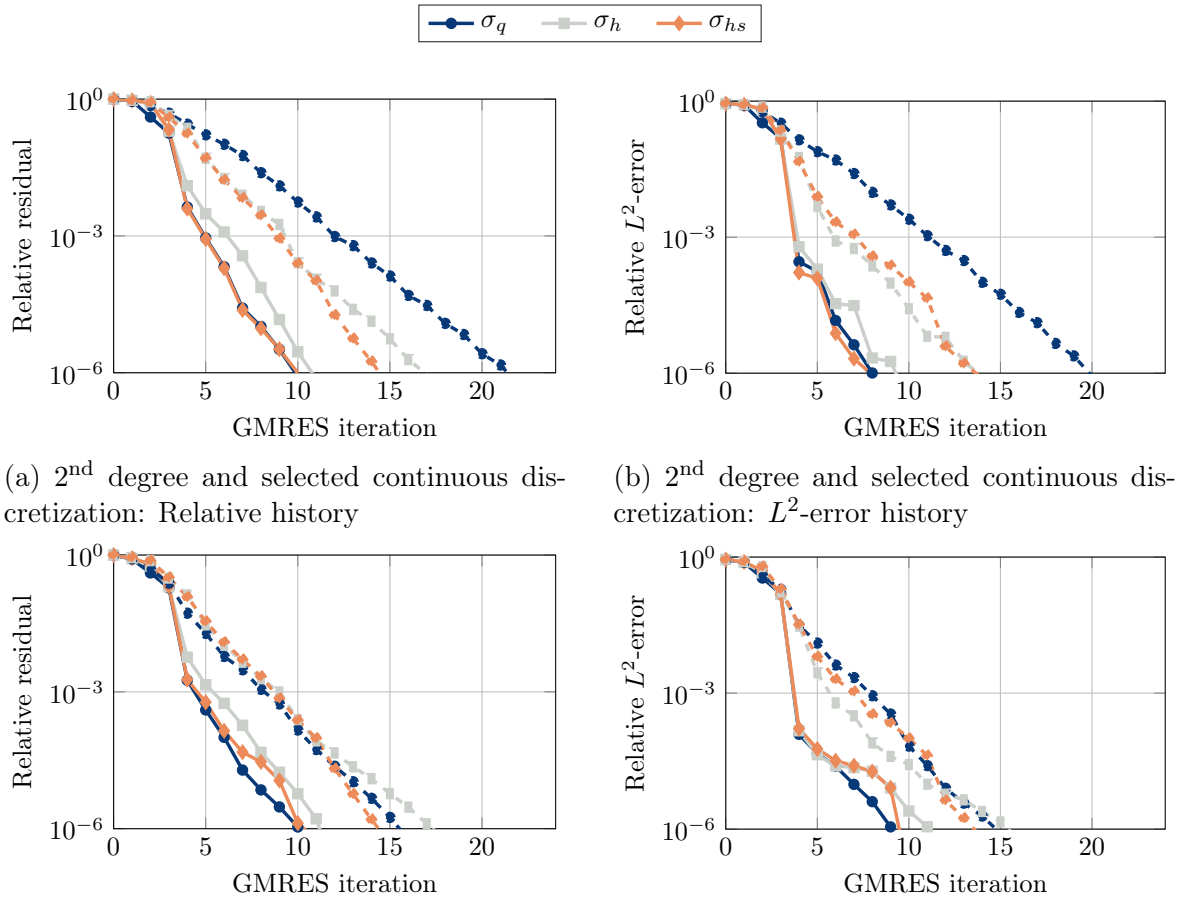
respectively. These functions are written for a PML in the x-direction with $x \in [L_x, L_x + \delta_{\text{PML}}]$, like the last line in Equation (3.50). The definitions are similar for the other PMLs. In the quadratic function, the parameter $\sigma^\star$ must be tuned. Here, the values $\sigma^\star = 86.435$ and 186 have been used for $N_{\text{PML}} = 1$ and 6, respectively.

Figure 3.8 shows the convergence history of the DDM process when the PML transmission conditions are tested with the different absorbing functions, and PML thicknesses corresponding to $N_{\text{PML}} = 1$ (dashed lines) and 6 (plain lines). The computations are performed for both selected discretizations, and second-degree polynomial basis functions. In all the cases, the hyperbolic absorbing function and $N_{\text{PML}} = 6$ have been used for the PMLs on the exterior border of the global domain. Therefore, for a given discretization, the reference numerical solution remains the same.

We observe that, with six-cells PMLs in the transmission conditions, the convergence is similar with the different absorbing functions for each discretization. With one-cell PMLs, the convergence is slower in all the cases. The differences between the absorbing functions remain rather small in the discontinuous case, but they are significant in the continuous case. These observations can be related to the quality of the PML as a good absorbing boundary treatment. The accuracy of the technique is not very sensitive to the choice of the absorbing function with thick layers, but the choice is much more critical with very thin layers. Therefore, we can expect that the choice is not critical in the DDM procedure with thick layers. In the remainder, shifted hyperbolic functions are used for both exterior conditions and transmission conditions.

4.4 Influence of the PML thickness in the transmission condition

In order to study the influence of the PML thickness in the transmission conditions on the efficiency of the procedure, the convergence history with PML thicknesses corresponding to $N_{\text{PML}} = 1$, 2, 4 or 6 mesh cells is presented in Figures 3.9 and 3.10. The results obtained with the standard impedance transmission condition proposed by Després [66] are also presented (dashed lines). In all the cases, PMLs with $N_{\text{PML}} = 6$ are used for the exterior border of the global domain, and the shifted hyperbolic absorbing function is used for both PML-based transmission and boundary conditions. The selected continuous and discontinuous discretizations have been tested with second and fourth degree basis functions for the u-fields.



(a) 2nd degree and selected continuous discretization: Relative history

(b) 2nd degree and selected continuous discretization: L^2-error history

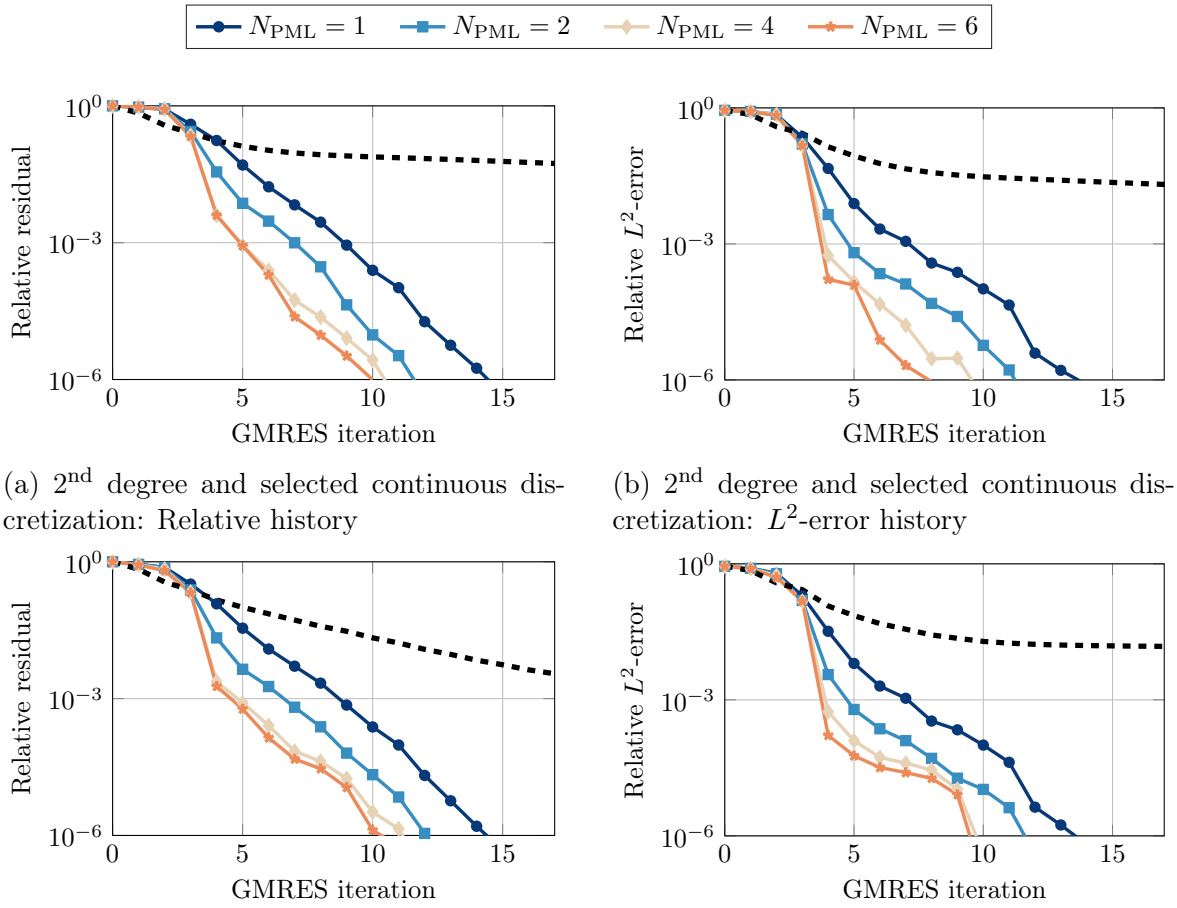
(c) 2nd degree and selected discontinuous discretization: Relative history

(d) 2nd degree and selected discontinuous discretization: L^2-error history

Figure 3.8: Convergence history of the second degree PML-based DDM for different PML types. The dashed curves show the convergence history of the one-layer-size PMLs while the plain ones show the convergence history of the six-layer-size PMLs.

In all the cases, the relative residual and the relative $L2$-error decrease with the number of iterations. They have approximately the same order of magnitude at each iteration. We observe that an increase of the PML thickness accelerate the convergence of the DDM process up to a particular point where an increase of N_{PML} does not change the convergence rate anymore. The convergence is not much faster with the $N_{\text{PML}} = 6$ than with $N_{\text{PML}} = 4$. For the cheapest PML, with only one mesh cell in the thickness, the convergence is slower than with thicker PMLs, but it is much faster than with standard impedance transmission conditions. The results are similar with second and fourth degree basis functions.

It is interesting to compare these results with those obtained using high-order ABCs (HABCs) presented in Section 4.2. Provided that the HABCs are complemented with appropriate corner compatibility conditions as presented, the overall performance of HABCs is very similar to the performance of the PML-based conditions proposed here. This is actually to be expected, as a formal equivalence between some type of HABCs (like CFABCs) and PMLs can be derived [100, 14]. Also note that the DDM based



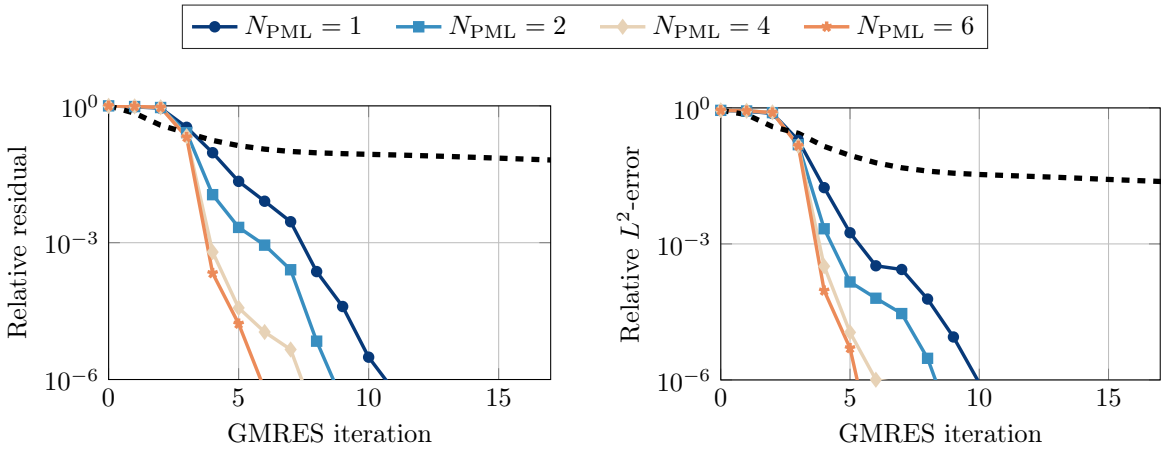
(a) 2nd degree and selected continuous discretization: Relative history

(b) 2nd degree and selected continuous discretization: L^2-error history

(c) 2nd degree and selected discontinuous discretization: Relative history

(d) 2nd degree and selected discontinuous discretization: L^2-error history

Figure 3.9: History of the relative GMRES residual (left) and the relative mono-domain L^2-error for different basis function orders and different number of layers in the PML. The dashed black curve corresponds to the results obtained when a 0^{th} order transmission condition is imposed on interface edges while a six-layer-size PML is still imposed on the boundary edges of the domain (the reference problem) (2nd degree).



(a) 4^{th} degree and selected continuous discretization: Relative history

(b) 4^{th} degree and selected continuous discretization: L^2-error history

(c) 4^{th} degree and selected discontinuous discretization: Relative history

(d) 4^{th} degree and selected discontinuous discretization: L^2-error history

Figure 3.10: History of the relative GMRES residual (left) and the relative mono-domain $L2$-error for different basis function orders and different number of layers in the PML. The dashed black curve corresponds to the results obtained when a 0^{th} order transmission condition is imposed on interface edges while a six-layer-size PML is still imposed on the boundary edges of the domain (the reference problem) (4^{th} degree).

on Perfectly Matched Discrete Layers (PMDLs) proposed in [14] implements a similar cross-point treatment at the discrete level, *i.e.* on the matrix form. Indeed, the node associated to a cross-point has to be known to build the matrix M referenced at the bottom of p. 77, under Equation (7) of [14]; see also Equation (2.2) of [132].

Before analyzing the influence of the wavenumber and the mesh density, let us also mention that the PML-based DDM scales as expected (and in the same way as HABC-based DDMs): the number of iterations increases linearly with the number of subdomains in each direction, which is normal for a one level domain decomposition solver without coarse grid.

4.5 Influence of the wavenumber and the mesh density

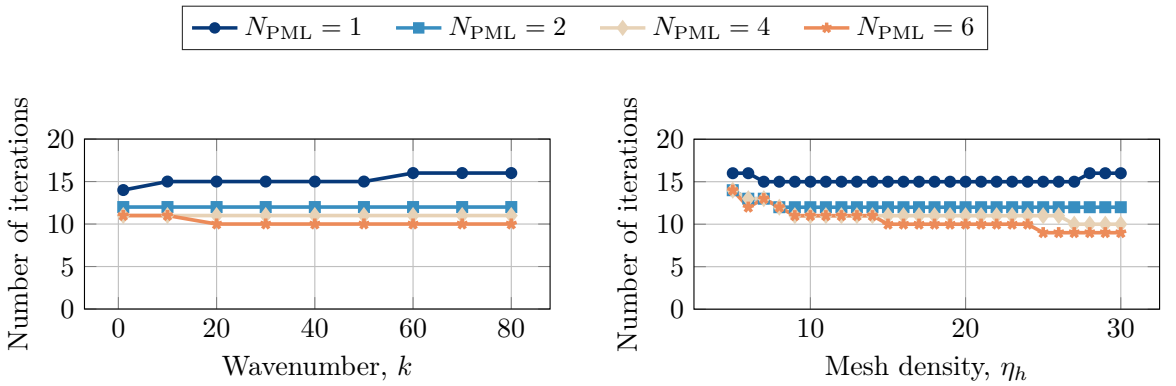
It is well known that the solution of high-frequency wave problems requires fine meshes with high-degree polynomial basis functions to decrease the dispersion error. Therefore, the efficiency of the solution procedures for large wavenumbers and fine meshes is a critical issue. Ideally, the influence the wavenumber and the mesh refinement on the convergence of the iterative procedure should be limited.

Figure 3.11 shows the number of GMRES iterations required to reach a relative residual lower than 10^{-6} as a function of the wavenumber k (left) and the characteristic number of vertices per wavelength η_h (right). The computations are performed for a given mesh density (number of mesh vertices by wavelength equal to 15) in the first case, and for a given wavenumber ($k = 4\pi$) in the second case. The results are presented for PML-based transmission conditions with $N_{\text{PML}} = 1, 2, 4$ or 6, and second and fourth degree basis functions for the u-fields.

In all the cases, the number of GMRES iterations slightly increases with the wavenumber and the mesh density for very thin PMLs (*i.e.* with only one or two mesh cells in the thickness), while it remains very stable for thick PMLs. The results are similar for second and fourth degree basis functions. These results indicate that the PML-based transmission conditions are efficient for high-frequency scattering problems, as soon as the layers are sufficiently thick.

5 Conclusion

In this chapter we have proposed a non-overlapping DDM with PML transmission conditions for the Helmholtz equation. The approach naturally takes into account cross-points for two-dimensional checkerboard domain partitions. It relies on Lagrange multipliers used for the weak coupling between subproblems defined on the rectangular subdomains and the surrounding PMLs. They are also used to compute the transmission variables for the DDM procedure. Two discretizations for the Lagrange multipliers and several stabilization strategies were compared. The best two converging approaches are the continuous discretization with additional corner equation, and the discontinuous discretization with higher polynomial degree and penalty. In addition to convergence rates, selecting the best approach among all presented discretizations and stabilization strategies might however depend on the user's software implementation and accuracy requirements. Indeed, $\mathbf{H}(\text{div})$-conforming basis functions might not be



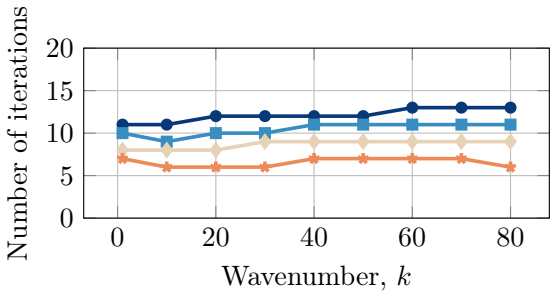
(a) 2nd degree and selected continuous discretization: wavenumber k

(b) 2nd degree and selected continuous discretization: characteristic number of vertices per wavelength η_h

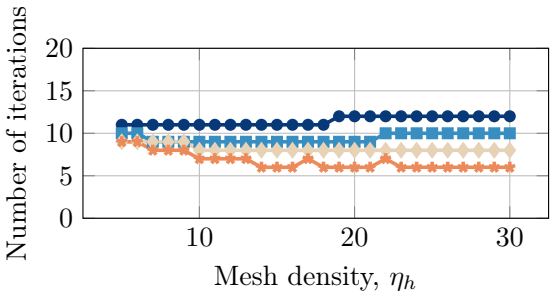
(c) 2nd degree and selected discontinuous discretization: wavenumber k

(d) 2nd degree and selected discontinuous discretization: characteristic number of vertices per wavelength η_h

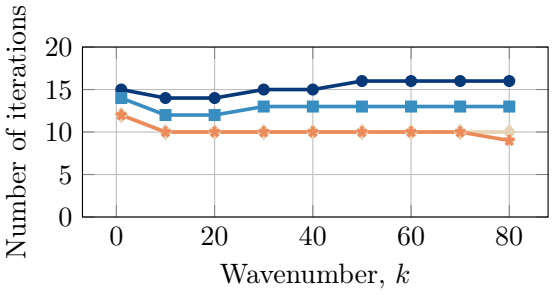
Figure 3.11a



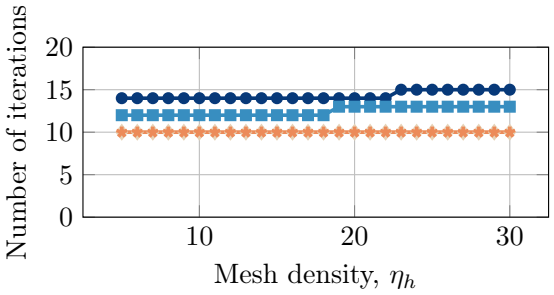
(e) 4^{th} degree and selected continuous discretization: wavenumber k



(f) 4^{th} degree and selected continuous discretization: characteristic number of vertices per wavelength η_h



(g) 4^{th} degree and selected discontinuous discretization: wavenumber k



(h) 4^{th} degree and selected discontinuous discretization: characteristic number of vertices per wavelength η_h

Figure 3.11b: Number of GMRES iterations needed to reach the relative residual of 10^{-6} as a function of the wavenumber k with a constant number of point by wavelength of 15 (left graphs) and as a function of the inverse of the characteristic number of vertices per wavelength η_h with a fixed wavenumber $k = 4\pi$ (right graphs), for different basis function orders and different number of layers in the interface PML.

implemented in all finite element codes; and adding corner treatments requires geometrical identifications and algebraic constraints that could be more or less straightforward to deal with depending on the computational framework.

In Chapter 4, the PML-based DDM will be applied to more challenging heterogeneous and three-dimensional test cases, and its performance compared with the HABC-based DDM that we will introduce in the previous chapter. In addition, the PML-based DDM will be also further extended to handle 2D and 3D elastodynamic simulations.

4 Performance comparisons

In this chapter, the performance of the methods presented in Chapters 2 and 3 is compared on acoustic applications and extensions to elastic applications. Two- and three-dimension mono-domain problems are considered first; domain decomposition resolutions are then addressed on both homogeneous and heterogeneous problems. Finally, the PML-based method of Chapter 3 is extended to elastic waves allowing the modelization of two- and three-dimension elastic problems.

1 Introduction

The accurate and fast resolutions of acoustic problems are crucial for applications such as ground characterization through inverse problems, as stated in the Introduction. Inverse problems require the resolution of a considerable number of direct problems. Therefore the resolution of these direct problems must be optimized. This is the goal of this chapter: numerical methods presented in Chapters 2 and 3 are applied in the computational frameworks GmshFEM and GmshDDM introduced in Part II to solve relevant acoustic and elastic problems.

This chapter is divided into six sections. First, Section 2 describes the computing environment in which the numerical benchmarks are run. Then 2D and 3D acoustic scattering problems in homogeneous media are considered in Section 3, and the mono-domain problem implementation in GmshFEM is studied with a focus on shared-memory parallel efficiency. Then, their DDM versions are addressed and the distributed-memory parallel efficiency is studied. While 2D numerical results are obtained with methods presented in Chapters 2 and 3, numerical results of 3D DDM scattering problem are presented using an 3D extension of the methods presented on Chapters 2 and 3. The 3D extension of cross-point treatment of HABC-based DDM are based on the corner treatment of HABC presented in Modave and al. [147]. Finally, a comparison of PML-based and HABC-based on 2D scattering problems is made to

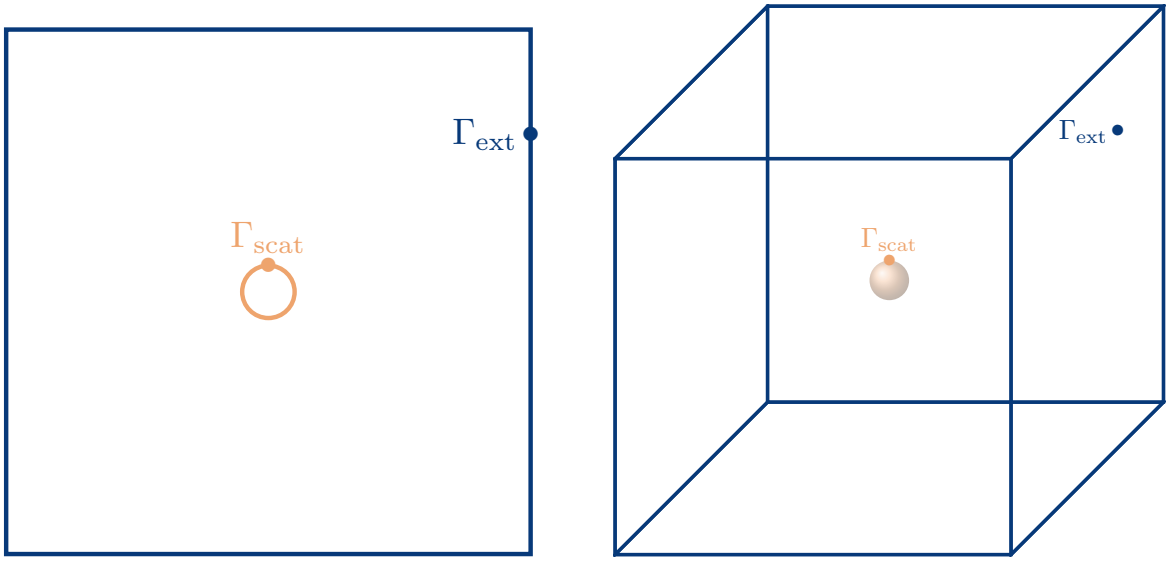
link the computation cost with the numerical accuracy of both methods. In Section 4, the resolution of heterogeneous underground acoustic problems using the DDM with cross-points treatment is presented. These problems are designed to use the maximal computational resources available in the running environment. The numerical results were obtained with GmshDDM (Chapter 6) and the related implementation of the test cases is available at the following address: `https://gitlab.onelab.info/gmsh/ddm/-/tree/master/examples/helmholtz/crossPoints`. In Section 5, 2D and 3D elastic scattering problems in homogeneous media are considered. These problems are similar to acoustic ones presented in Section 3. The study of these mono-domain problems in GmshFEM presents the shared-memory parallel efficiency. Finally, the DDM algorithm applied to heterogeneous problems similar to the acoustic 2D Marmousi and 3D Salt problems is presented in Section 6 as a future perspective of this thesis. Once again, the numerical results were obtained with GmshDDM (Chapter 6) and the related implementation of the test cases is available at the following address: `https://gitlab.onelab.info/gmsh/ddm/-/tree/master/examples/navier/crossPoints`.

2 Description of the computing environment

Computations are made on the NIC5 super-computer hosted at the University of Liège. This cluster consists of 70 compute nodes interconnected by a 100 Gbps Infiniband HDR interconnect. Each node is made of two 32 cores AMD Epyc Rome 7543 CPUs (second generation) running at the fixed frequency of 2.9 GHz and with 256 GiB of RAM. The memory is spread across sixteen 16 GiB DDH4 RAM modules at a frequency of 1600 MHz offering a data rate by RAM module of 3200 Mb/s. The motherboard has eight memory channels, and the maximum theoretical memory bandwidth is 190 GiB/s. One channel and eight cores build a NUMA node such that a cluster node is made of 8 NUMA nodes. One NUMA node's maximum theoretical memory bandwidth (*i.e.* on memory channel) is 23.75 GiB/s.

Before running any jobs on NIC5, this architecture must be considered, especially the thread placement that plays a vital role in the efficiency of GmshFEM. Indeed if threads move from one CPU core to another, data cache and NUMA data locality are lost. Therefore, threads are pinned to the cores for each run using the OpenMP environment variable `OMP_PLACES=cores`. An efficient NUMA node must contain all data needed to run a job assigned to its eight threads. Otherwise, data must be recovered for another NUMA node, and the overall efficiency is bounded by the per-channel bandwidth (*i.e.* 23.75 GiB/s) and not by the global memory bandwidth (*i.e.* 190 GiB/s).

The second important parameter defines the way threads are distributed over cores. This behavior is controlled using the OpenMP environment variable `OMP_PROC_BIND=spread` that asks to spread threads over all available cores such that from 1 to 8, threads are scattered on different NUMA nodes, then from 9 to 64, NUMA nodes are filled by keeping the number of threads by NUMA node well-balanced. It is the behavior used to run the applications of this chapter. The other available parameter is `OMP_PROC_BIND=close` that is used to ask to keep the thread as close as possible. Therefore from 1 to 8 threads, only the first NUMA node will be filled. Then for 9 to 15, the second NUMA node is filled, and so on. The spread policy is better for our



(a) Geometry of the 2D problem.

(b) Geometry of the 3D problem.

Figure 4.1: Geometries of the scattering problems.

application since it allows us to use the memory channel mechanism even with a small number of threads. For instance, a code running with eight threads has a maximum theoretical memory bandwidth of 190 GiB/s with the spreading policy, while only 23.75 GiB/s with the close one.

3 Acoustic scattering problems

This section studies academic acoustic scattering problems similar to those presented in Chapters 2 and 3. The scattering of an incident plane wave $u_{\text{inc}} = e^{\imath\, kx}$, traveling in the direction of the basis vector $\mathbf{e}_x$, on either a soft cylinder (2D problem) or a soft sphere (3D problem) is computed. These problems are both modeled by the following problem on the domains of Figures 4.1a or 4.1b.

$$\begin{cases} \Delta u + k^2 u = 0 & \text{on } \Omega, \\ u = -u_{\text{inc}} & \text{on } \Gamma_{\text{scat}}, \\ \partial_{\mathbf{n}} u - \mathcal{B} u = 0 & \text{on } \Gamma_{\text{ext}}, \end{cases} \tag{4.1}$$

where $\mathcal{B}$ is a boundary operator. The computational domain is made of a square box (resp. a cube box) of size 10 with a disk (resp. a sphere) of diameter size of 1 at the center of the box. The space between the box and the scattering object is meshed with first-order simplex elements (*i.e.* triangles in 2D and tetrahedra in 3D). The mesh size is chosen so that the number of points by wavelength equals 10.

In the following subsections, the multi-threaded efficiency of GmshFEM will be first studied on the mono-domain application of Problem 4.1. Then its domain decomposition resolution with both cross-point treatments will be presented. In addition, the efficiency of GmshDDM will be analyzed too.

Basis degree	2D	3D
2	1,164,044	789,172
4	4,652,496	6,222,228
6	10,465,356	20,873,608
8	18,602,624	//
10	29,064,300	//

(a) Problem size (number of DoFs) of the 2D and 3D acoustic scattering problem for different basis function degrees.

Basis degree	2D	3D
2	3.24	38.0
4	14.9	657
6	39.6	?
8	84.4	//
10	?	//

(b) Estimated memory needed to factorize the finite element matrix using MUMPS. The data are given in gibibyte (GiB). Character '?' means that MUMPS could not compute the estimated factorization memory cost.

Table 4.1: Number of degree of freedoms and estimated memory needed to store the LU factorization for the 2D and 3D acoustic scattering problems.

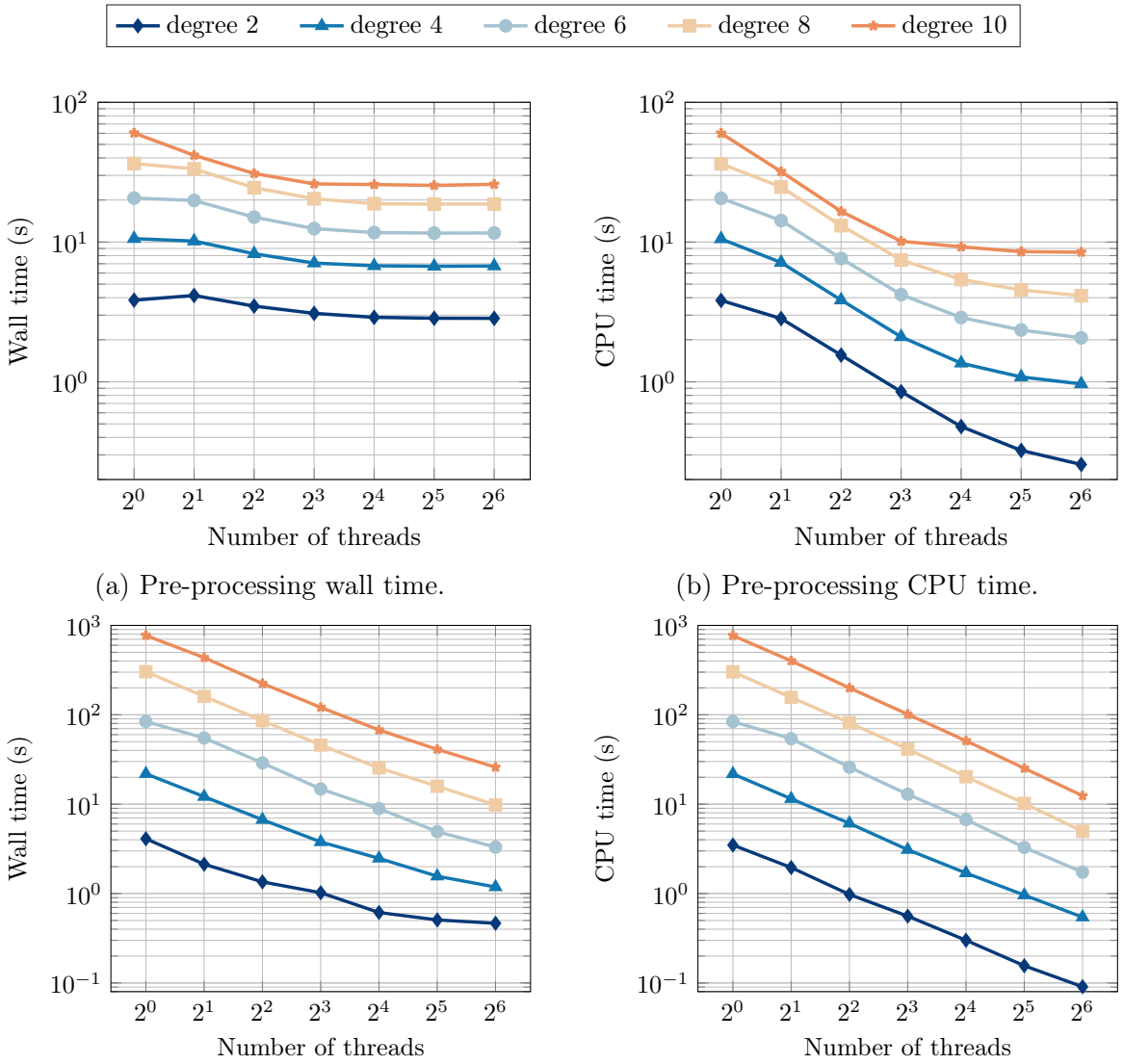
3.1 The mono-domain acoustic scattering problem

This section studies 2D and 3D mono-domain acoustic problems. The basis functions degrees are set to 2, 4, 6, 8, and 10,for the 2D problem and, 2, 4, and 6 for the 3D one. The Gauss quadratures used are exact to integrate a polynomial of degree twice the degree of the basis functions such that the mass term of Equation 4.1 is exactly integrated. Wavelengths are chosen such that the number of wavelengths over the geometry equals 50 for the 2D and 10 for the 3D problems. By doing so, and knowing that the discretization of 10 points par wavelength is kept, the total number of triangles for the 2D problem is 581,102, while the total number of tetrahedra for the 3D problem is 4,371,840. The resulting problem sizes are given in Table 4.1a which reports the number of degrees of freedom (*i.e.* the finite element system size) for each configuration.

The reported timings focus on the pre-processing and assembly part of GmshFEM. Indeed, as the sparse system is solved using the third-party library MUMPS [7, 8], we do not have any control over its multi-threaded efficiency. As explained earlier, the memory scaling in the 3D case (see Table 4.1b) is the primary limitation of mono-domain simulations, and motivate the DD approach, whose performance will be analyzed in the next section. In a nutshell, the pre-processing phase builds the dictionary of degrees of freedom and computes the pattern of the finite element matrix. The assembly phase is divided in two steps: first, an initialization step evaluates at all Gauss points the basis functions, the Jacobians and the functions involved in the formulation; then the entries of the finite element matrix are computed. These algorithmic phases will be explained in detail in Chapter 5.

For each simulation, a full node is reserved (*i.e.* 64 CPU cores), so timing cannot be skewed by other processes that may run on the same node. Figure 4.2 reports both wall and CPU times of the pre-processing (Figures 4.2a and 4.2b) and of the assembly part (Figures 4.2c and 4.2d) for the 2D benchmark. The same results for the 3D benchmark are reported by Figure 4.3.

The results show that the pre-processing phase does not scale very well. These



(a) Pre-processing wall time.

(b) Pre-processing CPU time.

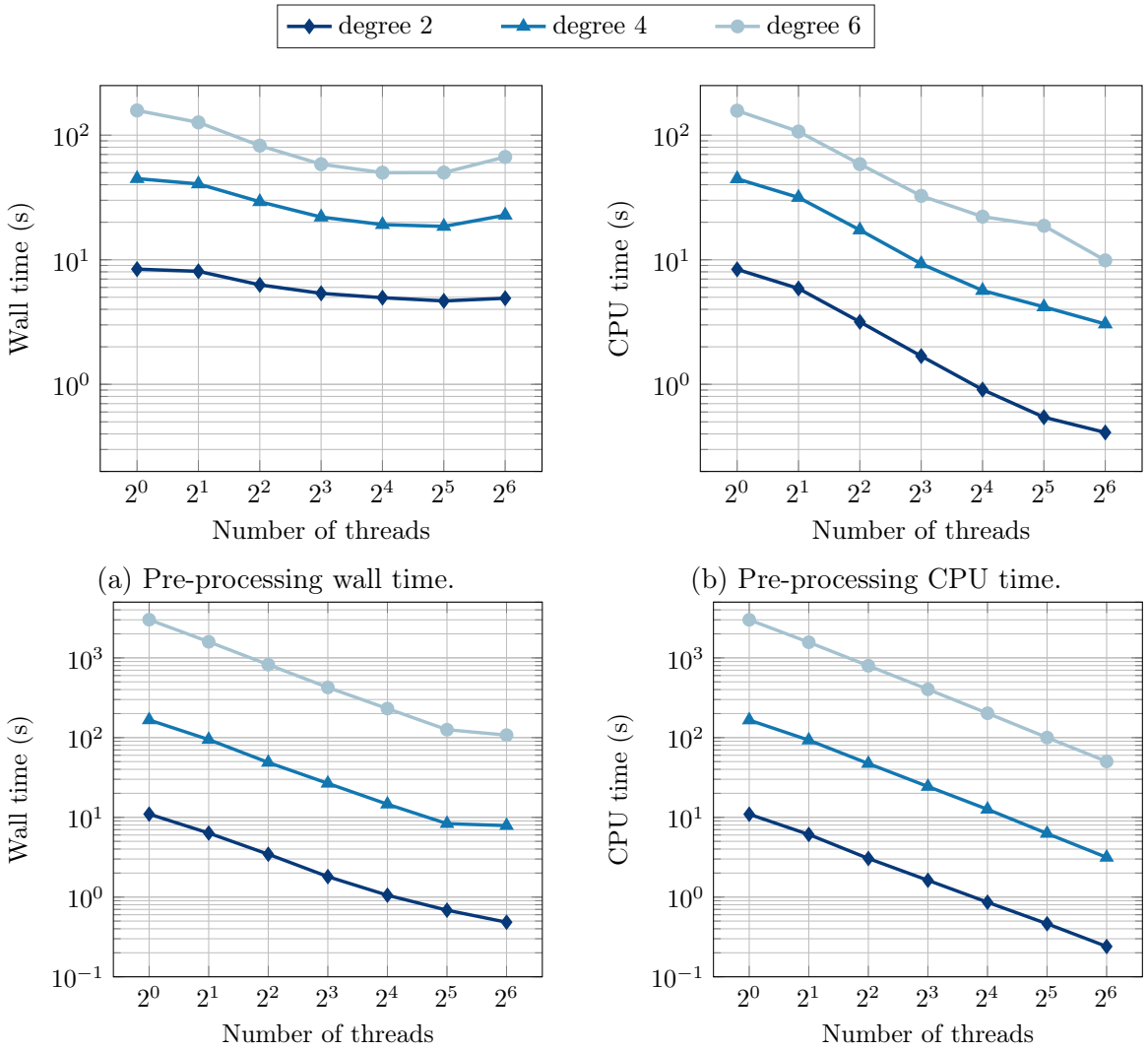
(c) Assembling wall time.

(d) Assembling CPU time.

Figure 4.2: Pre- and assembly time of the 2D scattering problem for different basis function degrees. Both wall and CPU time are reported such that the influence of the memory allocation can be estimated.

results are coherent with the expectation since sequential memory allocations dominate pre-processing. Indeed as will be explained in Section 3.1 of Chapter 5, the pre-processing is in charge of both building the degree of freedom dictionary and computing the pattern of the finite element matrix. These two operations need memory allocation during execution; therefore, the parallel efficiency cannot be optimal. Nevertheless, by looking at the CPU time instead of the wall time (Figures 4.2b and 4.3b), one can observe that the part of the pre-processing that can be parallelized (*e.g.* the Dirichlet condition evaluation, the DoF ordering algorithm, *etc.*) scale ut to eight threads.

The multi-threaded efficiency of the assembly process is significantly better than the pre-processing one. Nevertheless by looking to Figures 4.2c and 4.3c, the scaling



(a) Pre-processing wall time.

(b) Pre-processing CPU time.

(c) Assembling wall time.

(d) Assembling CPU time.

Figure 4.3: Pre- and assembly process time of the 3D scattering problem for different basis function degrees. Both wall and CPU time are reported such that the influence of the memory allocation can be estimated.

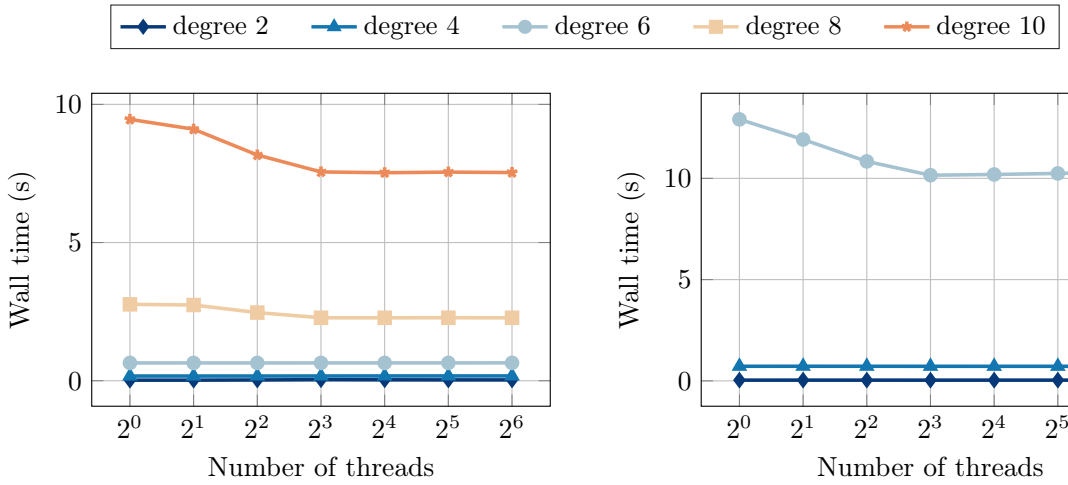
is not optimal when low-degree basis functions are used, especially in 2D. This can be explained by looking at Figure 4.4, which shows the part of the wall assembly time spent on allocating the finite element matrix based on the pattern computed in the pre-processing step and the amount that is spent on the initialization of the assembly process. The allocation of the matrix (one array of size equal to the number of non-zeros) is done, if necessary, at the beginning of the assembly part for code design purposes. The initialization of the assembly process computes the Jacobians, the basis functions, and allocates the array of finite element matrix indices. These two steps are mainly dominated by memory allocations of larger arrays, namely, the matrix array, the Jacobian array, and the indices array.

As expected, Figures 4.4a and 4.4b show that allocation times are almost constant no matter the number of threads used. Even if between 1 to 8 threads, the wall time of

the matrix allocation seems to decrease a bit, probably due to a kind of NUMA effect during the initialization of the array. On a single thread, the part of the time spend on this memory allocation is negligible over the total assembly time. Indeed, it from 0.63% with second-degree basis functions to 1.23% with tenth-degree basis functions for the 2D benchmark, while for the 3D benchmark, it goes from 0.33% to 0.43%. Nevertheless, when the number of threads increases, this effect cannot be neglected anymore. At 64 threads, the ratio of allocation time over the total assembly time goes from 6.42% to 29.05% for the 2D benchmark and from 4.76% to 9.68% for the 3D benchmark.

An analysis of Figures 4.4c and 4.4d indicates that the assembly initialization time is approximately reduced by 2 from 1 to 4 threads, for both benchmarks and all degrees. This is because the assembly initialization is not entirely dedicated to memory allocations but also some parallelized computations of Jacobian. Nevertheless, as our mesh is made of first-order simplex elements, the Jacobian computation is cheap such that memory allocations remain dominant in the initialization process. Opposite to the matrix allocation, the ratio of the initialization time over the total assembly time decreases when the basis function degree increases. On a single thread, it varies from 13.71% with second-degree basis functions to 0.67% with tenth-degree basis functions for the 2D benchmark and from 6.53% with second-degree basis functions to 0.43% with sixth-degree basis functions. Again, when the number of threads increases, the time spent on the assembly initialization over the total assembly time increases.

When both the matrix allocation wall time and the initialization wall time are subtracted from the total wall time of the assembly process as show in Figures 4.2c and 4.3c, Figures 4.4e and 4.4f are obtained. These remainder wall times are close to the CPU times reported on Figures 4.2d and 4.3d. Therefore it is decided to present in Figure 4.5 the multi-threaded parallel efficiency of the assembly algorithm on the CPU time to extract unparallelizable memory allocation effects.



(a) Matrix allocation wall time for the 2D benchmark.

(b) Matrix allocation wall time for the 3D benchmark.

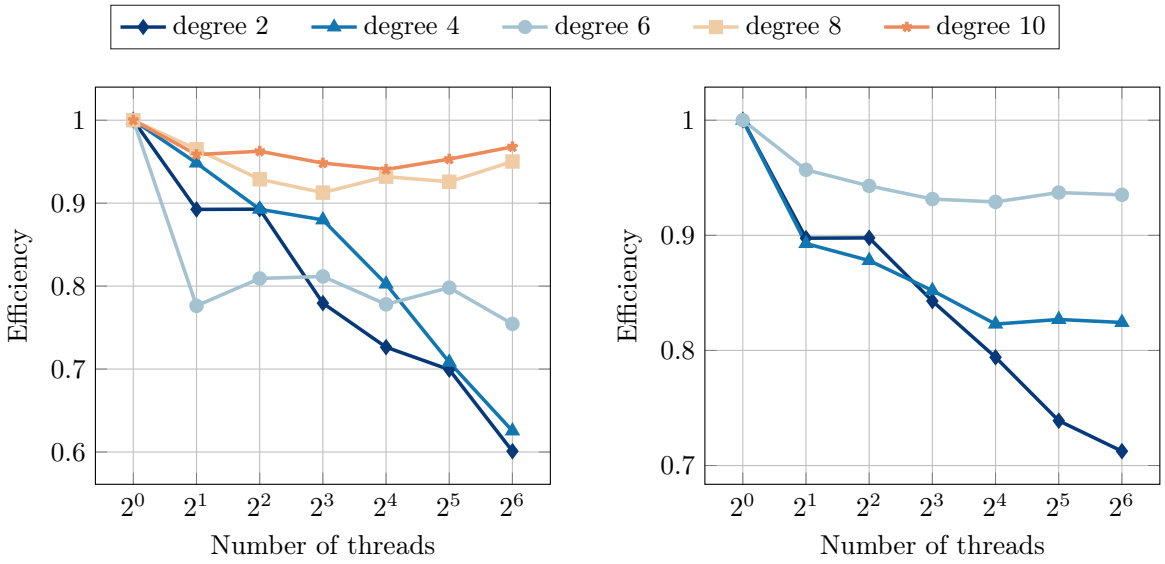
(c) Assembling initialization wall time for the 2D benchmark.

(d) Assembling initialization wall time for the 3D benchmark.

(e) Remainder wall time obtained by subtracting the total assembly wall time from the finite element matrix allocation wall time and the assembly initialization wall time of the 2D benchmark.

(f) Remainder wall time obtained by subtracting the total assembly wall time from the finite element matrix allocation wall time and the assembly initialization wall time of the 3D benchmark.

Figure 4.4: Comparison of the finite element matrix allocation wall time, the assembly initialization wall time, and the remainder wall time obtained by subtracting the total assembly wall time from the two previous ones.



(a) Efficiency of the 2D assembly process.

(b) Efficiency of the 3D assembly process.

Figure 4.5: Multi-threaded parallel efficiencies of the finite element assembly process computed on the CPU time for both 2D and 3D scattering problems.

Number of subdomain in the x-direction	Number of subdomain in the y-direction	Total number of subdomains	Number of CPUs	Wavenumber (k)
1	3	3	24	10 π
1	5	5	40	12.91 π
3	3	9	72	17.32 π
3	5	15	120	22.36 π
5	5	25	200	28.87 π

(a) 2D benchmark.

Number of subdomain in the x-direction	Number of subdomain in the y-direction	Number of subdomain in the z-direction	Total number of subdomains	Number of CPUs	Wavenumber (k)
1	1	3	3	24	π
1	1	5	5	40	1.19 π
1	3	3	9	73	1.44 π
1	3	5	15	120	1.71 π
3	3	3	27	216	2.08 π

(b) 3D benchmark.

Table 4.2: Parameters used for the weak scaling analysis.

3.2 The domain decomposition acoustic scattering problem

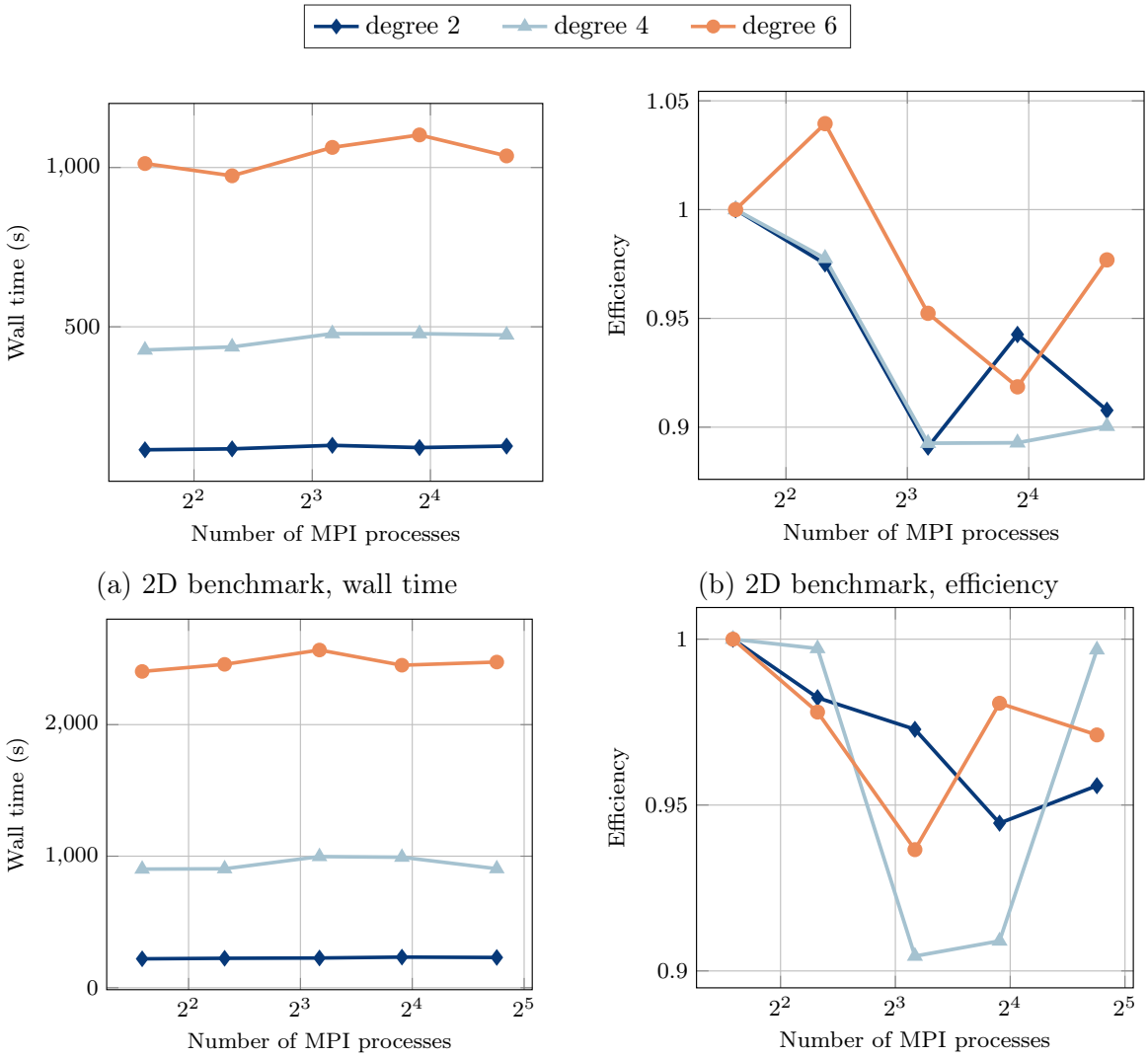
As shown in the previous section, when high-frequency simulations are considered, the number of unknowns and the LU factorization memory cost rapidly overtake the available memory on a single super-computer node. That is why the DD strategy is used.

In this section, the distributed memory parallelization efficiency is addressed on both the 2D and 3D scattering problems using the HABC-based or PML-based DDM with cross-point treatment presented in Chapters 2 and 3, respectively. Then a comparison of both approaches in terms of numerical precision and computational cost is carried out.

3.2.1 Distributed-memory parallel efficiency

A weak scalability test is assessed on the 2D and 3D scattering problems to study the computation efficiency. The total number of unknown is increased in a same way as the number of subdomain by refining the mesh; the wavenumber is chosen to keep 10 points by wavelength. Each subdomains are assigned to one MPI process. Furthermore, based on the shared-memory efficiency of Section 3.1, it is chosen to use eight cores by MPI process.

The square domain of Figure 4.1a and the cube domain of Figure 4.1b are partitioned in rectangles, or in rectangular parallelepipeds such that all cross-points have right angles. Moreover, the partition strategy is chosen such that the scattering object (the cylinder or the sphere) is contained inside only one subdomain. The problem size with the number of subdomains, the wavenumber, and the total number of CPUs used are listed in Table 4.2. Note that each subdomain is attached to a process placed on a unique node.



(a) 2D benchmark, wall time

(b) 2D benchmark, efficiency

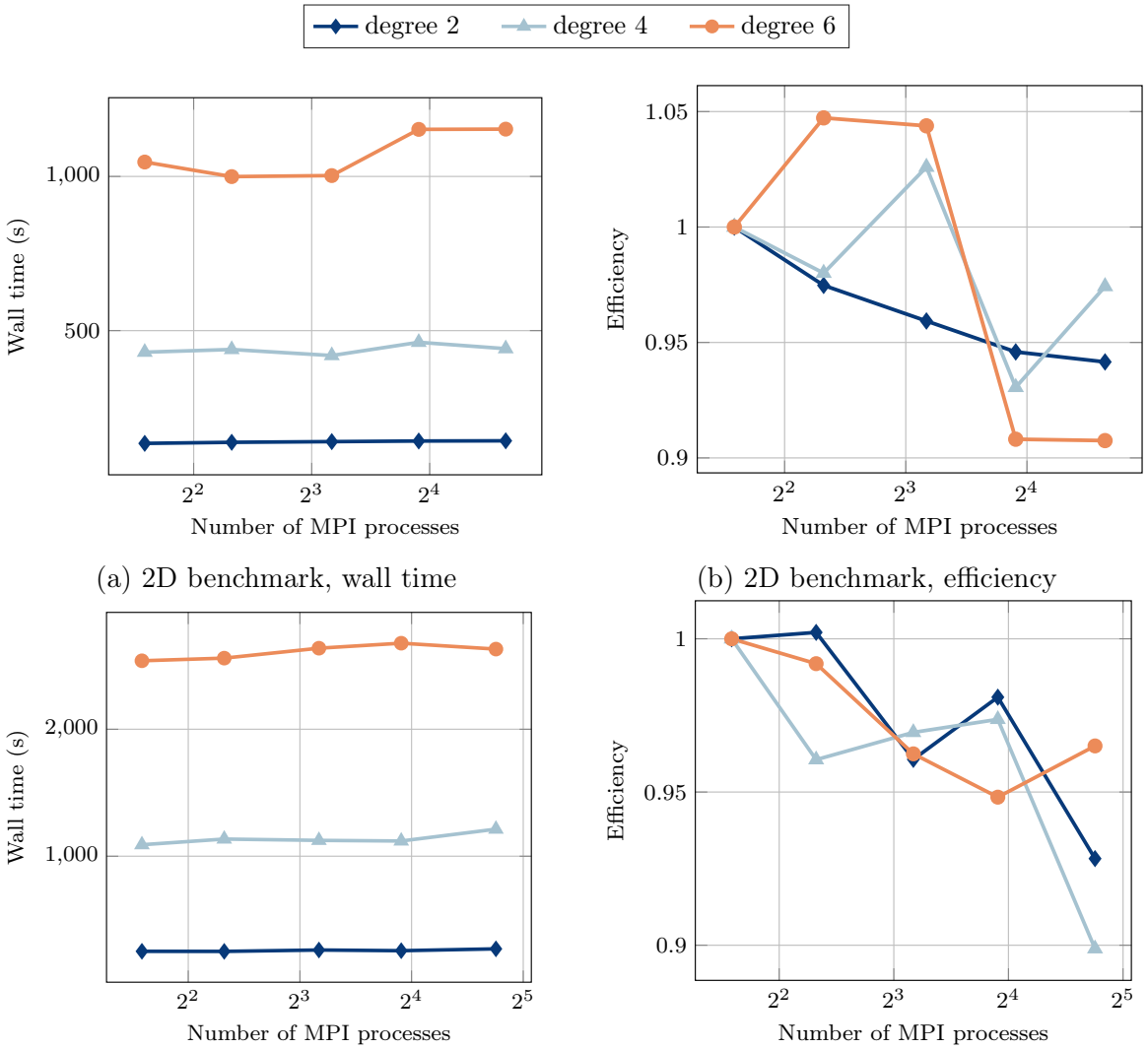
(c) 3D benchmark, wall time

(d) 3D benchmark, efficiency

Figure 4.6: Weak scaling analysis of GmshDDM using the HABC with cross-point treatment as presented in Chapter 2.

In Figures 4.6 and 4.7, the weak scaling analysis shows the efficiencies of the distributed-memory implementation when both methods presented in Chapters 2 and 3 are used. Furthermore, the benckmark is run using transmission condition of level 6 *i.e.*, namely the number of HABC auxiliary fields (N_{HABC}) or the number of PML layers (N_{PML}). In addition, second-, forth-, and sixth-degree basis functions are used. Concerning the HABC-based DDM, all simulations are run using the rotation parameter ϕ equal to 0.3π. Concerning the PML-based DDM, all computations are limited to the continuous approach with hyperbolic-shifted PML as presented in Chapter 3.

As seen in Figures 4.6b, 4.6d, 4.7b, and 4.7d, the distributed-memory efficiency is almost perfect for all configurations. The overall efficiencies are between 90% and 100%.



(a) 2D benchmark, wall time

(b) 2D benchmark, efficiency

(c) 3D benchmark, wall time

(d) 3D benchmark, efficiency

Figure 4.7: Weak scaling analysis of GmshDDM using the PML with cross-point treatment as presented in Chapter 3.

3.2.2 Comparison between HABC- and PML-based DDM

In this section a comparison with the HABC-based and PML-based DDM with cross-point treatment as presented in Chapters 2 and 3 is carried out. The analysis is done on the 2D scattering problems partitioned into a 3×3 grid. The frequency is set to 4π with a mesh density chosen such that they are 10 points by wavelength. Once again, the HABC rotation parameter ϕ equal to 0.3π, and the continuous approach PML-based DDM with hyperbolic-shifted type are considered.

First, let us consider a benchmark where a Sommerfeld radiation condition is enforced on the exterior boundaries, while a HABC or a PML with a varying N_{PML} or N_{HABC} is enforced on the interfaces. The GMRES residual is set to 10^{-8}. Fields are discretized using fourth-degree hierarchical basis functions. As can be seen in Figure 4.8 the PML-based DDM is faster at low-degree, namely at degree 1 and 2. Above, the HABC-based DDM becomes faster than the PML one. Furthermore, note that

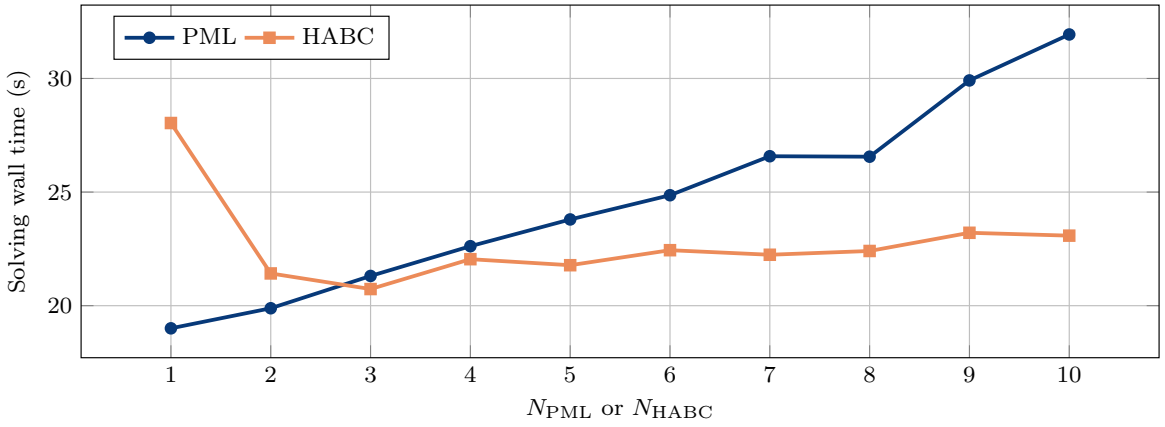


Figure 4.8: Influence of N_{PML} or N_{HABC} on the DDM the computational time when 0^{th} boundary condition is imposed on exterior boundary of the global domain.

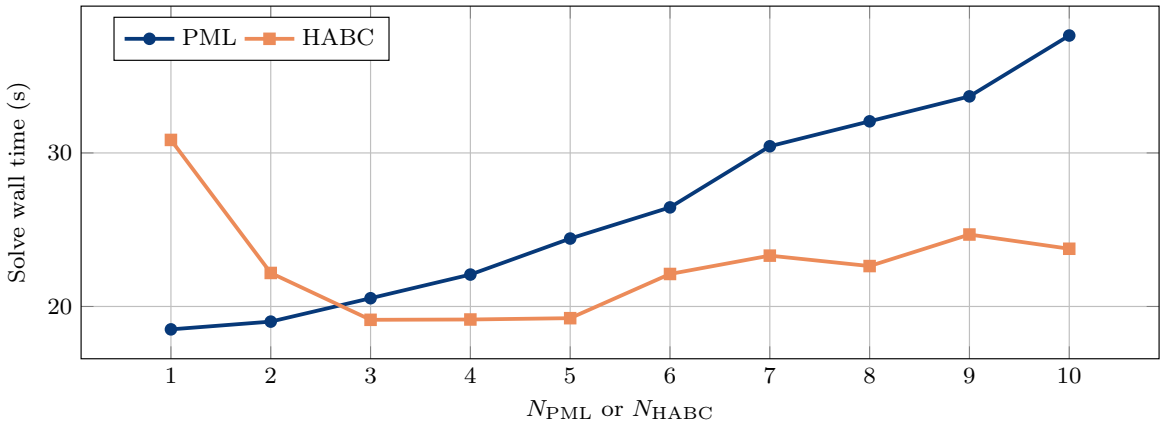


Figure 4.9: Influence of N_{PML} or N_{HABC} on the DDM the computational time when the exterior boundary condition is the same as the transmission boundary condition one.

the complexity of the PML-based DDM in terms of N_{PML} is higher than the HABC in terms of N_{HABC}. This behavior is explained by comparing the over-cost induced by the assembly in the surface PML region for the PML method versus the over-cost caused by the assembly on the interfaces for the HABC method. As the assembly cost on the surface is much higher than the one on line elements, the PML-based DDM is more influenced by the method degree than the HABC one.

The previous analysis is interesting since the mono-domain problem are the same because the exterior boundary condition is the same for both methods. Nevertheless, using high-order transmission conditions with a simple inefficient low-order condition on the exterior boundaries is a bit non-sense. Let us, therefore, run the same simulations but with the same interior and exterior boundary conditions to obtain the results as shown in Figure 4.9. The change in exterior condition does not impact the observation. The PML-based DDM is still more efficient than the HABC-based one for $N_{\text{PML}} = 1$ and 2; above, the HABC-based DDM is faster.

Since the exterior boundary condition is modified for each simulation, the numerical error at the convergence of the GMRES is not identical. Therefore a comparison

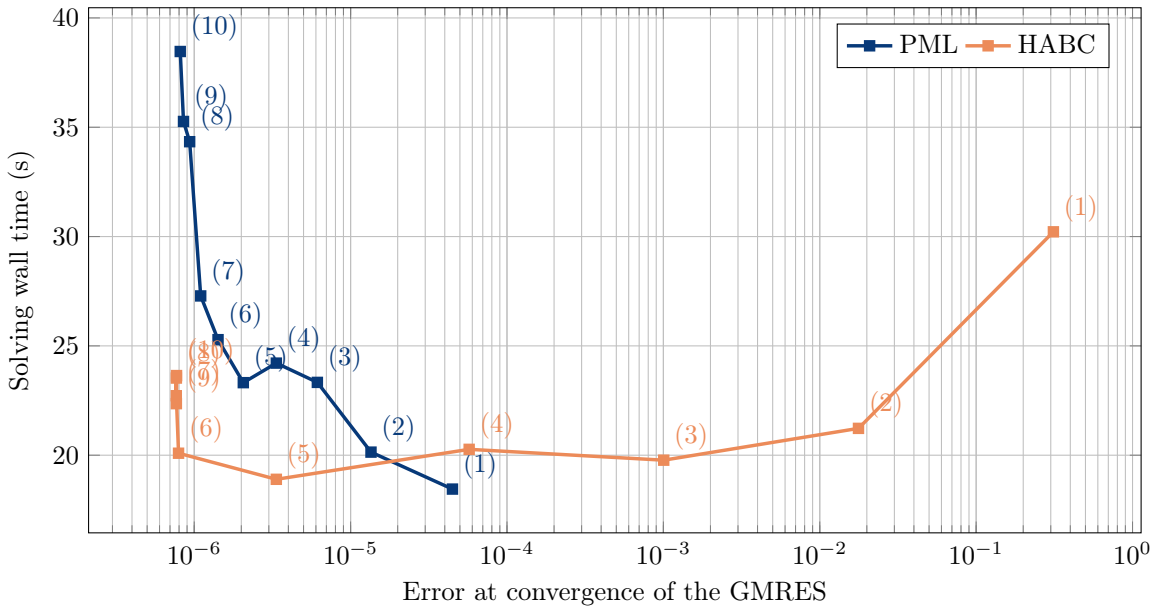


Figure 4.10: Comparison of HABC- and PML-based DDM in term of numerical error at convergence and computation time. The label next to the nodes indicates the value of N_{HABC} or N_{PML}. A basis function of degree 4 is used.

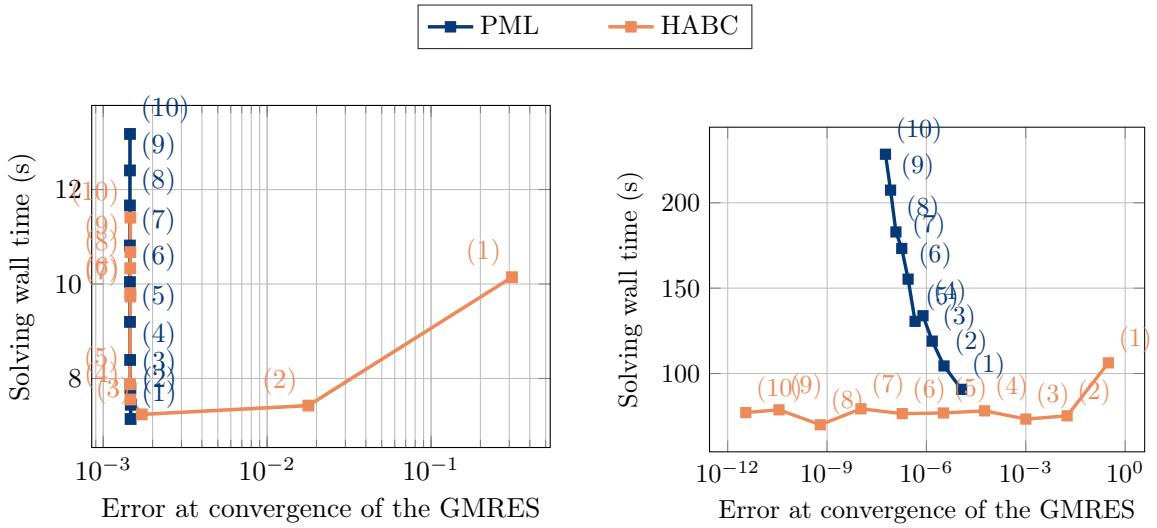
between the numerical error at convergence versus the solving time is presented in Figure 4.10. In practice, N_{HABC} and N_{PML} are increased from 1 to 10, while the numerical error at convergence and the solving time are measured for both methods.

As noticed, the PML-based method is more precise than the HABC one at a low-order. Indeed, the HABC with $N_{\text{HABC}} = 4$ has approximately the same precision (around 10^{-4}) as the PML with N_{PML}. Furthermore, the PML with $N_{\text{PML}} = 1$ is a bit faster than the HABC with $N_{\text{HABC}} = 4$. Nevertheless, the solving time of the PML-based approach grows faster than the HABC-based approach.

On Figure 4.11, the same analysis is reproduced using second- and eighth-degree basis functions. Once again, the PML-based DDM is more accurate than the HABC-based one at a low order, and this behavior is inverted at a high order.

Using second-degree basis functions (Figure 4.11a), the PML-based approach is stuck to an accuracy of about 10^{-3} for all considered degrees, probably because the accuracy of this method is better than the numerical error in this configuration. Therefore, the PML-based accuracy is bounded by this numerical error. As observed with the fourth-degree basis function, the HABC-based method reaches the same accuracy as the PML-based one at $N_{\text{HABC,PML}} = 4$. Then as for the PML-based approach, increasing N_{HABC} does not improve the efficiency due to the numerical error.

The results obtained with eighth-degree basis functions (Figure 4.11b) confirm that the PML-based approach is more accurate than the HABC-based on at low order. Nevertheless, the HABC-based method is faster than the PML-based one, except for $N_{\text{HABC}} = 1$. The error can decrease with N_{HABC} or N_{PML} because the numerical error does not bound the accuracy as with the second-degree basis functions. It can be noticed that the accuracy of the HABC-based method increases faster than the PML-based one and for a constant computational time. In addition, an increase in



(a) Degree 2

(b) Degree 8

Figure 4.11: Comparison of HABC- and PML-based DDM in terms of numerical error at convergence and computation time with basis functions of degree 2 and 8. The label next to the nodes indicates the level of the method N_{HABC} or N_{PML}.

N_{PML} also increases the computational time. Therefore, the HABC-based approach is recommended over the PML-based one at the eighth-degree basis functions.

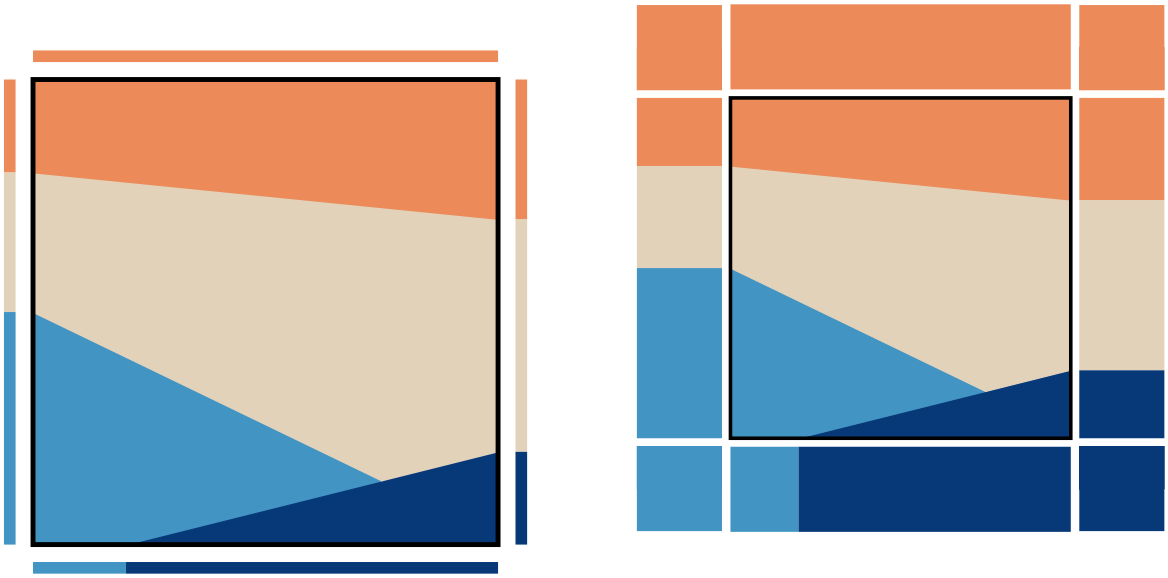
In conclusion, for 2D problems, the PML-based approach with a low order should be recommended over the HABC-based approach if a medium accuracy is required and low-degree basis functions are used. Conversely, the HABC-based approach is more suitable for high-precision requirements and when high-degree basis functions are used.

4 Acoustic problems in complex heterogeneous media

This section is devoted to the resolution of underground acoustic simulations. Therefore the material property, *i.e.* the sound velocity and consequently the wavenumber, is no longer constant. The accurate and fast resolution of these problems are crucial for applications such as ground characterization through inverse problems, as stated in the Introduction.

Figure 4.12 shows how the heterogeneous wavenumber distribution are considered inside methods presented in Chapters 2 and 3. For the HABC-based method, the wavenumber appearing in the mathematical expressions of Chapter 2 are simply evaluated at each Gauss points on the interface using a bi- or tri-linear interpolation of the material property map, without any substantial modification of the methods. For the PML-based method, the wavenumber distribution on each interface are extruded inside the edge PMLs and the wavenumber distrubution on each PML boundary are extruded inside the corner PMLs as depicted in Figure 4.12b.

In the following subsections, 2D and 3D underground benchmarks are presented. The 2D benchmark is based on the Marmousi model [47]. It is an artificial underground acoustic model developed in 1990 to represent complex structures comparable to those



(a) HABC-based DDM and heterogeneous wavenumber.

(b) PML-based DDM and heterogeneous wavenumber.

Figure 4.12: Treatment of the heterogeneous wavenumber distribution applied to the HABC- and PML-based DDM methods presented in Chapters 2 and 3. The examples show an schematic underground with four layers of material (one color by layer).

that are met in real applications. The 3D benchmark is also based on a synthetic data set called the Salt 3D model [81]. The frequency of each problem is chosen in both cases to fit the maximum resources available to a user on the NIC5 super-computer, *i.e.* 230 CPUs with a total memory of 920 GiB.

4.1 The Marmousi model

The Marmousi data is based on a heterogeneous sound velocity distribution in a 9192-meter-long and 2904-meter-deep underground. The sound velocity goes form 1500 m/s to 5500 m/s. The wave frequency is set to 100 Hz such that the wave number distribution is between 0.11 m^{-1} and 0.42 m^{-1} as Figure 4.13 depicts. This leads to a wavelength between 15 m and 57 m such that the number of wavelengths over the domain is between 96 and 368. The characteristic mesh size is set to 3 m such that the number of points by wavelengths is between 5 to 19, leading to 6.7 millions of straight triangles. All computations are made with fields discretized using forth-degree hierarchical basis functions. The integration scheme is exact for a polynomial of degree 9.

This mesh is partitioned into a grid of 10 by 4 leading to a total of 40 subdomains. Each of them is assigned to an MPI process that runs on 8 CPUs such that the total number of CPUs used is equal to 320, the maximal number of CPUs available to a user on NIC5. HABCs or PMLs with N_{HABC} or N_{PML} equal to 2, 4, and 6 are imposed on the exterior and interior boundaries. A punctual Dirichlet condition is enforced on each cross-point on the top boundary; see the blue dots of Figure 4.13. Note that an

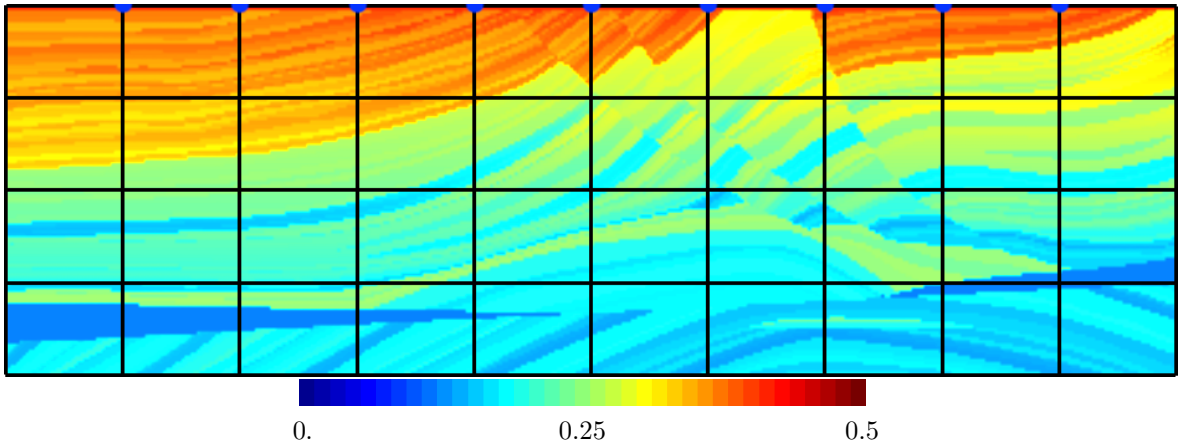


Figure 4.13: Wavenumber distribution of the 2D Marmousi benchmark. The blue dots on the top boundary indicate the position of the sources.

Transmission condition	# DoF (million)	Factorization size by subdomain (GiB)	Factorization time (min)	Convergence time (min)
Low order	≈ 54	≈ 3.9	1.8	7.0
HABC $N_{\text{HABC}} = 2$	≈ 55	≈ 4.0	2.6	7.1
HABC $N_{\text{HABC}} = 4$	≈ 56	≈ 4.0	3.2	6.0
HABC $N_{\text{HABC}} = 6$	≈ 57	≈ 4.1	3.7	6.9
PML $N_{\text{PML}} = 2$	≈ 56	≈ 4.1	2.1	7.4
PML $N_{\text{PML}} = 4$	≈ 58	≈ 4.3	2.4	8.5
PML $N_{\text{PML}} = 6$	≈ 59	≈ 4.4	2.4	8.4

Table 4.3: Orders of magnitude of the Marmousi benchmark. The table show the approximated total number of degrees of freedom, the approximated LU factorization size by subdomain, the subdomain problem factorization time, and the time required to converge to the residual of 10^{-6} using a GMRES solver.

absorbing condition is imposed on the four boundaries, even on the top one where the sources are enforced.

The resulting subdomain problem size is around 1.4 millions of DoF for the low-order methods; it slightly increases when the order increases but remains around 1.4 millions; the total number of unknowns is around 56 millions. The factorization of the subdomain finite element matrix depends on the method and N_{HABC} or N_{PML} (see Table 4.3). Note that the frequency of the problem could probably be pushed further than 100 Hz.

To obtain the results of Figure 4.14, the overall computational time is around 12 min no matter N_{HABC} or N_{PML} while the computation time using the zeroth-order transmission condition takes around 20 min to converge. The DDM algorithm converges in about 40 iterations (Figure 4.15), versus the 291 iterations required to converge with the zeroth-order transmission condition. The PML-based DDM takes more iterations to converge to a relative residual of 10^{-6}. As observed with numerical analysis of both methods in Chapters 2 and 3, the rate of convergence slightly increases with the degree of the method.

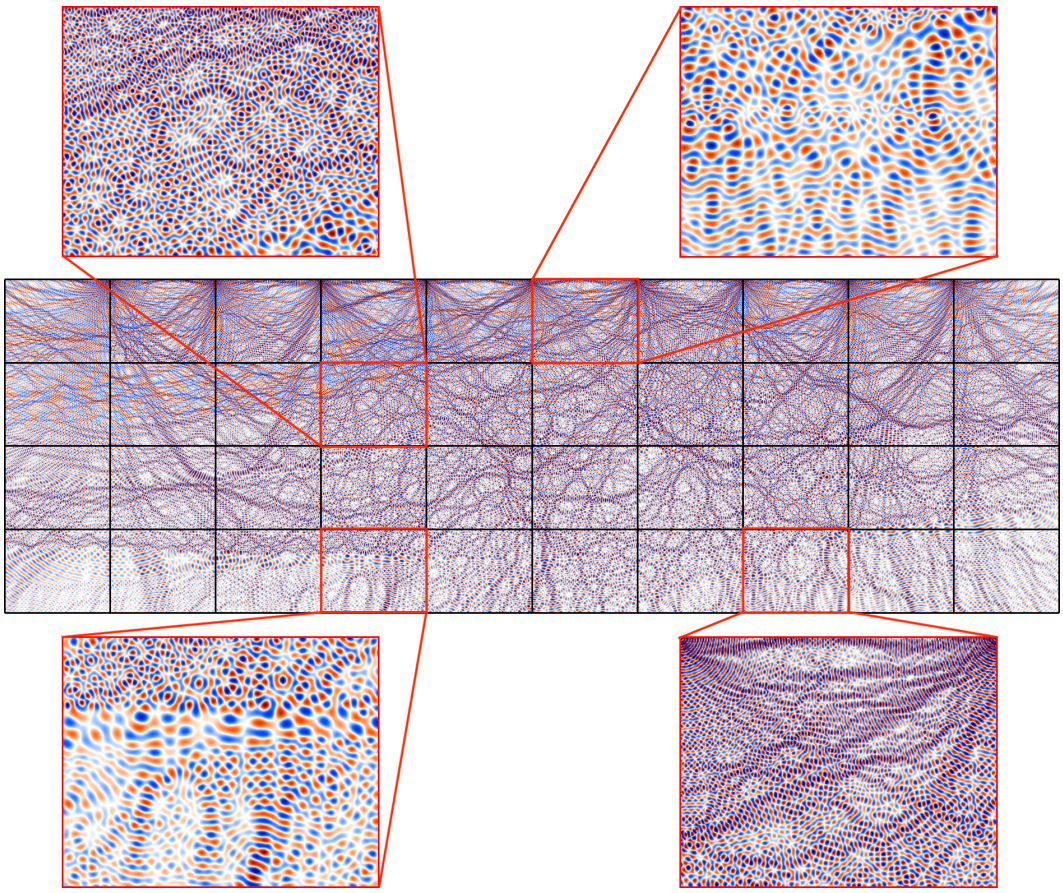
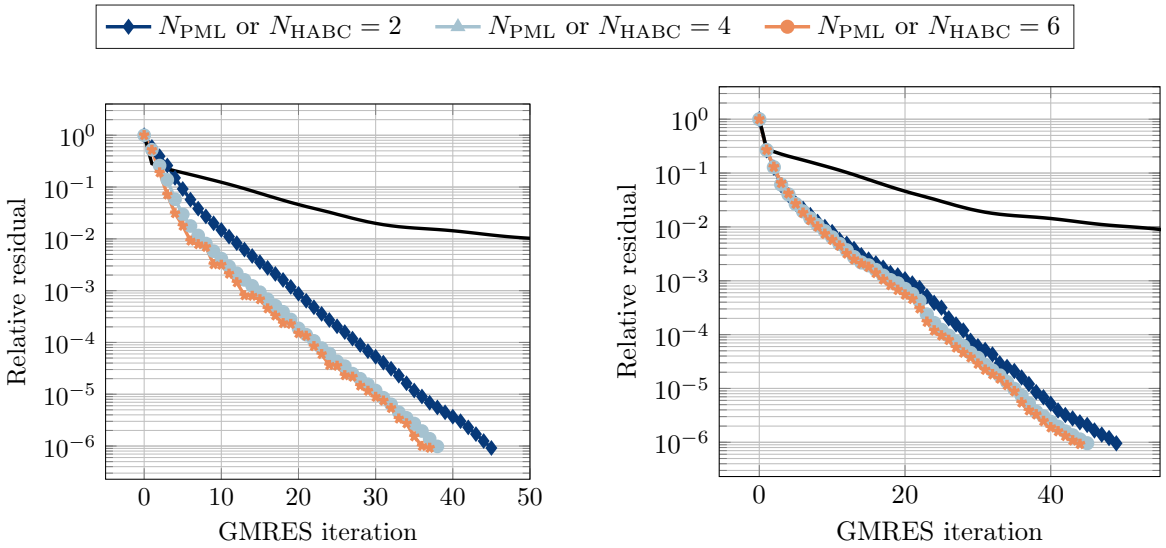


Figure 4.14: Real part of the numerical solution of the 2D Marmousi benchmark with at zoom over some subdomain solutions. The frequency of the benchmark is 100 Hz, leading to a number of wavelengths over the computational domain between 96 and 368.



(a) HABC-based DDM with cross-point treatment.

(b) PML-based DDM with cross-point treatment.

Figure 4.15: Convergence history of the 2D Marmousi benchmark. The black curve corresponds to the convergence history using the 0^{th}-order Sommerfeld condition as transmission operator.

4.2 The Salt 3D model

The Salt 3D data is a 3D model comparable to the 2D Marmousi one. It is a 13500-meter-long, 13500-meter-large, and 4180-meter-deep underground domain. The sound velocity goes form 1500 m/s to 4482 m/s. The wave frequency is set to 4 Hz such that the wave number distribution is between $5.6\,10^{-3}$ m^{-1} and $1.6\,10^{-2}$ m^{-1} as Figure 4.13 depicts. This leads to a wavelength between 375 m and 1120 m such that the number of wavelengths over the domain is between 12 and 36. The characteristic mesh size is set to 100 m such that the number of points by wavelength is between 4 to 11, leading to 3.6 millions of linear tetrahedra. All computations are made with fields discretized using third-degree hierarchical basis functions. The integration scheme is exact for a polynomial of degree 7.

This mesh is partitioned into a grid of 8 by 10 by 2, leading to a total of 160 subdomains. Once again, each subdomain is assigned to an MPI process that runs on 2 CPUs. HABCs with N_{HABC} equals to 2 and 4, and PMLs with N_{PML} equals to 1 are imposed on the exterior and interior boundaries. A punctual Dirichlet condition is enforced on each cross-point on the top boundary of Figure 4.16. Note that an absorbing condition is imposed on the six boundaries, even on the top one where the sources are enforced.

The resulting subdomain problem size is around 130,000 DoF for the low-order methods; it increases when the N_{HABC} or N_{PML} increase (see Table 4.4). Compared with the 2D Marmousi benchmark, the factorization memory consumption behaves differently since N_{HABC} or N_{PML} as a more significant impact on it. This is why, the HABC with $N_{\text{HABC}} = 6$, the PML with $N_{\text{PML}} = 4$ or 6 cannot be run on the NIC5 super-computer.

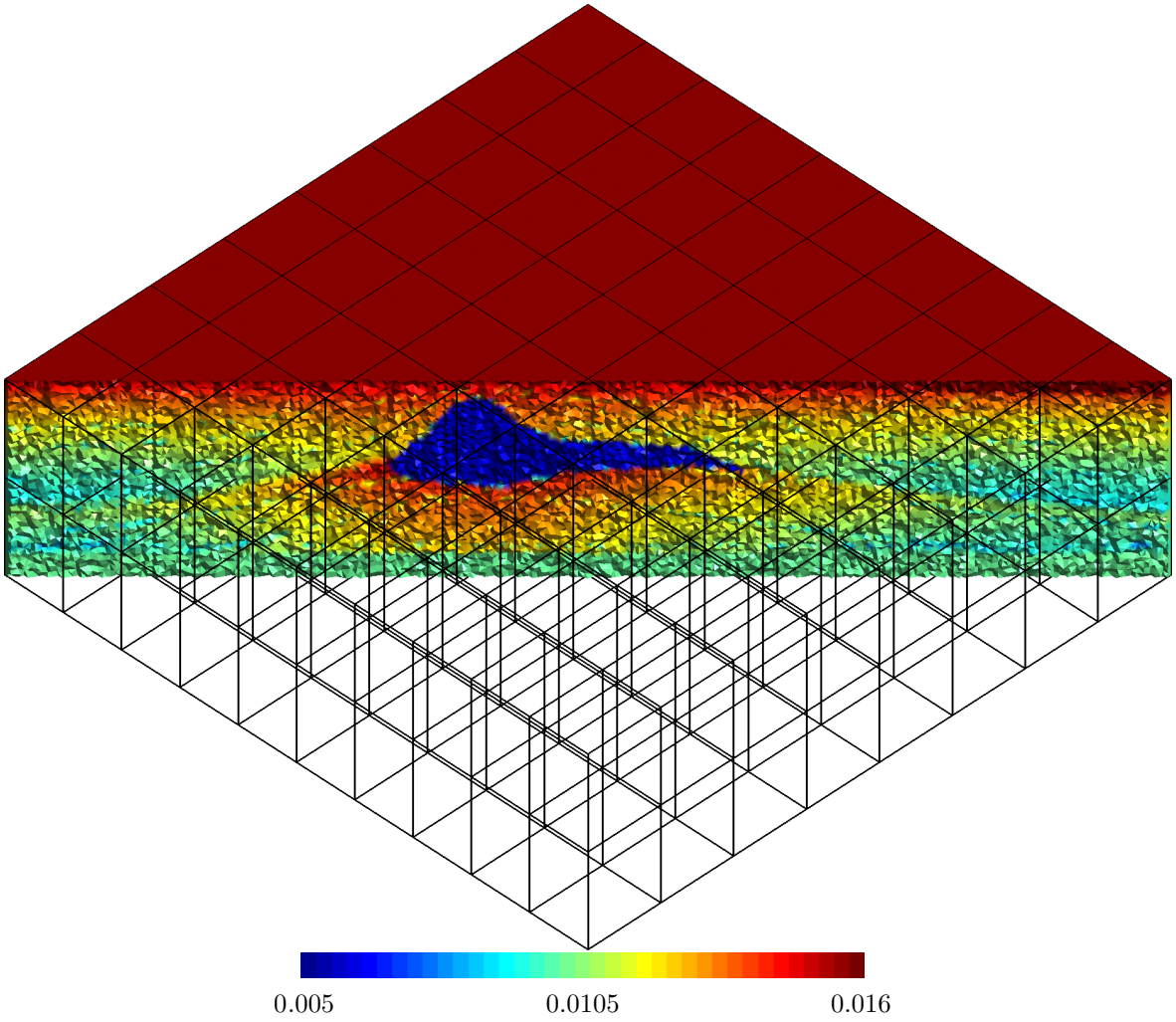


Figure 4.16: Wavenumber distribution of the 3D Salt benchmark.

Transmission condition	# DoF (million)	Factorization size by subdomain (GiB)	Factorization time (min)	Convergence time (min)
Low order	≈ 21	≈ 2.6	5.8	42.7
HABC $N_{\text{HABC}} = 2$	≈ 28	≈ 3.8	10.1	48.6
HABC $N_{\text{HABC}} = 4$	≈ 36	≈ 5.5	13.5	80.4
HABC $N_{\text{HABC}} = 6$	≈ 45	≈ 7.5	//	//
PML $N_{\text{PML}} = 1$	≈ 36	≈ 5.5	46.8	91.3
PML $N_{\text{PML}} = 2$	≈ 49	≈ 8.7	//	//

Table 4.4: Orders of magnitude of the Salt 3D benchmark. The table show the approximated total number of degrees of freedom, the approximated LU factorization size by subdomain, the subdomain problem factorization time, and the time required to converge to the residual of 10^{-6} using a GMRES solver.

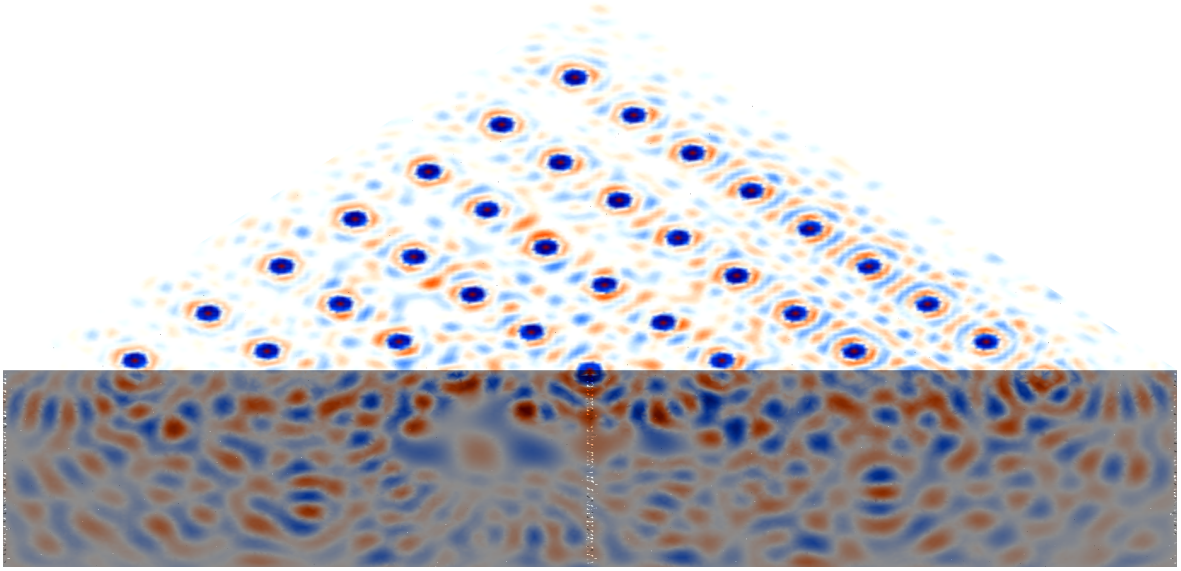
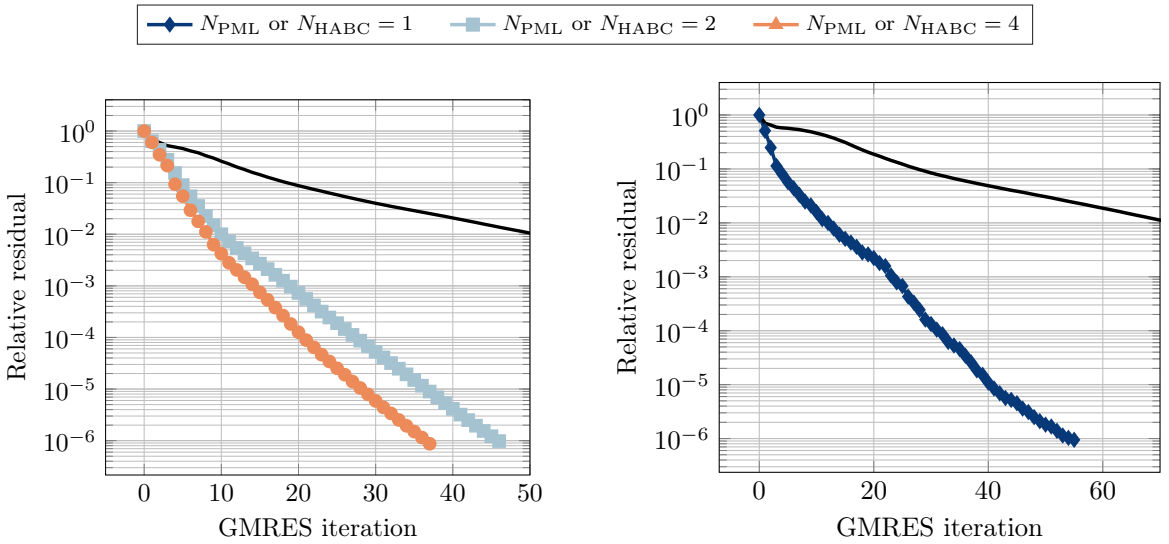


Figure 4.17: Real part of the numerical solution of the 3D Salt benchmark. The frequency of the benchmark is 4 Hz, leading to a number of wavelengths over the computational domain between 12 and 36.

To obtain the results of Figure 4.17, the overall computational time is around 50 min for the low-order method and around an hour for HABC methods. For this 3D problem, the PML method is slower (it takes around 2 hours to converge) and compared with the HABC method of the same degree, it requires more memory. The DDM algorithm converges in about 40 iterations as Figure 4.18 depicted, while it requires 243 iterations to converge with the zeroth-order transmission condition. The PML-based DDM takes more iterations to converge to a relative residual of 10^{-6}. As observed with the 2D Marmousi benchmark, the rate of convergence slightly increases with N_{HABC} or N_{PML}, but the increase is more important than with the 2D Marmousi benchmark.



(a) HABC-based DDM with cross-point treatment.

(b) PML-based DDM with cross-point treatment.

Figure 4.18: Convergence history of the 3D Salt benchmark. The black curve corresponds to the convergence history using the 0^{th}-order Sommerfeld condition as transmission operator.

5 The mono-domain elastic scattering problem

In this section, academic elastic scattering problems similar to those presented in Section 3 are studied. The scattering of an incident plane wave $\mathbf{u}_{\text{inc}} = e^{\imath\, kx}\mathbf{e}_y$, traveling in the direction of the basis vector $\mathbf{e}_x$, on either a soft cylinder (2D problem) or a soft sphere (3D problem) are computed. These problems are both modeled by the following problem on the domains of Figures 4.1a or 4.1b.

$$\begin{cases} \dfrac{1}{k_P^2}\,\mathbf{grad}\,\mathrm{div}\,\mathbf{u} - \dfrac{1}{k_S^2}\,\mathbf{curl}\,\mathbf{curl}\,\mathbf{u} - \mathbf{u} = \mathbf{0} & \text{in } \Omega_{\text{tot}}, \\ \mathbf{u} = -\mathbf{u}_{\text{inc}} & \text{on } \Gamma_{\text{scat}}, \\ \Gamma^t(\mathbf{u}) - \mathcal{B}\mathbf{u} = \mathbf{0} & \text{on } \Gamma_{\text{ext}}, \end{cases} \tag{4.2}$$

where $\mathcal{B}$ is a boundary operator and $\Gamma^t(\cdot)$ as defined in Equation 5.10. The computational domains and the meshes are identical to the acoustic problems ones (Figure 4.1). Once again, the mesh size is chosen such that the number of point by wavelength is equal to 10.

In the following subsections, the multi-threaded efficiency of GmshFEM will be first studied on the mono-domain application of Problem 4.2.

Equation 4.2 modeled using the same basis function degrees as done in the acoustic analysis. As done with acoustic scaling analysis, the P-wavenumber k_P are chosen such that the number of wavelengths over the geometry is equal to 50 for the 2D problem, and 10 for the 3D problem. The S-wavenumber k_S are defined as half of the P-wavelengths. The formulation is written using the `CompoundField` feature that will be presented in Chapter 5.

Basis degree	2D	3D
2	2,328,088	2,367,516
4	9,304,992	18,666,684
6	20,930,712	62,620,824
8	37,205,248	//
10	58,128,600	//

(a) Problem size (number of DoFs) of the 2D and 3D elastic scattering problem for different basis function degrees.

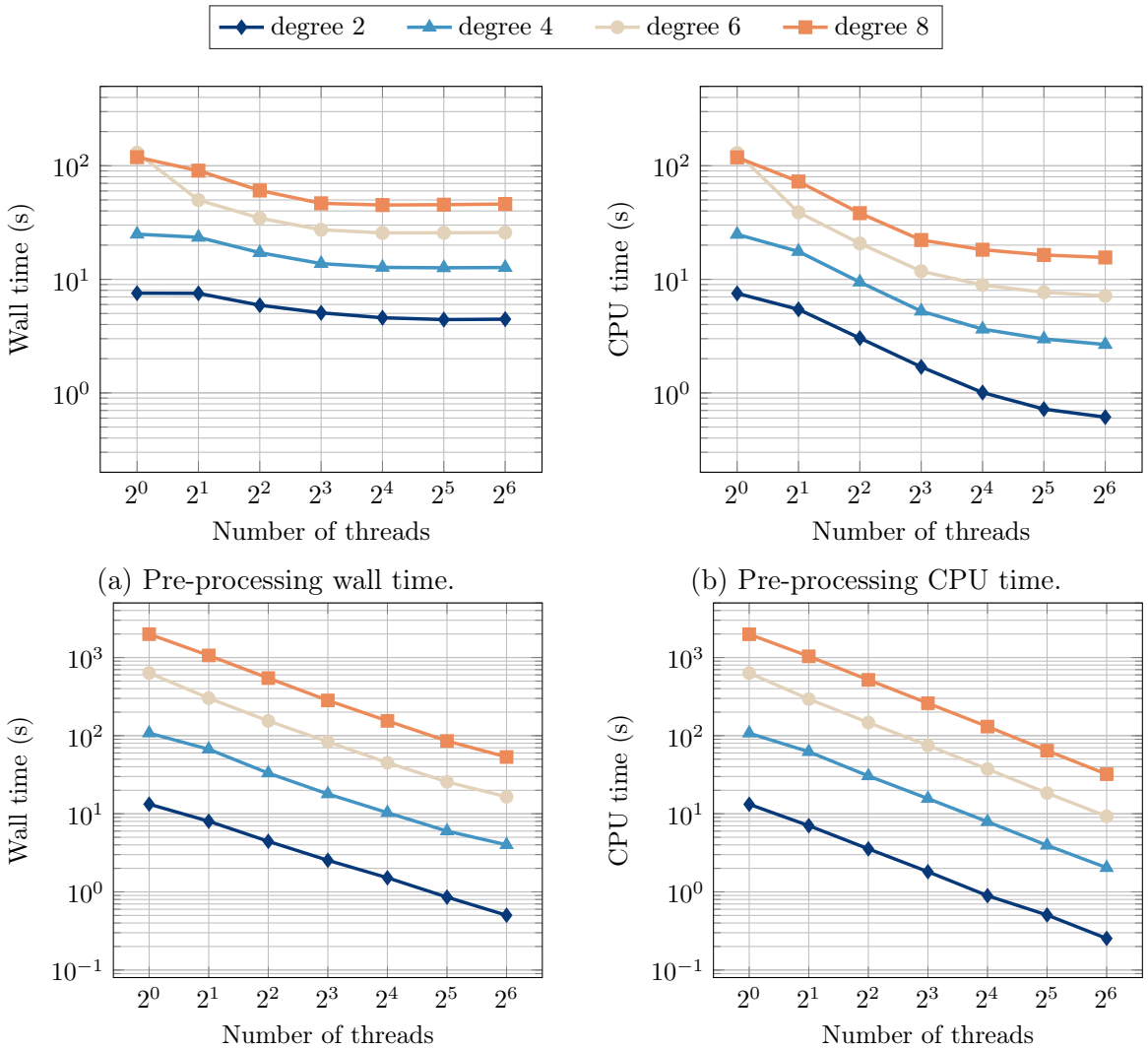
Basis degree	2D	3D
2	11.6	312
4	57.2	5089
6	155	//
8	332	//
10	//	//

(b) Estimated memory needed to factorize the finite element matrix using MUMPS. The data are given in gibibyte (GiB). Character '?' means that MUMPS could compute the estimated factorization memory cost; probably because it will be too huge.

Table 4.5: Magnitudes (number of degree of freedoms and estimated memory needed to store the LU factorization) of the 2D and 3D elastic scattering problems.

Tables 4.5 reports the problem size of both the 2D and 3D benchmark. Compared to those of the acoustic problems 4.1, the total number of unknowns is multiplied by a factor of 2 or 3, depending if the 2D or 3D problem is considered. Concerning the estimated matrix factorization memory size, the computed values are much greater than the equivalent acoustic problems of Chapter 4, and not only because of factor 2 or 3 on the number of DoFs. For instance, let us compare the 2D acoustic scattering problem at degree 6 which has around 10 millions of DoFs with the 2D elastic scattering problem at degree 4 which has around 9 millions. In this situation, even if the number of DoFs for this elastic problem is a bit lower than the acoustic one, and the basis function degree is lower, the estimated matrix factorization storage is 44 % greater. This is explained by the strong coupling between the elastic quantities over the elements that leads to a number of non-zero in the finite element matrix that is not simply multiplied by 2 or 3 depending on the problem dimension. As a result, for comparable acoustic and elastic problems (*i.e.* same mesh and same basis function degree) the finite element matrix sparsity is lower for the elastic problems than for the acoustic ones. Therefore, elastic problems are even harder to solve than acoustic ones.

Figures 4.19 and 4.20 report similar timings as done with the acoustic efficiency analysis, namely the wall and CPU times of the pre and assembly part. Similar behaviors can be observed, the pre-processing does not scale as good the assembly part. Nevertheless, two interesting observations could be pointed out. First, the difference between the wall and CPU times is not as important as in the acoustic study. This is simply because the arithmetic intensity of the elastic assembly process is greater than the acoustic one. Therefore the ratio of time spent in the memory allocation over the total assembly time is lower with elastic problems than with acoustic ones. Second, the assembly efficiency (see Figure 4.21) is better with elastic problems than with acoustic ones. This behavior is also explained by the arithmetic intensity that increases with elastic formulations.



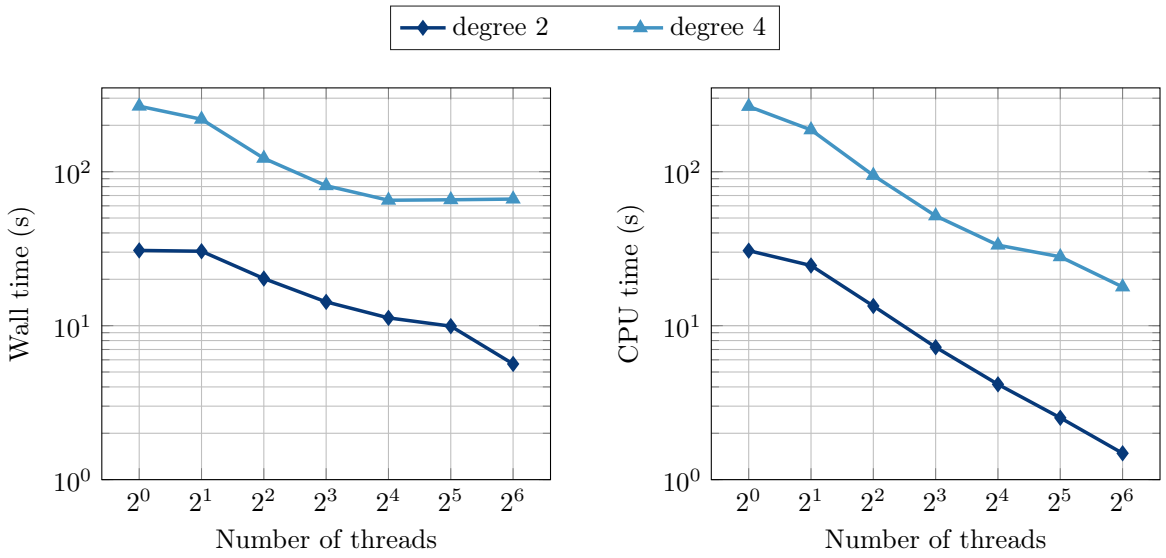
(a) Pre-processing wall time.

(b) Pre-processing CPU time.

(c) Assembling wall time.

(d) Assembling CPU time.

Figure 4.19: Pre- and assembly time of the 2D scattering problem for different basis function degrees. Both wall and CPU time are reported such that the influence of the memory allocation can be estimated.



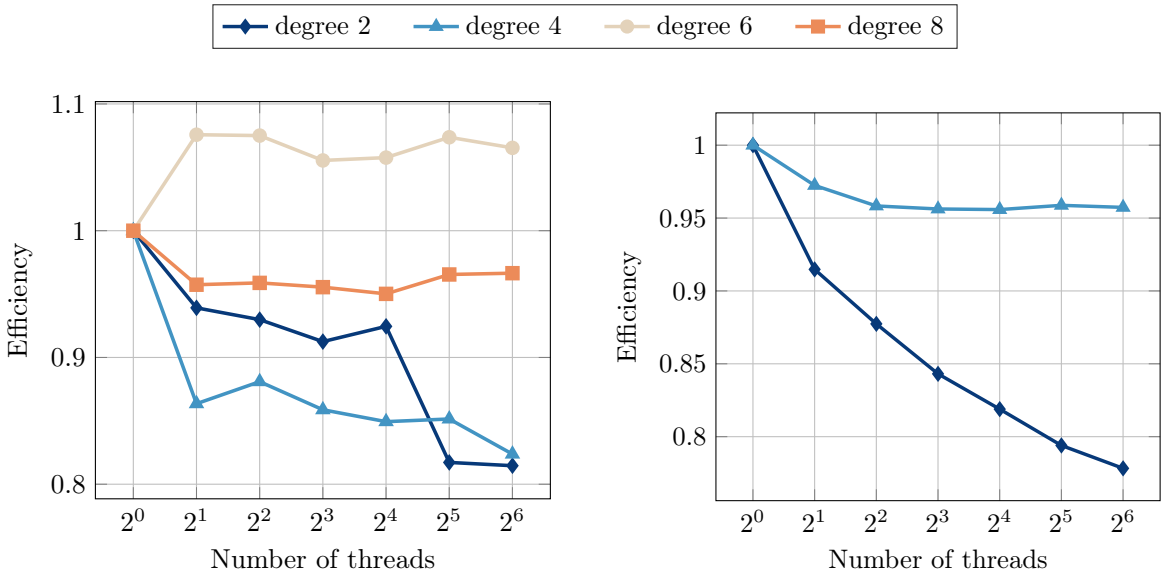
(a) Pre-processing wall time.

(b) Pre-processing CPU time.

(c) Assembling wall time.

(d) Assembling CPU time.

Figure 4.20: Pre- and assembly process time of the 3D scattering problem for different basis function degrees. Both wall and CPU time are reported such that the influence of the memory allocation can be estimated.



(a) Efficiency of the 2D assembly process.

(b) Efficiency of the 3D assembly process.

Figure 4.21: Multi-threaded parallel efficiencies of the finite element assembly process computed on the CPU time for both 2D and 3D scattering problem.

6 Elastic problems in complex heterogeneous media

In a similar way to what has been done with acoustic heterogeneous problems (Section 4), this section is devoted to the resolution of 2D and 3D underground elastic simulations. This time, two parameters are considered, the P-wave and the S-wave velocities, related to the primary and secondary waves of the elastodynamics equation.

The 2D benchmark is based on the Marmousi2 model [47]. It is an artificial underground acoustic model developed in 2002 [139, 140], as the elastic version of the acoustic Marmousi model. The 3D benchmark is also based the Salt 3D model [81] where the P-wave velocity is given by the model and the S-wave velocity is half the P-wave one.

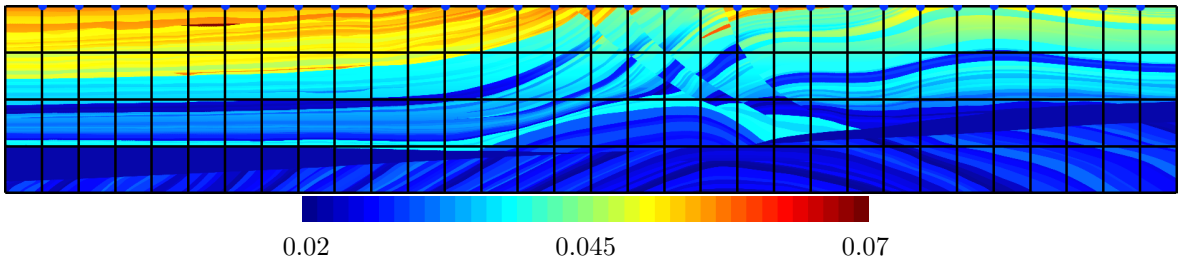
6.1 The Marmousi2 model

The Marmousi2 is a 17,000-meter-long and 2700-meter-deep underground model given P-wave and S-wave velocity maps. The P-wave velocity map between 1028 m/s to 4700 m/s, and a S-wave velocity between 368.08 m/s to 2802 m/s. The wave frequency is set to 15 Hz such that the P-wavenumber distribution is between 0.02 m^{-1} and 0.09 m^{-1} as Figure 4.22a depicts, and the S-wavenumber distribution is between 0.03 m^{-1} and 0.25 m^{-1} as shown in Figure 4.22b. This leads to a P-wavelength between 68 m and 313 m such that the number of wavelengths over the domain is between 54 and 248. The S-wavelengths are between 24 m and 186 m so that the number of wavelengths over the domain is between 91 and 692. The characteristic mesh size is set to 3 m such that the number of points by wavelengths is between 22 to 104 for the P-wave and between 8 to 62 for the S-wave. The resulting number of straight triangles is about 11.7 millions. Note that quadrangle elements are not used because they lead to less-sparse finite element matrices that increase the factorization memory requirement. Therefore even if quadrangle elements are theoretically more efficient to assemble, they are not used. Furthermore, note that the assembly time is negligible over the all computation time. All computations are made with fields discretized using forth-degree hierarchical basis functions. The integration scheme is exact for a polynomial of order 9.

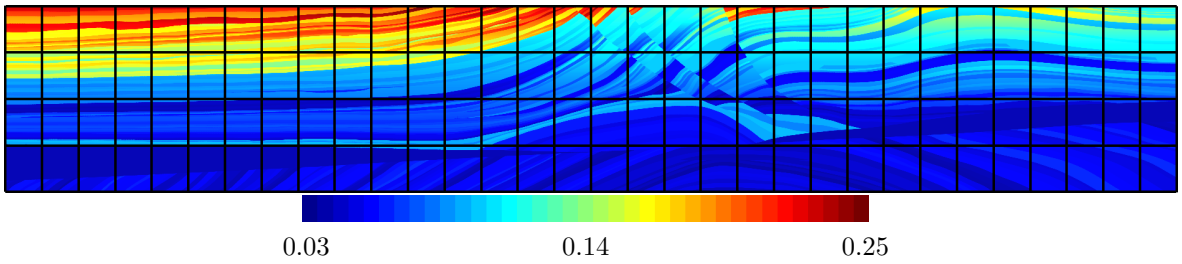
This mesh is partitioned into a grid of 32 by 4 leading to a total of 128 subdomains. Each of them is assigned to an MPI process that runs on 8 CPUs such that the total number of CPUs used is equal to 1024. This computation requires an extended reservation of 16 nodes of NIC5. PMLs of degrees 2, 4, and 6 are imposed on the exterior and interior boundaries. A punctual Dirichlet condition is enforced on each cross-point on the top boundary; see the blue dots of Figure 4.22a. Note that an absorbing condition is imposed on the four boundaries, even on the top one where the sources are enforced.

The resulting subdomain problem size is about 1.6 millions for the low-order methods; it increases to about 1.8 millions; the total number of unknowns is around 206 and 236 millions, respectively (see Table 4.6). The factorization of the subdomain finite element matrix takes around 10 GiB of memory for the PML problem with $N_{\text{PML}} = 6$.

To obtain the results of Figure 4.14, the overall computational time is around 12 min no matter the method degree. As observed with numerical analysis of both methods in Chapters 2 and 3, the rate of convergence slightly increases with N_{PML}

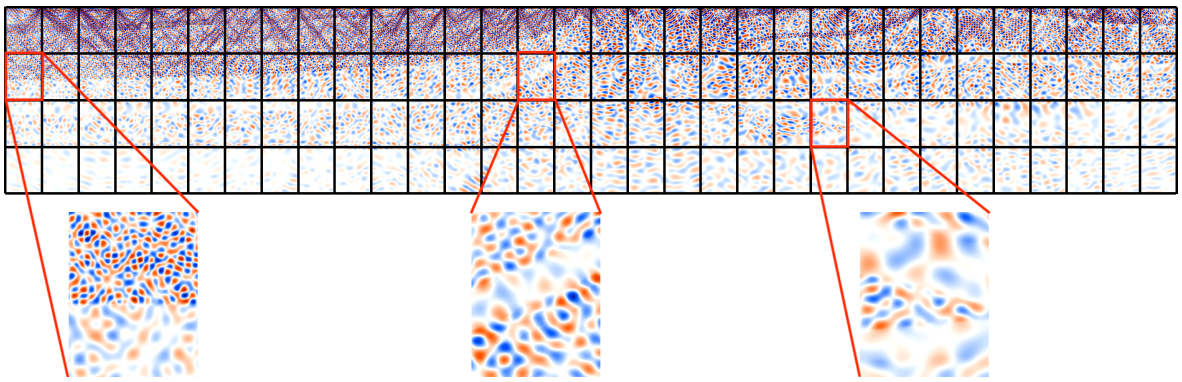


(a) P-wave wavenumber distribution.

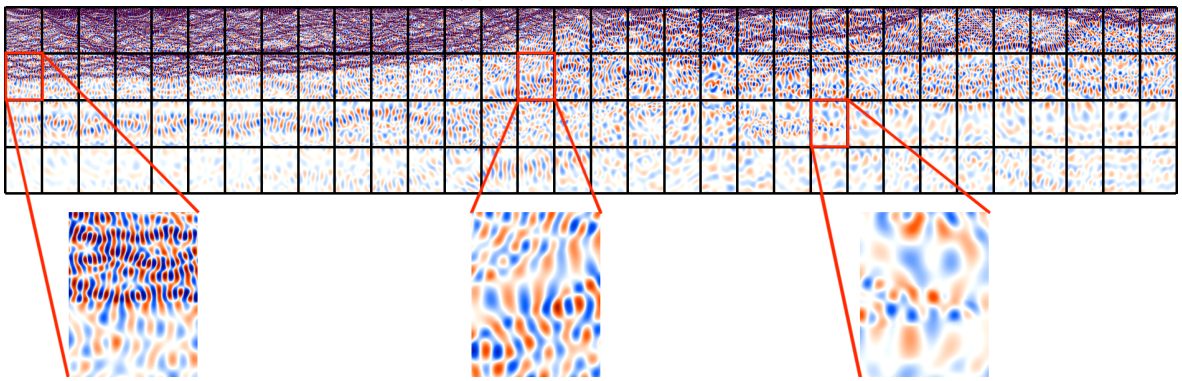


(b) S-wave wavenumber distribution.

Figure 4.22: Wavenumber distribution of the 2D Marmousi2 benchmark. The blue dots on the top boundary indicate the position of the source.



(a) The x-component.

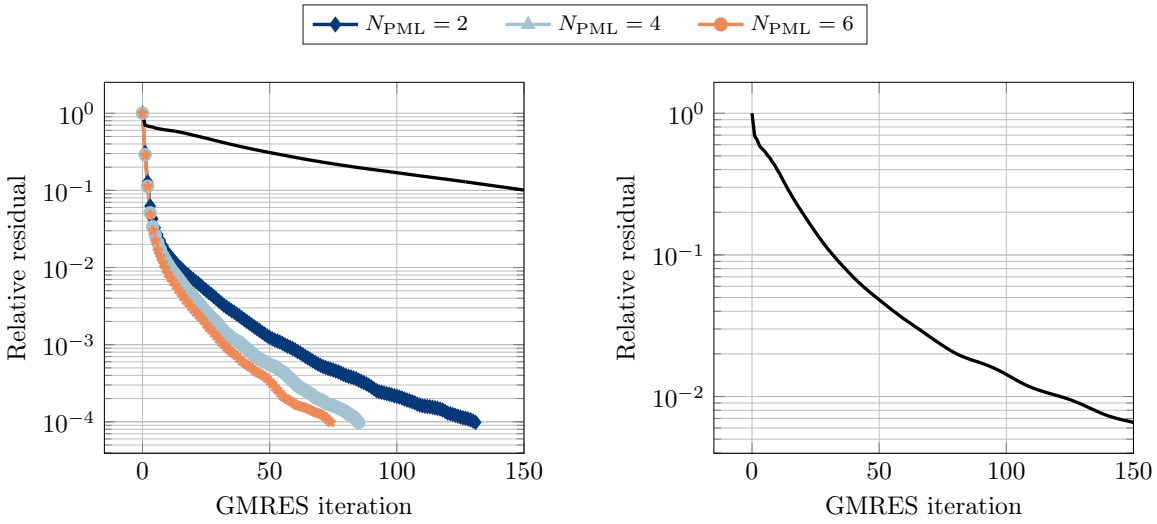


(b) The y-component.

Figure 4.23: Real part of the numerical solution of the 2D Marmousi2 benchmark with at zoom over some subdomain solutions. The frequency of the benchmark is 15 Hz, leading to a number of wavelengths over the computational domain between 91 and 692.

Transmission condition	# DoF (million)	Factorization size by subdomain (GiB)	Factorization time (min)	Convergence time (min)
Low order	≈ 190	≈ 8.0	1.9	33.6
PML $N_{\text{PML}} = 2$	≈ 207	≈ 8.8	2.2	9.4
PML $N_{\text{PML}} = 4$	≈ 221	≈ 9.2	2.2	7.8
PML $N_{\text{PML}} = 6$	≈ 236	≈ 10.0	2.5	5.7

Table 4.6: Orders of magnitude of the Marmousi2 benchmark. The table show the approximated total number of degrees of freedom, the approximated LU factorization size by subdomain, the subdomain problem factorization time, and the time required to converge to the residual of 10^{-5} using a GMRES solver.



(a) The 2D Marmousi2 benchmark.

(b) The elastic 3D Salt benchmark.

Figure 4.24: Convergence history of the 2D Marmousi2 and the elastic Salt 3D benchmarks. The black curve corresponds to the convergence history using the 0^{th}-order Lysmer-Kuhlemeyer condition as transmission operator.

(see Figure 4.24). To reach a relative residual of 10^{-4}, the elastic PML-based DDM takes 131, 85, or 74 iterations with a PML with N_{PML} equal to 2, 4, or 6, respectively. These rates of convergence are compared with a DDM using the simple zeroth-order Lysmer-Kuhlemeyer condition as transmission operator. This low-order method takes 902 iterations and three times the computation time of the PML-based method (around 36 min) to converge to the same residual.

6.2 The elastic Salt 3D model

The 3D benchmark is based on the acoustic Salt 3D data of Chapter 4, with the P-wave velocity chosen equal to the acoustic sound velocity of the Salt 3D model, and the S-wave velocity set to twice the P-wave velocity. The frequency is set to 2 Hz, such that the P-wave number distribution lies between $2.8\,10^{-3}$ m^{-1} and $8.3\,10^{-3}$ m^{-1} (see Figure 4.22). This leads to P-wavelengths between 750 m and 2241 m, so that the number of wavelengths over the domain is between 6 and 18. The S-wavelength is

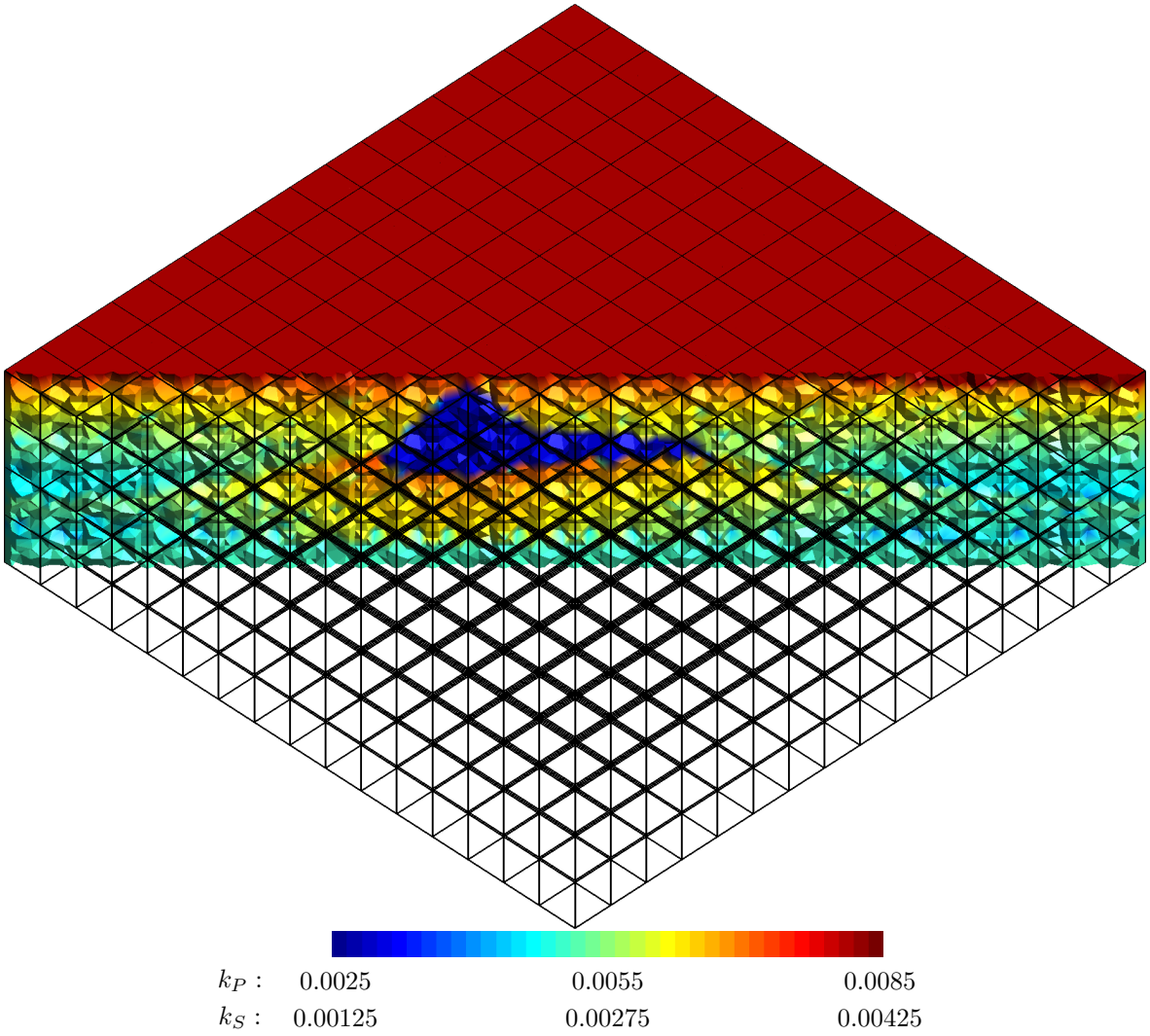
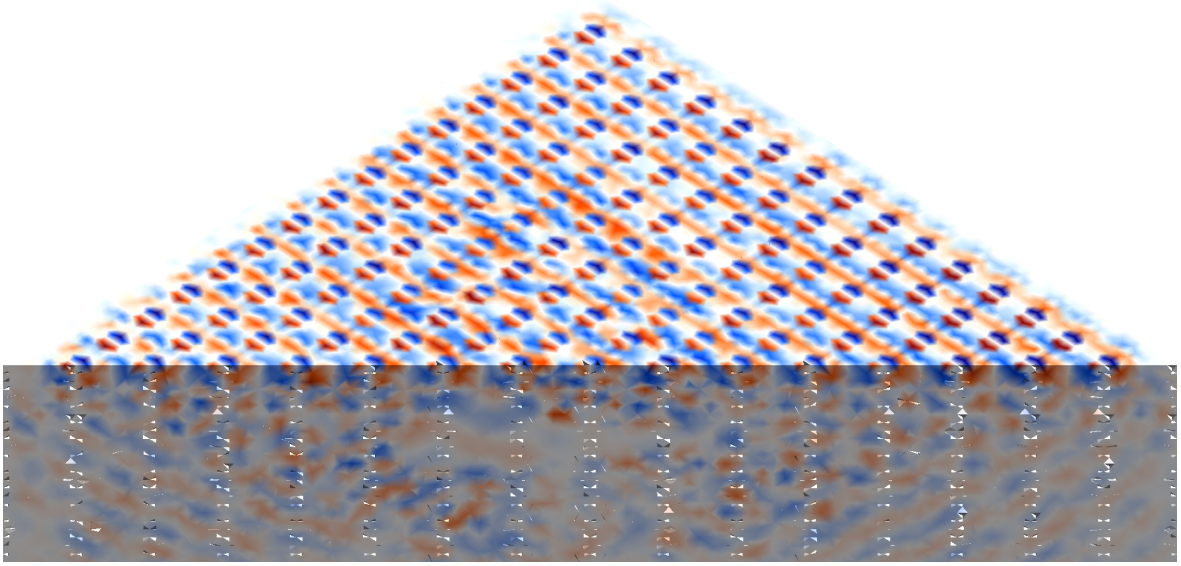


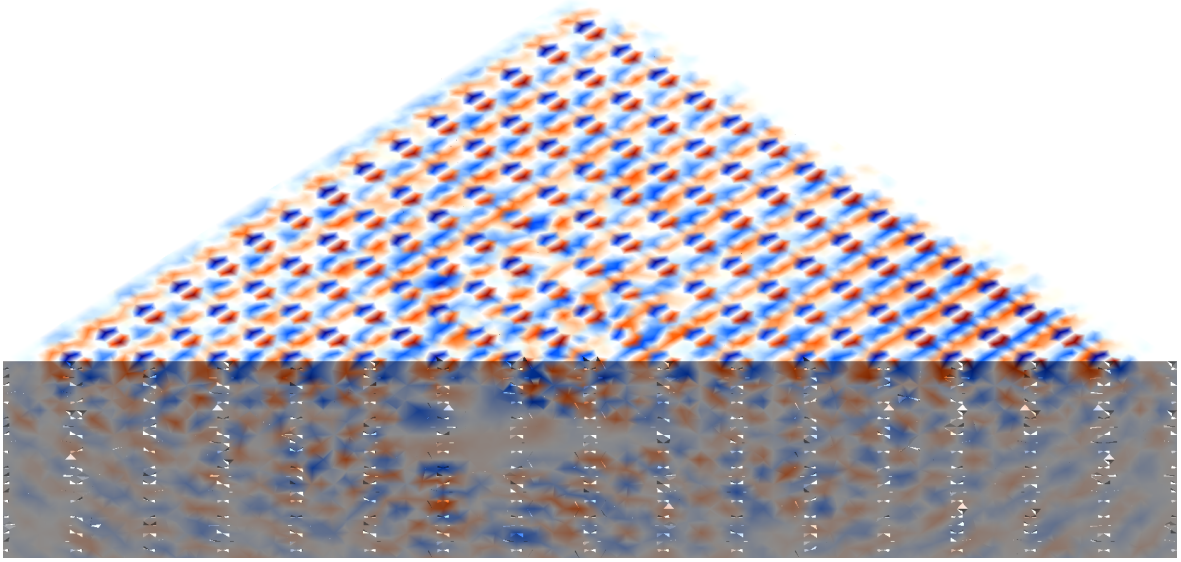
Figure 4.25: P- and S-wavenumber distribution of the elastic 3D Salt benchmark.

between 1500 m and 4482 m, with the number of wavelengths over the domain between 3 and 9. The characteristic mesh size is set to 200 m, such that the number of points by wavelength is between 4 to 22, leading to about 893000 tetrahedra. All computations are made with fields discretized using third-degree hierarchical basis functions. The integration scheme is chosen to be exact for polynomials of order 7.

This mesh is partitioned into a grid of 16 by 16 by 4, leading to a total of 1024 subdomains. A Dirichlet condition is enforced on each cross-point on the top boundary. A Zeroth-order transmission condition is enforced on the interior interfaces as well as on the exterior boundaries, including on the top boundary where the sources are enforced.



(a) The x-component.

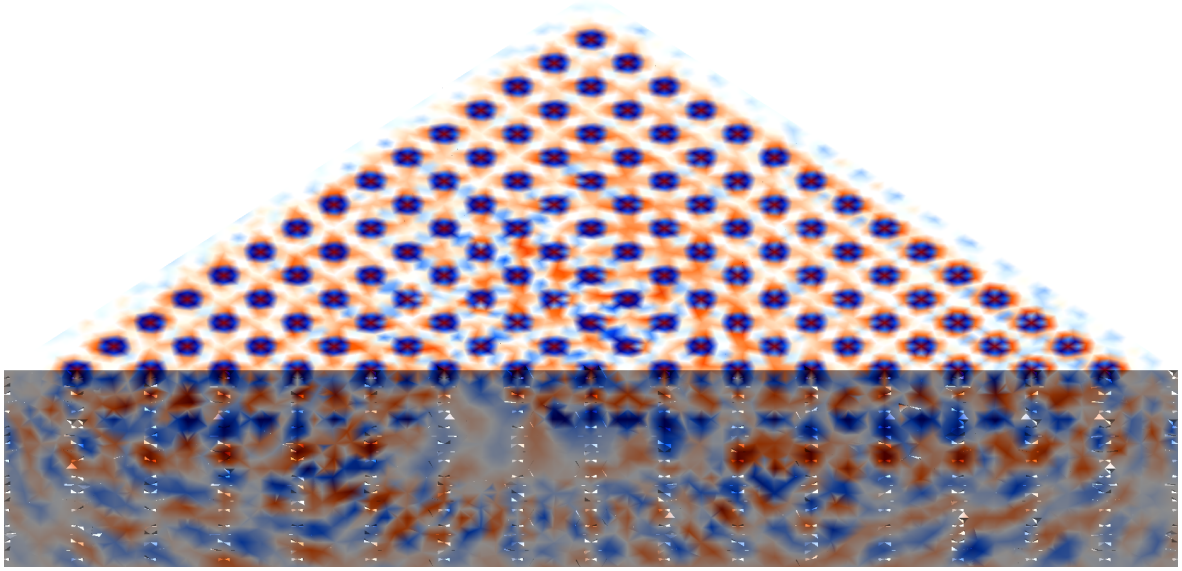


(b) The y-component.

Figure 4.26a

Transmission condition	# DoF (million)	Factorization size by subdomain (GiB)	Factorization time (min)	Convergence time (min)
Low order	≈ 22	≈ 0.4	0.5	6.0

Table 4.7: Orders of magnitude of the elastic Salt 3D benchmark. The table shows the approximated total number of degrees of freedom, the approximated LU factorization size by subdomain, the subdomain problem factorization time, and the time required to converge to the residual of 10^{-4} using a GMRES solver.



(c) The z-component.

Figure 4.26b: Real part of the numerical solution of the elastic 3D Salt benchmark. The frequency of the benchmark is 2 Hz, leading to a number of wavelengths over the computational domain between 6 and 18.

7 Conclusion

In this chapter, the performance of the methods presented in Chapters 2 and 3 has been compared on acoustic and elastic applications.

First of all, the study of 2D and 3D mono-domain acoustic and elastic scattering problem in homogeneous problems have demonstrated the shared-memory efficiency of GmshFEM (Sections 3.1 and 5).

Second, a study of the 2D and 3D acoustic scattering domain decomposition problem has shown the distributed efficiency of GmshDDM (Section 3.2). Besides, the comparison of the PML-based and HABC-based cross-point treatments of Chapters 2 and 3 has exhibited comparable efficiency and accuracy for both methods. Depending on the required precision and the finite element degree, one can be recommended over the other. Indeed, the PML-based method is more efficient (in terms of accuracy and computational time) at a low degree for a medium accuracy. Conversely, the HABC-based method is recommended for high accuracy and with high-degree.

Finally, 2D and 3D acoustic and elastic heterogeneous problems have been presented in Sections 4 and 6 to expose problems that can be run using the methods and libraries developed during this thesis.

Part II

Computational tools

5 GmshFEM, a finite element library

GmshFEM is an open-source C++ finite element library based on the application programming interface of Gmsh. Both share the same design philosophy: to be fast, light, and user-friendly. From the pre-possessing to the post-processing, each finite element processing stage is designed to be efficient for modern multi-core CPUs by combining an efficient multi-threaded parallelization with SIMD vectorization. The C++ interface allows the implementation of finite element problems in a natural manner, close to their mathematical definition, with as few programming artifacts as possible.

1 Introduction

Many open-source finite element codes and libraries are currently available, with varying degrees of generality, performance, robustness, and user-friendliness. Among many others, one can cite *e.g.* [68, 64, 13, 21, 164, 163, 6, 108, 92, 165, 9, 173, 104]... As with any software, most excel in one or two of these areas; but compromises are invariably necessary to balance *e.g.* generality and user-friendliness (the ability to deal quickly with various partial differential equations and physical models or to integrate the code in complex workflows) with raw performance. Robustness and code maturity are additional parameters to consider, as is community support.

In this chapter, we introduce the open-source C++ finite element library GmshFEM (`https://gitlab.onelab.info/gmsh/fem`) that was developed in the context of this thesis. GmshFEM is based on the application programming interface (API) of the open-source finite element mesh generator, pre- and post-processor Gmsh [93], and exhibits its own particular blend of features, sharing the same design philosophy as Gmsh: to be fast, light, and user-friendly [93].

To be fast, GmshFEM is designed to gain the most significant benefits of multi-

core CPUs by combining an efficient multi-threaded parallelization with SIMD vectorization. Particular attention was paid to data localities during all development, resulting in excellent performances on multi-core chips. Moreover, native support for complex arithmetic makes it a perfect framework for efficient time-harmonic finite element solvers.

To be light, GmshFEM extensively relies on Gmsh to manage geometries (with direct access to the CAD boundary representation), meshes (structured, unstructured, and hybrid with straight-sided or curved elements, in 1D, 2D, and 3D), interpolation (arbitrary order Lagrange or hierarchical bases for H^1 and $H(\mathbf{curl})$ [176]) and integration (numerical quadratures on all element shapes: lines, triangles, quadrangles, tetrahedra, prisms, hexahedra, pyramids). GmshFEM relies on its own internal sparse matrix storage, but interfaces with Eigen [102] for dense linear algebra and external solvers like PETSc [18] and MUMPS [7, 8] for large sparse solvers required for implicit formulations or SLEPc [111] for large sparse eigenproblem solvers.

Strongly inspired by the design of GetDP [92, 70], GmshFEM relies for generality and user-friendliness on a symbolic, high-level description of variational formulations which allows to define the problem to solve (including boundary conditions, source terms, etc.) in a natural mathematical manner, and is amenable to scripting without pre- or re-compilation, like GetDP [70] or FreeFEM++ [108] but unlike most other libraries [13, 21, 6, 163, 165, 104]. Moreover, this genericity implies neither the hard-coding of particular classes of PDEs [164], nor the use of a full-blown external scripting language like Python [173, 6].

The chapter is organized as follows. In Section 2 we start by describing how to express a finite element problem in GmshFEM, using the soft scattering Helmholtz problem described in Chapter 1, Section 2 as a guiding example. We then describe in Section 3 some implementation details for classical continuous finite elements, before presenting features not directly linked to time-harmonic wave simulation in Section 4. Numerical results are presented in Section 5 to highlight the efficiency of the library on modern multicore CPUs.

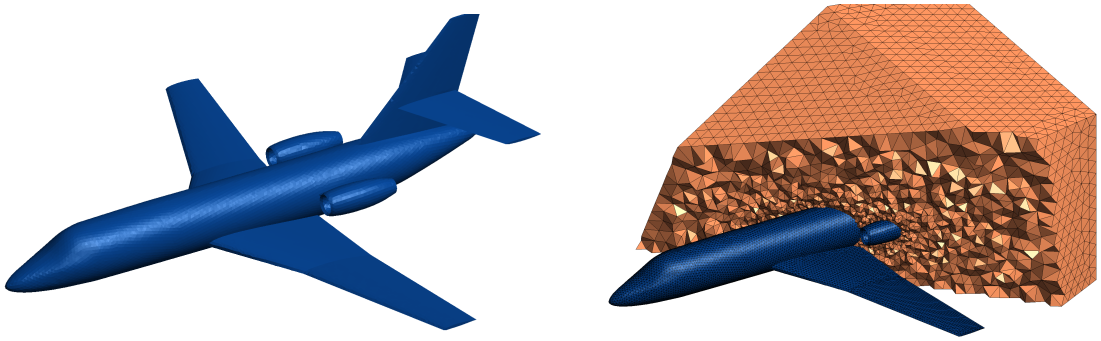
2 Symbolic definition of the problem

To illustrate how a finite element problem is set up in GmshFEM, let us consider the scattering of a time-harmonic incident acoustic plane wave $u_{\text{inc}} = \exp(\imath\,\mathbf{k}\cdot\mathbf{x})$ on a soft object Γ_{scat}, as presented in Chapter 1 in Equation (1.4). To keep things simple, a Sommerfeld boundary condition is enforced on the exterior boundary, such that (1.4) takes the following simpler form:

$$\begin{cases} \Delta u + k^2 u = 0 & \text{in } \Omega_{\text{tot}}, \\ u = -u_{\text{inc}} & \text{on } \Gamma_{\text{scat}}, \\ \partial_{\mathbf{n}} u - \imath\, k u = 0 & \text{on } \Gamma_{\text{ext}}, \end{cases} \tag{5.1}$$

with k a positive constant wavenumber and u the complex-valued scattered field.

The description of a finite element problem in GmshFEM is not based on the strong form (5.1) but rather on the associated variational formulation. The variational



(a) The Falcon plane used as a scattering object.

(b) The meshed domain Ω_{tot} around the plane.

Figure 5.1: Falcon plane as used in our example scattering problem (5.1).

formulation of Problem (5.1) reads: Find $u \in H^1(\Omega_{\text{tot}})$ such that

$$\int_{\Omega_{\text{tot}}} \mathbf{grad}\, u \cdot \mathbf{grad}\, \overline{v} - k^2 u\overline{v}\, \mathrm{d}\Omega_{\text{tot}} - \int_{\Gamma_{\text{ext}}} \imath\, ku\overline{v}\, \mathrm{d}\Gamma_{\text{ext}} = 0 \tag{5.2}$$

holds for every test function $v \in H^1(\Omega_{\text{tot}})$. The following sections describe how (5.2) is transcribed in the GmshFEM library. All the classes and functions of GmshFEM live in a `gmshfem` namespace such that the GmshFEM library can be used inside complex codes without name conflicts. On this simple problem, the "`using namespace gmshfem;`" directive was used for conciseness such that `gmshfem::` can be omitted.

For the example, the scattering object in Problem (5.1) is chosen to be a Falcon plane (see Figure 5.1a) that is 18.251 m long, 5.78 m high and 18.04 m large. The plane is put inside a box such that the domain Ω_{tot} is built by the intersection of the volume inside the box and the volume outside the plane. This volume is meshed by about 2.7M first order tetrahedra, with a total of about 473k nodes (see Figure 5.1b).

2.1 Geometric objects

In GmshFEM, all the data related to the geometry, the mesh, and the topology is manipulated with the help of child classes of the `GeometricObject` abstract class. Currently three child classes are defined:

- a `Domain` class (Figure 5.2a), to manipulate mesh entities as they appear in the mesh of the problem. For example, each time a function is piece-wise defined over a part of a mesh, a `Domain` object is involved.
- a `SkinLayer` class (Figure 5.2b), to build a layer of mesh elements defined inside a domain (the source) and adjacent to another domain (the tool).
- a `PeriodicLink` class (Figure 5.2c), to handle mesh entities that are linked by periodic conditions.

A `Domain` is based on the notion of physical group in Gmsh [93], which is identified by a pair of integers `(dim, tag)` corresponding to the geometric dimension of the

physical group, and a unique tag associated to it. A unique name can also be associated to the group. Therefore, a `Domain` class can be constructed either by giving this pair of integers or by giving the associated name. While the first method can be more convenient for automatically generated geometries with lots of different physical groups, *e.g.* a coil with an arbitrarily large number of windings, the second can be more readable for geometries with a small number of physical groups.

Among other operations, `Domain` objects can be manipulated using binary operations of set algebra: the union, the intersection, the symmetric difference, and the complement of domains are respectively defined using the bitwise OR operator `|`, the bitwise AND operator `&`, the bitwise XOR operator `^` and the bitwise NOT operator $\sim$. The bitwise OR assignment operator `|=`, the bitwise AND assignment operator `&=` and the bitwise XOR assignment operator `^=` are also overloaded.

Listing 5.1 shows how the three domains Ω, Γ_{scat} and Γ_{ext} used in Problem (5.1) are defined using GmshFEM, if we assume that physical entites named "omega", "gammaScat" and "gammaExt" are defined in the mesh.

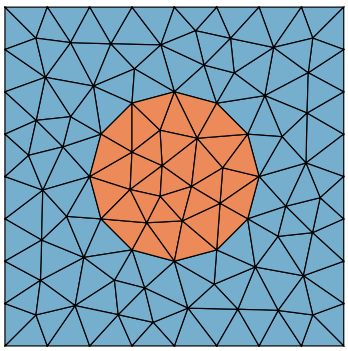
```
domain::Domain omega("omega");
domain::Domain gammaScat("gammaScat");
domain::Domain gammaExt("gammaExt");
```

Listing 5.1: `Domain` objects needed to model the scattering problem example.

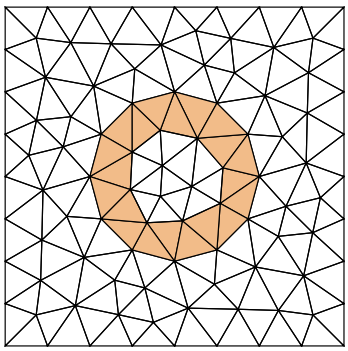
2.2 Function objects

All symbolic mathematical expressions in GmshFEM are managed with the help of a `Function<T_Scalar, T_Degree>` where:

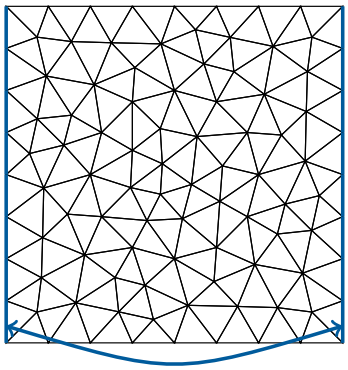
- `T_Scalar` is the algebraic structure (the real or complex set) with a desired pre-



(a) A mesh made of two regions: the inner disk and the exterior square. `Domain` classes can be used, for instance, to tag them.



(b) Example of `SkinLayer` class used to model elements inside the disk that are adjacent to the boundary between the disk and the exterior square domain.



(c) A `PeriodicLink` class that models a periodic condition between the left and the right boundary.

Figure 5.2: The three child classes of the `GeometricObject` abstract class that can be used in GmshFEM to define a finite element problem.

cision (single or double precision) of the function, *i.e.* `std::complex<double>`, `std::complex<float>`, `double` or `float`.

- `T_Degree` is the tensor degree of the function, *i.e.* 0 for a scalar, 1 for a vector, 2 for a tensor of order 2 and 4 for a tensor of order 4.

For simplicity, the tensor degree template argument can be omitted if one of the following alias are used:

- `ScalarFunction<T_Scalar>`,
- `VectorFunction<T_Scalar>`,
- `TensorFunction<T_Scalar>` and
- `TensorFunction<T_Scalar, 4 >`.

Functions store symbolic expressions in a tree structure that is evaluated at runtime, *e.g.* during pre-processing, assembly, residual calculation, or post-processing. The tree structure is designed to optimize the memory consumption of each symbolic expression evaluation. Each function can export a report in `HTML` format that shows the tree structure of the function and the peak and current memory consumption by point evaluation using the member function `exportExecutionTree`, that takes as arguments a domain of evaluation, an output file name and an optional output path.

In addition, piece-wise functions can also be defined using classes:

- `ScalarPiecewiseFunction<T_Scalar>`,
- `VectorPiecewiseFunction<T_Scalar>`,
- `TensorPiecewiseFunction<T_Scalar>` and
- `TensorPiecewiseFunction<T_Scalar, 4 >`,

that are filled-in using the member function `addFunction`, which takes as arguments a `Function` object of the same tensor degree as the `PiecewiseFunction` object and a domain of definition. These `PiecewiseFunction` classes are very useful for defining properties such as material parameters that vary from one physical group to another in a model.

On the one hand, these classes are versatile, allowing the writing of various functions such as transcendental functions, interpolation functions, mesh coordinates evaluations, and finite element field evaluations. The `C++` arithmetic operators are overloaded for these classes, which makes it possible to write expressions in a compact form close to their mathematical definition. New functions can, of course, be added by users, going as far as hard-coding the full terms necessary for *e.g.* assembling complex variational formulation or post-processing computations. This versatility can, for example, be advantageously used for multilevel methods, where a function can embed another solver; or for optimization or inverse problems, where automatic differentiation engines can be efficiently interfaced.

On the other hand, the tree structure allows for writing external parsers that can directly feed complex functional expressions to GmshFEM, allowing to embed GmshFEM in other codes and benefit from its high-performance C++ numerical kernel without having to recompile the library; and still keep the user-friendliness and flexibility of a scripted code, without incurring the runtime costs of scripted languages. This is directly inspired by the design of GetDP [70], where all expressions are analyzed once during a parsing phase and executed at runtime later on.

In our example scattering problem, the plane wave incident function $u_{\text{inc}} = \exp(\iota \mathbf{k} \cdot \mathbf{x}) = \cos(\mathbf{k} \cdot \mathbf{x}) + \iota \sin(\mathbf{k} \cdot \mathbf{x})$ must be defined. We chose an angle of incidence such that $\mathbf{x} = (0.3, 1, -0.2)$. Listing 5.2 shows how this function can be defined using the GmshFEM function setting. The `x<T_Scalar>()`, `y<T_Scalar>()` and `z<T_Scalar>()` functions return the x-, y- and z-coordinate of each considered point when it will be evaluated on a domain and the `vector<T_Scalar>()` function is used to create a vector. Note that the clause `using Scalar std::complex<double>;` is used such that `Scalar` stands as a shorter alias of `std::complex<double>`.

```
typedef std::complex<double> Scalar;
Scalar im(0.,1.);
function::VectorFunction<Scalar> xVec =
  function::vector<Scalar>(
    0.3 * function::x<Scalar>(),
    1. * function::y<Scalar>(),
    -0.2 * function::z<Scalar>()) / std::sqrt(0.3*0.3+1.+0.2*0.2);
function::VectorFunction<Scalar> kVec =
  function::vector<Scalar>(k, k, k) / std::sqrt(3.);
function::ScalarFunction<Scalar> uInc =
  function::cos(kVec * xVec) + im * function::sin(kVec * xVec);
```

Listing 5.2: Definition of a scalar function used as boundary condition to model the scattering problem example.

2.3 Field objects

The `Field` class is designed to store information about a finite element field, its associated discrete function space, and the degrees of freedom (DoF) map. To write any finite element problem based on its variational form, one or several instantiations of the `Field` class are employed. Once the interpolation coefficients, *i.e.* the degrees of freedom values, have been computed, these `Field` objects can be evaluated, for instance, to be used in other formulations, to be saved as post-processing views with the Gmsh API, or be exchanged across subdomains in domain decomposition algorithms. A `Field` object is fully compatible with `Function` classes defined in the previous section such that 0-form and 3-form fields can be considered as scalar functions, and 1-from and 2-form fields can be considered as vector functions.

A `Field<T_Scalar, T_Form>` class has two templates: the first one, `T_Scalar` is defined in the previous section, and the second, `T_Form` defines the mathematical differential form of the field. The differential form is either `Form0` for continuous scalar fields, `Form1` for vector curl-conform fields that ensure the continuity of the tangential trace between elements, `Form2` for vector div-conform fields that ensure the continuity of the normal trace between elements and `Form3` for scalar fields without any continuity

constraint between elements. For instance, 0-forms (scalar fields) are needed in our acoustic scattering example, while 1-forms will be needed for the electromagnetic time-harmonic wave problems.

The simplest fields are instantiated in GmshFEM with three arguments: a name, a domain of definition (through a `Domain` object), and the identifier of a discrete function space. For non-isoparametric interpolations, a fourth argument specifies the desired interpolation order. GmshFEM currently supports both Lagrange basis functions, arbitrary order hierarchical basis functions for H^1 and $\mathbf{H}(\mathbf{curl})$ [176], and a constant by element basis function for L^2 space.

The `Field` class also manages Dirichlet or periodic boundary conditions with the help of `addConstraint()` and `addPeriodicConstraint()` member functions, respectively. The member function `addConstraint()` takes two arguments: a `Domain` object where the Dirichlet condition is imposed and a `Function` object corresponding to the condition itself. Note that the function must be of the same tensor degree as the field, *i.e.* as `ScalarFunction` for 0-form and 3-form fields and a `VectorFunction` for 1-form and 2-form fields. The `addPeriodicConstraint()` also takes two arguments: a `PeriodicLink` object giving the links between a master and a slave geometric region and a `T_Scalar` coefficient such that the value of the field on the slave region is equal to the field value on the master region times the coefficient.

In our acoustic scattering example, the discrete scalar field u^h approximating the solution u of the variational form (5.2) is transcribed into a `Field` object templated over the `Scalar` type defined in Listing 5.2, as shown in Listing 5.3. It is defined over the open domain approximating Ω, *i.e.* the `Domain` object named "omega" defined in Listing 5.1, and we chose to interpolate it with a hierarchical basis function of order 3. This basis function order is used to decrease the pollution effect presented in Chapter 1, Section 4. In addition, we impose the incident plane wave defined in Listing 5.2 as Dirichlet condition on the scattered boundary named "gammaScat" defined in Listing 5.1.

```
field::Field<Scalar, form::Form0>
  u("u", omega, functionSpaceH1::HierarchicalH1, 3);
// Dirichlet BC: u = -u_inc on gammaScat
u.addConstraint(gammaScat, -uInc);
```

Listing 5.3: The 0-form field defined to model the acoustic wave scattering problem example.

2.4 Formulation objects

The `Formulation` object stores the symbolic representation of the variational formulation of the problem 5.2. It can evaluate linear and bilinear forms, store the corresponding matrix systems, and request their solution through external linear algebra packages [7, 8, 18, 111]. The `Formulation<T_Scalar>` class is templated on the `T_Scalar` type defined in Section 2.2.

For continuous Galerkin finite elements formulations, the `Formulation` object provides the `integral` member function, whose two first `Function` arguments are the inner product arguments describing one term in the variational formulation: the first can involve any linear function of an unknown field, denoted by `dof()`, or any function

independent of any unknown field; the second must involve a linear function of the test function, denoted by `tf()`. If the first argument involves an unknown field, it evaluates a bilinear form; otherwise, it leads to evaluating a linear form. Each argument involving an unknown field is modeled as an instance of an `Equation` class that stores an `UnknownField` object containing information about the unknown field and the possible derivative operator applied to it, two `Function` objects multiplying the `UnknownField` object by the left side and by the right side and the two associated products (*e.g.* a scalar product, a vector product, a tensor product). The third argument specifies the domain over which the integration is performed, *i.e* a `Domain` or a `SkinLayer` object, the fourth specifies the quadrature rule (*e.g.* `Gauss8` for a Gauss quadrature suited for integrating 8^{th} order polynomials). Finally, the last argument is optional and specifies the definition of the product used between the first and the second argument; if the value is set to `term::ProductType::Scalar`, then no conjugation is applied, and therefore for complex arguments, the product is not an inner product. Otherwise, if it is set to `term::ProductType::Hermitian` (the default value) and the second argument is conjugated. Note that with real arguments, both product types are equivalent and correspond to the definition of an inner product. The product depends on the tensor degree of the left l and right r argument. It corresponds to a scalar multiplication, lr, for scalar arguments; a scalar product (*i.e.* a dot product), $\mathbf{l} \cdot \mathbf{r} = (\mathbf{l})_i(\mathbf{r})_i$, for vector arguments; and a twice contracted product (*i.e.* a double dot product), $\mathbb{L} : \mathbb{R} = \text{tr}\,(\mathbb{L}\mathbb{R}^T) = (\mathbb{L})_{ij}(\mathbb{R})_{ij}$, for tensor arguments.

Let us imagine a problem where a field `u` is involved. It is essential to understand the difference between a term defined in a formulation with an expression containing `dof(u)` and another containing only `u`. When the field is tagged using the function `dof()`, it is considered an unknown field in the corresponding formulation. When the field is not tagged, it is considered a simple function that will be evaluated during the assembly of the system and not as an unknown. For instance, the term containing the inner product "`dof(u), tf(u)`" is considered as a bilinear term that will be a part of the left-hand side of the system, while the term "`u, tf(u)`" is processed as a linear term that will be a part of the right-hand side of the system, *i.e.* a source term.

Some terms like the $\int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, \overline{v}\, \mathrm{d}\Omega$ of (5.2) require to express the exterior derivative of the unknown field, *i.e.* the gradient of a 0-from, the rotational of a 1-form or the divergence of a 2-form. This exterior derivative is expressed in GmshFEM using the function `d()` where the argument is either `dof(u)` or `u` depending if we are dealing with an unknown field or the evaluation of a known field. Every type of field can be used with the exterior derivative function `d()`. Furthermore, field-type-specific operators are also defined for convenience (`grad()`, `curl()` and `div()`); they can be used as the operator `d()` and will produce the exact same results.

An arbitrary number of `integral` terms can be specified: they are all summed to produce the final discrete variational formulation. This symbolic expression of the variational form is identical to the one introduced by GetDP [72] and allows to handle coupled and mixed formulations seamlessly. For simple implicit formulations, the `Formulation` provides three member functions encapsulating the pre-processing phase, *i.e.* the identification of degrees of freedom and constraints (`pre()`), the assembly of the linear system (`assemble()`) and the solution of the linear system (`solve()`) or the solution of the eigenvalue system (`eigensolve()`).

In our acoustic scattering example, three bilinear terms (two in the volume, one on the artificial boundary) encode the discrete version of the variational form (5.2) in a `Formulation` object that we chose to name "helmholtz": see Listing 5.4. Note that for convenience's sake, we introduce functions `dof()`, `tf()` and `grad()` defined into the namespace `equation` and all functions defined into the namespace `function` into the current namespace. As the variational form (5.2) is expressed with Hermitian products, the last argument can be ignored as the Hermitian product is the default value.

```
using equation::dof;
using equation::tf;
using equation::grad;
using namespace function;

problem::Formulation<Scalar> formulation("helmholtz");

formulation.integral(grad(dof(u)), grad(tf(u)), omega, "Gauss8");
formulation.integral(- k*k * dof(u), tf(u), omega, "Gauss8");
formulation.integral(- im*k * dof(u), tf(u), gammaExt, "Gauss8");

formulation.pre();
formulation.assemble();
formulation.solve();
```

Listing 5.4: Definition of the formulation for the acoustic wave scattering problem example.

2.5 Post-processing functions

Once a problem is solved, fields can be post-processed with the help of any `Function`, using a variety of operations. Currently three main post-processing features are available: direct data export on a mesh, evaluation of integrals and evaluation at arbitrary points.

The simplest operation consists in exporting the data as a Gmsh post-processing view through the `save()` function. Three versions of `save()` are defined. The first one takes three mandatory arguments and six optional ones. The mandatory arguments are: the function to export (*i.e.* any `Function` objects), a geometric object where the function should be evaluated (i.e. a `Domain` or a `SkinLayer` object) and a file name. The optional arguments are: a post-processing file format (the default value is `msh`), an output path (the default is set to the current path), a boolean to activate if the function has to be save in memory and not on disk, a "step" that specifies the identifier (≥ 0) of the data in a sequence, a "time" argument associating a time value with the data and a "partition" that allows one to specify data in several sub-sets. The second version takes almost the same arguments as the first one, except the second parameter; the geometric object is substituted by a `Mesh` object that models geometric primitives such as circles, disks, lines, planes and spheres. The last version is a more efficient version of the first one for 0-form fields discretized using iso-parametric or first order hierarchical bases. It takes a 0-from field as mandatory argument and the same six optional ones as the previous versions.

The integration post-processing function, `integrate()`, is declined in two versions that both take three mandatory arguments and return the evaluation of the integral.

The first one takes a `Function` object as integrand of the integral, a domain of integration (*i.e.* a `Domain` or a `SkinLayer` object) and a quadrature rule. The second version allows to integrate over a `Mesh` object.

The last post-processing function, `evaluate()`, allows to evaluate any `Function` object f, given as first argument, on any point $P : (p_x, p_y, p_z)$, where p_x, p_y and p_z are given by the last three arguments. The function returns the evaluation of $f(P)$. Note that the return type of `integrate()` or `evaluate()` depends of the tensor degree of the function argument. Both functions called with a `Function<T_Scalar, T_Degree>` return an object of type `MathObject<T_Scalar, T_Degree>::Object`. Note that `MathObject<T_Scalar, Degree::Degree0>::Object` is equivalent to the type `T_Scalar`.

```
const double rho = 1.2; // [kg / m^3]
const double f = 200; // [Hz]
const double pi = 3.14159265359;

VectorFunction<Scalar> v = - im / (2. * pi * rho * f) * grad(u);

post::save(u, omega, "u");
post::save(v, omega, "v");

post::Sphere sphere("sphere", -18., -0.7, 0.272, 0.8, 100);
Scalar power =
  post::integrate(u * v * function::normal<Scalar>(),
  sphere, "Gauss8");
msg::info << "Power = " << power << "[W]" << msg::endl;

post::Plane cut("cut", -19., -2.23, 0.275, // cut-plane origin
                       20.38, 0., 0., // x vector
                       0., 5.55+2.23, 0, // y vector
                       500, 190); // x & y discretization
post::save(u, cut, "u_cut", "pos");
post::save(v, cut, "v_cut", "pos");
```

Listing 5.5: Some post-processing operations applied to the acoustic wave scattering problem example.

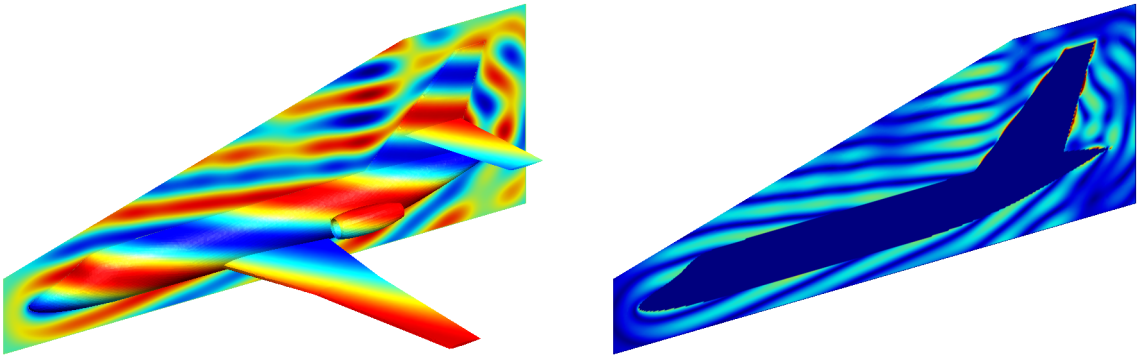
The application of those post-processing functions is shown in Listing 5.5. For our example, we chose the save the pressure field u and the local particle velocity $\mathbf{v}$ on the domain Ω_{tot}. The local particle velocity is given by Equation 1.25 of Chapter 1,

$$\mathbf{v} = -\frac{\iota}{2\pi\rho_0 f}\,\mathbf{grad}\,u, \tag{5.3}$$

where $\rho_0 = 1.2$ kg/m^3 and $f = 200$ Hz. Figure 5.3 shows fields u and $\mathbf{v}$ evaluated on the median plane of the Falcon. Furthermore, we also compute the sound power given by

$$P = \int_S u\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S, \tag{5.4}$$

where S is a sphere of radius 0.8 centered on the nose plane $(-18, -0.7, 0.272)$ and $\mathbf{n}$ is its outgoing normal. The `Sphere` object is constructed with a name, the coordinate of its center, its radius, and the number of points used to discretize it. Finally, the



(a) The real part of the pressure field. (b) The norm of the particle velocity field.

Figure 5.3: The solution to the acoustic scattering problem depicted on the median plane.

solution is evaluated and exported on the median cut modeled by a plane `Plane` object defined by a name, the coordinate of the origin of the cut-plane, the coordinate of its base vectors $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ and the number of discretized points in the $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ direction, respectively. Figure 5.3 shows the output pressure and particle velocity fields.

3 The C++ implementation

For implicit, low-order continuous Galerkin finite element formulations, the most time-consuming part of the finite element process (CAD and meshing aside) resides in the solution of the resulting large, sparse linear systems. However, high-order finite element methods, which are increasingly used for complex simulations to alleviate the slow grid convergence of the state-of-the-art (usually second order) methods provided by most industrial codes, the mere process of assembling the finite element matrices, or computing the residuals or the time iterates, rapidly becomes a bottleneck in the computer implementation.

While high-order finite elements naturally lead to increased arithmetic intensity, since the local element-wise matrices become larger and denser, the number of quadrature points also dramatically increases. In such cases, the best way to achieve good performance is to reformulate all the quadratures as dense matrix-matrix products [6, 137], and by pre-computing as many of the underlying matrices as possible. While a natural decomposition resides in the separation of the metric-dependent and metric-independent parts in the Galerkin terms [137], many codes trade off accuracy and generality for performance by assuming, for example, that all parts of the integrands are interpolated using the same bases as the unknown fields [6]. This is not suitable for *e.g.* strongly nonlinear problems or multiscale problems, though, as the coefficients do not have the same regularity. The cost of pre-computing and storing local matrices is exacerbated when using hierarchical bases, or vector-valued basis functions for *e.g.* $\mathbf{H}(\mathbf{curl})$ or $\mathbf{H}(\mathrm{div})$, where the basis functions depend on the orientation of the elements [176, 137]. Storing all unassembled matrices in such cases rapidly leads to prohibitive memory requirements, even in cases where direct linear solvers eventually factorize the matrix.

In GmshFEM, the efficient evaluation of linear and bilinear forms is based on a compromise between processing time and memory usage. Good parallel performance is obtained by carefully analyzing the effect of spatial and temporal data locality.

3.1 The pre-processing phase

Before any evaluation of linear or bilinear forms occurs, GmshFEM performs a pre-processing phase, which builds a dictionary of degrees of freedom (identified by keys) based on the input mesh, the fields with their associated function spaces, and the finite element formulation. A unique tag is associated with each unknown DoF, corresponding to an equation number. Tags are chosen, for example, such that "bubble" degrees of freedom (which only depend on the element and are not shared between mesh entities) are explicitly identified, which helps assemble them without locks due to their one-element locality. For implicit formulations, the pre-processing phase also allocates the arrays used to store the finite element matrices in a *compressed row storage* (CRS) format. The sizes of these arrays are known by computing the pattern of the finite element matrix by assuming that all local matrices (resulting from the integration over one element) are dense. The parallel efficiency of the pre-processing is limited by the performance of the hash map used to identify DoFs, locks required by the calculation of the global matrix pattern, and the memory allocation needed to instantiate `Dof` objects and the vectors used by the CRS format.

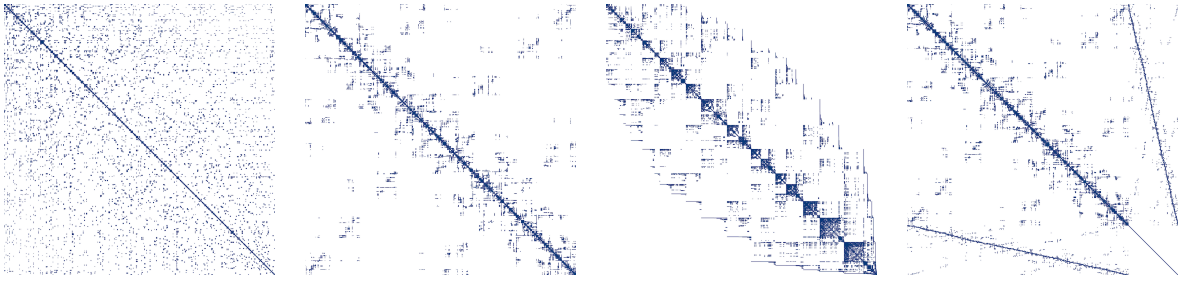
The pattern of the finite element matrix depends on the DoF ordering algorithm, which can have an impact on the performance of the assembly. Indeed, to some extent, the spatial and temporal locality can be optimized by choosing a good DoF ordering. Currently, four algorithms are available in GmshFEM:

- No DoF ordering: DoFs are considered in the order returned by the Gmsh API.
- Hilbert sort (space filling curve) ordering: this keeps DoFs that are close spatially close in memory [105, 171].
- Reverse Cuthill–McKee (RCM) ordering: this reduces the matrix bandwidth and thus also improves memory locality [62].
- Default DoF ordering: this forces the bubble DoFs to be stored at the end of the matrix while the other DoFs are sorted using the Hilbert sort algorithm.

The effect of DoF ordering on the pattern of a finite element matrix corresponding to a small 3D scattering problem like the one described in the previous section can be observed in Figure 5.4. As can be seen, with the default DoF ordering algorithm, the block matrix corresponding to the bubble DoFs appears in the lower right corner.

3.2 The function tree structure

As explained in Section 2.2, each function is stored in memory as a tree structure. For instance, the incident plane wave 5.2 used in the scattering problem is described by the tree structure of Figure 5.5. Every tree node corresponds to a basic operation, *e.g.* arithmetic operation, transcendental func-



(a) No DoF sort (b) Hilbert (c) RCM (d) Default

Figure 5.4: Spy plots of a small version of the example problem (5.2) (6600 DoFs) for different DoF ordering algorithms.

tion, field evaluation, *etc.*; they are all an instance of child classes of the interface `ExecutionTree<T_Scalar>` class. For instance, a node could be an instance of `NullaryNode<T_Op>`, `UnaryNode<T_Op>`, `BinaryNode<T_Op>`, `TernaryNode<T_Op>`, and so on, depending on the number of parameters, *i.e.* the arity of the associated function. Furthermore, advanced nodes such as `FieldNode<T_Op>` for operations related to finite element fields, `AnalyticalNode<T_Op>` for pre-defined analytic expressions or `ScalarTypeNode<T_Op>` for functions that will change the scalar type of a function such as `real()` or `imag()` (that take a complex function in argument and return a real function), are also defined.

All nodes all templated on a type `T_Op`, which corresponds to a child of an `Operation` class. For instance, a `BinaryNode` class must be associated through a `BinaryOperator` class as template type. The `Operation` classes have to override the call operator `operator()` that is responsible for the computation of the needed function. For example, the first node of Figure 5.5 is a `BinaryNode<Add>` object, namely a `BinaryNode` instantiated with a `BinaryOperator` class named `Add` as template type. This node is responsible for the addition of two scalar parameters `a` and `b`, thus the `Add<T_Scalar>` class must override the operator `operator()` as shown in Listing 5.6. The function takes several arguments: first, a vector named `values` that will contain the result of the node evaluation, two vectors of parameters (*i.e.* `a` and `b`), two vectors containing the spatial coordinates and the reference local coordinates of where the function has to be evaluated, and finally, the type of element and the entity on which the function is evaluated.

```
void operator()(std::vector< T_Scalar > &values,
  const InputVector< T_Scalar > &a,
  const InputVector< T_Scalar > &b,
  const std::vector< scalar::Precision< T_Scalar > > &points,
  const std::vector< scalar::Precision< T_Scalar > > &gaussPoints,
  const int elementType,
  const std::pair< int, int > &entity) const override
{
#pragma omp for
  for(auto i = 0ULL; i < values.size(); ++i) {
    values[i] = a[i] + b[i];
  }
}
```

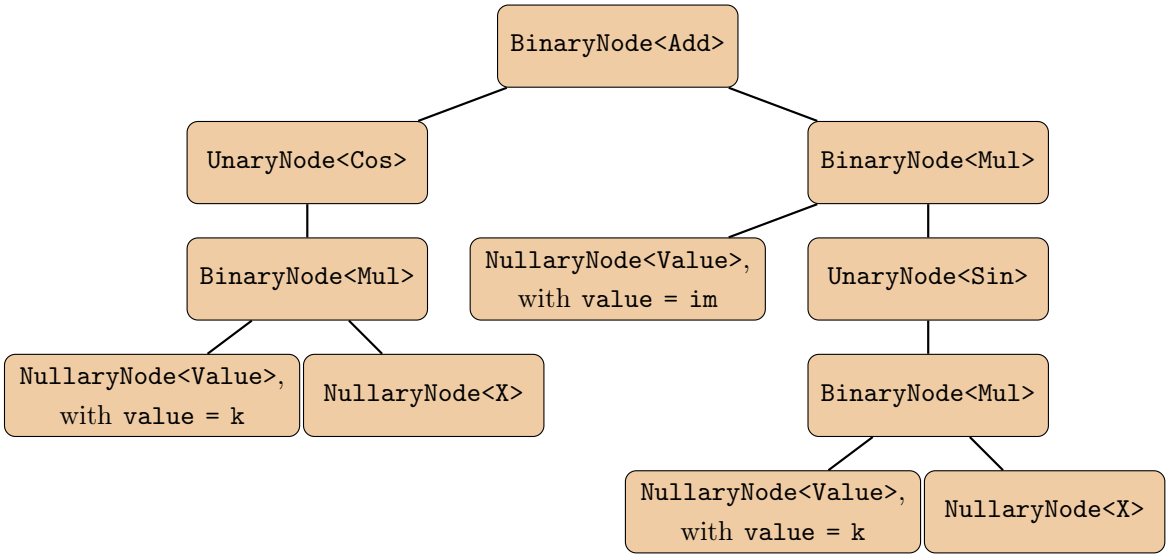


Figure 5.5: Function tree by GmshFEM to represent the incident plane wave of Listing 5.2.

Listing 5.6: Exemple of overriden call operator that compute the addition of two scalar parameters `a` and `b`.

Nodes are optimized to reduce memory consumption and reduce slow allocation calls during evaluation. For instance, an evaluation of the incident plane wave function of Figure 5.5 has a peek memory consumption by evaluation point of 16 B, *i.e.* the memory needed to store a complex number in double precision; that means only one extra vector of complex numbers is allocated during its evaluation. Nodes are optimized such that: on the one hand, constant expressions are not evaluated at each evaluation point but only once; on the other hand, every operator returns some information that the node can use to determine if it can reuse some intermediate arrays. For instance the addition operator of Listing 5.6 can be optimized as

```
values[i] = values[i] + b[i];
```

where the input vector `a` uses the same vector as the output one. Furthermore, this operator can be optimized a bit more if leaves `a` and `b` are identical,

```
values[i] = values[i] + values[i];
```

such that in this case, the `BinaryNode` does not need to allocate arrays to evaluate its operator.

By inheritance, users can define their own operations. The simple but also less flexible method is to define a new operator by inheriting from the appropriated `Operator` class (definition of a custom node). For instance, if someone wants to define a new operation that takes one argument, it has to create a class that inherits from the `UnaryOperator` mother class and overrides the call operator as done in Listing 5.6. This method is quite simple to implement and can keep the memory optimization of

node classes. Nevertheless for some advanced uses, the user also has to redefine its own node class which is a bit less simple (but still affordable for users accustomed to C++) but offers enormous possibilities.

Finally, note that every post-processing function presented in Section 2.5 is also related to this function evaluation algorithm. Indeed, both data exportations, integral evaluations, or point evaluations are carried out by evaluating a given function.

3.3 The assembly phase

The assembly process can be seen as an algorithm that combines information of different nature and stores the result in a finite element matrix. For instance, let us consider the local stiffness matrix of the variational form (5.2) over any element Ω_e of the domain Ω_{tot}. This local stiffness matrix is given by the following general expression:

$$A_{ij} = \int_{\Omega_e} \left[f(\mathbf{x}) \mathbb{J}^{-T} \mathbf{grad}\, \hat{\phi}_i \right]^T \overline{\left[g(\mathbf{x}) \mathbb{J}^{-T} \mathbf{grad}\, \hat{\phi}_j \right]} \det(\mathbb{J})\, \mathrm{d}\Omega_e. \tag{5.5}$$

where $f(\mathbf{x})$ and $g(\mathbf{x})$ are functions equal to one for the stiffness matrix of the variational form (5.2), $\hat{\phi}_i$ and $\hat{\phi}_j$ are the basis functions associated with the i^{th} or j^{th} degree of freedom over Ω_{tot}, and $\mathbb{J} = {}^{\partial x_i}/_{\partial \hat{x}_j}$ is the Jacobian matrix mapping the reference coordinates ($\hat{\mathbf{x}}$) of the reference element to the mesh coordinates ($\mathbf{x}$). The integrand is a combination of geometric, *i.e.* metric-dependent, information (the Jacobian matrix $\mathbb{J}$ and its determinant $\det(\mathbb{J})$), metric-independent basis function data ($\mathbf{grad}\, \hat{\phi}_i$ and $\mathbf{grad}\, \hat{\phi}_j$), and arbitrary function evaluations ($f(\mathbf{x})$ and $g(\mathbf{x})$). The metric-dependent and metric-independent data are currently returned by the Gmsh API, while the function evaluations are computed by the tree structure presented in Section 2.2.

Depending on the nature of the field, the change of coordinates for the reference element $\hat{K}$ to the mesh element K is different. Given function f or $\mathbf{f}$ in the mesh element coordinate system $\mathbf{x}$ and $\hat{f}$ or $\hat{\mathbf{f}}$ is the reference element coordinate system $\hat{\mathbf{x}} = \mathbf{\Phi}^{-1}(\mathbf{x})$, with $\mathbf{\Phi} : \hat{K} \to K$, the following changes of coordinate must be applied [176]:

- mapping of functions in H^1: $f(\mathbf{x}) = (\hat{f} \circ \mathbf{\Phi}^{-1})(\mathbf{x})$,
- mapping of functions in $\mathbf{H}(\mathbf{curl})$: $\mathbf{f}(\mathbf{x}) = (\mathbb{J}^{-T} \hat{\mathbf{f}} \circ \mathbf{\Phi}^{-1})(\mathbf{x})$,
- mapping of functions in $\mathbf{H}(\mathrm{div})$: $\mathbf{f}(\mathbf{x}) = (\mathbb{J} \det(\mathbb{J})^{-1} \hat{\mathbf{f}} \circ \mathbf{\Phi}^{-1})(\mathbf{x})$,
- mapping of functions in L^2: $f(\mathbf{x}) = (\det(\mathbb{J})^{-1} \hat{f} \circ \mathbf{\Phi}^{-1})(\mathbf{x})$.

The example equation (5.5) applies the $\mathbf{H}(\mathbf{curl})$ mapping to the basis functions, as the gradient of a function defined in H^1 is computed. In GsmhFEM, the mapping function is managed by the `FieldEvaluator< T_Scalar, T_Form>` classes that take as argument the Jacobian matrix, its determinant, and the basis function. Mathematically the `FieldEvaluator` classes are operators defined as:

$$\begin{aligned} \texttt{FieldEvaluator} : \hat{K} &\to K \\ \hat{\phi} &\mapsto \phi \end{aligned} \tag{5.6}$$

where $\hat{\phi}$ is a basis function in the reference space, and ϕ is the corresponding basis function in the mesh space.

Once basis functions are expressed in the mesh space, expression of type $F(\mathbf{x}) \star_L \phi \star_R G(\mathbf{x})$, where F and G are scalar, vector or tensor functions, and $\star_L$ and $\star_R$ are any product compatible with the tensor degree of F, ϕ and G, must be evaluated. Note that we consider here that $\star_L$ and $\star_R$ also induce the priority of operations such that depending on their definition, the product should be understood as $F(\mathbf{x}) \star_L (\phi \star_R G(\mathbf{x}))$ or $(F(\mathbf{x}) \star_L \phi) \star_R G(\mathbf{x})$. For instance, in the expression (5.5), products $f(\mathbf{x})\phi_i$ and $g(\mathbf{x})\phi_j$, where f and g are scalar functions, have to be computed. In GmshFEM, these products are computed by the `EquationEvaluator` classes templated on lots of parameters: the scalar type (*i.e.* `T_Scalar`) the tensor degrees of F and G, the differential form of ϕ and the definition of products $\star_L$ and $\star_R$ with the associated priority. Mathematically, these classes are operators that take the basis function and the functions F and G and return a function of tensor degree depending on their parameters. Let us call $L(\mathbf{x})$ the function returned by the left expression and $R(\mathbf{x})$ the function returned by the left expression such that any local matrix assembly can be expressed by

$$A_{ij} = \int_{\Omega_e} L(\mathbf{x})^T \overline{R(\mathbf{x})} \det(\mathbb{J}) \, \mathtt{d}\Omega_e. \tag{5.7}$$

For the left or right expression, the appropriate `FieldEvaluator` and `EquationEvaluator` are respectively returned by the left or right `Equation` object instantiated during the declaration the term as done in Listing 5.4.

This integral is then evaluated using a numerical quadrature rule, leading to a weighted sum (with weights w_q) of evaluations of the integrand at integration points $\mathbf{x}_q$:

$$A_{ij} \approx \sum_{q=1}^{Q} L(\mathbf{x}_q)^T \overline{R(\mathbf{x}_q)} \det(\mathbb{J}_q) w_q. \tag{5.8}$$

Four kinds of data are therefore needed to assemble such a term over an element:

- Integration points and weights,
- Geometric information represented by the Jacobian matrix and its determinant evaluated at integration points,
- Basis functions evaluation at integration points,
- Arbitrary function evaluated at the integration points.

GmshFEM retrieves the first three data directly from the Gmsh API. For efficiency, Gmsh and GmshFEM deal with such data for groups ("buckets") of elements of the same type so that data can be accessed efficiently in contiguous chunks of memory, suitable for optimized vectorized operations. Three criteria define these buckets.

The first distinction is made on the type and geometrical order of the elements (*e.g.* lines, triangles, quadrangles, tetrahedra, ..., straight-sided or curved). Indeed, the integration points expressed in the reference coordinate system and their associated weights are identical for a given type if the same quadrature order is used. Furthermore, before assembling elements of the same type using the same quadrature order, basis functions can be pre-computed at integration points for all possible orientations. Then during the assembly process, each element has a tag that identifies its orientation.

```
forall elementTypes do
    basisFunctionsData ← ComputeNeededBasisFunctions();
    entities ← GetEntitiesHavingCurrentElementType();
    forall entities do
        precomputedNeededFunctions();
        geometricData ← ComputeNeededGeometricData();
        fieldPairs ← GetFieldPairsDefinedOverCurrentEntity();
        forall fieldPairs do
            dofIndices ← ComputeDOFIndices();
            AssembleAndStoreInMatrix(basisFunctionsData, geometricData,
             dofIndices);
        end
    end
end
```

Figure 5.6: Pseudo-code of the assembly algorithm.

The second distinction is made by geometrical entities, for which metric-dependent information is computed in a single pass by Gmsh. Moreover, as variational form integrals are defined over geometric entities, mathematical functions appearing in their integrands can be pre-computed for all integration points at this stage.

The third distinction depends on the formulation. When a bilinear term is defined, it involves a pair of fields, an unknown field, and test functions associated with the same field or with another field. Once the problem is discretized, this pair corresponds to a block in the finite element matrix. Therefore, it is suitable to assemble terms pair by pair to avoid unnecessary displacements in memory that will negatively impact the cache efficiency of the program.

Once this data is pre-computed (in parallel) and stored in arrays, the assembly proceeds in parallel for elements belonging to the same bucket. Each thread is responsible for contiguous elements, and on each element, linear algebra operations are handled by the Eigen [102] `C++` template library for linear algebra. As threads assemble contiguous elements, they combine contiguous parts of the pre-computed data. As a result, spatial memory locality is maximized as data needed to process an element are always close to each other, and temporal memory locality is also maximized as the needed data to assemble the next element are close to the data used to assemble the current one.

Finally, the local matrix elements are pushed into the global matrix stored in CRS format at locations given by pre-computed indexed arrays. As mentioned above, a single atomic addition directive is applied to avoid race conditions, except for bubble DoFs. The whole assembly procedure is summarized in Figure 5.6.

4 Other features

In order to present other features developed in GmshFEM, we have to step a bit outside the main framework of this thesis. In Sections 4.1 and 4.2, the plane geometry

of Figure 5.1a remains unchanged but the physics is adapted to model time-harmonic elastodynamic and electromagnetic waves, respectively. In Sections 4.3 and 4.4, the electrostatic modeling of a cylindrical capacitor is briefly considered to shed light on some other interesting features of GmshFEM, which can be useful in a variety of applications: infinite and axisymmetric transformations and the introduction of global quantities.

4.1 Elastodynamic time-harmonic waves

Let us consider the counterpart of the scattering problem 5.1 in elastodynamics. The keep the strong parallel with the other wave problem examples, let us consider the unphysical problem of a plane inside a solid medium hit by a P-incident plane elastic wave $\mathbf{u}_{\text{inc}} = u_{\text{inc}}\,\mathbf{x}$. This problem is modeled by the following vectorial Navier equation,

$$\begin{cases} \dfrac{1}{k_P^2}\,\mathbf{grad}\,\mathrm{div}\,\mathbf{u} - \dfrac{1}{k_S^2}\,\mathbf{curl}\,\mathbf{curl}\,\mathbf{u} - \mathbf{u} = \mathbf{0} & \text{in } \Omega_{\text{tot}}, \\ \mathbf{u} = -\mathbf{u}_{\text{inc}} & \text{on } \Gamma_{\text{scat}}, \\ \Gamma^t(\mathbf{u}) - \dfrac{\iota}{k_P}\mathbb{I}_n\mathbf{u} - \dfrac{\iota}{k_S}\mathbb{I}_\tau\mathbf{u} = \mathbf{0} & \text{on } \Gamma_{\text{ext}}, \end{cases} \tag{5.9}$$

where $k_P^2 = \rho\omega^2/(\lambda + 2\mu)$ and $k_S^2 = \rho\omega^2/\mu$ are the P-wave and S-wave wavenumbers, with λ and μ are Lamé parameters, $\mathbb{I}_n = \mathbf{n}\otimes\mathbf{n}$, $\mathbb{I}_\tau = \mathbb{I} - \mathbb{I}_n$, $\mathbf{u}$ is the unknown displacement field and $\Gamma^t(\cdot)$ is the trace operator defined on a boundary Γ as

$$\Gamma^t : \mathbf{u} \mapsto \mathbf{u}^t := 2\mu\partial_{\mathbf{n}}\mathbf{u}|_\Gamma + \lambda\mathbf{n}\,\mathrm{div}\,\mathbf{u}|_\Gamma + \mu\mathbf{n}\times\mathbf{curl}\,\mathbf{u}|_\Gamma. \tag{5.10}$$

To express the variational formulation of Problem 5.9, let us use the following 4^{th} order elastic tensor $\mathbb{C}$

$$(\mathbb{C})_{ijkl} := \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}), \tag{5.11}$$

$$= \left(\frac{1}{k_P^2} - \frac{2}{k_S^2}\right)\delta_{ij}\delta_{kl} + \frac{1}{k_S^2}(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{5.12}$$

where δ_{ij} is the Kronecker delta, such that variational formulation is: Find $\mathbf{u} \in [H^1(\Omega_{\text{tot}})]^3$ such that

$$\int_{\Omega_{\text{tot}}} (\mathbb{C} : \mathbf{grad}\,\mathbf{u}) : \mathbf{grad}\,\overline{\mathbf{v}} - \rho\omega^2\mathbf{u}\cdot\overline{\mathbf{v}}\;\mathrm{d}\Omega_{\text{pla}} = 0 \tag{5.13}$$

holds for every test function $\mathbf{v} \in [H^1(\Omega_{\text{tot}})]^3$.

For this elastodynamic problem, the 4^{th}-degree elastic tensor $\mathbb{C}$ must be defined as done in Listing 5.7. A constant 4^{th}-order tensor object of type `MathObject<Scalar, Degree::Degree4>::Object` is created using the definition (5.12), then a 4^{th}-tensor-degree function, *i.e.* a `TensorFunction<Scalar, 4>` object, is instantiated using this constant object. In addition, the vector incident plane wave is also defined using the acoustic incident wave definition, and the direction of propagation is defined in Listing 5.2.

```
typename MathObject<Scalar, Degree::Degree4>::Object Ctmp;
for(int i = 0; i < 3; ++i) {
  for(int j = 0; j < 3; ++j) {
    for(int k = 0; k < 3; ++k) {
      for(int l = 0; l < 3; ++l) {
        if(i == j && k == l) {
          Ctmp(i,j)(k,l) += 1./(kP*kP) - 2./(kS*kS);
        }
        if(i == k && j == l) {
          Ctmp(i,j)(k,l) += 1./(kS*kS);
        }
        if(i == l && j == k) {
          Ctmp(i,j)(k,l) += 1./(kS*kS);
        }
      }
    }
  }
}
function::TensorFunction<Scalar, 4> C(Ctmp);
function::VectorFunction<Scalar> uIncVec = uInc * xVec;
function::VectorFunction<Scalar> n = function::normal<Scalar>();
function::TensorFunction<Scalar> In = function::dyadic(n,n);
function::TensorFunction<Scalar> It =
       function::identity<Scalar>() - function::dyadic(n,n);
```

Listing 5.7: Definition of a the 4th degree elastic tensor and the normal function.

As suggested for the elastic problem, each component of the $\mathbf{u}$ field is defined H^1. Such vector fields are modeled by a `CompoundField< Scalar, form::Form0, 3>` object (see Listing 5.8), where the last template parameter corresponds to the dimension on the vector field, here 3. Once this field is instantiated, it can be used as any other field presented before; for instance, the boundary condition can be applied on the plane using the `addConstraint()` member function.

```
field::CompoundField<Scalar, form::Form0, 3>
    u("u", omega, functionSpaceH1::HierarchicalH1, 2);
// Dirichlet BC: u = -u_inc on gammaScat
u.addConstraint(gammaScat, -uIncVec);
```

Listing 5.8: The 0-form compound field defined to model the elastodynamic wave problem.

Listing 5.9 presents the formulation of the elastodynamic problem 5.13. Once again, this problem transcription is very close to the variational formulation (5.13). The differential operator `grad` applied here corresponds to the vector gradient, while in the acoustic example, the `grad` operator is the classical scalar gradient. Note that the `dyadic()` function is used to compute the dyadic product $\mathbf{n} \otimes \mathbf{n}$ and that the identity matrix is returned with the function `identity()`.

```
using equation::dof;
using equation::tf;
using equation::grad;
using namespace function;

problem::Formulation<Scalar> formulation("navier");
```

```
formulation.integral(C*grad(dof(u)), grad(tf(u)), omega, "Gauss8");
formulation.integral(-dof(u), tf(u), omega, "Gauss8");
formulation.integral(-im/kP * In * dof(u), tf(u),
                                                gammaExt, "Gauss8");
formulation.integral(-im/kS * It * dof(u), tf(u),
                                                gammaExt, "Gauss8");

formulation.pre();
formulation.assemble();
formulation.solve();
```

Listing 5.9: Definition of the formulation for the elastodynamic wave problem.

Finally, similar post-processing as shown in Listing 5.5 can be applied to the elastodynamic problem, where the displacement field is represented: see Listing 5.10 and Figure 5.7a.

```
post::save(u, omega, "u");
post::save(u, cut, "u_cut", "pos");
```

Listing 5.10: Some post-processing operations applied to the elastodynamic wave problem.

4.2 Electromagnetic time-harmonic waves

Once again, let us consider the counterpart of the scattering problem 5.1, this time in electromagnetics. This problem is modeled by the following vectorial Maxwell equation [175],

$$\begin{cases} \mathbf{curl}\,\mathbf{curl}\,\mathbf{e} - k^2\mathbf{e} = \mathbf{0} & \text{in } \Omega_{\text{tot}}, \\ \gamma^T(\mathbf{e}) = -\gamma^T(\mathbf{e}_{\text{inc}}) & \text{on } \Gamma_{\text{scat}}, \\ \gamma^t(\mathbf{curl}\,\mathbf{e}) + \imath\, k\gamma^T(\mathbf{e}) = \mathbf{0} & \text{on } \Gamma_{\text{ext}}, \end{cases} \tag{5.14}$$

where $\mathbf{e}$ is the unknown scattered electric field, $\mathbf{e}_{\text{inc}}$ is the incident plane wave enforced on the scattering object (*i.e.* the plane) and $\gamma^t(\cdot)$ and $\gamma^T(\cdot)$ are the trace operators defined on a boundary Γ as

$$\gamma^t : \mathbf{e} \mapsto \mathbf{e}^t := \mathbf{n} \times \mathbf{e}|_\Gamma \qquad \text{and,} \qquad \gamma^T : \mathbf{e} \mapsto \mathbf{e}^T := \mathbf{n} \times (\mathbf{e}|_\Gamma \times \mathbf{n}), \tag{5.15}$$

where $\mathbf{a} \times \mathbf{b}$ designed the cross product. The last equation of (5.14) is the equivalent of the Sommerfeld condition for the electromagnetic problem, *i.e.* the Silver-Müller radiation condition [175].

Once again, the description of a finite element problem in GmshFEM is based on the variational formulation of Problem 5.14, namely: Find $\mathbf{e} \in \mathbf{H}(\mathbf{curl})(\Omega_{\text{tot}})$ such that

$$\int_{\Omega_{\text{tot}}} \mathbf{curl}\,\mathbf{e} \cdot \mathbf{curl}\,\overline{\mathbf{v}} - k^2\mathbf{e} \cdot \overline{\mathbf{v}}\, \mathrm{d}\Omega_{\text{tot}} - \int_{\Gamma_{\text{ext}}} \imath\, k\, [\mathbf{n} \times (\mathbf{e} \times \mathbf{n})] \cdot \overline{\mathbf{v}}\, \mathrm{d}\Gamma_{\text{ext}} = 0 \tag{5.16}$$

holds for every test function $\mathbf{v} \in \mathbf{H}(\mathbf{curl})(\Omega_{\text{tot}})$.

The geometric objects as described in Listing 5.1 remain valid, but the incident plane wave must be modified. For this electromagnetic problem, we inject a vector

plane wave polarized in the direction $\mathbf{p}_e$: $(1, -0.3, 0)$ such that $\mathbf{p}_e$ is orthogonal to the direction of propagation $\mathbf{x}$. Therefore, we instantiate a `VectorFunction` object defined as the product of `uInc` by the polarized vector $(1, -0.3, 0)$ divided by its norm as shown in Listing 5.11. Note that to write the Silver-Müller condition, the normal function must also be defined using the `normal<T_Scalar>()` function.

```
function::VectorFunction<Scalar> eInc =
  uInc * function::vector<Scalar>(1.,-0.3,0.) / std::sqrt(1.09);
function::VectorFunction<Scalar> n = function::normal<Scalar>();
```

Listing 5.11: Definition of a vector function used as boundary condition to model the elastodynamic scattering problem.

The electric field $\mathbf{e}$ is a 1-form field in $\mathbf{H}(\mathbf{curl})$, and is thus defined as a `Field<Scalar, form::Form1>` object, with associated function space `functionSpaceHCurl` (see Listing 5.12. The Dirichlet boundary condition is enforced using the same `addConstraint()` member function as in the acoustic and elastodynamic cases; note that the trace operators can be omitted as they are implicitly handled at the discrete level by the choice of the `HierarchicalHCurl` basis functions.

```
field::Field<Scalar, form::Form1>
  e("e", omega, functionSpaceHCurl::HierarchicalHCurl, 2);
// Dirichlet BC: gamma^T(e) = - gamma^T(e_inc) on gammaScat
e.addConstraint(gammaScat, -eInc);
```

Listing 5.12: The 1-form field defined to model the elastodynamic wave scattering problem.

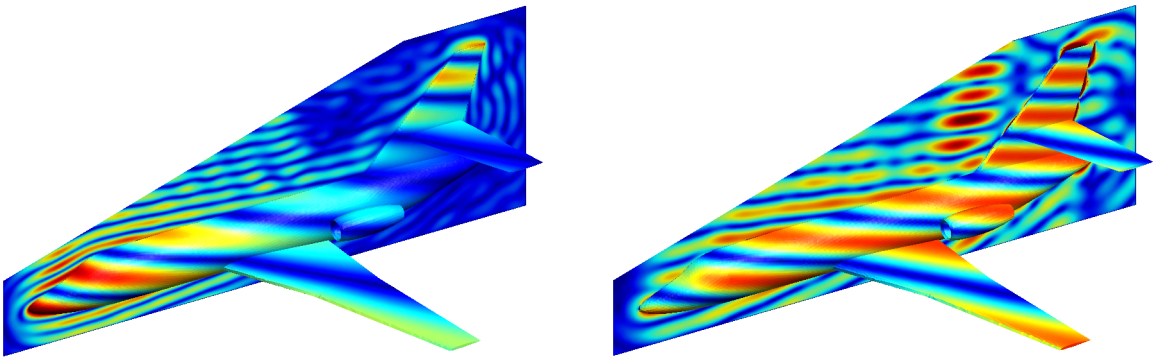
Listing 5.13 presents the formulation of the electromagnetic problem 5.16. It is again close to the formal mathematical expression; it should just be noted that the vector product is implemented by an overloading of the modulo operator `%`.

Finally, Listing 5.14 and Figure 5.7b present the same kind of post-processing as done in the elastodynamic example.

```
using equation::dof;
using equation::tf;
using equation::curl;
using namespace function;

problem::Formulation<Scalar> formulation("maxwell");

formulation.integral(curl(dof(e)), curl(tf(e)), omega, "Gauss8");
formulation.integral(- k*k * dof(e), tf(e), omega, "Gauss8");
formulation.integral(- im*k * n % (dof(e) % n), tf(e),
                                                gammaExt, "Gauss8");

formulation.pre();
formulation.assemble();
formulation.solve();
```

Listing 5.13: Definition of the formulation for the elastomagnetic wave scattering problem.

```
post::save(e, omega, "e");
post::save(e, cut, "e_cut", "pos");
```



(a) The real part of the norm of the displacement field computed by the elastodynamic example.

(b) The real part of the norm of the electric field computed by the electromagnetic example.

Figure 5.7: The solution of the electromagnetic and elastodynamic scattering problem depicted on the cut-plane.

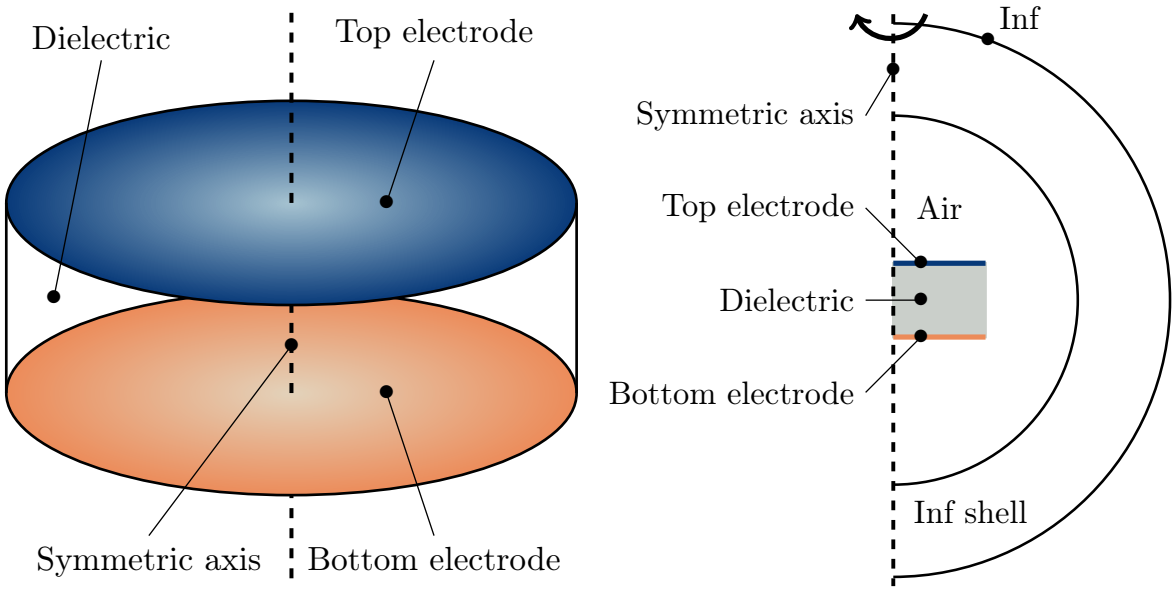
Listing 5.14: Some post-processing operations applied to the elastomagnetic wave scattering problem.

4.3 Infinite shell and axisymmetric transformations

In order to introduce other interesting features of GmshFEM, let us consider the simple electrostatic modeling of a planar capacitor, as shown in Figure 5.8a. The capacitor is made of two parallel disks separated by a dielectric material of relative permittivity $\varepsilon_r = 5$, surrounded by air. An electric potential of $+1$ and -1 Volt is enforced on the top and bottom electrodes.

A change of coordinates from the 3D Cartesian system $\hat{\mathbf{x}} : (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ to a 3D cylindrical system $\hat{\mathbf{x}}' : (\mathbf{e}_r, \mathbf{e}_\theta, \mathbf{e}_z)$, combined with the axisymmetry of the problem, allows to model the problem in the plane $\hat{\mathbf{x}}_{2D} : (\mathbf{e}_x, \mathbf{e}_y)$, with $\mathbf{e}_x = \mathbf{e}_r$ and $\mathbf{e}_y = \mathbf{e}_z$ (see Figure 5.8b) provided that the Jacobian is modified accordingly [110, 109]. Furthermore, an additional Jacobian modification can be considered to handle the homogeneous boundary condition on the electric field at infinity, through the introduction of an "infinite" shell transformation. To this end, the "Air" domain in Figure 5.8b is extended by a layer "Inf shell", that maps its exterior boundary to infinity [110, 116].

In GmshFEM, all of these Jacobian modifications are handled by the `Domain` class which can take as last optional argument an object inherited from the abstract class: `JacobiansModificator`. Three classes are currently implemented to describe infinite shell transformations (`PolarShell`, `CylindricalShell` and `SphericalShell`), the `Axisymmetry` class encodes the asymmetric transformation, and the `AxisymmetryShell` class combines the asymmetric and polar shell transformations. Listing 5.15 shows how the Jacobian transformations are defined for the capacitor example. The `AxisymmetryShell` takes as arguments the radius of the inner and outer shell boundary of the shell.



(a) The plane capacitor that is symmetric around the dashed axis.

(b) The axisymmetric model of the capacitor where the computational domain (*i.e.* the inner half-disk) is surrounded by an infinite shell.

Figure 5.8: The 3D capacitor and the corresponding 2.5D plane model.

Every time a function is called by taking one of these domains in argument, the associated Jacobian transformation is considered (if needed). In the same way, any FE terms or post-processing functions will be evaluated using the Jacobian transformations associated with the domain passed as argument.

```
domain::Axisymmetry<Scalar> axisymmetry;
domain::AxisymmetryShell<Scalar> axisymmetryShell(2., 2.5);

domain::Domain air("air", axisymmetry);
domain::Domain dielectric("dielectric", axisymmetry);
domain::Domain top("top", axisymmetry);
domain::Domain bottom("bottom", axisymmetry);
domain::Domain axis("axis", axisymmetry);
domain::Domain shell("inf shell", axisymmetryShell);
domain::Domain inf("inf", axisymmetryShell);

domain::Domain all = air | shell | dielectric;
```

Listing 5.15: Definition of the axisymmetric and infinite shell domains.

Note that this problem is also a good example to showcase the use of piecewise functions to declare the permittivity in Listing 5.16. First a `ScalarPiecewiseFunction<T_Scalar>` object is instantiated, then values are assigned depending on the domain of definition.

```
const double eps0 = 8.854187e-12; // [F / m]
const double epsr = 5;

function::ScalarPiecewiseFunction<Scalar> eps;
eps.addFunction(eps0, air | shell);
```

```
eps.addFunction(eps0 * epsr, dielectric);
```

Listing 5.16: The permittivity piecewise function.

4.4 Global quantities

In many applications one needs to impose non-local constraints, or to link non-local (e.g. integral) quantities through external circuit-type equations. So-called global quantities [74] allow for a natural handling of these non-local constraints and external circuit couplings. Global quantities are associated with global basis functions, introduced at the function space level, which are linear combinations of the classical elementary shape functions.

In the case of the capacitor example, a global electric voltage can be defined on each electrode, to which a global electric charge (its dual) is associated. A field named "v" is instantiated in Listing 5.17, with a homogeneous Dirichlet condition at infinity. Two `GlobalQuantity` objects are then instantiated (one by electrode), by taking a name and a domain of definition as argument. Finally, they are both associated with the field "v".

```
field::Field<Scalar, form::Form0> v("v", all,
                                    functionSpaceH1::Lagrange);
v.addConstraint(inf,  0.);

field::GlobalQuantity<Scalar> vTop("vTop", top);
field::GlobalQuantity<Scalar> vBottom("vBottom", bottom);
v.assignGlobalQuantity(vTop);
v.assignGlobalQuantity(vBottom);
```

Listing 5.17: The electric potential field and two global quantities use to impose the potential on each electrode.

The formulation of the capacitor problem, as shown in Listing 5.18 is written as usual (cf. Listing 5.16), with a homogeneous Neumann condition on the axis of symmetry. The two global quantities are integrated into it by fixing the primal values, *i.e.* the global electric potential. The dual values, *i.e.* the global electric charges on the electrodes, can be obtained once the system is solved.

```
problem::Formulation<Scalar> formulation("capacitor");

formulation.integral(eps * grad(dof(v)), grad(tf(v)), all,
                                                    "Gauss4");
formulation.integral(0., tf(v), axis, "Gauss4");

formulation.globalTerm(vTop, field::FixedComponent::Primal, -1.);
formulation.globalTerm(vBottom, field::FixedComponent::Primal, 1.);

formulation.pre(); formulation.assemble(); formulation.solve();

msg::info << "The charge on the inner electrode " <<
                    vTop.getDualValue() << "[C/rad]" << msg::endl;
msg::info << "The charge on the outer electrode " <<
                  vBottom.getDualValue() << "[C/rad]" << msg::endl;
```

Listing 5.18: The capacitor formulation.

5 Efficiency studies

In this section, the efficiency of the function tree evaluation is analyzed, based on the benchmarks introduced in Sections 3 and 5 of Chapter 4. In addition, a comparison with GetDP is presented.

5.1 Efficiency study of function tree evaluations

An important feature of GmshFEM is the function tree evaluation presented in Section 3.2. In order to test the efficiency of the function tree evaluation, the scattering benchmarks of Section 3, Chapter 4, are slightly modified by replacing the simple Sommerfeld radiation with a PML around the square domain. The PML parameters defined in (1.28) must be computed for each integration Gauss point inside the PML region, which leads to a realistic stress-test for the function tree evaluator.

The efficiency of the function tree is studied on the 2D benchmark at order 10, with a Gauss quadrature allowing to integrate exactly polynomials of order up to 20. The efficiency of the function tree evaluation is assessed by comparing it with the use of a custom node defined by inheritance (cf. Section 3.2), which hardcodes the same mathematical expression for the tensor PML parameters $\mathbb{D}$. This allows a comparison between the user-friendly function definition, the optimized node definition, and the maximum performance obtained by native code.

Listings 5.19 presents the definition of the tensor PML parameters $\mathbb{D}$ using the natural definition through the GmshFEM functions. Three parameters are supposed to be known: the domain size (`L`), the pml size (`pmlSize`), and the wavenumber (`k`). The parameters $\mathbb{D}$ is a `TensorPiecewiseFunction` such that it definition is specified inside each PML region surrounded the computational domain (*i.e.* the four edge PMLs and the four corner PMLs). The implementation is related to the mathematical definition of the PML parameter as presented in Chapter 1. First the distance inside the PMLs are declared (`distSigma*`), followed by the absorption functions (`sigma*`). Finally the stretching functions are defined (`k*`) and the dissipation tensor (`D`).

```
ScalarFunction< Scalar >
  distSigmaN, distSigmaW, distSigmaS, distSigmaE;
distSigmaN = y< Scalar >() - L/2.;
distSigmaW = -L/2. - x< Scalar >();
distSigmaS = -L/2. - y< Scalar >();
distSigmaE = x< Scalar >() - L/2.;

ScalarFunction< Scalar > sigmaN, sigmaW, sigmaS, sigmaE;
sigmaN = 1. / (pmlSize - distSigmaN) - 1./ pmlSize;
sigmaW = 1. / (pmlSize - distSigmaW) - 1./ pmlSize;
sigmaS = 1. / (pmlSize - distSigmaS) - 1./ pmlSize;
sigmaE = 1. / (pmlSize - distSigmaE) - 1./ pmlSize;

Scalar im(0., 1.);
```

```
ScalarFunction< Scalar > kN, kW, kS, kE;
kN = 1. + im * sigmaN / k;
kW = 1. + im * sigmaW / k;
kS = 1. + im * sigmaS / k;
kE = 1. + im * sigmaE / k;

TensorPiecewiseFunction< Scalar > D;
D.addFunction(tensorDiag< Scalar >(kN, 1./kN, kN), omegaN);
D.addFunction(tensorDiag< Scalar >(1./kW, kW, kW), omegaW);
D.addFunction(tensorDiag< Scalar >(kS, 1./kS, kS), omegaS);
D.addFunction(tensorDiag< Scalar >(1./kE, kE, kE), omegaE);
D.addFunction(tensorDiag< Scalar >(kN/kW, kW/kN, kN*kW), omegaNW);
D.addFunction(tensorDiag< Scalar >(kS/kW, kW/kS, kS*kW), omegaSW);
D.addFunction(tensorDiag< Scalar >(kS/kE, kE/kS, kS*kE), omegaSE);
D.addFunction(tensorDiag< Scalar >(kN/kE, kE/kN, kN*kE), omegaNE);
```

Listing 5.19: The tensor PML parameters $\mathbb{D}$ defined with the natural definition through the GmshFEM functions.

In comparison, Listing 5.20 shows how an optimized hard-coded version of the parameter $\mathbb{D}$ can be defined. All the computations are defined inside a single node, bypassing the cost of traveling through the execution tree. While some user-friendliness and the ability to wrap the higher level functions in a scripting language are lost, the code should still be understandable for users familiar with C++. As explained in Section 3.2, a new operation class `Pml` is inherited for the base class `NullaryOperation`. In this new class, the computation of $\mathbb{D}$ is done in the `operator()` function. In addition a function called `pml()` that returns a tensor function built over our new class `Pml` is defined. Once defined, the tensor piece-wise function object can be filled by calling our function `pml()` with the appropriate parameters.

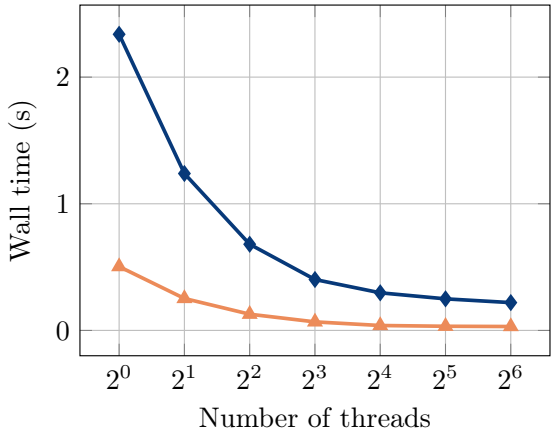
```
template< class T_Scalar >
class Pml :
  public NullaryOperation< T_Scalar, Degree::Degree2 >
{
 private:
  int _region;
  scalar::Precision< T_Scalar > _pmlSize;
  scalar::Precision< T_Scalar > _k;
  scalar::Precision< T_Scalar > _l;

 public:
  Pml(const std::string &region,
      const scalar::Precision< T_Scalar > &pmlSize,
      const scalar::Precision< T_Scalar > &k,
      const scalar::Precision< T_Scalar > &l) :
    _pmlSize(pmlSize), _k(k), _l(l)
  {
    if(region == "N")       _region = 0b0001;
    else if(region == "W")  _region = 0b0010;
    else if(region == "S")  _region = 0b0100;
    else if(region == "E")  _region = 0b1000;
    else if(region == "NW") _region = 0b0011;
    else if(region == "SW") _region = 0b0110;
    else if(region == "SE") _region = 0b1100;
    else if(region == "NE") _region = 0b1001;
```

```
  }

  Pml(const Pml &other) :
    NullaryOperation< T_Scalar, Degree::Degree2 >(other),
    _region(other._region), _pmlSize(other._pmlSize),
    _k(other._k), _l(other._l)
  {
  }

  void operator()(
    OutputVector< T_Scalar, Degree::Degree2 > &values,
    const std::vector< scalar::Precision< T_Scalar >,
    numa::allocator<scalar::Precision<T_Scalar>>> &points,
    const std::vector<scalar::Precision<T_Scalar>> &gaussPoints,
    const int elementType,
    const std::pair< int, int > &entity) const
  {
    T_Scalar im(0., 1.);
#pragma omp for
    for(auto i = 0ULL; i < values.size(); ++i) {
      T_Scalar kN = 0., kW = 0., kS = 0., kE = 0.;

      if(_region & 0b0001) {
        scalar::Precision< T_Scalar > distSigmaN =
          points[3*i + 1] - _l/2.;
        scalar::Precision< T_Scalar > sigmaN =
          1. / (_pmlSize - distSigmaN) - 1./ _pmlSize;
        kN = 1. + im * sigmaN / _k;
      }
      if(_region & 0b0010) {
        scalar::Precision< T_Scalar > distSigmaW =
          -_l/2. - points[3*i + 0];
        scalar::Precision< T_Scalar > sigmaW =
          1. / (_pmlSize - distSigmaW) - 1./ _pmlSize;
        kW = 1. + im * sigmaW / _k;
      }
      if(_region & 0b0100) {
        scalar::Precision< T_Scalar > distSigmaS =
          -_l/2. - points[3*i + 0];
        scalar::Precision< T_Scalar > sigmaS =
          1. / (_pmlSize - distSigmaS) - 1./ _pmlSize;
        kS = 1. + im * sigmaS / _k;
      }
      if(_region & 0b1000) {
        scalar::Precision< T_Scalar > distSigmaE =
          points[3*i + 0] - _l/2.;
        scalar::Precision< T_Scalar > sigmaE =
          1. / (_pmlSize - distSigmaE) - 1./ _pmlSize;
        kE = 1. + im * sigmaE / _k;
      }

      if(_region & 0b0001)
        values[i] << kN,0.,0., 0.,1./kN, 0.,0.,0.,kN;
      else if(_region & 0b0010)
        values[i] << 1./kW,0.,0., 0.,kW,0., 0.,0.,kW;
      else if(_region & 0b0100)
```

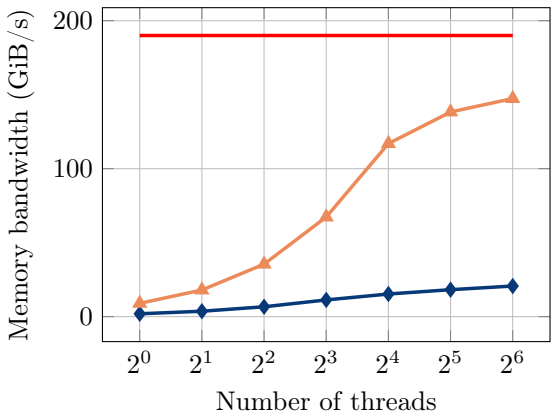
```
        values[i] << kS,0.,0., 0.,1./kS,0., 0.,0.,kS;
      else if(_region & 0b1000)
        values[i] << 1./kE,0.,0., 0.,kE,0., 0.,0.,kE;

      else if(_region & 0b0011)
        values[i] << kN/kW,0.,0., 0.,kW/kN,0., 0.,0.,kN*kW;
      else if(_region & 0b0110)
        values[i] << kS/kW,0.,0., 0.,kW/kS,0., 0.,0.,kS*kW;
      else if(_region & 0b1100)
        values[i] << kS/kE,0.,0., 0.,kE/kS,0., 0.,0.,kS*kE;
      else if(_region & 0b1001)
        values[i] << kN/kE,0.,0., 0.,kE/kN,0., 0.,0.,kN*kE;
    }
  }

  bool isConstant() const override { return false; }

  std::string name() const override { return "pml"; }

  bool operator==(
    const NullaryOperation< T_Scalar, Degree::Degree2 > &other)
    const override { return false; }
};

template< class T_Scalar >
Function< T_Scalar, Degree::Degree2 > pml(
  const std::string &region,
  const scalar::Precision< T_Scalar > &pmlSize,
  const scalar::Precision< T_Scalar > &k,
  const scalar::Precision< T_Scalar > &l)
{
  return Function< T_Scalar, Degree::Degree2 >(
    new NullaryNode< Pml< T_Scalar > >(
      Pml< T_Scalar >(region, pmlSize, k, l)
    )
  );
}

TensorPiecewiseFunction< Scalar > D;
D.addFunction(pml< Scalar >("N", pmlSize, k, l), omegaN);
D.addFunction(pml< Scalar >("W", pmlSize, k, l), omegaW);
D.addFunction(pml< Scalar >("S", pmlSize, k, l), omegaS);
D.addFunction(pml< Scalar >("E", pmlSize, k, l), omegaE);
D.addFunction(pml< Scalar >("NW", pmlSize, k, l), omegaNW);
D.addFunction(pml< Scalar >("SW", pmlSize, k, l), omegaSW);
D.addFunction(pml< Scalar >("SE", pmlSize, k, l), omegaSE);
D.addFunction(pml< Scalar >("NE", pmlSize, k, l), omegaNE);
```

Listing 5.20: The tensor PML parameters $\mathbb{D}$ hard-coded to gain the maximum of performanc .

Figure 5.9 shows both the evaluation wall time (Figure 5.9a) and the measured memory bandwidth (Figure 5.9b). First, the evaluation time with the natural definition through the GmshFEM functions is about five times slower than the optimized version. This decrease in performance is expected and, in our opinion, utterly accept-



(a) Evaluation wall time.



(b) Measured memory bandwidth. The red curve corresponds to the maximal theoretical bandwidth of NIC5.

Figure 5.9: Comparison of PML parameters $\mathbb{D}$ using the natural definition through the GmshFEM functions (blue) or by defining a custom node by inheritance (orange).

able given the user-friendliness and flexibility offered by the GmshFEM functions. In addition, compared to the total assembly time of Figure 4.2c in Chapter 4 the function evaluations are really negligible. Thus for most applications, the natural definition will not be a bottleneck in the overall performance of the resolution.

The hard-coded function possibility offers an elegant and well-integrated way to get closer to the maximum performance for specific applications that are performance-critical. Indeed, Figure 5.9b shows the measured memory bandwidth of both approaches. One can notice that the hard-coded version allows reaching a memory bandwidth close to the theoretical one of NIC5 (about 77%).

5.2 Comparison with GetDP

Finally, we compare the performance of GmshFEM with the performance of GetDP [70] by measuring both codes' total pre-processing and assembly time. The considered problem is the 2D scattering problem with the PML around the domain. GmshFEM and GetDP do not have the same definition of hierarchical basis functions, so the comparison is made using first-order iso-parametric basis functions, leading to 532,031 unknowns. An exact Gauss quadrature to integrate a fourth-order polynomial is used in both codes, which are both compiled on the NIC5 cluster with optimization flags switched on.

The results are reported on Figure 5.10. The single-thread comparison shows that GmshFEM takes a bit less than half of the GetDP time to build the finite element matrix of our problem. Furthermore, as the assembly algorithm in GetDP is not a multithreaded, the performance difference increases as expected as the number of threads is increased. The sum of the pre-processing assembly wall time can be reduced by 87% compared to GetDP when the 64 cores of NIC5 are used.

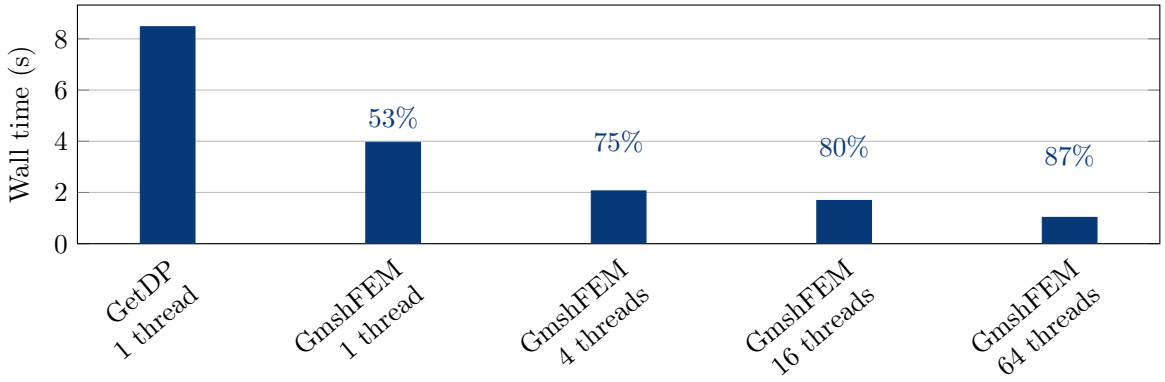


Figure 5.10: Comparison with GetDP

6 Conclusion and perspectives

In this chapter, we introduced GmshFEM, an open-source C++ finite element library based on the Gmsh API. After an introduction to the basic philosophy of the library with an acoustic example, we briefly presented several more advanced features that distinguish GmshFEM from other FEM libraries. The analysis of the function tree evaluation algorithm showed that its efficiency is of the same order of magnitude as that of an optimized version of the function. Furthermore, we showed that the opportunity to implement a user-defined node in the execution tree, that replaces most of the function tree structure, leads to a performance close to the maximal theoretical interpretation of the hardware - at the cost of course the user-friendliness and interactivity that the evaluation tree provides. A brief comparison with GetDP allowed to highlight the substantial performance gains on the high-order finite element problems treated in this thesis.

The next chapter will present the GmshDDM domain decomposition library, based on GmshFEM, which was used to solve all the numerical examples presented in Chapters 2, 3 and 4.

6 GmshDDM, the domain decomposition library

GmshDDM is an open-source C++ domain decomposition library based on GmshFEM. Considered as an extension of GmshFEM, they both share the same design philosophy: to be fast, light, and user-friendly with problem implementations close to their mathematical definitions. Combining the efficient multi-threaded parallelization and SIMD vectorization of GmshFEM with a distributed memory parallelization of the domain decomposition algorithm implemented in GmshDDM offers a suitable environment for solving large-scale time-harmonic wave problems.

1 Introduction

In this chapter we introduce GmshDDM (`https://gitlab.onelab.info/gmsh/ddm.git`), an open-source C++ domain decomposition library that is based on the GmshFEM library presented in Chapter 5. GmshDDM is a small add-on to GmshFEM (about 2000 lines of C++ code) that offers features to implement optimized Schwarz domain decomposition solvers. It shares with GmshFEM the same design philosophy and functionalities: in particular, it manages both straight-sided or curved meshes in 1D, 2D, or 3D on all element shapes supported by GmshFEM and supports all GmshFEM interpolations. While GetDP inspired GmshFEM, GmshDDM is inspired by GetDDM [182]: it relies on a high-level symbolic definition of the DDM formulation and hides most technical aspects of the DDM algorithm, which allows users to focus on the mathematical definition of their problems.

GmshDDM supports a two-level parallelization strategy based on both a distributed- and a shared-memory architecture to scale to large problems. While the subdomain computations, *i.e.* the pre-processing, the assembling, and the solving of each subproblem are parallelized using OpenMP through GmshFEM, GmshDDM im-

plements a distributed-memory parallelization version of the DDM algorithm using the *Message Passing Interface* (MPI) protocol. The parallel iterative solver relies on the external solver PETSc [18].

The chapter will start in Section 2 by describing how the DDM version of the acoustic scattering problem presented in Chapter 5 is implemented using the GmshDDM framework. We then present in Section 3 the implementation of the DDM algorithm as introduce in Chapter 1, Section 6. As a parallel of Section 2, the DDM version of the electromagnetic and elastodynamic examples of Chapter 5, Section 4 is presented in Section 4.

2 Symbolic definition of the problem

To illustrate how a finite element domain decomposition problem is set up in GmshDDM, let us consider the partitioned version into N subdomains of the acoustic scattering problem (5.1). To keep things simple, a simple 0^{th}-order transmission condition is enforced on the interfaces betweeen sudomains such that (5.1) is expressed as

$$\begin{cases} \Delta\, u_n + k^2 u_n = 0 & \text{in } \Omega_n, \\ u_n = -u_{\text{inc}} & \text{on } \Gamma_{n,\text{scat}}, \\ \partial_{\mathbf{n}} u_n - \imath\, k u_n = g_{n,m} & \text{on } \Gamma_{n,\text{ext}}, \end{cases} \tag{6.1}$$

where the domains and notations are defined in Chapter 1, Section 6 and $u_{\text{inc}} = \exp(\imath\, \mathbf{k}\cdot\mathbf{x})$ as before. Furthermore, the interface fields are defined as

$$g_{m,n} := \begin{cases} 0 & \text{on } \Gamma_{n,\text{ext}} \cap \partial\Omega_{\text{tot}} \\ -\, g_{n,m} - 2\,\imath\, k u_n & \text{on each } \Sigma_{n,m}. \end{cases} \tag{6.2}$$

In a nutshell, the interface fields vanish on the subdomain boundaries that coincide with the boundary of the global problem.

As in GmshFEM, the problem definition in GmshDDM is based on a variational formulations of both (6.1) and (6.2). For each subdomain Ω_n, the variational formulation of (6.1) reads: Find $u_n \in H^1(\Omega_n)$ such that

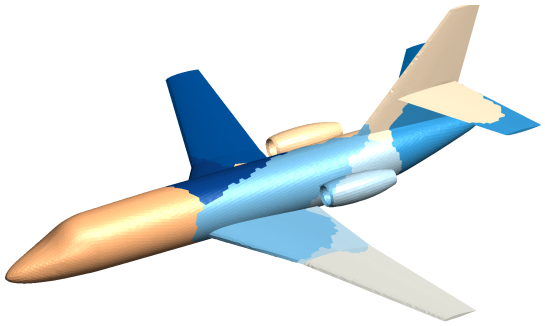
$$\int_{\Omega_n} \mathbf{grad}\, u_n \cdot \mathbf{grad}\, \overline{v}_n - k^2 u_n \overline{v}_n \,\mathsf{d}\Omega_n - \int_{\Gamma_{n,\text{ext}}} \imath\, k u_n\, \overline{v}_n + \sum_{m\in\mathfrak{N}_n} g_{n,m}\, \overline{v}_n \,\mathsf{d}\Gamma_{n,\text{ext}} = 0, \tag{6.3}$$

holds for every test function $v \in H^1(\Omega_n)$. In addition, for each interface $\Sigma_{n,m}$, the variational formulation of (6.2) reads: Find $g_{m,n} \in H^1(\Sigma_{n,m})$ such that

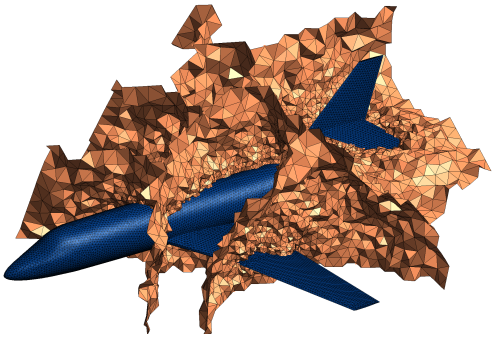
$$\int_{\Sigma_{n,m}} g_{m,n}\, \overline{h}_{m,n} + g_{n,m}\, \overline{h}_{m,n} + 2\,\imath\, k u_n\, \overline{h}_{m,n} \,\mathsf{d}\Sigma_{n,m} = 0, \tag{6.4}$$

holds for every test function $h_{m,n} \in H^1(\Sigma_{n,m})$. We will detail in the following subsections how the variational formulations (6.3) and (6.4) are transcription in the GmshDDM library.

Following the namespace convention of GmshFEM, all classes and functions related to GmshDDM live in a `gmshddm` namespace. To avoid confusions, using-directives



(a) The computational domain is partitioned into ten subdomains.



(b) The interfaces created by the partitioner of Gmsh. An interface (in orange) is created between subdomains.

Figure 6.1: The partition of the Falcon plane returned by the mesh partitoner of Gmsh.

"`using namespace gmshfem;`" or "`using namespace gmshddm;`" are not used in what follows so that functionalities of GmshFEM and GmshDDM can be clearly differentiated.

In order to setup a distributed solver in GmshDDM, the mesh of Figure 5.1b has first to be partitioned using Gmsh mesh partitioner. Figure 6.1 shows such a partitioning in ten subdomains, obtained through Gmsh using the third-part library Metis [120]. It is important to note that when partitioning the mesh, Gmsh rebuilds the full boundary representation (B-Rep) of the partitioned geometry, so that each subdomain is defined by a closed set of surfaces; each surface is defined by a closed set of curves; and each curve is defined by defined by two points. All of these new (discrete) entities are either a part of an existing geometrical entity such as the boundary of the plane (see Figure 6.1a) or are an interface between two subdmains (see Figure 6.1b). An entity that already exists before the mesh partition is called a parent entity as, after the partition process, it is cut into child entities that inherit from the parent entity. For instance, the physical entity associated with the plane's boundary is transferred to the child entities.

Finally, contrary to GetDDM, GmshDDM is based on the assumption that the meshes match at the interfaces (*i.e.* with the same entities, same elements, and same nodes, with the same tags). In other words, at least the interface skeleton (*i.e.* the set of all interfaces) must be meshed globally, but the interior of each subdomain can be meshed independently.

2.1 Geometric objects

The geometric objects needed to describe Problem (6.1) can be sorted into two subsets: domains related to individual subdomains and domains related to the interfaces between subdomains. While the first ones are tagged by the subdomain index n, the second ones are tagged by a pair of indices n and m corresponding to an interface. This is why GmshDDM introduces two new classes: a `Subdomain` class to model the first kind of geometric objects and `Interface` class to model the second kind.

The `Subdomain` class internally stores an array of GmshFEM `Domain` objects while

the `Interface` class contains an array of sets of GmshFEM `Domain` objects. To access these GmshFEM objects, the `Subdomain` class overrides the call operator with one unsigned integer argument `n` and returns the `Domain` object associated to the n^{th} subdomain. In a similar way, the `Interface` class overrides the call operator with two unsigned integer arguments `n` and `m` and returns the `Domain` object associated to the interface between the n^{th} and the m^{th} subdomain. Moreover, all operations available on `Domain` objects are also available on `Subdomain` or `Interface` objects.

Listing 6.1 presents how `Subdomain` objects and `Interface` objects are instantiated using the `buildSubdomainOf()` and the `buildInterface()` functions. The `buildSubdomainOf()` simply takes a GmshFEM `Domain` object associated to the parent entities and returns the corresponding `Subdomain` object. The `buildInterface()` automatically returns the `Interface` object of the current model. It also returns the topology as a vector of vectors, such as a subdomain `n` has all subdomains of `topology[n]` as neighbors. Note that we define the domain $\Gamma_{n,\text{bnd}} = \Gamma_{n,\text{ext}} \cap \partial\Omega_{\text{tot}}$ as the part of the boundary of Ω_n that belongs to the exterior boundary of the global problem.

While this method is the easiest way to instantiate geometric objects, the topology can also be specified manually. Both `Subdomain` and `Interface` objects can be instantiated by taking as argument a number of subdomains. Then using the overridden call operator, the domain of each subdomain or interface can be specified manually. For instance let us assume that we have a mesh partitioned into N partitions such that a physical entity named "`omega_n`" where `n` $= 0, 1, \cdots, N-1$ is the tag of the subdomain, then an subdomain object "`omega`" can manually created as Listing 6.2 shows.

```
gmshddm::domain::Subdomain omega =
gmshddm::domain::Subdomain::buildSubdomainOf(
                                    gmshfem::domain::Domain("omega"));
gmshddm::domain::Subdomain gammaScat =
gmshddm::domain::Subdomain::buildSubdomainOf(
                                  gmshfem::domain::Domain("gammaScat"));
gmshddm::domain::Subdomain gammaBnd =
gmshddm::domain::Subdomain::buildSubdomainOf(
                                   gmshfem::domain::Domain("gammaExt"));

std::vector< std::vector< unsigned int > > topology;
gmshddm::domain::Interface sigma =
gmshddm::domain::Interface::buildInterface(topology);
```

Listing 6.1: Geometric objects needed to model the DDM version of the scattering problem example.

```
gmshddm::domain::Subdomain omega(100);
for(int n = 0; n < 100; ++n) {
  omega(n) = gmshfem::domain::Domain("omega_"+std::to_string(n));
}
```

Listing 6.2: A subdomain object manually instantiated.

2.2 Subdomain and interface field objects

In a similar fashion as geometric objects, field objects can be sorted in two subsets: the `SubdomainField<T_Scalar, T_Form>` objects and the `InterfaceField<T_Scalar, T_Form>` objects. They follow the same logic; while the subdomain fields are tagged by only one index n and store a vector of GmshFEM fields of type `Field<T_Scalar, T_Form>`, the interface fields are tagged by a pair of indices n and m and store a vector of set of GmshFEM fields of type `Field<T_Scalar, T_Form>`. The internal GmshFEM fields can also be accessed using the call operator of the `SubdomainField` or `InterfaceField` class.

A `SubdomainField` or an `InterfaceField` object is instantiated using a name, a `Subdomain` or an `Interface` object, respectively, and a basis function definition as presented in Section 2.3.

To model our acoustic scattering example, two fields have to be defined in Listing 6.3: the subdomain field u of (6.3) and the interface field g of (6.4).

```
gmshddm::field::SubdomainField<Scalar, gmshfem::form::Form0>
        u("u", omega, gmshfem::functionSpaceH1::HierarchicalH1, 3);
gmshddm::field::InterfaceField<Scalar, gmshfem::form::Form0>
        g("g", sigma, gmshfem::functionSpaceH1::HierarchicalH1, 3);
```

Listing 6.3: The subdomain field and the interface field needed to model our acoustic scattering problem in GmshDDM.

2.3 Formulation objects

GmshDDM has its own `Formulation<T_Scalar>` class. Once again, a `Formulation` is built around a vector of GmshFEM `Formulation` objects to store the subdomain formulations, and a vector of set of GmshFEM `Formulation` objects to store the interface formulations. They are both accessible with the appropriate call operators.

Listing 6.4 shows how our acoustic scattering problem can be solved using GmshDDM. Once the GmshDDM formulation is instantiated with a name and the topology vector, a call to the member function `addInterfaceField()` is used to specify which interface field must be considered as an unknown of the interface problem (1.68) defined in Chapter 1. Then, the formulation of the DDM problems can be written by considering a loop over the number of subdomains and a loop over the interfaces of each subdomain. Inside these loops, the subdomain and interface formulations expressions are defined using the tools presented in Chapter 5. At this level, physical sources and artificial sources that may appear in the variational formulation must be tagged with the member function `physicalSource()` and `artificialSource()`. For computational costly functions, `physicalSourceTerm()` and `artificialSourceTerm()` can be alternatively used to tag an `integral()` term as physical source or artificial source. They both take in argument the unique tag returned by each `integral()` member function. Inside the loop over the subdomains, Dirichlet conditions are enforced on every subdomain field using the `addConstraint()` member function introduced in Section 2.3.

```
using gmshfem::equation::dof;
using gmshfem::equation::tf;
```

```
using gmshfem::equation::grad;
using namespace gmshfem::function;

gmshddm::problem::Formulation<Scalar>
                              formulation("helmholtzDDM", topology);
formulation.addInterfaceField(g);

for(unsigned int n = 0; n < 10; ++n) {
  u(n).addConstraint(gammaScat(n), -uInc);

  formulation(n).integral(grad(dof(u(n))), grad(tf(u(n))),
                                                omega(n), "Gauss8");
  formulation(n).integral(- k*k * dof(u(n)), tf(u(n)),
                                                omega(n), "Gauss8");
  formulation(n).integral(- im*k * dof(u(n)), tf(u(n)),
                                               gammaBnd(n), "Gauss8");

  for(unsigned int m : topology[n]) {
    formulation(n).integral(- im*k * dof(u(n)), tf(u(n)),
                                                 sigma(n,m), "Gauss8");
    formulation(n).integral(-formulation.artificialSource(g(n,m)),
                                    tf(u(n)), sigma(n,m), "Gauss8");

    formulation(n,m).integral(dof(g(m,n)), tf(g(m,n)),
                                                 sigma(n,m), "Gauss8");
    formulation(n,m).integral(formulation.artificialSource(g(n,m)),
                                  tf(g(m,n)), sigma(n,m), "Gauss8");
    formulation(n,m).integral(2.*im*k * u(n), tf(g(m,n)),
                                                 sigma(n,m), "Gauss8");
  }
}

formulation.pre();
formulation.solve("gmres");
```

Listing 6.4: The formulation of our acoustic scattering problem in GmshDDM.

Finally, the DDM formulation is pre-processed using the member function `pre()`. The pre-processing initializes the interface problem (*i.e.* Equation (1.68) of Chapter 1) by computing the right-hand side. In a distributed-memory run, the pre-processing also initializes the MPI communication and the distribution of unknowns between processes. The solving step is called with the member function `solve()`, that takes an iterative solver name as argument. In addition, optional arguments corresponding to the relative tolerance (default value set to 10^{-6}), the maximum number of iterations (default value set to 1000) can be passed to the `solve()` member function, and a flag, initially deactivated, that indicates if the subdomain matrices have the same pattern, regardless of whether physical sources and artificial sources are activated or not.

Once the iterative solver has converged, all post-processing functionalities presented in Section 2.5 of Chapter 5 can be applied to any GmshFEM field returned by the appropriated call operator on both `SubdomainField` or `InterfaceField` objects.

3 The C++ implementation

The C++ implementation of the DDM algorithm presented in Section 6 follows the pseudo-code of Figure 1.3. It is organized in two phases: the pre-processing and the solving phases. The pre-processing phase, corresponding to the first step of the pseudo-code of Figure 1.3, initializes the right-hand side of the interface problem, `b`. When MPI is used, the pre-processing also initializes the data structure needed to communicate interface field DoFs between the processes. The solving phase, corresponding to the second and third steps of the pseudo-code of Figure 1.3, finally computes the solution of the interface problem (1.68) using an iterative solver such as a GMRES [169], and once converged, evaluates the final solution.

In each of the previous steps, the physical and artificial sources must be activated or deactivated depending on the algorithm step. From step one to step three of Figure 1.3, the physical sources must be successively turned on, then turned off, and finally turned on again, while the artificial sources must be switched off on step one, then switched on for the last two steps. This is why it is essential to tag both physical and artificial sources in the GmshDDM formulation.

The following subsections detail these two phases when subdomains are spread over different processes. We focus on the work done by only one process, and we denote data assigned to this process as *local* data (*e.g.* local fields, local subdomain formulations, ...).

3.1 The pre-processing phase

First of all, the pre-processing function activates the physical sources and deactivates the artificial sources before calling the GmshFEM `pre()` function of each local subdomain formulation and each local interface formulation to build the dictionary of DoFs and the pattern of the finite element matrix. Let us suppose that subdomain n is assigned to our process: then the pre-processing of all interface fields $g_{n,m}$ for all m in $\mathfrak{N}_n$ is carried out at this stage. At the same time, each local associated interface field $g_{m,n}$ is instantiated but not pre-processed.

These associated interface fields are then filled in by the DoF dictionary generated by the other processes. Therefore our process has to communicate DoF dictionaries of its local interface fields and receive DoF dictionaries from other processes to assign them to its local associated interface fields. These communications exchange, through MPI, vectors of elements of type `GmshFEM::RawDofKey` is a light structure representing the internal `GmshFEM::Dof` class. Exchanging these vectors is enough if the mesh on the interface is identical on both subdomains and if, for DoFs associated with entities other than nodes (*i.e.* edges and faces), the edge- and face tags correspond on both subdomains. Since generating the edge- and face tags beforehand can be cumbersome, they are also exchanged automatically to build an interface mapping structure.

Once every local field is pre-processed, local subdomain- and interface formulations are assembled and solved, possibly taking advantage of the multi-threading functionalities of GmshFEM.

3.2 The solving phase

The solving phase starts by switching on the artificial sources and turning off the physical sources. Then, depending on the last optional argument of the `solve()` member function that indicates if the subdomain matrices have the same pattern as explained before, each local subdomain system is regenerated and factorized, or only the right-hand side is regenerated such that the factorization computed in the pre-processing phase is reused.

The iterative solving step is carried on by either PETSc [18] iterative solvers such as GMRES [169] or BiCGSTAB [187], or a built-in Jacobi solver using the PETSc vector and matrix objects, depending on the solver selected by the first parameter of `solve()`. These iterative methods are matrix-free, in the sense that they only need to be able to compute the application of the matrix on a vector. The *shell matrix* feature of PETSc is used to instantiate a matrix-free object where a built-in function detailed hereafter implements the matrix-vector product.

The matrix-vector product function applies the solving steps presented in Figure 1.3. The function takes as argument a PETSc vector `X` and returns another PETSc vector `Y` corresponding to the product of the iteration matrix $\mathcal{A}$ with `X`. In our application, the input vector `X` corresponds to the artificial source vector `g` at the current iterative step of the solver, and we thus chose to ignore the input vector and work directly with the interface fields that contain precisely the same information. For each call to the matrix-vector product function, DoF values of local interface fields are exchanged to the neighbor processes, and local associated interface fields are filled in by the DoF values received. Then for each local subdomain formulation and local interface formulation, local right-hand sides are assembled, and local systems are solved by reusing matrix factorizations already computed in the previous step. Finally, a local artificial source vector $\mathbf{g}_n$ is built, collecting all DoF values of local interface fields and passed to PETSc that assembles the output vector `Y` according to its own parallelization strategy.

Once the solver has converged, both physical- and artificial sources are activated to compute the final solution.

4 Extension to other wave problems

To make a parallel with the elastodynamic and electromagnetic wave problems presented in Section 4 of Chapter 5, let us briefly introduce their DDM version in this section.

4.1 Elastodynamic waves

Let us consider the elastodynamic problem (5.9) posed on the same partitioned domain with a 0^{th} order transmission conditions enforced on the interfaces:

$$\begin{cases} \dfrac{1}{k_P^2}\,\mathbf{grad}\,\mathrm{div}\,\mathbf{u}_n - \dfrac{1}{k_S^2}\,\mathbf{curl}\,\mathbf{curl}\,\mathbf{u}_n - \mathbf{u}_n = \mathbf{0} & \text{in } \Omega_n, \\ \mathbf{u}_n = -\mathbf{u}_{\text{inc}} & \text{on } \Gamma_{n,\text{scat}}, \\ \Gamma^t(\mathbf{u}_n) - \dfrac{\iota}{k_P}\mathbb{I}_n\mathbf{u}_n - \dfrac{\iota}{k_S}\mathbb{I}_\tau\mathbf{u}_n = \mathbf{g}_{n,m} & \text{on } \Gamma_{n,\text{ext}}, \end{cases} \tag{6.5}$$

where Γ^t is the trace operator (5.10). Furthermore, the interface fields are now defined as

$$\mathbf{g}_{m,n} := \begin{cases} \mathbf{0} & \text{on } \Gamma_{n,\text{ext}} \cap \partial\Omega_{\text{tot}} \\ -\mathbf{g}_{n,m} - \dfrac{\iota}{k_P}\mathbb{I}_n\mathbf{u}_n - \dfrac{\iota}{k_S}\mathbb{I}_\tau\mathbf{u}_n & \text{on each } \Sigma_{n,m}. \end{cases} \tag{6.6}$$

The domain definition of Listing 6.1 and the function definition of Listing 5.7 are used without modification. The subdomain and interface fields become respectively a `SubdomainCompoundField` and a `InterfaceCompoundField` object, following the idea of Section 4.1 of Chapter 5. Finally in Listing 6.5 the formulation is adapted according to the Navier problem (6.5) and its interface problem (6.6).

```
gmshddm::field::SubdomainCompoundField
  <Scalar, gmshfem::form::Form0, 3>
    u("u", omega, gmshfem::functionSpaceH1::HierarchicalH1, 3);
gmshddm::field::InterfaceCompoundField
  <Scalar, gmshfem::form::Form0, 3>
    g("g", sigma, gmshfem::functionSpaceH1::HierarchicalH1, 3);

gmshddm::problem::Formulation<Scalar>
                                  formulation("navierDDM", topology);
formulation.addInterfaceField(g);

for(unsigned int n = 0; n < 10; ++n) {
  u(n).addConstraint(gammaScat(n), -uIncVec);

  formulation(n).integral(C * grad(dof(u(n))), grad(tf(u(n))),
                                                  omega(n), "Gauss8");
  formulation(n).integral(- dof(u(n)), tf(u(n)),
                                                  omega(n), "Gauss8");
  formulation(n).integral(- im/kP * In * dof(u(n)),
                                  tf(u(n)), gammaBnd(n), "Gauss8");
  formulation(n).integral(- im/kS * It * dof(u(n)), tf(u(n)),
                                              gammaBnd(n), "Gauss8");

  for(unsigned int m : topology[n]) {
    formulation(n).integral(- im/kP * In * dof(u(n)),
                                    tf(u(n)), sigma(n,m), "Gauss8");
    formulation(n).integral(- im/kS * It * dof(u(n)), tf(u(n)),
                                      sigma(n,m), "Gauss8");
    formulation(n).integral(- formulation.artificialSource(g(n,m)),
                                    tf(u(n)), sigma(n,m), "Gauss8");

    formulation(n,m).integral(dof(g(m,n)), tf(g(m,n)),
                                      sigma(n,m), "Gauss8");
    formulation(n,m).integral(formulation.artificialSource(g(n,m)),
                                  tf(g(m,n)), sigma(n,m), "Gauss8");
    formulation(n,m).integral(2.*im*(lambda+2.*mu)*kP*In * u(n),
                                  tf(g(m,n)), sigma(n,m), "Gauss8");
    formulation(n,m).integral(2.*im*mu*kS*It * u(n), tf(g(m,n)),
                                      sigma(n,m), "Gauss8");
  }
}

formulation.pre(); formulation.solve("gmres");
```

Listing 6.5: The field definition and the formulation of our elastodynamic scattering problem in GmshDDM.

4.2 Electromagnetic waves

The electromagnetic problem (5.14) posed on a partitioned domain made of N subdomains with a 0^{th}-order transmission condition enforced on the interfaces is expressed as

$$\begin{cases} \mathbf{curl}\,\mathbf{curl}\,\mathbf{e}_n - k^2\mathbf{e}_n = \mathbf{0} & \text{in } \Omega_n, \\ \gamma^T(\mathbf{e}_n) = -\gamma^T(\mathbf{e}_{\text{inc}}) & \text{on } \Gamma_{n,\text{scat}}, \\ \gamma^t(\mathbf{curl}\,\mathbf{e}_n) + \iota\, k\gamma^T(\mathbf{e}_n) = \mathbf{g}_{n,m} & \text{on } \Gamma_{n,\text{ext}}, \end{cases} \tag{6.7}$$

where γ^t and γ^T are the trace operators (5.15) and $\mathbf{g}$ is now a vector interface field. Furthermore, the interface fields are defined as

$$\mathbf{g}_{m,n} := \begin{cases} \mathbf{0} & \text{on } \Gamma_{n,\text{ext}} \cap \partial\Omega_{\text{tot}} \\ -\mathbf{g}_{n,m} + 2\,\iota\, k\gamma^T(\mathbf{e}_n) & \text{on each } \Sigma_{n,m}. \end{cases} \tag{6.8}$$

Again, the domain definitions of Listing 6.1 and the function definitions of Listing 5.11 can be kept as is. The subdomain and interface fields must be adapted in a similar way as done in Listing 5.12. The subdomain field `u` becomes a 1-form subdomain field `e`, and the same modification is applied to the interface field `g`. The formulation is then adapted (see Listing 6.6) to correspond to the Maxwell problem (6.7) with its interface problem (6.8).

```
gmshddm::field::SubdomainField<Scalar, gmshfem::form::Form1> e("e",
  omega, gmshfem::functionSpaceHCurl::HierarchicalHCurl, 1);
gmshddm::field::InterfaceField<Scalar, gmshfem::form::Form1> g("g",
  sigma, gmshfem::functionSpaceHCurl::HierarchicalHCurl, 1);

gmshddm::problem::Formulation<Scalar>
                                formulation("maxwellDDM", topology);
formulation.addInterfaceField(g);

for(unsigned int n = 0; n < 10; ++n) {
  e(n).addConstraint(gammaScat(n), -eInc);

  formulation(n).integral(curl(dof(e(n))), curl(tf(e(n))),
                                                  omega(n), "Gauss8");
  formulation(n).integral(- k*k * dof(e(n)), tf(e(n)),
                                                  omega(n), "Gauss8");
  // n_ is the normal (to avoid conflict with the integer n)
  formulation(n).integral(- im*k * n_ % (dof(e(n)) % n_), tf(e(n)),
                                              gammaBnd(n), "Gauss8");

  for(unsigned int m : topology[n]) {
    formulation(n).integral(- im*k * n_ % (dof(e(n))%n_), tf(e(n)),
                                                sigma(n,m), "Gauss8");
    formulation(n).integral(- formulation.artificialSource(g(n,m)),
                                  tf(e(n)), sigma(n,m), "Gauss8");
```

```
    formulation(n,m).integral(dof(g(m,n)), tf(g(m,n)),
                                          sigma(n,m), "Gauss8");
    formulation(n,m).integral(formulation.artificialSource(g(n,m)),
                              tf(g(m,n)), sigma(n,m), "Gauss8");
    formulation(n,m).integral(-2.*im*k * n_ % (e(n) % n_),
                              tf(g(m,n)), sigma(n,m), "Gauss8");
  }
}

formulation.pre(); formulation.solve("gmres");
```

Listing 6.6: The field definition and the formulation of our eletromagnetic scattering problem in GmshDDM.

5 Conclusion and perspectives

In this chapter we briefly overviewed GmshDDM, an open-source `C++` domain decomposition library based on GmshFEM. After an introduction to the basic philosophy of the library with an acoustic example, its application to electromagnetic and elastodynamic wave problems was presented. Two perspectives are closely related to those of GmshFEM: the development of a multi-language API and the offloading on GPUs. The second, in particular, is technically and mathematically challenging, as GPUs will entail using a much larger number of smaller subdomains, where dense algebra could be used. Going to the limit of one element per subdomain leads to a method that is formally close to modern *hybridized* version of the *Discontinuous Galerkin methods* (DG) where fluxes, *i.e.* the continuity conditions between elements, plays the same role as the transmission conditions [69]. Another perspective is to include automatic support for overlapping DDMs, by extending the Gmsh partitioning interface. Finally, GmshDDM itself could be extended beyond one-level optimized Schwarz methods by including classical two- or multi-level DDMs for elliptic or parabolic problems, e.g. by interfacing the HPDDM framework [117, 118].

Conclusion and perspectives

This thesis focused on the efficient numerical solution of frequency-domain wave propagation problems using finite element methods.

In the first part of the manuscript, the development of efficient domain decomposition methods was addressed, with the aim of overcoming the limitations of state-of-the art direct and iterative solvers. To this end, in Chapter 2, a non-overlapping substructured domain decomposition method with high-order absorbing conditions used as transmission conditions (HABC DDM) has first been extended to deal with cross-points, where more than two subdomains meet. The handling of cross-points is a well-known issue for non-overlapping HABC DDMs. Our methodology proposes an efficient solution for lattice-type domain partitions, where the domains meet at right angles. The method is based on the introduction of suitable relations and additional transmission variables at the cross-points, and its effectiveness was demonstrated on several test cases. In Chapter 3, a similar non-overlapping substructured DDM was then proposed, but with Perfectly Matched Layers (PMLs) instead of HABCs used as transmission conditions. The proposed approach naturally considers cross-points for two-dimensional checkerboard domain partitions through Lagrange multipliers used for the weak coupling between subproblems defined on rectangular subdomains and the surrounding PMLs. Two discretizations for the Lagrange multipliers and several stabilization strategies have been proposed and compared. The best two converging approaches are continuous discretization with an additional corner equation and discontinuous discretization with a higher polynomial degree and penalty. In Chapter 4, the performance of the methods proposed in the first two chapters is compared on test cases of increasing complexity, from two-dimensional wave scattering in homogeneous media to three-dimensional wave propagation in highly heterogeneous media. While the theoretical developments were carried out for the scalar Helmholtz equation for acoustic wave propagation, Chapter 4 also presents results obtained on the extension of the method to elastic wave problems, highlighting the potential for further generalizations to other physical contexts.

The second part of the manuscript was devoted to the presentation of the computational tools developed during the thesis and which were used to produce all the numerical results presented in Part I. Chapter 5 introduces GmshFEM, a new `C++` finite element library based on the application programming interface (API) of the open-source finite element mesh generator, pre-, and post-processor Gmsh [93], while Chapter 6 presents the domain decomposition library GmshDDM, based on GmshFEM. Both GmshFEM and GmshDDM are designed to be fast, light, and user-friendly. Strongly inspired by the design of the finite element code GetDP [92, 70] and domain decomposition code GetDDM [182], developed in the ACE laboratory, GmshFEM and GmshDDM can be seen as their modern upgrades that have allowed to efficiently run the high-frequency simulations presented in the first part of the manuscript.

The thesis opens up many avenues for further research.

In the context of DDMs for time-harmonic wave problems, a first perspective is to consider an alternative approach without Lagrange multipliers, where the continuity of the Dirichlet traces between the subdomain and the edge PMLs and between the edge and corner PMLs is imposed strongly through the definition of appropriate function spaces at the discrete level. This idea is being currently investigated. Another promising avenue to deal with cross-points is making use of the multi-trace formalism as proposed in [55]. Currently the resulting transmission conditions are however nonlocal and their effectiveness for high-frequency problems needs to be assessed. To bypass the cross-point issue, but at the cost of a higher number of unknowns in each subdomain, one could also investigate overlapping DDMs. In their non-substructured form [38] their numerical implementation is easy, but the increase in local factorization cost is substantial and the number of unknowns for the iterative solver is huge, which can become an issue for Krylov subspace techniques with long recurrences such as GMRES. Investigating short recurrence solvers such as TFQMR [170] or substructured versions of overlapping DDMs would thus be of great interest. Another strategy altogether is to investigate first order formulations, which bypass the cross point issue when discretized with mixed finite elements. The issue of the increased number of unknowns could be mitigated through the use of modern hybridization techniques such as Hybridized Discontinuous Galerkin (HDG) [97, 41, 5] or Hybrid High Order (HHO) [162, 52] methods. Finally, while for the applications treated in this thesis a regular decomposition of the computational domain was appropriate, the development of high-order DDMs that allow to efficiently deal with the "jagged" interfaces resulting from the automatic mesh partitioning of unstructured grids should be investigated. One strategy could take inspiration from [67], which is designed for low-order conditions. Another possibility would be to investigate the development of immersed transmission conditions [144].

Next, the extension to other physical contexts should be further investigated. While the extension to isotropic elastodynamics was briefly presented in Chapter 4, handling anisotropic cases will be challenging. Recent advances in HABCs [141, 142] and in half-space matching methods [42] for elasticity could serve as a starting point for HABC DDMs, as designing stable PMLs for the anisotropic case have proved elusive. The extension to electromagnetics and flow acoustics on the other hand should be more straightforward, since stable PML formulations already exist for these problems.

More fundamentally, even if the DDM significantly gains in efficiency thanks to the cross-point treatment, as a one-level method its strong scalability is still intrin-

sically dependent on the number of subdomains. The design of multi-level DDMs is an open problem for high-frequency wave propagation, however. Indeed, designing a "coarse grid" (the second level in a two-level DDM) is notoriously challenging, as coarse spaces cannot really be "coarse" in the high-frequency regime since they need to capture the highly oscillatory nature of the solution [61, 14, 44]. While sweeping preconditioners [188, 189, 180] provide an interesting alternative, with some potential for parallelism [63], their overall parallel scalability is also limited. Improvements in the scalability of the DDM are crucial in view of its use in inverse problem optimization loops, *e.g.* for underground characterization. A promising idea are then multi-step one-shot methods, which iterate on the forward problem solution and on the inverse problem for model parameters in the same loop [40].

Many perspectives also exist with respect to further development of the computational tools presented in the second part of the manuscript. With the growth in popularity of accelerators, in particular graphical processing units (GPUs), the offloading of some computations in both GmshFEM and GmshDDM should be investigated. First, developing a GPU-friendly version of the DDM algorithm could be exciting from a computation method point of view. One idea would be to consider tiny subdomains of only a few high-order elements, where the resulting finite element dense subdomain matrix could be inverted on the GPUs. Combined with a short-recurrence Krylov solver such as TFQMR, all the iterative solution could thus take place on GPU. In this approach, the DDM becomes similar to a HDG method where fluxes, *i.e.* the continuity conditions between elements, plays the same role as the transmission conditions. Second, one should investigate the offloading on GPU of the dense linear algebra calculations in GmshFEM, currently handled by Eigen. For sufficiently high order interpolations the benefit could be substantial in particular on modern unified memory architectures, where the bottleneck of moving data from CPU to GPU memory and back, vanishes [90]. Finally, one could also imagine porting the tree-based function evaluator to GPU by adding support for GPU kernel nodes. This offloading of function evaluations on GPUs could be very interesting for problems that require many finite element matrix assemblies, such as nonlinear and/or time-domain simulations.

Beside these purely computational improvements, several developments are still necessary to bring GmshFEM closer in terms of feature parity with GetDP. In particular, one can mention the handling of function spaces involving several different types of basis functions (as required for example for the strong coupling of fields and potentials [91]), the coupling with circuit equations [73] and the handling of moving meshes [71].

Finally, developing a unified Python and Julia API for GmshFEM and GmshDDM, similar to the API of Gmsh, would be extremely useful. It would allow to write and run all the formulations that can already be handled by the codes without requiring familiarity with `C++`, but still with an equivalent syntax such that the mathematics behind the formulations can be easily understood. More importantly, developing an API for interpreted languages such as Python and Julia offers the possibility to use GmshFEM and GmshDDM without recompilation, which opens exciting opportunities for users not familiar with code development.

Bibliography

[1] Xavier Adriaens. *Inner product preconditioned optimization methods for full waveform inversion*. PhD thesis, 2023.

[2] Xavier Adriaens, François Henrotte, and Christophe Geuzaine. Adjoint state method for time-harmonic scattering problems with boundary perturbations. *Journal of Computational Physics*, 428:109981, 2021.

[3] Xavier Adriaens, Ludovic Métivier, and Christophe Geuzaine. A trust-region newton method for frequency-domain full-waveform inversion. In *First International Meeting for Applied Geoscience & Energy*, pages 757–761. Society of Exploration Geophysicists, 2021.

[4] Xavier Adriaens, Ludovic Métivier, and Christophe Geuzaine. Inner product preconditioned trust-region methods for frequency-domain full waveform inversion. *Available at SSRN 4258740*, 2022.

[5] Emmanuel Agullo, Luc Giraud, Alexis Gobé, Matthieu Kuhn, Stéphane Lanteri, and Ludovic Moya. High order hdg method and domain decomposition solvers for frequency-domain electromagnetics. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 33(2):e2678, 2020.

[6] Martin Alnæs, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris Richardson, Johannes Ring, Marie E. Rognes, and Garth N. Wells. The FEniCS project version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[7] Patrick R Amestoy, Iain S Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[8] Patrick R Amestoy, Abdou Guermouche, Jean-Yves L'Excellent, and Stéphane Pralet. Hybrid scheduling for the parallel solution of linear systems. *Parallel computing*, 32(2):136–156, 2006.

[9] Robert Anderson, Julian Andrej, Andrew Barker, Jamie Bramwell, Jean-Sylvain Camier, Jakub Cerveny, Veselin Dobrev, Yohann Dudouit, Aaron Fisher, Tzanio Kolev, et al. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021.

[10] Xavier Antoine. Fast approximate computation of a time-harmonic scattered field using the on-surface radiation condition method. *IMA Journal of Applied Mathematics*, 66(1):83–110, 2001.

[11] Xavier Antoine and Marion Darbas. Integral equations and iterative schemes for acoustic scattering problems, 2016.

[12] Xavier Antoine, Marion Darbas, and Ya Yan Lu. An improved surface radiation condition for high-frequency acoustic scattering problems. *Computer Methods in Applied Mechanics and Engineering*, 195(33-36):4060–4074, 2006.

[13] Daniel Arndt, Wolfgang Bangerth, Denis Davydov, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Jean-Paul Pelteret, Bruno Turcksin, and David Wells. The deal. II finite element library: Design, features, and insights. *Computers & Mathematics with Applications*, 81:407–422, 2021.

[14] Ali Vaziri Astaneh and Murthy N. Guddati. A two-level domain decomposition method with accurate interface conditions for the Helmholtz problem. *International Journal for Numerical Methods in Engineering*, 107(1):74–90, 2016.

[15] Sergey Asvadurov, Vladimir Druskin, Murthy N. Guddati, and Leonid Knizhnerman. On optimal finite-difference approximation of PML. *SIAM Journal on Numerical Analysis*, 41(1):287–305, 2003.

[16] Ismaïl Badia. *Couplage par décomposition de domaine optimisée de formulations intégrales et éléments finis d'ordre élevé pour l'électromagnétisme*. PhD thesis, Université de Liège, Belgium, Université de Lorraine, France, 2022.

[17] Ismaïl Badia, Boris Caudron, Xavier Antoine, and Christophe Geuzaine. A well-conditioned weak coupling of boundary element and high-order finite element methods for time-harmonic electromagnetic scattering by inhomogeneous objects. *SIAM Journal on Scientific Computing*, 44(3):B640–B667, 2022.

[18] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.11, Argonne National Laboratory, 2019.

[19] Alain Bamberger, Björn Engquist, Laurence Halpern, and Patrick Joly. Higher order paraxial wave equation approximations in heterogeneous media. *SIAM Journal on Applied Mathematics*, 48(1):129–154, 1988.

[20] Gang Bao and Haijun Wu. Convergence analysis of the perfectly matched layer problems for time-harmonic Maxwell's equations. *SIAM journal on numerical analysis*, 43(5):2121–2143, 2005.

[21] Peter Bastian, Markus Blatt, Andreas Dedner, Nils-Arne Dreier, Christian Engwer, René Fritze, Carsten Gräser, Christoph Grüninger, Dominic Kempf, Robert Klöfkorn, et al. The Dune framework: Basic concepts and recent developments. *Computers & Mathematics with Applications*, 81:75–112, 2021.

[22] Ushnish Basu and Anil K. Chopra. Perfectly matched layers for time-harmonic elastodynamics of unbounded domains: theory and finite-element implementation. *Computer methods in applied mechanics and engineering*, 192(11-12):1337–1375, 2003.

[23] Klaus-Jürgen Bathe. The inf–sup condition and its evaluation for mixed finite element methods. *Computers & structures*, 79(2):243–252, 2001.

[24] Alvin Bayliss, Charles Goldstein, and Eli Turkel. On accuracy conditions for the numerical computation of waves. *Journal of Computational Physics*, 59(3):396–404, 1985.

[25] Alvin Bayliss and Eli Turkel. Radiation boundary conditions for wave-like equations. *Communications on Pure and applied Mathematics*, 33(6):707–725, 1980.

[26] Éric Béchet, Nicolas Moës, and Barbara Wohlmuth. A stable Lagrange multiplier space for stiff interface conditions within the extended finite element method. *International Journal for Numerical Methods in Engineering*, 78(8):931–954, 2009.

[27] Jean-David Benamou and Bruno Després. A domain decomposition method for the helmholtz equation and related optimal control problems. *Journal of Computational Physics*, 136(1):68–82, 1997.

[28] Abderrahmane Bendali and Yassine Boubendir. Non-overlapping domain decomposition method for a nodal finite element method. *Numerische Mathematik*, 103(4):515–537, 2006.

[29] Jean-Pierre Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185–200, 1994.

[30] Jean-Pierre Bérenger. Perfectly matched layer for the FDTD solution of wave-structure interaction problems. *IEEE Transactions on antennas and propagation*, 44(1):110–117, 1996.

[31] Jean-Pierre Bérenger. Three-dimensional perfectly matched layer for the absorption of electromagnetic waves. *Journal of computational physics*, 127(2):363–379, 1996.

[32] Hadrien Bériot and Axel Modave. An automatic perfectly matched layer for acoustic finite element simulations in convex domains of general shape. *International Journal for Numerical Methods in Engineering*, 122(5):1239–1261, 2021.

[33] A Bermúdez, L Hervella-Nieto, Andrés Prieto, and Rodolfo Rodríguez. Perfectly matched layers for time-harmonic second order elliptic problems. *Archives of Computational Methods in Engineering*, 17(1):77–107, 2010.

[34] Alfredo Bermúdez, Luis Hervella-Nieto, Andrés Prieto, and Rodolfo Rodríguez. An exact bounded PML for the Helmholtz equation. *Comptes Rendus Mathematique*, 339(11):803–808, 2004.

[35] Alfredo Bermúdez, Luis Hervella-Nieto, Andrés Prieto, and Rodolfo Rodríguez. An optimal perfectly matched layer with unbounded absorbing function for time-harmonic acoustic scattering problems. *Journal of Computational Physics*, 223(2):469–488, May 2007.

[36] Alfredo Bermúdez, Luis Hervella-Nieto, Andrés Prieto, and Rodolfo Rodríguez. An exact bounded perfectly matched layer for time-harmonic scattering problems. *SIAM Journal on Scientific Computing*, 30(1):312–338, 2008.

[37] Daniele Boffi, Franco Brezzi, Michel Fortin, et al. *Mixed finite element methods and applications*, volume 44. Springer, 2013.

[38] Marcella Bonazzoli, Xavier Claeys, Frédéric Nataf, and Pierre-Henri Tournier. How does the partition of unity influence SORAS preconditioner? *arXiv preprint arXiv:2212.03132*, 2022.

[39] Marcella Bonazzoli, Victorita Dolean, Ivan G Graham, Euan A Spence, and Pierre-Henri Tournier. Two-level preconditioners for the Helmholtz equation. In *International Conference on Domain Decomposition Methods*, pages 139–147. Springer, 2017.

[40] Marcella Bonazzoli, Houssem Haddar, and Tuan Anh Vu. Convergence analysis of multi-step one-shot methods for linear inverse problems. *arXiv preprint arXiv:2207.10372*, 2022.

[41] Marie Bonnasse-Gahot, Henri Calandra, Julien Diaz, and Stéphane Lanteri. Hybridizable discontinuous Galerkin method for the 2-D frequency-domain elastic wave equations. *Geophysical Journal International*, 213(1):637–659, 2018.

[42] Anne-Sophie Bonnet-Ben Dhia, Simon N Chandler-Wilde, Sonia Fliss, Christophe Hazard, Karl-Mikael Perfekt, and Yohanes Tjandrawidjaja. The complex-scaled half-space matching method. *SIAM Journal on Mathematical Analysis*, 54(1):512–557, 2022.

[43] Anne-Sophie Bonnet-Ben Dhia, Sonia Fliss, and Antoine Tonnoir. The halfspace matching method: A new method to solve scattering problems in infinite media. *Journal of Computational and Applied Mathematics*, 338:44–68, 2018.

[44] Niall Bootland, Victorita Dolean, Pierre Jolivet, and Pierre-Henri Tournier. A comparison of coarse spaces for Helmholtz problems in the high frequency regime. *Computers & Mathematics with Applications*, 98:239–253, 2021.

[45] Yassine Boubendir, Xavier Antoine, and Christophe Geuzaine. A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation. *Journal of Computational Physics*, 231(2):262–280, 2012.

[46] Yassine Boubendir and Dawid Midura. Non-overlapping domain decomposition algorithm based on modified transmission conditions for the helmholtz equation. *Computers & Mathematics with Applications*, 75(6):1900–1911, 2018.

[47] A. Brougois, Marielle Bourget, Patriek Lailly, Michel Poulet, Patrice Ricarte, and Roelof Versteeg. Marmousi, model and data. In *EAEG workshop-practical aspects of seismic data inversion*, pages cp–108. European Association of Geoscientists & Engineers, 1990.

[48] Xiao-Chuan Cai and Olof B. Widlund. Domain decomposition algorithms for indefinite elliptic problems. *SIAM Journal on Scientific and Statistical Computing*, 13(1):243–258, 1992.

[49] Goong Chen and Jianxin Zhou. Boundary element methods. *176*, 1992.

[50] Junqing Chen and Zhiming Chen. An adaptive perfectly matched layer technique for 3-d time-harmonic electromagnetic scattering problems. *Mathematics of computation*, 77(262):673–698, 2008.

[51] 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, 1994.

[52] Matteo Cicuttin and Christophe Geuzaine. Numerical investigation of a 3D Hybrid High-Order Method for the Indefinite Time-Harmonic Maxwell Problem. *HAL preprint hal-03808094*, 2022.

[53] Matteo Cicuttin, Anthony Royer, Peter Binde, and Christophe Geuzaine. Electrostatic discharge simulation using a gpu-accelerated dgtd solver targeting modern graphics processors. *IEEE Transactions on Magnetics*, 58(9):1–4, 2022.

[54] Xavier Claeys. Non-local variant of the Optimised Schwarz Method for arbitrary non-overlapping subdomain partitions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 55(2):429–448, 2021.

[55] Xavier Claeys and Emile Parolin. Robust treatment of cross-points in optimized Schwarz methods. *Numerische Mathematik*, pages 1–38, 2022.

[56] Francis Collino. *Conditions absorbantes d'ordre élevé pour les équations de Maxwell dans des domaines rectangulaires*. PhD thesis, INRIA, France, 1993.

[57] Francis Collino and al. High order absorbing boundary conditions for wave propagation models. straight line boundary and corner cases. In *Second International Conference on Mathematical and Numerical Aspects of Wave Propagation (Newark, DE, 1993)*, pages 161–171, 1993.

[58] Francis Collino, Souad Ghanemi, and Patrick Joly. Domain decomposition method for harmonic wave propagation: a general presentation. *Computer methods in applied mechanics and engineering*, 184(2-4):171–211, 2000.

[59] Francis Collino, Patrick Joly, and Matthieu Lecouvez. Exponentially convergent non overlapping domain decomposition methods for the Helmholtz equation. *ESAIM: Mathematical Modelling and Numerical Analysis*, 54(3):775–810, 2020.

[60] Francis Collino and Peter Monk. The perfectly matched layer in curvilinear coordinates. *SIAM Journal on Scientific Computing*, 19(6):2061–2090, 1998.

[61] Lea Conen, Victorita Dolean, Rolf Krause, and Frédéric Nataf. A coarse space for heterogeneous Helmholtz problems based on the Dirichlet-to-Neumann operator. *Journal of Computational and Applied Mathematics*, 271:83–99, 2014.

[62] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, 1969.

[63] Ruiyang Dai, Axel Modave, Jean-François Remacle, and Christophe Geuzaine. Multidirectionnal sweeping preconditioners with non-overlapping checkerboard domain decomposition for helmholtz problems. *Journal of Computational Physics*, page 110887, 2022.

[64] Electricité de France. Finite element *Code_Aster*, analysis of structures and thermomechanics for studies and research. www.code-aster.org, 1989–2017.

[65] Armel de La Bourdonnaye, Charbel Farhat, Antonini Macedo, Frédéric Magoules, François-Xavier Roux, et al. A non-overlapping domain decomposition method for the exterior Helmholz problem. *Contemporary Mathematics*, 218:42–66, 1998.

[66] Bruno Després. *Méthodes de décomposition de domaines pour les problemes de propagation d'ondes en régime harmonique. Le théoreme de Borg et l'équation de Hill vectorielle.* PhD thesis, Université Paris 9, France, 1991.

[67] Bruno Després, Anouk Nicolopoulos, and Bertrand Thierry. Corners and stable optimized domain decomposition methods for the Helmholtz problem. *Numerische Mathematik*, 149(4):779–818, 2021.

[68] Guido Dhondt and Klaus Wittig. CALCULIX: A free software three-dimensional structural finite element program. www.calculix.de, 2022.

[69] Vít Dolejší and Miloslav Feistauer. *Discontinuous galerkin method*, volume 48. Springer, 2015.

[70] Patrick Dular and Christophe Geuzaine. GetDP reference manual: the documentation for GetDP, a general environment for the treatment of discrete problems. http://getdp.info, 1997.

[71] Patrick Dular, Christophe Geuzaine, MV Ferreira da Luz, N Sadowski, and JPA Bastos. Connection boundary conditions with different types of finite elements applied to periodicity conditions and to the moving band. *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering*, 2001.

[72] Patrick Dular, Christophe Geuzaine, François Henrotte, and Willy Legros. A general environment for the treatment of discrete problems and its application to the finite element method. *IEEE Transactions on Magnetics*, 34(5):3395–3398, 1998.

[73] Patrick Dular, Christophe Geuzaine, and Willy Legros. A natural method for coupling magnetodynamic h-formulations and circuit equations. *IEEE transactions on magnetics*, 35(3):1626–1629, 1999.

[74] Patrick Dular, Willy Legros, and André Nicolet. Coupling of local and global quantities in various finite element formulations and its application to electrostatics, magnetostatics and magnetodynamics. *IEEE Transactions on Magnetics*, 34(5):3078–3081, 1998.

[75] Björn Engquist and Andrew Majda. Absorbing boundary conditions for numerical simulation of waves. *Proceedings of the National Academy of Sciences*, 74(5):1765–1766, 1977.

[76] Björn Engquist and Lexing Ying. Sweeping preconditioner for the helmholtz equation: hierarchical matrix representation. *Communications on pure and applied mathematics*, 64(5):697–735, 2011.

[77] Björn Engquist and Lexing Ying. Fast algorithms for high frequency wave propagation. In *Numerical Analysis of Multiscale Problems*, pages 127–161. Springer, 2012.

[78] Alexandre Ern and Jean-Luc Guermond. *Finite Elements I: Approximation and Interpolation*, volume 73. Springer, 2021.

[79] Alexandre Ern and Jean-Luc Guermond. *Finite Elements II: Galerkin approximation, elliptic and mixed PDEs*, volume 73. Springer Nature, 2021.

[80] Oliver G. Ernst and Martin J. Gander. Why it is difficult to solve Helmholtz problems with classical iterative methods. *Numerical analysis of multiscale problems*, pages 325–363, 2012.

[81] John Etgen and T. Kunz. The 3D SEG Salt Model – Dreams and Reality. In *EAEG/SEG Summer Workshop-Construction of 3-D Macro Velocity-Depth Models*, pages cp–96. European Association of Geoscientists & Engineers, 1994.

[82] Charbel Farhat, Philip Avery, Radek Tezaur, and Jing Li. FETI-DPH: a dual-primal domain decomposition method for acoustic scattering. *Journal of Computational Acoustics*, 13(03):499–524, 2005.

[83] Charbel Farhat, Antonini Macedo, and Michel Lesoinne. A two-level domain decomposition method for the iterative solution of high frequency exterior Helmholtz problems. *Numerische Mathematik*, 85(2):283–308, 2000.

[84] Martin J. Gander and Laurence Halpern. A simple finite difference discretization for Vent-cell transmission conditions at cross points. In *Proceedings of the 26th International Domain Decomposition Conference*, 2020.

[85] Martin J. Gander, Frédéric Magoules, and Frédéric Nataf. Optimized Schwarz methods without overlap for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 24(1):38–60, 2002.

[86] Martin J. Gander and Kévin Santugini. Cross-points in domain decomposition methods with a finite element discretization. *Electronic Transactions on Numerical Analysis*, 45:219—-240, 2016.

[87] Martin J. Gander and Hui Zhang. Optimized Schwarz methods with overlap for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 38(5):A3195–A3219, 2016.

[88] Martin J. Gander and Hui Zhang. A class of iterative solvers for the Helmholtz equation: factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized Schwarz methods. *Siam Review*, 61(1):3–76, 2019.

[89] Mahadevan Ganesh and Charles Morgenstern. High-order FEM domain decomposition models for high-frequency wave propagation in heterogeneous media. *Computers & Mathematics with Applications*, 75(6):1961–1972, 2018.

[90] Lars Gebraad and Andreas Fichtner. Seamless GPU acceleration for C++ based physics with the Metal Shading Language on Apple's M series unified chips. *arXiv e-prints*, pages arXiv–2206, 2022.

[91] Christophe Geuzaine. *High order Hybrid finite element schemes for Maxwell's equations taking thin structures and global quantities into account.* PhD thesis, Université de Liège, Belgium, 2001.

[92] Christophe Geuzaine. GetDP: a general finite-element solver for the de Rham complex. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 7, pages 1010603–1010604. Wiley Online Library, 2007.

[93] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79:1309–1331, May 2009.

[94] Dan Givoli. Non-reflecting boundary conditions. *Journal of computational physics*, 94(1):1–29, 1991.

[95] Dan Givoli. Recent advances in the DtN FE method. *Archives of Computational Methods in Engineering*, 6(2):71–116, 1999.

[96] Ivan Graham, E. Spence, and Eero Vainikko. Domain decomposition preconditioning for high-frequency Helmholtz problems with absorption. *Mathematics of Computation*, 86(307):2089–2127, 2017.

[97] Roland Griesmaier and Peter Monk. Error analysis for a hybridizable discontinuous Galerkin method for the Helmholtz equation. *Journal of Scientific Computing*, 49(3):291–310, 2011.

[98] Zhang Guan-Quan. High order approximation of one-way wave equations. *Journal of computational Mathematics*, pages 90–97, 1985.

[99] Murthy N. Guddati. Arbitrarily wide-angle wave equations for complex media. *Computer Methods in Applied Mechanics and Engineering*, 195(1-3):65–93, 2006.

[100] Murthy N. Guddati and Keng-Wit Lim. Continued fraction absorbing boundary conditions for convex polygonal domains. *International Journal for Numerical Methods in Engineering*, 66(6):949–977, 2006.

[101] Murthy N. Guddati and John L. Tassoulas. Continued-fraction absorbing boundary conditions for the wave equation. *Journal of Computational Acoustics*, 8(01):139–156, 2000.

[102] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. `http://eigen.tuxfamily.org`, 2010.

[103] Thomas Hagstrom, Reginal P. Tewarson, and Aron Jazcilevich. Numerical experiments on a domain decomposition algorithm for nonlinear elliptic boundary value problems. *Applied Mathematics Letters*, 1(3):299–302, 1988.

[104] Alexandre Halbach. Sparselizard - the user friendly finite element c++ library. `www.sparselizard.org`, 2017.

[105] Chris Hamilton. Compact hilbert indices. *Dalhousie University, Faculty of Computer Science, Technical Report CS-2006-07*, 2006.

[106] Isaac Harari and Uri Albocher. Studies of FE/PML for exterior problems of time-harmonic elastic waves. *Computer methods in applied mechanics and engineering*, 195(29-32):3854–3879, 2006.

[107] Frank D. Hastings, John B. Schneider, and Shira L. Broschat. Application of the perfectly matched layer (PML) absorbing boundary condition to elastic wave propagation. *The Journal of the Acoustical Society of America*, 100(5):3061–3069, 1996.

[108] Frédéric Hecht. New development in FreeFem++. *Journal of numerical mathematics*, 20(3-4):251–266, 2012.

[109] François Henrotte, Hakim Hedia, N. Bamps, André Genon, André Nicolet, and Willy Legros. A new method for axisymmetrical linear and nonlinear problems. *IEEE transactions on magnetics*, 29(2):1352–1355, 1993.

[110] François Henrotte, Benoît Meys, Hakim Hedia, Patrick Dular, and Willy Legros. Finite element modelling with transformation techniques. *IEEE transactions on magnetics*, 35(3):1434–1437, 1999.

[111] Vicente Hernandez, Jose E. Roman, and Vicente Vidal. SLEPc: A scalable and flexible toolkit for the solution of eigenvalue problems. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):351–362, 2005.

[112] Fang Q. Hu. On absorbing boundary conditions for linearized Euler equations by a perfectly matched layer. *Journal of computational physics*, 129(1):201–219, 1996.

[113] Frank Ihlenburg. *Finite element analysis of acoustic scattering*. Springer, 1998.

[114] Frank Ihlenburg and Ivo Babuška. Finite element solution of the Helmholtz equation with high wave number Part i: The h-version of the fem. *Computers & Mathematics with Applications*, 30(9):9–37, 1995.

[115] Frank Ihlenburg and Ivo Babuška. Finite element solution of the Helmholtz equation with high wave number Part ii: the hp version of the fem. *SIAM Journal on Numerical Analysis*, 34(1):315–358, 1997.

[116] Johninson F. Imhoff, Gerard Meunier, Xavier Brunotte, and Jean-Claude Sabonnadiere. An original solution for unbounded electromagnetic 2D-and 3D-problems throughout the finite element method. *IEEE Transactions on Magnetics*, 26(5):1659–1661, 1990.

[117] Pierre Jolivet, Frédéric Hecht, Frédéric Nataf, and Christophe Prud'Homme. Scalable domain decomposition preconditioners for heterogeneous elliptic problems. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11, 2013.

[118] Pierre Jolivet, Jose E. Roman, and Stefano Zampini. KSPHPDDM and PCHPDDM: extending PETSc with advanced Krylov methods and robust multilevel overlapping Schwarz preconditioners. *Computers & Mathematics with Applications*, 84:277–295, 2021.

[119] Douglas Jones. Surface radiation conditions. *IMA journal of applied mathematics*, 41(1):21–30, 1988.

[120] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[121] Riyad Kechroud, Xavier Antoine, and Azzeddine Soulaïmani. Numerical accuracy of a Padé-type non-reflecting boundary condition for the finite element solution of acoustic scattering problems at high-frequency. *International Journal for Numerical Methods in Engineering*, 64(10):1275–1302, 2005.

[122] Joseph B. Keller and Dan Givoli. Exact non-reflecting boundary conditions. *Journal of computational physics*, 82(1):172–192, 1989.

[123] Seungil Kim and Hui Zhang. Optimized schwarz method with complete radiation transmission conditions for the Helmholtz equation in waveguides. *SIAM Journal on Numerical Analysis*, 53(3):1537–1558, 2015.

[124] Jung-Han Kimn and Marcus Sarkis. Restricted overlapping balancing domain decomposition methods and restricted coarse problems for the Helmholtz problem. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1507–1514, 2007.

[125] Gregorya Kriegsmann, Allen Taflove, and Koradarr Umashankar. A new formulation of electromagnetic wave scattering using an on-surface radiation boundary condition approach. *IEEE Transactions on Antennas and Propagation*, 35(2):153–161, 1987.

[126] Prem K. Kythe. *An introduction to boundary element methods*. CRC press, 2020.

[127] Matthieu Lecouvez. *Méthodes itératives de décomposition de domaine sans recouvrement avec convergence géométrique pour l'équation de Helmholtz*. PhD thesis, Ecole Polytechnique, France, 2015.

[128] Matthieu Lecouvez, Bruno Stupfel, Patrick Joly, and Francis Collino. Quasi-local transmission conditions for non-overlapping domain decomposition methods for the helmholtz equation. *Comptes Rendus Physique*, 15(5):403–414, 2014.

[129] Wei Leng and Lili Ju. An additive overlapping domain decomposition method for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 41(2):A1252–A1277, 2019.

[130] EL Lindman. "free-space" boundary conditions for the time dependent wave equation. *Journal of computational physics*, 18(1):66–78, 1975.

[131] Pierre-Louis Lions. On the Schwarz alternating method. III: a variant for nonoverlapping subdomains. In *Third international symposium on domain decomposition methods for partial differential equations*, volume 6, pages 202–223. SIAM Philadelphia, 1990.

[132] Sébastien Loisel. Condition number estimates for the nonoverlapping optimized Schwarz method and the 2-Lagrange multiplier method for general domains and cross points. *SIAM Journal on Numerical Analysis*, 51(6):3062–3083, 2013.

[133] Steffen Marburg and Bodo Nolte. *Computational acoustics of noise propagation in fluids: finite and boundary element methods*, volume 578. Springer, 2008.

[134] Philippe Marchner. *Non-reflecting boundary conditions and domain decomposition methods for industrial flow acoustics*. PhD thesis, Université de Liège, Belgium, Université de Lorraine, France, 2022.

[135] Philippe Marchner, Xavier Antoine, Christophe Geuzaine, and Hadrien Bériot. Construction and numerical assessment of local absorbing boundary conditions for heterogeneous time-harmonic acoustic problems. *SIAM Journal on Applied Mathematics*, 82(2):476–501, 2022.

[136] Nicolas Marsic and Herbert De Gersem. Convergence of classical optimized non-overlapping Schwarz method for Helmholtz problems in closed domains. *arXiv preprint arXiv:2001.01502*, 2020.

[137] Nicolas Marsic and Christophe Geuzaine. Efficient finite element assembly of high order Whitney forms. *IET Science, Measurement & Technology*, 9(2):204–210, 2015.

[138] Nicolas Marsic, Christophe Geuzaine, and Herbert De Gersem. Transmission operators for the non-overlapping schwarz method for solving helmholtz problems in rectangular cavities. *Computers & Mathematics with Applications*, 138:37–64, 2023.

[139] Gary S. Martin, Shawn Larsen, and Kurt Marfurt. Marmousi-2: An updated model for the investigation of AVO in structurally complex areas. In *2002 SEG Annual Meeting*. OnePetro, 2002.

[140] Gary S. Martin, Robert Wiley, and Kurt J. Marfurt. Marmousi2: An elastic upgrade for Marmousi. *The leading edge*, 25(2):156–166, 2006.

[141] Vanessa Mattesi, Marion Darbas, and Christophe Geuzaine. A high-order absorbing boundary condition for 2D time-harmonic elastodynamic scattering problems. *Computers & Mathematics with Applications*, 77(6):1703–1721, 2019.

[142] Vanessa Mattesi, Marion Darbas, and Christophe Geuzaine. A quasi-optimal non-overlapping domain decomposition method for two-dimensional time-harmonic elastic wave problems. *Journal of Computational Physics*, 401:109050, 2020.

[143] Fausto A. Milinazzo, Cedric A. Zala, and Gary H. Brooke. Rational square-root approximations for parabolic equation algorithms. *The Journal of the Acoustical Society of America*, 101(2):760–766, 1997.

[144] Rajat Mittal and Gianluca Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[145] Raj Mittra and Omar M. Ramahi. Absorbing boundary conditions for the direct solution of partial differential equations arising in electromagnetic scattering problems. *Progress In Electromagnetics Research*, 2:133–173, 1990.

[146] Axel Modave, Andreas Atle, Jesse Chan, and Tim Warburton. A GPU-accelerated nodal discontinuous Galerkin method with high-order absorbing boundary conditions and corner/edge compatibility. *International Journal for Numerical Methods in Engineering*, 112(11):1659–1686, 2017.

[147] Axel Modave, Christophe Geuzaine, and Xavier Antoine. Corner treatments for high-order local absorbing boundary conditions in high-frequency acoustic scattering. *Journal of Computational Physics*, 401:109029, 2020.

[148] Axel Modave, Anthony Royer, Xavier Antoine, and Christophe Geuzaine. A non-overlapping domain decomposition method with high-order transmission conditions and cross-point treatment for helmholtz problems. *Computer Methods in Applied Mechanics and Engineering*, 368:113162, 2020.

[149] Andrea Moiola and Euan A. Spence. Is the Helmholtz equation really sign-indefinite? *SIAM Review*, 56(2):274–312, 2014.

[150] Thomas G. Moore, Jeffery G. Blaschak, Allen Taflove, and Gregory A. Kriegsmann. Theory and application of radiation boundary operators. *IEEE Transactions on Antennas and Propagation*, 36(12):1797–1812, 1988.

[151] Frédéric Nataf. Interface connections in domain decomposition methods. In *Modern methods in scientific computing and applications*, pages 323–364. Springer, 2002.

[152] Frédéric Nataf and Francis Nier. Convergence rate of some domain decomposition methods for overlapping and nonoverlapping subdomains. *Numerische mathematik*, 75(3):357–377, 1997.

[153] Frédéric Nataf, Francois Rogier, and Eric de Sturler. *Optimal interface conditions for domain decomposition methods*. PhD thesis, CMAP Ecole Polytechnique, France, 1994.

[154] Jean-Claude Nédélec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35(3):315–341, 1980.

[155] Jean-Claude Nédélec. A new family of mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 50(1):57–81, 1986.

[156] Jean-Claude Nédélec. *Acoustic and electromagnetic equations: integral representations for harmonic problems*, volume 144. Springer Science & Business Media, 2013.

[157] Anouk Nicolopoulos-Salle. *Formulations variationnelles d'équations de Maxwell résonantes et problèmes aux coins en propagation d'ondes*. PhD thesis, Sorbonne université, France, 2019.

[158] Émile Parolin. *Non-overlapping domain decomposition methods with non-local transmission operators for harmonic wave propagation problems*. PhD thesis, Institut Polytechnique de Paris, France, 2020.

[159] Clemens Pechstein. A unified theory of non-overlapping Robin-Schwarz methods–continuous and discrete, including cross points. *arXiv preprint arXiv:2204.03436*, 2022.

[160] Zhen Peng and Jin-Fa Lee. Non-conformal domain decomposition method with second-order transmission conditions for time-harmonic electromagnetics. *Journal of Computational Physics*, 229(16):5615–5629, 2010.

[161] A. Piacentini and N. Rosa. An improved domain decomposition method for the 3d Helmholtz equation. *Computer methods in applied mechanics and engineering*, 162(1-4):113–124, 1998.

[162] Daniele A Di Pietro, Alexandre Ern, and Simon Lemaire. A review of hybrid high-order methods: formulations, computational aspects, comparison with other methods. *Building bridges: connections and challenges in modern approaches to numerical partial differential equations*, pages 205–236, 2016.

[163] Christophe Prud'Homme, Vincent Chabannes, Vincent Doyeux, Mourad Ismail, Abdoulaye Samake, and Gonçalo Pena. Feel++: A computational framework for galerkin methods and advanced numerical methods. In *ESAIM: Proceedings*, volume 38, pages 429–455. EDP Sciences, 2012.

[164] Peter Råback and Mika Malinen. Overview of Elmer. `http://www.elmerfem.org`, 2016.

[165] Yves Renard and Konstantinos Poulios. GetFEM: Automated FE modeling of multiphysics problems based on a generic weak form language. *ACM Transactions on Mathematical Software (TOMS)*, 47(1):1–31, 2020.

[166] Thomas D. Rossing. *Springer handbook of acoustics*. Springer, 2014.

[167] Anthony Royer, Eric Béchet, and Christophe Geuzaine. GmshFEM: An Efficient Finite Element Library Based On Gmsh. In *14th World Congress on Computational Mechanics (WCCM), ECCOMAS Congress 2020*, 2021.

[168] Anthony Royer, Christophe Geuzaine, Eric Béchet, and Axel Modave. A non-overlapping domain decomposition method with perfectly matched layer transmission conditions for the helmholtz equation. *Computer Methods in Applied Mechanics and Engineering*, 395:115006, 2022.

[169] Youcef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[170] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[171] Shankar P. Sastry, Emre Kultursay, Suzanne M. Shontz, and Mahmut T. Kandemir. Improved cache utilization and preconditioner efficiency through use of a space-filling curve mesh element- and vertex-reordering technique. *Engineering with Computers*, 30(4):535–547, 2014.

[172] Achim Schädle and Lin Zschiedrich. Additive Schwarz method for scattering problems using the PML method at interfaces. In *Domain decomposition methods in science and engineering XVI*, pages 205–212. Springer, 2007.

[173] Joachim Schöberl. C++ 11 implementation of finite elements in NGSolve. Technical report, 2014.

[174] Hermann Amandus Schwarz. *Ueber einen Grenzübergang durch alternirendes Verfahren*. Zürcher u. Furrer, 1870.

[175] Samuel Silver. *Microwave antenna theory and design*. Number 19. Iet, 1984.

[176] Pavel Solin, Karel Segeth, and Ivo Dolezel. *Higher-order finite element methods*. Chapman and Hall/CRC, 2003.

[177] Christiaan C. Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *Journal of Computational Physics*, 241:240–252, 2013.

[178] Christiaan C. Stolk. An improved sweeping domain decomposition preconditioner for the Helmholtz equation. *Advances in Computational Mathematics*, 43(1):45–76, 2017.

[179] Bruno Stupfel. Improved transmission conditions for a one-dimensional domain decomposition method applied to the solution of the Helmholtz equation. *Journal of Computational Physics*, 229(3):851–874, 2010.

[180] Matthias Taus, Leonardo Zepeda-Núñez, Russell J. Hewett, and Laurent Demanet. L-Sweeps: A scalable, parallel preconditioner for the high-frequency Helmholtz equation. *Journal of Computational Physics*, 420:109706, 2020.

[181] Michael Eugene Taylor. Pseudodifferential operators (pms-34). In *Pseudodifferential Operators (PMS-34)*. Princeton University Press, 2017.

[182] Bertrand Thierry, Alexandre Vion, Simon Tournier, Mohamed El Bouajaji, David Colignon, Nicolas Marsic, Xavier Antoine, and Christophe Geuzaine. GetDDM: An open framework for testing optimized Schwarz methods for time-harmonic wave problems. *Computer Physics Communications*, 203:309–330, 2016.

[183] Lonny L. Thompson. A review of finite-element methods for time-harmonic acoustics. *The Journal of the Acoustical Society of America*, 119(3):1315–1330, 2006.

[184] Andrea Toselli. Some results on overlapping Schwarz methods for the Helmholtz equation employing perfectly matched layers. In *Domain Decomposition Methods in Sciences and Engineering: Eleventh International Conference London, UK*, pages 539–545. Citeseer, 1998.

[185] Andrea Toselli and Olof Widlund. *Domain decomposition methods-algorithms and theory*, volume 34. Springer Science & Business Media, 2004.

[186] Eli Turkel and Amir Yefet. Absorbing PML boundary layers for wave-like equations. *Applied Numerical Mathematics*, 27(4):533–557, 1998.

[187] Henk A. Van der Vorst. Bi-cgstab: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, 13(2):631–644, 1992.

[188] Alexandre Vion and Christophe Geuzaine. Double sweep preconditioner for optimized Schwarz methods applied to the Helmholtz problem. *Journal of Computational Physics*, 266:171–190, 2014.

[189] Alexandre Vion and Christophe Geuzaine. Improved sweeping preconditioners for domain decomposition algorithms applied to time-harmonic Helmholtz and Maxwell problems. *ESAIM: Proceedings and Surveys*, 61:93–111, 2018.

[190] Leonardo Zepeda-Núñez and Laurent Demanet. The method of polarized traces for the 2D Helmholtz equation. *Journal of Computational Physics*, 308:347–388, 2016.

[191] Leonardo Zepeda-Núñez, Adrien Scheuer, Russell J. Hewett, and Laurent Demanet. The method of polarized traces for the 3D Helmholtz equation. *Geophysics*, 84(4):T313–T333, 2019.

[192] Olgierd C. Zienkiewicz, D.W. Kelly, and Pete Bettess. The Sommerfeld (radiation) condition on infinite domains and its modelling in numerical procedures. In *Computing Methods in Applied Sciences and Engineering, 1977, I*, pages 169–203. Springer, 1979.