http://lib.ulg.ac.be http://matheo.ulg.ac.be

Simulation of geophysical wave propagation using domain decomposition techniques

Auteur : Royer, Anthony
Promoteur(s) : Geuzaine, Christophe
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil physicien, à finalité approfondie
Année académique : 2016-2017
URI/URL : http://hdl.handle.net/2268.2/2531

Avertissement à l'attention des usagers :

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative"(BOAI, 2002), l'utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s'en servir de données pour un logiciel, ou s'en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d'auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l'utilisateur s'engage à respecter les droits moraux de l'auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l'autorisation préalable et expresse des auteurs ou de leurs ayants droit.

Université de Liège - Faculté des Sciences Appliquées

Simulation of Geophysical Wave Propagation Using Domain Decomposition Techniques

Travail de fin d'études réalisé en vue
de l'obtention du grade de master
"Ingénieur Civil physicien"
par Anthony Royer.

Promoteur: Christophe Geuzaine.

Année Académique 2016-2017

Je tenais tout d'abord à remercier Monsieur le Professeur C. Geuzaine pour l'aide et les conseils qu'il m'a apporté tout au long de la réalisation de ce travail de fin d'étude.

Je voudrais ensuite remercier les membres du jury et plus partipulièrement MM M. Arnst, R. Boman, E. Béchet et C. Geuzaine pour leurs cours. En effet, le cours de *Modelling with partial differential equations* de MM. M. Arnst et R. Boman, ainsi que le cours *Multiphysics integrated computational project* de MM. C. Geuzaine et R. Boman, m'ont permis de trouver ma voie dans la modélisation et les méthodes numériques. Plus récemment, le projet de modélisation de M. E. Béchet a confirmé ma passion pour ces matières.

Contents

Introduction **6**

1 Schwarz methods **8**

1.1 Sequential Schwarz algorithm 8

1.2 Parallel Schwarz algorithm 9

1.3 Optimized Schwarz methods 12

1.3.1 First optimized Schwarz method 12

1.3.2 Application to the Helmholtz problem 12

1.3.3 Others interface conditions 15

1.4 Helmholtz equation using optimized Schwarz 17

1.4.1 Weak formulation 17

1.4.2 Schwarz method implementation 18

2 Mesh partitioning tool **20**

2.1 Global mesh partitioning 20

2.1.1 METIS partitioning 20

2.1.2 Implementation 21

2.2 Creation of partitioned meshes 24

2.3 Creation of partition boundaries 24

2.4 Creation of the topology structure file for GetDP 28

2.4.1 Structure of the GetDP code 28

2.4.2 The partition file 29

2.5 Files structure of a domain decomposition problem 32

2.6 Efficiency 33

2.7 Parallel version 34

3 Algorithm validation **36**

3.1 Two dimensional problem 37

3.1.1 Analytical solution 37

3.1.2 Convergence 41

3.1.3 Number of iterations 42

3.2 Three-dimensional problem 44

3.2.1 Analytical solution 44

3.2.2 Convergence 45

3.2.3 Number of iterations 46

3.3 Influence of transmission conditions 46

3.3.1 Similar shape 47

3.3.2 Not similar shape 48

3.4 Parallel scaling performance 49

4 Numerical simulation of p-wave in the ground **51**

4.1 Seismic measurement 51

4.2 Physical phenomena 52

4.2.1 Underground waves 52

4.2.2 Air wave 53

4.2.3 Surface waves 54

4.3 Numerical results 54

4.3.1 Two-dimensional simulations 54

4.3.2 Three-dimensional simulation 55

Conclusion **59**

A Propagative and evanescent modes **61**

A.1 Evanescent modes 61

A.2 Propagative modes 62

A.3 Between propagative and evanescent modes 62

B Absorbing boundary conditions **63**

B.1 Without ABC 64

B.2 Sommerfeld radiation condition 65

B.3 Bayliss–Turkel condition 66

B.3.1 First order 67

B.3.2 Second order 67

C Transmission conditions **69**

C.1 Optimal transmission operator 69

C.2 Impedance transmission conditions 70

C.3 Second order impedance transmission conditions 70

D Numerical simulations **72**

D.1 Two dimensional 72

List of Figures **76**

Bibliography **80**

Introduction

Wave problems are often encountered in several fields of physics. Acoustic, electromagnetic, seismology and mechanical waves in solids or fluids, *inter alia*. These problems can be solved using their harmonic solutions that correspond to solutions subjected to harmonic excitations. If equations are linear, these harmonic solutions constitute a basis for their corresponding problem. Finding them is therefore, important to characterize the problem.

The finite element method is often used to solve these problems because it is possible to deal with complex geometries and non-uniform material properties. To solve problems using the finite element method, the domain need to be bounded. Nevertheless, some problems are unbounded, for example, a radiation problem in the free-space. Therefore, a new boundaries are added to truncate the domain in order to solve the problem using the finite element method. New boundary conditions must be added to these new non-physical boundaries to avoid reflection. Conditions such absorbing boundary conditions (ABC) and more recently perfectly matched layers (PML) are used for this propose. Furthermore, a fairly fine mesh needs to be used to properly represent the wave behavior. In a three-dimensional problem, this can lead to a significant number of complex unknowns, especially at high frequencies. Thus, using direct sparse solvers is not suitable for these kinds of problems, while iterative solvers converge slowly or worse, diverge. Domain decomposition methods are used to overcome this problem. The idea behind these methods is to subdivide the domain into smaller subdomains to which a direct sparse solver can be applied to solve the global problem by an iterating algorithm on subdomains.

The rate of convergence of domain decomposition methods is strongly linked to transmission conditions (TC) imposed between adjacent subdomains. Therefore, it is crucial to achieve a superior design of the TC.

If the implementation is parallel, there may also be another consequence of using a domain decomposition method. Since the resolution is divided into subdomains, an efficiency parallel implementation can be built. The use of a decomposition of domain is especially advantageous for large problem (several million degrees of freedom), and may even be necessary.

The following work focuses the Helmholtz equation, which can be obtained, for instance in acoustic, fluid or uncoupled solid mechanics. The solutions are expressed by scalar complex values, while electromagnetic waves or coupled mechanical waves involve vector complex

values. Moreover, the numerical implementation is based on the GetDDM[1] package.

The work is divided in four chapters and some appendices that discuss or clarify some points. The first chapter introduces the theoretical notions relating to the Schwarz domain decomposition method. The second presents the mesh partitioning tool that is developed in the framework of this work in order to automatically partition a mesh. This tool creates the partitioned meshes files and an the topology structure. The third chapter examines the numerical efficiency of such a method effects. The final chapter presents real examples of a seismic problem.

[1] `http://onelab.info/wiki/GetDDM`

Chapter 1

Schwarz methods

Hermann Schwarz (1843 - 1921) was a German mathematician who worked on an existence and uniqueness proof of the Poisson problem applied to complex geometries. Currently, the Lax-Milgram theorem exists that provides sufficient conditions for the existence and uniqueness of a solution to a problem expressed in its weak formulation. However, this theorem was not discovered until 1954, so the only mathematical tool available to Schwarz was the Fourier transform, which can only be applied to simple geometries. Schwarz had the idea to divide a complex geometry into simple subdomains to which the Fourier transform could be applied.

This chapter reviews the history of the Schwarz algorithm to obtain the current formulation that will be used in this work.

1.1 Sequential Schwarz algorithm

The first version of the Schwarz algorithm was sequential and applied to Poisson problems. it is introduced here by considering the following problem:

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{1.1}$$

where $\Omega = \Omega_1 \cup \Omega_2$ is the domain in Figure 1.1 and u is a function from Ω to $\mathbb{R}$.

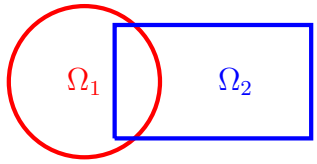

Figure 1.1: A complex geometry where the Schwarz algorithm is applied.

Once this problem is posed, the sequential Schwarz algorithm operates iteratively on u_1^n (resp. u_2^n), which corresponds to the nth iteration of the local solution on Ω_1 (resp. Ω_2) and can be expressed as:

$$\begin{aligned} -\Delta u_1^{n+1} &= f \quad \text{in } \Omega_1 \\ u_1^{n+1} &= 0 \quad \text{on } \partial\Omega_1 \cap \partial\Omega \\ u_1^{n+1} &= u_2^n \quad \text{on } \partial\Omega_1 \cap \overline{\Omega_2}, \end{aligned} \tag{1.2}$$

then,

$$\begin{aligned} -\Delta u_2^{n+1} &= f \quad \text{in } \Omega_2 \\ u_2^{n+1} &= 0 \quad \text{on } \partial\Omega_2 \cap \partial\Omega \\ u_2^{n+1} &= u_1^{n+1} \quad \text{on } \partial\Omega_2 \cap \overline{\Omega_1}. \end{aligned} \tag{1.3}$$

This algorithm is interesting because it reduces the size of the finite elements matrix, which subsequently a direct solver to be used. However, its weakness is its sequential nature; at the $(n+1)$th iteration the $(n+1)$th solution on subdomain 1 is computed and then the $(n+1)$th solution on subdomain 2 can be computed using the $(n+1)$th solution on subdomain 1, etc. To make it parallel, P.L. Lions proposed another version of the Schwarz algorithm in 1989.

1.2 Parallel Schwarz algorithm

The P.L. Lions parallel version of the Schwarz algorithm applied on N subdomains has the following form:

$$\begin{aligned} -\Delta u_i^{n+1} &= f \quad \text{in } \Omega_i \\ u_i^{n+1} &= 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega \\ u_i^{n+1} &= u_{N+1-i}^n \quad \text{on } \partial\Omega_i \cap \overline{\Omega_{N+1-i}}. \end{aligned} \tag{1.4}$$

where u_i is the local solution in subdomain i. This parallel algorithm is called the Jacobi Schwarz method (JSM) because there is a link with the Block Jacobi method used to solve linear systems (see [8] p.6).

This is a parallel Schwarz algorithm because at each iteration, every local solution can be computed simultaneously. The diagram (Figure 1.2) illustrates how such a method could be implemented to solve a Poisson problem on two subdomains using two CPUs. An other important remark is that solutions computed are local. Mathematically, two operators have to be introduce to build the global solution.

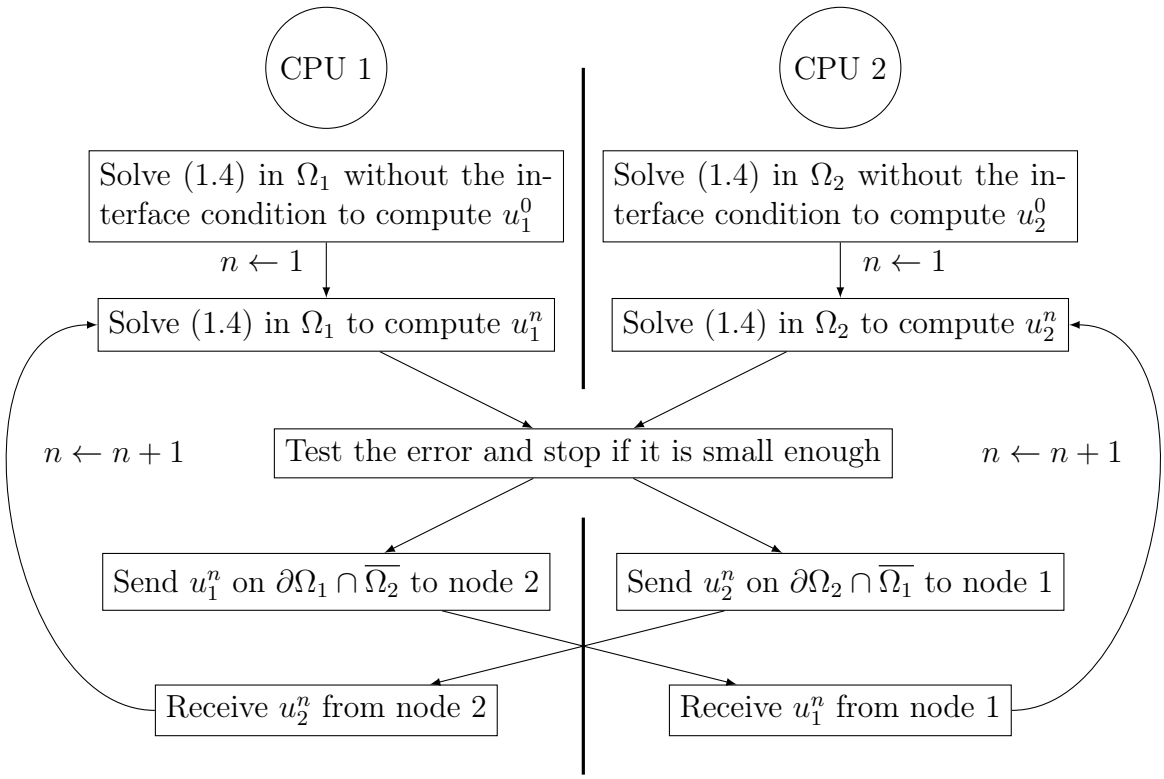

Figure 1.2: Diagram of parallel Schwarz algorithm.

The first operator is called the extension operator $E_i(u_i)$ and is defined on every subdomain. It extends the solution u_i in Ω_i to Ω by zero outside Ω_i. The second one is called the partition of unity $\chi_i(x)$ defined in Ω_i. It is equal to zero for $x = \partial\Omega_i \backslash \partial\Omega$ and it is positive anywhere else. The value of the partition of unity is defined such that:

$$u = \sum_{i=1}^{2} E_i(\chi_i u_i). \tag{1.5}$$

This partition of unity allows that the global solution is a weighted average of local solutions on every point of the domain. For example, in the domain represented in Figure 1.1, the partition of unity would be equal to:

$$\chi_1(x) = \begin{cases} 0 & \forall x \in \partial\Omega_1 \backslash \partial\Omega \\ \alpha & \forall x \in \Omega_1 \cap \Omega_2 \\ 1 & \text{otherwise} \end{cases} \text{ and, } \chi_2(x) = \begin{cases} 0 & \forall x \in \partial\Omega_2 \backslash \partial\Omega \\ 1-\alpha & \forall x \in \Omega_2 \cap \Omega_1 \\ 1 & \text{otherwise} \end{cases}, \tag{1.6}$$

where α is a real constant.

The JSM solves the local solution u_i at each iteration. Two others versions of this algorithm exist which compute a local correction v_i of the global solution u at each time step. The first one is the Restricted Additive Schwarz method (RAM) and the second is the Additive Schwarz method (ASM). In this section only the RAS method is discussed.[1]

The RAM starts by computing the local solution in the same way as JSM:

$$\begin{aligned} -\Delta u_i^0 &= f \quad \text{in } \Omega_i \\ u_i^0 &= 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega, \end{aligned} \tag{1.7}$$

next, it forms the global solution u^0 using (1.5). Subsequently, the iterative solver begins by computing the residual:

$$r^n := f + \Delta u^n \tag{1.8}$$

and uses it to compute the local correction v_i^n,

$$-\Delta v_i^n = r^n \quad \text{in } \Omega_i, \quad v_i^n = 0 \quad \text{on } \partial\Omega_i. \tag{1.9}$$

Finally, the global solution is corrected using:

$$u^{n+1} = u^n + \sum_{i=1}^{N} E_i(\chi_i v_i^n) \tag{1.10}$$

where N is the number of subdomains.

To establish that the RAS and the JSM algorithms are equivalent, it is necessary to prove that

$$u^n = \sum_{i=0}^{N} E_i(\chi_i w_i^n) \tag{1.11}$$

where w_i is the local solution given by the JSM algorithm and u^n is the global solution given by the RAS algorithm. Because the two algorithms start by resolving the decoupled problem on the subdomains, it can be assumed that:

$$u^0 = \sum_{i=0}^{N} E_i(\chi_i w_i^0) \tag{1.12}$$

Let us suppose that (1.11) is true for n and prove that it is also true for $n+1$. According to (1.10) we have

$$u^{n+1} = \sum_{k=1}^{N} E_i(\chi_i(u^n + v_i^n)) \tag{1.13}$$

The goal now is to prove that $u^n + v_i^n$ satisfies (1.4) and is equal to w_i^{n+1},

$$\begin{aligned} -\Delta(u^n + v_i^n) &= -\Delta(u^n) + r^n = -\Delta(u^n) + f + \Delta(u^n) = f \quad \text{in } \Omega_i \\ u^n + v_i^n &= u^n \quad \text{on } \partial\Omega_i \cap \overline{\Omega_{N+1-i}}. \end{aligned} \tag{1.14}$$

[1]For more information about ASM, see [8], [6] and [24]

Equation 1.14 matches perfectly with (1.2) if $u^n = u^n_{N+1-i}$ on $\partial\Omega_i \cap \overline{\Omega_{N+1-i}}$. This is the case due to the definition of the partition of unity χ_i, which is null on $\partial\Omega_i \cap \overline{\Omega_{N+1-i}}$ and furthermore, χ_{N+1-i} is equal to one on this domain. It is therefore proved that $u^n + v^n_i$ is equal to the solution of the JSM algorithm, w^{n+1}_i.

1.3 Optimized Schwarz methods

1.3.1 First optimized Schwarz method

In practice, the parallel Schwarz methods discussed above have a disadvantage because to observe a convergence, an overlap between the neighbor subdomains is necessary. For a large problem with many subdomains, this overlap can introduce a non-negligible amount of additional computation. For this reason, an optimized Schwarz version that can be applied to non-overlapping subdomains was developed by P.L. Lions. This method replaces the Dirichlet interface condition in (1.4) with a Robin condition:

$$\begin{aligned} -\Delta u^{n+1}_i &= f \quad \text{in } \Omega_i \\ u^{n+1}_i &= 0 \quad \text{on } \partial\Omega_i \cap \partial\Omega \\ \left(\frac{\partial}{\partial \mathbf{n_i}} + \alpha\right) u^{n+1}_i &= \left(\frac{\partial}{\partial \mathbf{n_i}} + \alpha\right) u^n_{N+1-i} \quad \text{on } \partial\Omega_i \cap \overline{\Omega_{N+1-i}}, \end{aligned} \tag{1.15}$$

where $\mathbf{n_i}$ is the outer normal on the boundary of the subdomain i and α is a positive constant.

1.3.2 Application to the Helmholtz problem

The Helmholtz equation represents the harmonic solutions of a wave equation. It is an elliptic partial differential equation. This equation can easily be obtained from a general wave equation, such as:

$$\frac{1}{c^2}\frac{\partial^2 u(\mathbf{x},t)}{\partial t^2} - \Delta u(\mathbf{x},t) = f(\mathbf{x},t) \tag{1.16}$$

where c is the speed of the wave and u and f are a scalar function (although they can be a vector function as well in the case of Maxwell or coupled elastic equations). Because we are interesting by the harmonic solutions, it is assumed that the solution and the excitation follow the general form:

$$u(\mathbf{x},t) = u(\mathbf{x})e^{-i\omega t} \quad \text{and,} \quad f(\mathbf{x},t) = f(\mathbf{x})e^{-i\omega t} \tag{1.17}$$

where $i^2 = -1$ and ω is the angular frequency. Note that the convention chosen uses a time dependency e^{-iwt}, while some authors have used e^{iwt}. This choice affects subsequent

developments. Substituting (1.17) for (1.16) results in:

$$-\frac{\omega^2}{c^2}u(\mathbf{x})e^{-i\omega t} - \Delta u(\mathbf{x})e^{-i\omega t} = f(\mathbf{x})e^{-i\omega t} \tag{1.18}$$

Using $k^2 = \omega^2/c^2$, ithe non-homogeneous Helmholtz equation is obtained:

$$(-\Delta - k^2)u(\mathbf{x}) = f(\mathbf{x}). \tag{1.19}$$

In a majority of the cases, $f(\mathbf{x}) = 0$ in the wave equation leads to the homogenous Helmholtz equation:

$$(-\Delta - k^2)u(\mathbf{x}) = (\Delta + k^2)u(\mathbf{x}) = 0. \tag{1.20}$$

The main difference between the optimized Schwarz method (1.15) applied to the Poisson problem and the same algorithm applied to the Helmholtz problem is the sign of the operator. In the Poisson problem, the operator is positive definite while for the Helmholtz problem it is indefinite. This implies that the finite element stiffness matrix remains symmetric, but is not longer positive definite. For example, in the case of a one-dimensional finite element method with five nodes applied to the Helmholtz problem, one can compute the stiffness matrix as follows:

$$K = \begin{pmatrix} -4.3333 & -8.1667 & 0 & 0 & 0 \\ -8.1667 & -8.6667 & -8.1667 & 0 & 0 \\ 0 & -8.1667 & -8.6667 & -8.1667 & 0 \\ 0 & 0 & -8.1667 & -8.6667 & -8.1667 \\ 0 & 0 & 0 & -8.1667 & -4.3333 \end{pmatrix}, \tag{1.21}$$

which has the following eigenvalues $(-22.2289; -14.9492; -5.8379; 1.9492; 6.4002)$. As can be observed, the eigenvalues are either positive or negative and thus, the matrix is indefinite.

Thus, it is necessary to determined whether the indefinite operator changes the convergence of the method by computing the errors $e^n_{1,2} = u^n_{1,2} - u$ induced by the Schwarz method on a homogeneous Helmholtz equation. Because $u^n_{1,2}$ and u verify the equation and the operators are linear, the error verifies (1.15) as well. The domain is the plane $\Omega = \mathbb{R}^2$ divided into two domains $\Omega_1 = \{\mathbb{R}^-, \mathbb{R}\}$ and $\Omega_2 = \{\mathbb{R}^+, \mathbb{R}\}$. The non-overlapping interface is $\Gamma = \{(x, y) \in \mathbb{R}^2 : x = 0\}$. To complete the problem definition, the incoming Sommerfeld radiation condition is added:

$$\lim_{r\to\infty} \sqrt{r}\left(\frac{\partial u}{\partial r} - iku\right) = 0 \tag{1.22}$$

where $r = \sqrt{x^2 + y^2}$.

Due to the symmetry of the problem, one should the partial Fourier transform along y to algorithm 1.15,

$$\begin{aligned} &\frac{\partial^2 \hat{e}^{n+1}_{1,2}}{\partial x^2} + (k^2 - \tilde{k}^2)\hat{e}^{n+1}_{1,2} = 0, \quad \forall (x, \tilde{k}) \in \{\Omega_{1,2}, \mathbb{R}\} \\ &\left(\pm\frac{\partial}{\partial x} + \alpha\right)\hat{e}^{n+1}_{1,2}(0, \tilde{k}) = \left(\pm\frac{\partial}{\partial x} + \alpha\right)\hat{e}^{n}_{2,1}(0, \tilde{k}), \end{aligned} \tag{1.23}$$

where $\hat{e}_{1,2}(x,\tilde{k}) = (\mathcal{F}e_{1,2})(x,\tilde{k})$. The plus sign in front of the derivative along x is applied to compute the solution on the 1st subdomain and the minus sign is applied on the 2nd subdomain. Equation 1.23 has the well-known solution:

$$\hat{e}_{1,2}^{n+1} = C_0 e^{\lambda(\tilde{k})x} + C_1 e^{-\lambda(\tilde{k})x} \tag{1.24}$$

where

$$\lambda(\tilde{k}) = \begin{cases} \sqrt{\tilde{k}^2 - k^2} & \forall\ |\tilde{k}| \geq k, \\ i\sqrt{k^2 - \tilde{k}^2} & \forall\ |\tilde{k}| < k. \end{cases} \tag{1.25}$$

Since the incoming Sommerfeld radiation condition excludes growing modes and outgoing modes, it is as follows

$$\begin{aligned} \hat{e}_1^{n+1}(x,\tilde{k}) &= \begin{cases} e_1^{n+1}(0,\tilde{k})e^{\lambda(\tilde{k})x} & \forall|\tilde{k}| \geq k \\ e_1^{n+1}(0,\tilde{k})e^{-\lambda(\tilde{k})x} & \forall|\tilde{k}| < k \end{cases} \\ \hat{e}_2^{n+1}(x,\tilde{k}) &= \begin{cases} \hat{e}_2^{n+1}(0,\tilde{k})e^{-\lambda(\tilde{k})x} & \forall|\tilde{k}| \geq k \\ \hat{e}_2^{n+1}(0,\tilde{k})e^{\lambda(\tilde{k})x} & \forall|\tilde{k}| < k \end{cases} \end{aligned} \tag{1.26}$$

A first important conclusion can be made from these results. The local error induced by the Schwarz algorithm, and thus the global error, depends only on the error that is made at the interfaces. Therefore, if the interface solutions converge, the global error also converges. From an algorithmic perspective, it is possible to apply the stop criterion only at the interface errors.

To determine the convergence of such a problem, on should compute the derivative of (1.26) at the interface:

$$\begin{aligned} \left.\frac{\partial \hat{e}_1^{n+1}}{\partial x}\right|_{x=0} &= \begin{cases} \lambda(\tilde{k})\hat{e}_1^{n+1}(0,\tilde{k}) & \forall|\tilde{k}| \geq k \\ -\lambda(\tilde{k})\hat{e}_1^{n+1}(0,\tilde{k}) & \forall|\tilde{k}| < k \end{cases} \\ \left.\frac{\partial \hat{e}_2^{n+1}}{\partial x}\right|_{x=0} &= \begin{cases} -\lambda(\tilde{k})\hat{e}_2^{n+1}(0,\tilde{k}) & \forall|\tilde{k}| \geq k \\ \lambda(\tilde{k})\hat{e}_2^{n+1}(0,\tilde{k}) & \forall|\tilde{k}| < k \end{cases} \end{aligned} \tag{1.27}$$

and plug (1.27) into the Robin condition of problem (1.23) to obtain:

$$\begin{aligned} (\pm\lambda(\tilde{k}) + \alpha)\hat{e}_1^{n+1}(0,\tilde{k}) &= -(\pm\lambda(\tilde{k}) - \alpha)\hat{e}_2^{n}(0,\tilde{k}) \\ (\pm\lambda(\tilde{k}) + \alpha)\hat{e}_2^{n}(0,\tilde{k}) &= -(\pm\lambda(\tilde{k}) - \alpha)\hat{e}_1^{n-1}(0,\tilde{k}) \end{aligned} \tag{1.28}$$

Thus, it is easy to verify that:

$$\begin{aligned} \hat{e}_1^{n+1}(0,\tilde{k}) &= \frac{(\pm\lambda(\tilde{k}) - \alpha)^2}{(\pm\lambda(\tilde{k}) + \alpha)^2}\hat{e}_1^{n-1}(0,\tilde{k}) \\ \hat{e}_2^{n+1}(0,\tilde{k}) &= \frac{(\pm\lambda(\tilde{k}) - \alpha)^2}{(\pm\lambda(\tilde{k}) + \alpha)^2}\hat{e}_2^{n-1}(0,\tilde{k}), \end{aligned} \tag{1.29}$$

where the convergence factor appears,

$$\rho(\tilde{k}) = \begin{cases} \left| \dfrac{\lambda(\tilde{k}) - \alpha}{\lambda(\tilde{k}) + \alpha} \right| & \forall |\tilde{k}| \geq k \\ \left| \dfrac{\lambda(\tilde{k}) + \alpha}{\lambda(\tilde{k}) - \alpha} \right| & \forall |\tilde{k}| < k \end{cases} \tag{1.30}$$

Figure 1.3 displays the convergence factor in the function of the reduced Fourier number ($s = \tilde{k}/k$). Therefore, the algorithm proposed by P.L. Lions, which use a real value for α, is not useful for the Helmholtz problem. Indeed, the convergence factor is equal to one for the propagative modes (see Appendix A) and thus, the solution will not converge.

To conclude, more sophisticated interface conditions need to be used when applying a non-overlapping domain decomposition method to the Helmholtz problem.

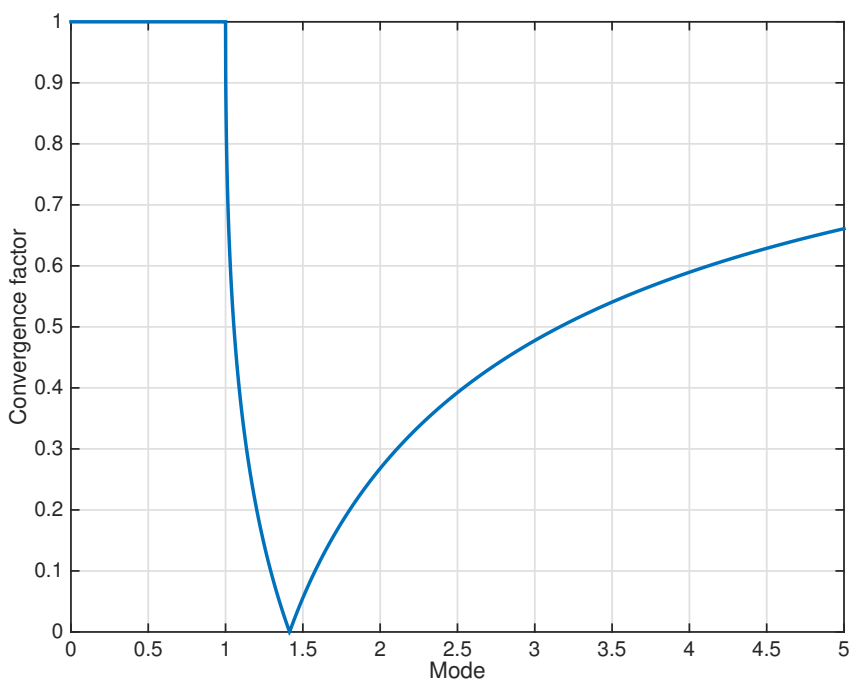

Figure 1.3: Convergence factor of the problem (1.15) for the Helmholtz equation applied on two partitions dividing the plane ($\alpha = k$).

1.3.3 Others interface conditions

Because a more complex operator is needed to solve the Helmholtz problem, the real number α is replaced by a more general one. Using the same notation from paper[23], the new

algorithm can be rewritten as follows:

$$\begin{aligned} -\Delta u_i^{n+1} &= f \quad \text{in } \Omega_i \\ u_i^{n+1} &= 0 \quad \text{on } \partial\Omega_i \cup \partial\Omega \\ \left(\frac{\partial}{\partial \mathbf{n_i}} + \mathcal{S}\right) u_i^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n_i}} + \mathcal{S}\right) u_{N+1-i}^{n} \quad \text{on } \partial\Omega_i \cup \overline{\Omega_{N+1-i}}, \end{aligned} \tag{1.31}$$

where $\mathcal{S}$ is the operator. Many operators have been proposed (see Appendix C) such as the evanescent modes damping algorithm [4] that uses,

$$\mathcal{S} = -ik + \chi \tag{1.32}$$

where $i^2 = -1$, k is the wave number and χ is a real constant. The convergence factor with this operator is presented in Figure 1.4. Using this kind of operator, an adequate convergence factor appears for the propagative and evanescence modes, except for $s = 1$ where the convergence factor is equal to one. Nevertheless, it can be concluded that this parallel Schwarz algorithm converges for the Helmholtz problem when the appropriate transmission condition is used.

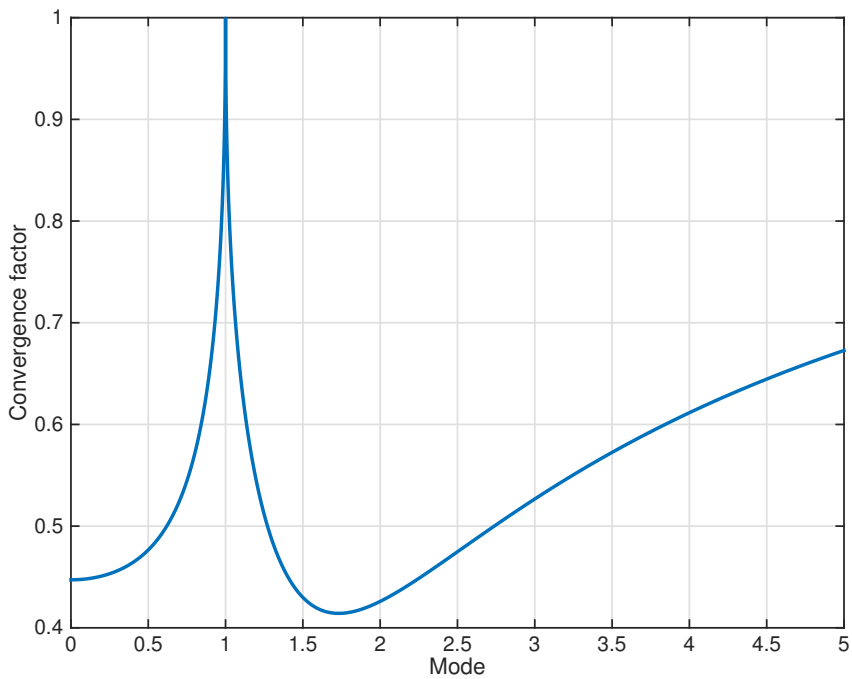

Figure 1.4: Convergence factor of the problem (1.31) using the transmission operator (1.32) with $\mathcal{S} = -ik + k$ in the case of the Helmholtz equation applied on two partitions dividing the real plane.

1.4 Helmholtz equation using optimized Schwarz

1.4.1 Weak formulation

Using all previous discussions, the Helmholtz problem can be formulated. To remain general, the Helmholtz problem is applied to a volume Ω partitioned in N subdomains called Ω_i. The boundaries of Ω_i are divided into three types:

- $\Gamma_{i,d}$ where Dirichlet conditions are applied;
- $\Gamma_{i,n}$ where Neumann conditions are applied;
- $\Gamma_{i,Inf}$, the non-physical boundary used to truncate an infinite domain where absorbing conditions are imposed (see Appendix B).

In addition, the boundary interfaces between subdomains i and j are called Σ_{ij}, to which a transmission condition is applied.

With this geometry in mind, the Helmholtz problem using an optimized Schwarz method can be expressed as:

$$\begin{aligned}
(\Delta + k^2)u_i^{n+1}(\mathbf{x}) &= 0 \quad \text{in, } \Omega_i \\
u_i^{n+1}(\mathbf{x}) &= f(\mathbf{x}) \quad \text{on, } \Gamma_{i,d} \\
\partial_{\mathbf{n}} u_i^{n+1}(\mathbf{x}) &= e(\mathbf{x}) \quad \text{on, } \Gamma_{i,n} \\
\partial_{\mathbf{n_i}} u_i^{n+1}(\mathbf{x}) + \mathcal{B} u_i^{n+1}(\mathbf{x}) &= 0 \quad \text{on, } \Gamma_{i,Inf} \\
\partial_{\mathbf{n_i}} u_i^{n+1}(\mathbf{x}) + \mathcal{S} u_i^{n+1}(\mathbf{x}) &= g_{ij}^n \quad \text{on, } \Sigma_{ij}
\end{aligned} \tag{1.33}$$

where $f(\mathbf{x})$ and $e(\mathbf{x})$ are the Dirichlet and Neumann conditions and g_{ij}^n is the interface unknown on Σ_{ij}, which can be updated according to:

$$g_{ji}^{n+1} = -g_{ij}^n + 2\mathcal{S} u_i^{n+1}(\mathbf{x}) \quad \text{on, } \Sigma_{ij}. \tag{1.34}$$

To implement problems (1.33) and (1.34) using a finite element method, their weak formulations are needed. The weak formulations of the volume and the interface problems are as follows:

- Find $u_i^{n+1}(\mathbf{x})$ in $H_0^1(\Omega_i)$ such that,

$$\begin{aligned}
&\int_{\Omega_i} \nabla u_i^{n+1}(\mathbf{x}) \cdot \nabla v_i(\mathbf{x}) - k^2 u_i^{n+1}(\mathbf{x}) v_i(\mathbf{x})\, d\Omega_i + \int_{\Gamma_{i,Inf}} \mathcal{B} u_i^{n+1}(\mathbf{x}) v_i(\mathbf{x})\, d\Gamma_{i,Inf} \\
&\quad - \int_{\Gamma_{i,n}} e(\mathbf{x}) v_i(\mathbf{x})\, d\Gamma_{i,n} + \sum_j \int_{\Sigma_{ij}} \mathcal{S} u_i^{n+1}(\mathbf{x}) v_i(\mathbf{x})\, d\Sigma_{ij} = \sum_j \int_{\Sigma ij} g_{ij}^n(\mathbf{x}) v_i(\mathbf{x})\, d\Sigma_{ij}
\end{aligned} \tag{1.35}$$

for all v_i in $H_0^1(\Omega_i)$.

- Find $g_{j,i}^{n+1}(\mathbf{x})$ in $H_0^1(\Sigma_{ij})$ such that,

$$\int_{\Sigma_{ij}} g_{ji}^{n+1}(\mathbf{x}) h_{ji}(\mathbf{x})\, d\Sigma_{ij} = -\int_{\Sigma_{ij}} g_{ij}^{n}(\mathbf{x}) h_{ji}(\mathbf{x})\, d\Sigma_{ij} - \int_{\Sigma_{ij}} \mathcal{S} u_i^{n+1}(\mathbf{x}) h_{ji}(\mathbf{x})\, d\Sigma_{ij} \quad (1.36)$$

for all $h_{j,i}$ in $H_0^1(\Sigma_{ij})$.

It is important that the absorbing boundary condition and the transmission condition possess the form presented in Equation 1.33, because it can be strongly enforced in the weak formulation.

1.4.2 Schwarz method implementation

Following the same idea used to find the RAS algorithm (1.10), the solution to problem 1.33 can be decomposed in two terms $u_i^{n+1} = v_i^{n+1} + w_i^{n+1}$. This decomposition is allowed by the superposition principle, which is a consequence of the linearity of the problem. The first term is the solution in which the interface's unknown is set to zero which leads to:

$$\begin{aligned}
(\Delta + k^2) v_i^{n+1}(\mathbf{x}) &= 0 \quad \text{in, } \Omega_i \\
v_i^{n+1}(\mathbf{x}) &= f(\mathbf{x}) \quad \text{on, } \Gamma_{i,d} \\
\partial_{\mathbf{n}} v_i^{n+1}(\mathbf{x}) &= e(\mathbf{x}) \quad \text{on, } \Gamma_{i,n} \\
\partial_{\mathbf{n_i}} v_i^{n+1}(\mathbf{x}) + \mathcal{B} v_i^{n+1}(\mathbf{x}) &= 0 \quad \text{on, } \Gamma_{i,Inf} \\
\partial_{\mathbf{n_i}} v_i^{n+1}(\mathbf{x}) + \mathcal{S} v_i^{n+1}(\mathbf{x}) &= 0 \quad \text{on, } \Sigma_{ij}.
\end{aligned} \quad (1.37)$$

This term is independent on the iteration and can be written as v_i. The second term corresponds to the problem without boundary conditions, except the interface's unknown,

$$\begin{aligned}
(\Delta + k^2) w_i^{n+1}(\mathbf{x}) &= 0 \quad \text{in, } \Omega_i \\
w_i^{n+1}(\mathbf{x}) &= 0 \quad \text{on, } \Gamma_{i,d} \\
\partial_{\mathbf{n}} w_i^{n+1}(\mathbf{x}) &= 0 \quad \text{on, } \Gamma_{i,n} \\
\partial_{\mathbf{n_i}} w_i^{n+1}(\mathbf{x}) + \mathcal{B} w_i^{n+1}(\mathbf{x}) &= 0 \quad \text{on, } \Gamma_{i,Inf} \\
\partial_{\mathbf{n_i}} w_i^{n+1}(\mathbf{x}) + \mathcal{S} w_i^{n+1}(\mathbf{x}) &= g_{ij}^{n} \quad \text{on, } \Sigma_{ij}.
\end{aligned} \quad (1.38)$$

According to Equation 1.34, the interface's unknown is given by:

$$\begin{aligned}
g_{ji}^{n+1} &= -g_{ij}^{n} + 2\mathcal{S} u_i^{n+1}(\mathbf{x}) \\
&= -g_{ij}^{n} + 2\mathcal{S} w_i^{n+1}(\mathbf{x}) + 2\mathcal{S} v_i(\mathbf{x}),
\end{aligned} \quad (1.39)$$

which can be rewritten by introducing a new operator $\mathcal{A}$:

$$g_{ji}^{n+1} = \mathcal{A} g_{ij}^{n} + 2\mathcal{S} v_i(\mathbf{x}). \quad (1.40)$$

This expression can be compared to the Jacobi method used to solve a linear system such as $Ax = b$, where A is a square matrix of size m, x is the unknown vector of size m and b is the right-hand-side of size m. First, the matrix A is decomposed by $A = M - N$, where M is an easily inversed matrix (for example a diagonal matrix). Using this decomposition, the linear problem can be written as:

$$\begin{aligned} Mx &= Nx + b \\ x &= M^{-1}Nx + M^{-1}b \end{aligned} \tag{1.41}$$

The Jacobi method proposes to solve such a linear problem iteratively according to,

$$\begin{cases} x_0 = 0 \\ x_{n+1} = M^{-1}Nx_n + M^{-1}b \end{cases}, \tag{1.42}$$

which converges if the spectral radius of the matrix $M^{-1}N$ is smaller than one.

Indeed, if $\mathcal{A}$ is such that its spectral radius is smaller than one, which in fact depends on the choice of the transmission operator, Equation 1.40 is a one step of a Jacobi method that corresponds to the linear system:

$$(\mathcal{I} - \mathcal{A})g = 2\mathcal{S}v_i(\mathbf{x}), \tag{1.43}$$

where $\mathcal{I}$ is the identity operator. This kind of problem can be solved using every iterative solver.

Finally, the Schwarz domain decomposition method, as implemented in the GetDDM package, is formulated by:

i Compute the initial solution v_i without the transition condition according to (1.37);

ii Iterate until the convergence is reached on the following problem:

  1 Compute w_i^{n+1} using Equation 1.38;

  2 Use Equation 1.40 to compute g^{n+1}, the set of interface unknowns.

iii Compute the final solution $u_i = v_i + w_i^{n+1}$.

Chapter 2

Mesh partitioning tool

To implement a parallel Schwarz algorithm on a domain Ω, the algorithm needs to be divided into subdomains Ω_i and the topology must to be known. For example, the boundary Γ_{ij} between subdomains i and j need to be identified. This is the purpose of the mesh partitioning tool developed in this work. Thus, the partitioning tool performs:

- the decomposition of a global mesh file written in the Gmsh[1] format into other mesh files that contain the subdomain meshes;
- the creation of the new entities that correspond to the boundaries of the subdomains;
- the creation of a file that contains the topology of the problem to be used by the finite element solver GetDP.[2]

This chapter presents several algorithms and uses a pseudo-code to clarify the lecture. When actually implemented, the C++ language is used.

2.1 Global mesh partitioning

2.1.1 METIS partitioning

METIS is a set of functions used for partition a graph or a mesh[19]. It was developed in 1995 by the Karypis Lab at the University of Minnesota.

[1]Gmsh is a free 3D finite element grid generator developed by the C.Geuzaine and J.F. Remacle[12].

[2]GetDP is a free finite element solver developed by P. Dular and C. Geuzaine at the University of Liège[10].

The partitioning algorithm, as described in [16], [17] and [18], works in three steps. The first step uses an initial graph as input and creates iteratively smaller graphs by collapsing the set of adjacent vertices into a single vertex. When the number of vertices reaches in the hundreds, the next step begins. These hundreds of vertices are divided into N sets, where N is the desired number of partitions. This step uses a heuristic method. Due to the small number of vertices, the partitioning occurs quickly. The last step consists of iteratively re-forming the graph by performing the inverse of the first phase. After each iteration, an optimization process is conducted at the boundaries of the partitions to smooth them. Figure 2.1 illustrates a schematic representation of this algorithm.

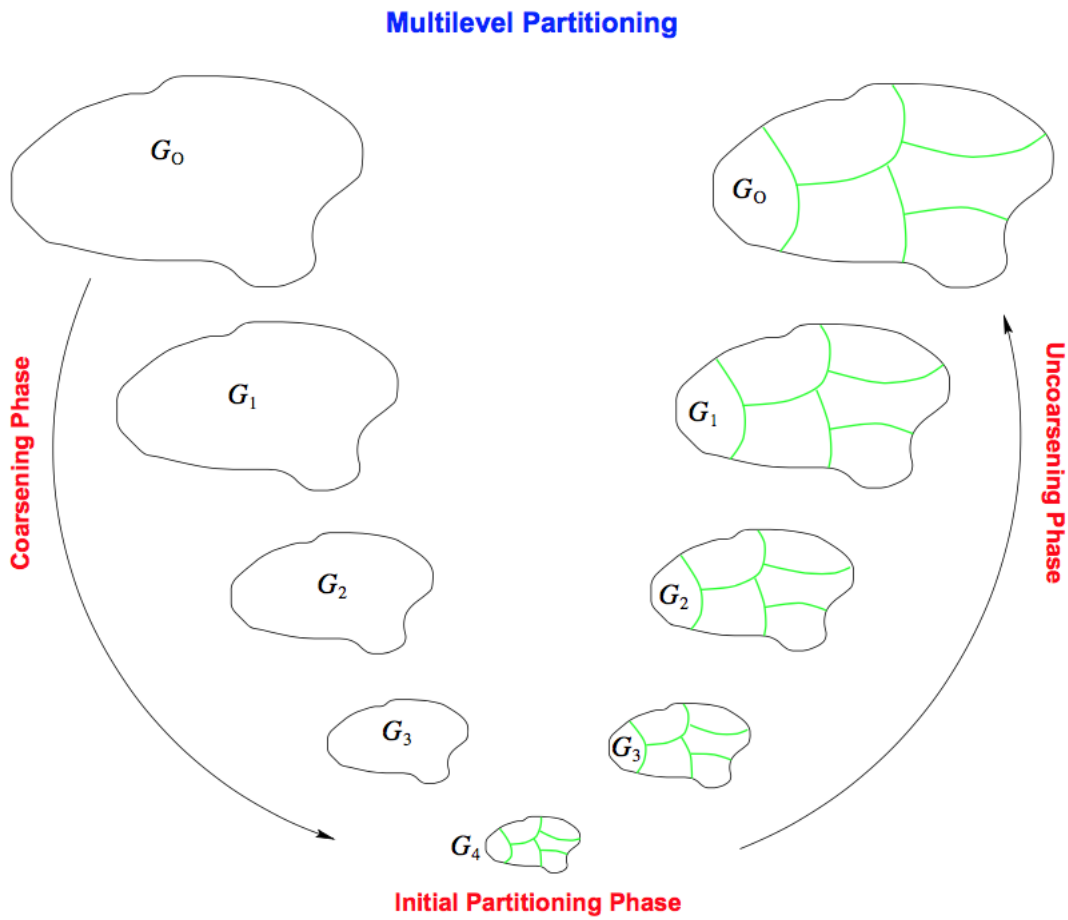

Figure 2.1: The METIS partitioning algorithm (image of George Karypis that comes from [19]).

2.1.2 Implementation

The METIS library can directly partition a mesh. The format that it uses to represent a mesh is composed by two arrays of integers, `eptr` and `eind`. The `eptr` has a length equal to

the number of elements in the mesh plus one. For the element `n`, `eind[eptr[n]]` contains the node's tag of the first node of element `n`. The following integers of `eind[]` represent the tag of the others nodes of element `n` until `eind[eptr[n+1]]` is reached, which corresponds to the first node of the next element. Figure 2.2 displays an example of a mesh that corresponds to the following arrays:

```
int eptr[] = {0,4,7};
int eind[] = {1,2,3,4,2,3,4};
```

Listing 2.1: eptr and eind format.

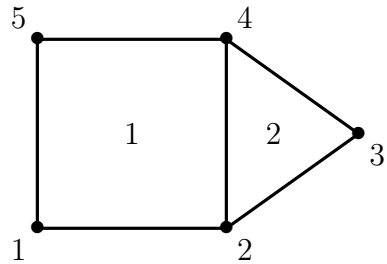

Figure 2.2: Example of mesh that corresponds to the METIS mesh structure in Listing 2.1.

In the present case, the mesh structure is included in a `GModel` class provided by the Gmsh library. Therefore, the first step uses this class to build the pair of arrays necessary to METIS. The function that accomplishes this is called `GModelToGraph`. The implemented algorithm is presented in Listing 2.2.

First, a map that contains the correspondence between the tags of the elements and the tags of their corresponding nodes is filled; we loop over all the geometrical entities that are contained in the `GModel` class (regions, faces, edges and vertices). Inside an entity, the function `elementsToNodes` is used to fill the map with a particular kind of element. Finally, this map is used to create the METIS mesh structure. The `eptr` array is created by counting the number of nodes possessed by element `i` and assign this value to `eptr[i+1]`. Subsequently, the `eind` array is filled by assigning all the node's tags contained in the map.

```
void GModelToGraph(GModel gModel, int eptr[], int eind[])
{
    //multimap that store the all element's tags with their corresponding node
    's tags
    multimap<int, int> elementsToNodes;

    //Loop over regions
    for reg overAll regions in gModel
    {
        fillElementsToNodes(elementsToNodes, all tetrahedra in reg);
        fillElementsToNodes(elementsToNodes, all hexahedra in reg);
        fillElementsToNodes(elementsToNodes, all prisms in reg);
        fillElementsToNodes(elementsToNodes, all pyramids in reg)
        fillElementsToNodes(elementsToNodes, all trihedra in reg)
        fillElementsToNodes(elementsToNodes, all polyhedra in reg);
    }
```

```
    //Loop over faces
    for fac overAll faces in gModel
    {
        fillElementsToNodes(elementsToNodes, all triangles in fac);
        fillElementsToNodes(elementsToNodes, all quadrangles in fac));
        fillElementsToNodes(elementsToNodes, all polygons in fac));
    }

    //Loop over edges
    for edg overAll edges in gModel
    {
        fillElementsToNodes(elementsToNodes, all lines in edg);
    }

    //Loop over vertices
    for ver overAll verticies in gModel
    {
        fillElementsToNodes(elementsToNodes, all points in ver);
    }

    //create mesh format for METIS
    int numVerticesByElm = 0;
    int i = 0;
    eptr[0] = 0;

    for iterator overAll elementsToNodes
    {
        //Count the number of nodes that have each elements
        numVerticesByElm = (count(firstValue of iterator) of elementsToNodes);
        eptr[i+1] = numVerticesByElm;
        i++;
    }

    int i = 0;
    for iterator overAll elementsToNodes
    {
        eind[i] = (secondValue of iterator);
        i++;
    }
}

void fillElementsToNodes(multimap<int, int> elementsToNodes, setOf elements)
{
    for iterator overAll elements
    {
        int tag = (getNum() of iterator);

        for j = 0:1:(getNumVertices() of iterator)
        {
             MVertex vertex = (getVertex(j) of iterator);
             add(getNum() of vertex) to elementsToNodes[tag];
        }
    }
```

}

Listing 2.2: Creation of the pair of arrays for METIS partitioning.

The second step uses the function `METIS_PartMeshDual` to partition the mesh. The input is:

- the number of elements contained in the global mesh;
- the number of nodes contained in the global mesh;
- the METIS mesh structure `eptr` and `eind`;
- two parameters used if all nodes are not equally important (this is not used);
- the number of nodes that two elements must have in common to create an edge between them in the graph structure (this value is set equal to the dimension of the global mesh);
- the number of partitions to create;
- some other options.

The output of this function is an array that associates the element's tags with its partition number.

2.2 Creation of partitioned meshes

At this stage, the partitioned meshes can be built. The algorithm is quite simple. We loop over all the elements of the global mesh and we assign this element to the correct partitioned mesh according to the output array provided by the METIS function. The function performed in this step is called `createNewModels`.

The function contains other processes to keep or create new physical entities. Although they are not detailed here, it is important to note that a subdomain `i` is included in one of the physical entities named `_Omega_i`.

2.3 Creation of partition boundaries

The last step creates the partition boundaries. First, it is necessary to illustrate a partition boundary using an example. If a two-dimensional mesh is cut into three subdomains,

then the partitioned mesh has three mesh classes (GModel) that contain the elements corresponding to their partition and boundaries that already existed in the global mesh. In this context, the partition boundaries are the new boundaries that appear between each subdomain. These boundaries do not have physical meaning; they are only of significant to the domain decomposition algorithm. Figure 2.3 illustrates this example.

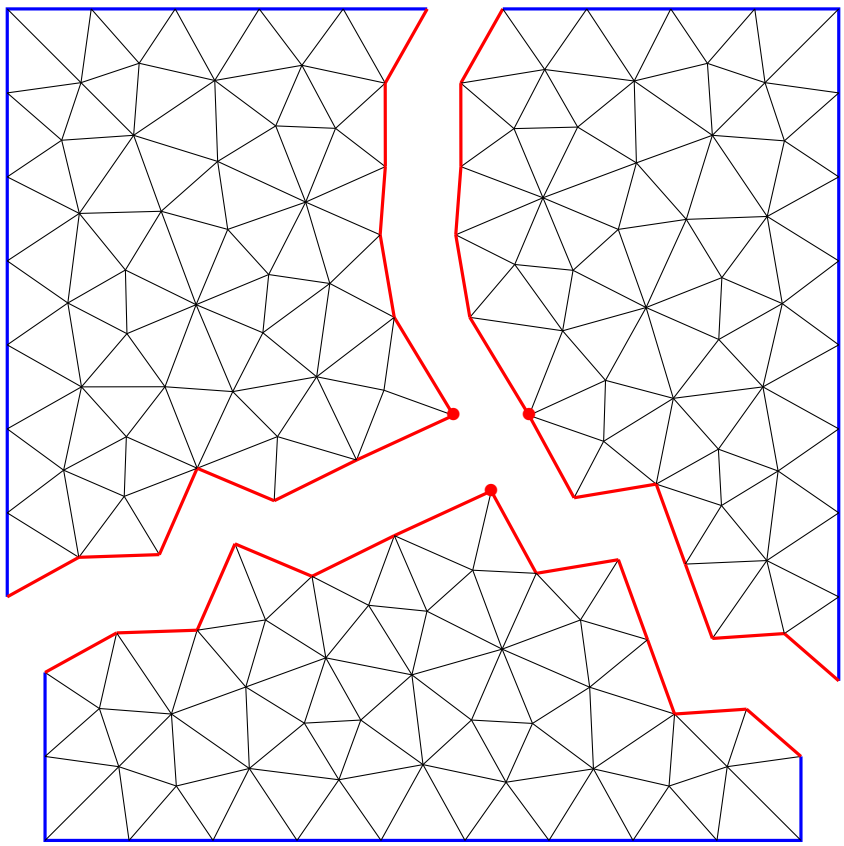

Figure 2.3: Three subdomains where the thick blue lines are the physical existing boundaries and the red ones are the partition boundaries.

The algorithm developed to create these new boundaries works in two steps. It has been implement in function `createPartitionBoundaries`. The first one creates three maps that contain element boundaries (one map for faces, another for edges and the last for vertices) and adjacent elements. For example, using the same mesh as Figure 2.2, the edge map will contain the edge between node 2 and node 4 and will associate to this edge the elements 1 and 2, because they are neighbors to this edge. Listing 2.3 explains how this algorithm is coded for the edge map in this case. The algorithm is similar for the faces and the vertices maps.

```
int createPartitionBoundaries(GModel model)
{
  map<MEdge, vector<MElement> > edgeToElement;

    if(meshDim == 3)
    {
```

```
        //We loop over all region entities that is contained in the model
    corresponding to the global mesh.
        for it overAll regions in model
        {
            fillit_(edgeToElement, all tetrahedra in it);
            fillit_(edgeToElement, all hexahedra in it);
            fillit_(edgeToElement, all prisms in it);
            fillit_(edgeToElement, all pyramids in it);
            fillit_(edgeToElement, all trihedra in it);
            fillit_(edgeToElement, all polyhedra in it);
        }
    }

    if(meshDim == 2)
    {
        for it overAll faces in model
        {
            fillit_(edgeToElement, all triangles in it);
            fillit_(edgeToElement, all quadrangles in it);
            fillit_(edgeToElement, all polygons in it);
        }
    }
}

void fillit_(map<MEdge, vector<MElement> > edgeToElement, setOf elements)
{
  for iterator overAll elements
    {
        MElement theElement = iterator;

        for j = 0:1:(getNumEdges() of theElement)
        {
             MEdge edge = (getEdge(j) of theElement);
             add(theElement) to edgeToElement[edge];
        }
    }
}
```

Listing 2.3: Creation of the edgeToElement map.

The second step loops over all boundaries of the elements (faces, then edges and finally vertices) and calls the function `assignPartitionBoundary` that creates the new boundaries.

First, this function loops over all elements of a boundary type and verifies the partition number of these elements. If all the elements belong to the same partition, the boundary of the elements is not a boundary of the subdomain and therefore, the function stops. If not, the function continues and creates a partition entity of a geometrical degree greater that of the boundary we tried to create and determines if whether this partition entity has already been created. If it exists, the function stops because this boundary is included in a subdomain boundary of a higher degree. If none of them exist, the new partition entity can be created. A new mesh element is created and is associated with the new or the existing partition entity.

At this stage, particular attention is paid on the orientation of the boundaries. Their normals are imposed externally and tangents are imposed in accordance with the normals. Listing 2.4 presents the algorithm of the `assignPartitionBoundary` for the edge boundaries.

```
void assignPartitionBoundary(GModel *model, MEdge currentEdge, set<
    partitionEdge> pedges, vector<MElement> elements, set<partitionFace> pfaces
    )
{
    vector<int> elementsPartition;
    add(getPartition() of elements[0]) to elementsPartition;

    for i =1:1:(size() of elements)
    {
        bool found = false;
        for j =0:1:(size() of elementsPartition)
        {
            if (getPartition() of elements[i] == elementsPartition[j])
            {
                found = true;
                break;
            }
        }

        if (!found)
        {
          add(getPartition() of elements[i]) to elementsPartition;
        }
    }

    //If there is less than two partitions touching the edge: stop
    if (size() of elementsPartition < 2)
    {
        return;
    }

    //Creation of a partition face having elementsPartition has partition
    numbers
    partitionFace pf(model, 1, elementsPartition);

    //If the edge is in a partitionFace (pfaces contains all the partition
    faces that have been created before)
    if (find(pf) in pfaces == true)
    {
        return;
    }

    //Creation of the partition edge having elementsPartition has partition
    numbers
    partitionEdge pe(model, 1, elementsPartition);

    partitionEdge ppe;
    if (find(pe) in pedges == false)
    {
        ppe = partitionEdge(model, -(int)pedges.size()-1, elementsPartition);
```

```
        add ppe in pedges;
        add ppe in model;
    }
    else
    {
        ppe = pedges[pe];
    }

    //Creation of a new mesh line
    MLine line(getVertex() of currentEdge);
    //and assign them to the new entity if pe is not find in pedges or to the
   existing entity if it is.
    add line to ppe;
}
```

Listing 2.4: Creation of a new edge boundary.

2.4 Creation of the topology structure file for GetDP

2.4.1 Structure of the GetDP code

The domain decomposition algorithm is coded using five GetDP `.pro` files:

`SchwarzMacros.pro` contains several useful macros, such as a macro to compute the artificial source; namely, the interface condition terms.

`Schwarz.pro` implements the resolution using the Schwarz algorithm, as described in Chapter 1.

`Helmholtz.pro` codes the weak formulation of the Helmholtz problem with the definition of its function space and the post operations that are useful for this kind of problem. This file depends on the type of problem studied. Other files can be used like `Elasticity.pro` for mechanical waves or `Maxwell.pro` for electromagnetic waves. Listing 2.5 displays an example of the formulation of the following Helmholtz problem in the GetDP language.

$$\begin{aligned}(\Delta + k^2)u(\mathbf{x}) &= 0 \quad \text{in } \Omega_i \\ \partial_{\mathbf{n}} u(\mathbf{x}) &= -iku(\mathbf{x}) \quad \text{on } \Gamma_{Inf,i} \\ u(\mathbf{x}) &= g(\mathbf{x}) \quad \text{on } \Gamma_{Dir,i} \\ \partial_{\mathbf{n}} u(\mathbf{x}) &= -iku(\mathbf{x}) \quad \text{on } \Sigma_i.\end{aligned} \tag{2.1}$$

`partition.pro` is the file that is created by the partitioning tool which contains the topology description and the work distribution rules for each CPU.

`xxxxxx.pro` (where `xxxxxx` can be replaced by any name) is the file that contains the specific data relevant to our problem, like the Dirichlet of Neumann conditions, the domain constants (the conductivity, the wave number, the permittivity, ...). This file is exceedingly dependent on the geometry of the studied problem.

```
//Volume terms
Galerkin { [Dof{Grad u~{i}}, {Grad u~{i}} ];
    In Omega~{i}; Jacobian JVol; Integration I1; }
Galerkin { [ - k[]^2 * Dof{u~{i}}, {u~{i}} ];
    In Omega~{i}; Jacobian JVol; Integration I1; }
//Sommerfeld radiation conditions
Galerkin { [ - I[] * k[] * Dof{u~{i}}, {u~{i}} ];
    In GammaInf~{i}; Jacobian JSur Integration I1; }
//Transmission conditions
Galerkin { [ - I[] * k[] * Dof{u~{i}} , {u~{i}} ];
    In Sigma~{i}; Jacobian JSur; Integration I1; }
```

Listing 2.5: Formulation of the Helmholtz problem in the GetDP language where `u~{i}` is the solution, `k[]` the wave number, `I[]` the imaginary number, `Omega~i` the subdomain `i`, `GammaInf~i` the boundary of the subdomain `i` where the Sommerfeld condition holds and `Sigma~i` the boundaries of the subdomain `i` where the TC are applied.

2.4.2 The partition file

As previously noted, the partition file (`partition.pro`) contains the definition of the partitioned topology (which is placed in a GetDP `Group` section) and the work distribution rules (which are placed in a GetDP `Function` section). This file is automatically generated by the partitioning tool.

The `Group` section

The `Group` section specifies the physical groups. There are eight kinds of physical groups that must be defined:

`Omega_i` defines the subdomain volume `i`.

`Sigma_i_j` defines the boundary between subdomain `i` and `j`. Note that if `Sigma_i_j` is defined, `Sigma_j_i` must also be defined even if it is the same.

`Sigma_i` defines all transmission boundaries of subdomain `i`.

`BndSigma_i_j` are boundaries of the transmission boundary between subdomain `i` and `j`. For example, it is a line that separates two subdomains. The points at which the line ends are the boundaries of sigma.

`BndGammaInf_i_j` are boundaries of the transmission boundary between subdomain `i` and `j` that coincide with the boundary `GammaInf`, where the Sommerfeld radiation conditions hold.

`BndGammaInf_i` are sets of `BndGammaInf_i_j` for all `j`.

`BndGammaD_i_j` are boundaries of the transmission boundary between subdomain `i` and `j` that coincide with the boundary `GammaD`, where the Dirichlet conditions hold.

`BndGammaD_i` are sets of `BndGammaD_i_j` for all `j`.

In addition to these physical groups, an array `D()` contains all subdomains, some arrays `D_i()` contain all neighbor subdomains to subdomain i, and the number of subdomains `N_DOM` refers to the size of the array texttt D().

Listing 2.6 displays the `Group` section generated by the partitioning tool for the partition in Figure 2.3. Numbers in the `Region` object are the tag of the physical regions defined in the geometry.

```
Group{
  Omega_1 = Region[{3}];
  Omega_2 = Region[{2}];
  Omega_0 = Region[{1}];

  Sigma_1_2 = Region[{6}];
  Sigma_2_1 = Region[{6}];
  BndSigma_1_2 = Region[{7}];
  BndSigma_2_1 = Region[{7}];
  BndGammaInf_1_2 = Region[{}];
  BndGammaInf_2_1 = Region[{}];
  BndGammaD_1_2 = Region[{}];
  BndGammaD_2_1 = Region[{}];
  BndGammaInf_1 = Region[{}];
  BndGammaInf_2 = Region[{}];
  BndGammaD_1 = Region[{}];
  BndGammaD_2 = Region[{}];
  Sigma_1_0 = Region[{5}];
  Sigma_0_1 = Region[{5}];
  BndSigma_1_0 = Region[{7}];
  BndSigma_0_1 = Region[{7}];
  BndGammaInf_1_0 = Region[{}];
  BndGammaInf_0_1 = Region[{}];
  BndGammaD_1_0 = Region[{}];
  BndGammaD_0_1 = Region[{}];
  BndGammaInf_1 = Region[{}];
  BndGammaInf_0 = Region[{}];
  BndGammaD_1 = Region[{}];
  BndGammaD_0 = Region[{}];
  Sigma_2_0 = Region[{4}];
  Sigma_0_2 = Region[{4}];
  BndSigma_2_0 = Region[{7}];
  BndSigma_0_2 = Region[{7}];
```

```
  BndGammaInf_2_0 = Region[{}];
  BndGammaInf_0_2 = Region[{}];
  BndGammaD_2_0 = Region[{}];
  BndGammaD_0_2 = Region[{}];
  BndGammaInf_2 = Region[{}];
  BndGammaInf_0 = Region[{}];
  BndGammaD_2 = Region[{}];
  BndGammaD_0 = Region[{}];

  Sigma_1 = Region[{5, 6}];
  Sigma_2 = Region[{4, 6}];
  Sigma_0 = Region[{4, 5}];

  BndSigma_2 = Region[{7}];
  BndSigma_1 = Region[{7}];
  BndSigma_0 = Region[{7}];

  D() = {0, 1, 2};
  N_DOM = #D();
  D_0 = {1, 2};
  D_2 = {1, 0};
  D_1 = {2, 0};
}
```

Listing 2.6: Group section for the partition shown in Figure 2.3.

The `Function` section

This section defines three important arrays. The first one is called `myD()` and it contains subdomains of which each CPU is in charge. The second and the last one are called `ListOfFields()` and `ListOfConnectedFields()`, respectively and contain a list of tags that correspond to a set of boundary transmissions; for example, if CPU `i` has a subdomain `i` which is connected to another subdomain `j` owned by the CPU `j`. The boundary transmission `Sigma_i_j` is associated `tag_g_i_j` on CPU `i` and with `tag_g_j_i`, on CPU `j`. Using these arrays, CPU `i` knows that it has to compute the artificial source on `Sigma_i_j` and associated its to `tag_g_i_j` and retrieve the artificial source `tag_g_i_j` that comes from CPU `j`. Equation uses to create the tag is (2.2). It is supposed that a subdomain can have more than one 1,000 neighbor subdomains.

$$\texttt{tag_g_i_j} = 1000i + j \tag{2.2}$$

Listing 2.7 displays the `Function` section generated by the partitioning tool shown in Figure 2.3.

```
Function {
  myD = {} ; // the domains that I'm in charge of
  myD_0 = {};
  myD_1 = {};
```

```
  myD_2 = {};
  ListOfFields = {};
  ListOfConnectedFields = {};

  For idom In {0:N_DOM-1}
    If (idom % MPI_Size == MPI_Rank)
      myD() += D(idom);
      myD~{idom} += D~{idom}();
    EndIf
  EndFor
  For ii In {0:#myD()-1}
    i = myD(ii);
    If(#myD~{i}() == 2)
      Printf("We can do sweeping!");
    EndIf
    For jj In {0:#myD~{i}()-1}
      j = myD~{i}(jj);

      tag_g~{i}~{j} = i * 1000 + j;
      tag_g~{j}~{i} = j * 1000 + i;

      ListOfFields() += tag_g~{i}~{j};
      ListOfConnectedFields() += 1;
      ListOfConnectedFields() += tag_g~{j}~{i};
      If(ANALYSIS == 0)
        g_in~{i}~{j}[ Sigma~{i}~{j} ] = ComplexScalarField[XYZ[]]{ tag_g~{j}~{
    i} };
      EndIf
      If(ANALYSIS == 1)
        g_in~{i}~{j}[ Sigma~{i}~{j} ] = ComplexVectorField[XYZ[]]{ tag_g~{j}~{
    i} };
      EndIf
    EndFor
  EndFor
}
```

Listing 2.7: Function section for the partition shown in Figure 2.3.

2.5 Files structure of a domain decomposition problem

The file structure of a domain decomposition problem is presented in Figure 2.4. This structure remains the same for every problem and is composed of three types of files:

- The blue one, which contains the structure of the Schwarz algorithm and the definition of the problem type. Depending on the problem, we have to choose between `Helmoltz.pro`, `Elasticity.pro` and `Maxwell.pro`.
- The red one, which contains the files generated by the partitioning tool. The global

mesh is the input file (created by the user) and the desired number of partitions. The output is composed of a `partition.pro` file and partitioned meshes `mesh_i.msh`.

- The green one, which contains a file created by the user. It defines constants and regions used in the problem, such as regions that represent the boundary of the infinity, the Dirichlet sources, the wave number, etc. `partition.pro` and the problem definition file must be included in it.

To run such a problem, the `problem.pro` file has to be given to the GetDP solver.

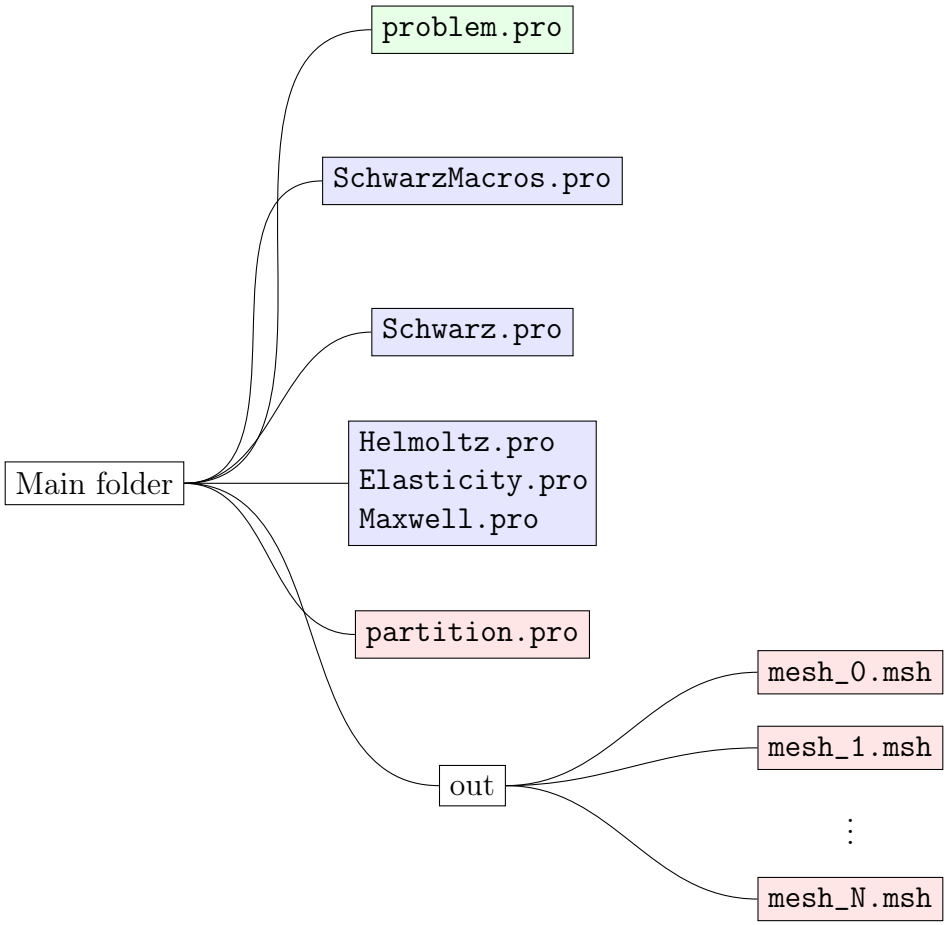

Figure 2.4: File structure of a domain decomposition problem (the green file needs to be created by the user; the blue one contains the ready to use files and the red one contains the files generated by the partitioning tool).

2.6 Efficiency

Figure 2.5 exhibits the influence of the number of partitions on the execution time of the partitioning tool. Measurements were conducted on a mesh made of 125,000 vertices. The

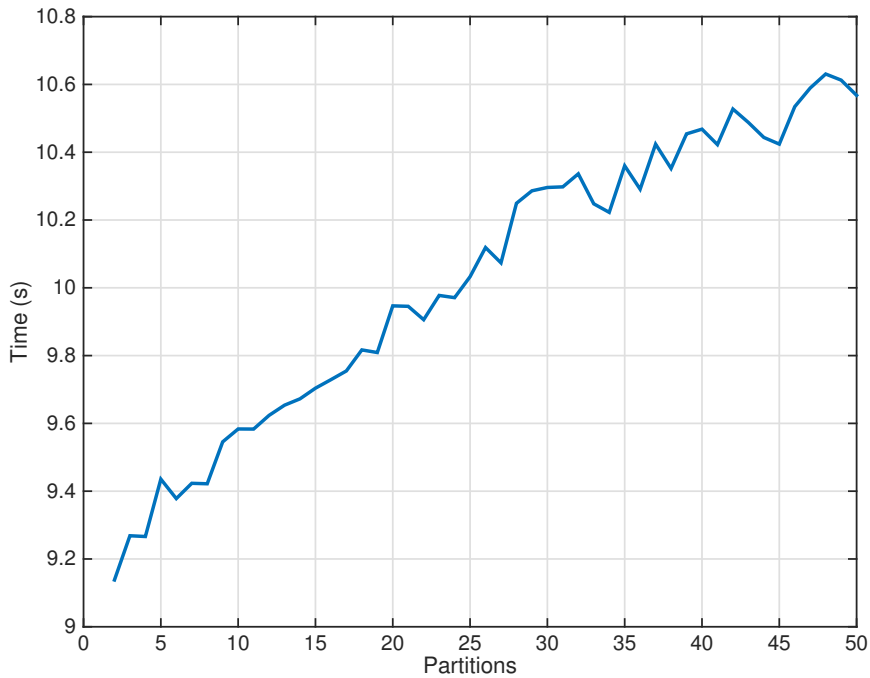

Figure 2.5: Influence of the number of partitions on the execution time of the partitioning tool.

curve appears linear. Thus, it can be concluded that the time complexity of the algorithm is linear with the number of partitions. However, the slope is not high (about 0.025) and thus, the number of partitions is not principally affecting execution time.

On the other hand, Figure 2.6 exhibits the influence of the number of nodes. It is also linear.

It can be concluded that the time complexity of the partitioning algorithm is:

$$T = \mathcal{O}(pn) \tag{2.3}$$

where T is the execution time, n the number of nodes and p the number of partitions.

2.7 Parallel version

A parallel version was developed which uses parMETIS[3] library (the parallel version of METIS). Currently, only the partitioning step that uses parMETIS is parallelised. The following steps such as the creation of the new entities or the writing of files are performed sequentially. Thus, the parallel version is not especially interesting as the partitioning step requires less than 5% of the total computing time.

[3]http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview

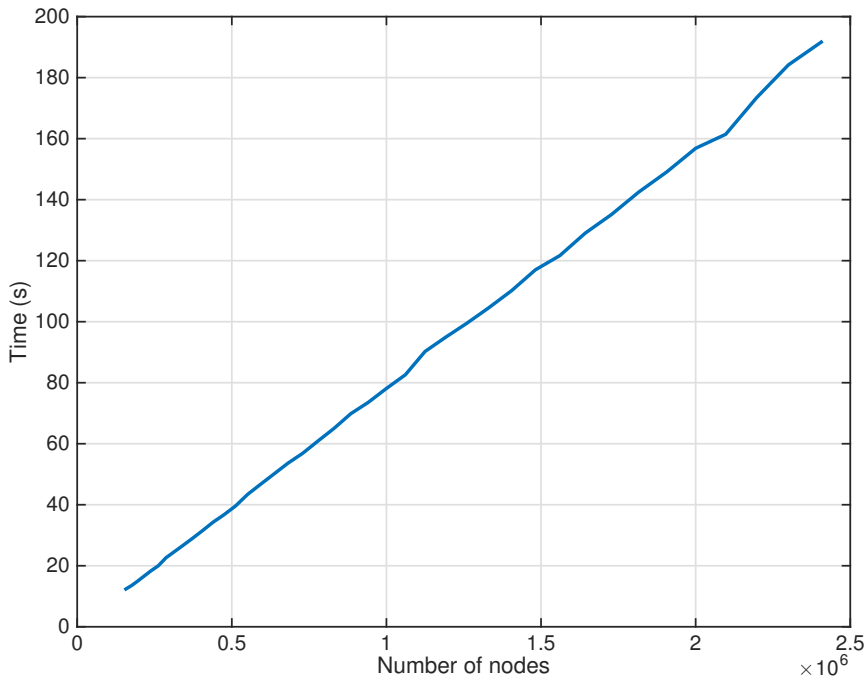

Figure 2.6: Influence of the number of nodes on the execution time of the partitioning tool.

In future developments, all the code could be parallelize all the code. Nevertheless, the algorithm, as presently developed, is difficult to parallelize.

Chapter 3

Algorithm validation

In this chapter, the algorithm is validated on a real problem composed of two concentric circles. The domain is the space between the two circles (see Figure 3.1). The small soft circle is illuminated by a plane wave coming from the left, and the solution computed is the scattering wave. Non-homogeneous Dirichlet conditions that equate the scattering wave to the incident wave are applied to the soft circle. The large circle is not a physical boundary. It appears because the domain needs to be truncated for the numerical computation. Thus, the Sommerfeld radiation condition is applied to allow the wave to propagate to infinity.

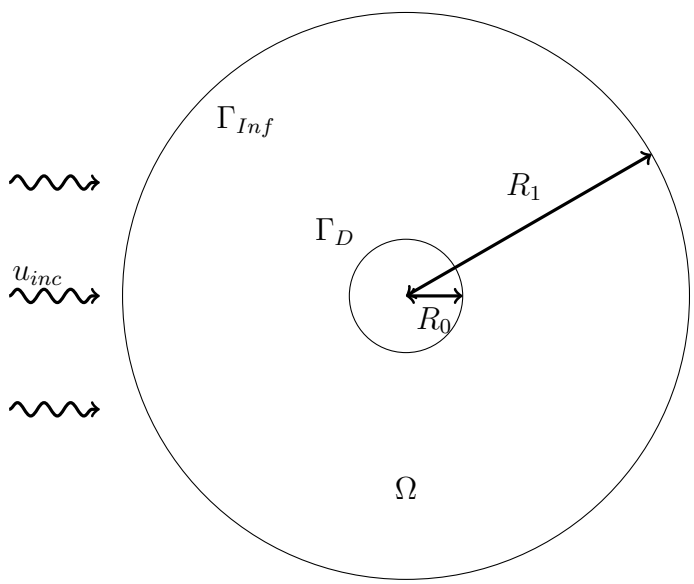

Figure 3.1: Problem composed by two concentric circles.

All computations were carried by computational resources 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.

3.1 Two dimensional problem

3.1.1 Analytical solution

To compute the error of the numerical solution, the analytical solution must be found. The analytical problem applied to the geometry defined in Figure 3.1 has the following form:

$$\begin{aligned} (\Delta + k^2)u(x,y) &= 0 \quad \text{in, } \Omega \\ u(x,y) &= -u_{inc}(x,y) \quad \text{on, } \Gamma_D \\ \lim_{r\to\infty} r^{1/2}\left(\frac{\partial}{\partial r} - ik\right)u(x,y) &= 0 \quad \text{where, } r = \sqrt{x^2+y^2}. \end{aligned} \tag{3.1}$$

and where $u(x,y)$ is the scattering wave, k the wave number, $i^2 = -1$ and $u_{inc}(x,y) = e^{-ikx}$ is a classical plane wave.

Due to the symmetry of the problem, it is natural to switch to polar coordinates (r,θ) rather of the cartesian coordinates (x,y). Equation 3.1 becomes:

$$\begin{aligned} \left(\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2}{\partial\theta^2} + k^2\right)u(r,\theta) &= 0 \quad \text{in, } \Omega \\ u(R_0,\theta) &= -u_{inc}(R_0,\theta) \\ \lim_{r\to\infty} r^{1/2}\left(\frac{\partial}{\partial r} - ik\right)u(r,\theta) &= 0 \quad \text{where, } r = \sqrt{x^2+y^2}. \end{aligned} \tag{3.2}$$

where $u_{inc}(r,\theta) = e^{-ikr\cos(\theta)}$.

Equation 3.2 is typically solved by decomposing the solution into two functions by separating of variables,

$$u(r,\theta) = R(r)\Theta(\theta) \tag{3.3}$$

which leads to two ordinary differential equations,

$$\left\{\begin{aligned} \frac{\partial^2\Theta}{\partial\theta^2} + \alpha^2\Theta &= 0, \\ r^2\frac{\partial^2 R}{\partial r^2} + r\frac{\partial R}{\partial r} + (k^2r^2 - \alpha^2)R &= 0, \end{aligned}\right. \tag{3.4}$$

where $\alpha = -\frac{1}{\Theta}\frac{\partial^2\Theta}{\partial\theta^2}$ is an *a priori* undefined number (real or complex) that links these two equations.

The solution of the first equation of (3.4) has the form :

$$\Theta(\theta) = A\cos(\alpha\theta) + B\sin(\alpha\theta) \tag{3.5}$$

where A and B are real constants. Two conditions must be imposed:

- The solution needs to be periodic, $\Theta(\theta) = \Theta(\theta + 2\pi)$ which restricts α to only integers;
- The solution needs to be symmetric with respect to Ox, which imposes that $B = 0$.

Thus, the angular solution for a particular α is:

$$\Theta_\alpha(\theta) = A_\alpha \cos(\alpha\theta) \quad \text{with, } \alpha \in \mathbb{Z}. \tag{3.6}$$

The second equation of (3.4) has the homogeneous solution:

$$R(r) = CJ_\alpha(kr) + DY_\alpha(kr) \tag{3.7}$$

where $J_\alpha(kr)$ and $Y_\alpha(kr)$ are the first and the second Bessel functions. There is restriction on C and D because (3.7) must verify the Sommerfeld radiation condition,

$$\lim_{r\to\infty} r^{1/2} \left(\frac{\partial}{\partial r} - ik\right) (CJ_\alpha(kr) + DY_\alpha(kr)) = 0. \tag{3.8}$$

Using recurrence relations that are satisfied by all Bessel functions,

$$\frac{dZ_\alpha}{dx} = \frac{1}{2}\left(Z_{\alpha-1} - Z_{\alpha+1}\right), \tag{3.9}$$

where Z_α can be J_α or Y_α, and asymptotic forms ($r \gg |\alpha - 1/4|$),

$$\begin{cases} J_\alpha(kr) &= \sqrt{\dfrac{2}{k\pi r}} \cos\left(kr - (2\alpha+1)\frac{\pi}{4}\right), \\ Y_\alpha(kr) &= \sqrt{\dfrac{2}{k\pi r}} \sin\left(kr - (2\alpha+1)\frac{\pi}{4}\right) \end{cases} \tag{3.10}$$

Equation 3.8 becomes:

$$\begin{aligned} \lim_{r\to\infty} \sqrt{\frac{2}{k\pi}} \Bigg(& \frac{kC}{2} \cos\left(kr - (2\alpha-1)\frac{\pi}{4}\right) + \frac{kD}{2} \sin\left(kr - (2\alpha-1)\frac{\pi}{4}\right) \\ & - \frac{kC}{2} \cos\left(kr - (2\alpha+3)\frac{\pi}{4}\right) - \frac{kD}{2} \sin\left(kr - (2\alpha+3)\frac{\pi}{4}\right) \\ & - ikC \cos\left(kr - (2\alpha+1)\frac{\pi}{4}\right) - ikD \sin\left(kr - (2\alpha+1)\frac{\pi}{4}\right)\Bigg) = 0, \end{aligned} \tag{3.11}$$

and after simplifying,

$$\begin{aligned} \lim_{r\to\infty} \sqrt{\frac{2}{k\pi}} \Big(& -kC \sin\left(kr - (2\alpha+1)\frac{\pi}{4}\right) + kD \cos\left(kr - (2\alpha+1)\frac{\pi}{4}\right) \\ & - ikC \cos\left(kr - (2\alpha+1)\frac{\pi}{4}\right) - ikD \sin\left(kr - (2\alpha+1)\frac{\pi}{4}\right)\Big) = 0. \end{aligned} \tag{3.12}$$

This equation is verified if $D = iC$. Thus, the radial solution for a particular α is

$$R_\alpha(r) = C_\alpha H_\alpha^{(1)}(kr) \quad \text{with, } \alpha \in \mathbb{Z}, \tag{3.13}$$

where $H_\alpha^{(1)}(kr) = J_\alpha(kr) + iY_\alpha(kr)$ is the Hankel function of the first kind.

Finally, combining (3.6) and (3.13) according to (3.3), the solution of the scattering wave is

$$u(r,\theta) = \sum_{\alpha=0}^{\infty} u_\alpha(r,\theta) = \sum_{\alpha=0}^{\infty} E_\alpha H_\alpha^{(1)}(kr)\cos(\alpha\theta), \tag{3.14}$$

where the constant $E_\alpha = A_\alpha C_\alpha$ can be determined using the non-homogeneous Dirichlet conditions on Γ_d. For this purpose, the Dirichlet conditions need to be expressed by the sum of the Bessel functions using the Jacobi–Anger expansion,

$$u_{inc}(r,\theta) = e^{-ikr\cos(\theta)} = J_0(kr) + 2\sum_{\alpha=1}^{\infty}(-i)^\alpha J_\alpha(kr)\cos(\alpha\theta). \tag{3.15}$$

This expression allows us to find:

$$E_\alpha = \begin{cases} \dfrac{-J_0(kR_0)}{H_0^{(1)}(kR_0)} & \text{if } \alpha = 0, \\ \dfrac{-2(-i)^\alpha J_\alpha(kR_0)}{H_\alpha^{(1)}(kR_0)} & \text{otherwise.} \end{cases} \tag{3.16}$$

Figure 3.2 displays the analytical scattering wave and the total wave that corresponds to the sum of the scattering and the incident wave. To compute the scattering wave, the sum in (3.14) is evaluated until $\alpha = 100$. Therefore, it is not the exact solution because the sum is not evaluated until infinity, but according to Figure 3.3 the solution becomes more exact as more and more terms are added.

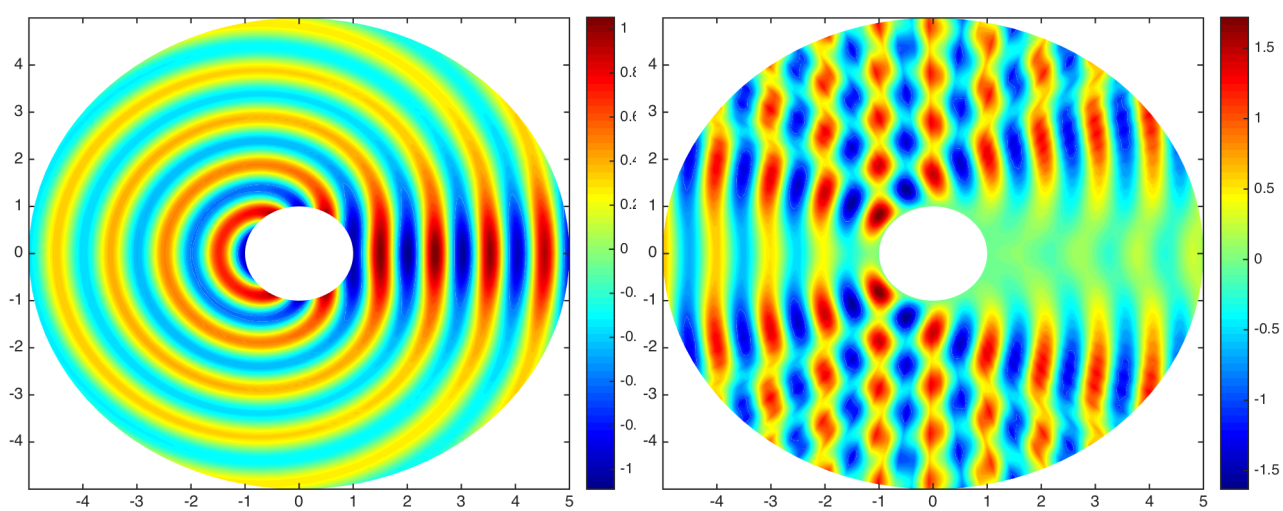

Figure 3.2: Scattering wave (left) and total wave (right) corresponding to the sum of the scattering wave and the incident wave ($k = 2\pi$).

Note that this solution is the solution of the unbounded problem. The solution studied in the following numerical method is a bounded one (bounded by Γ_{Inf}). To evaluate the error

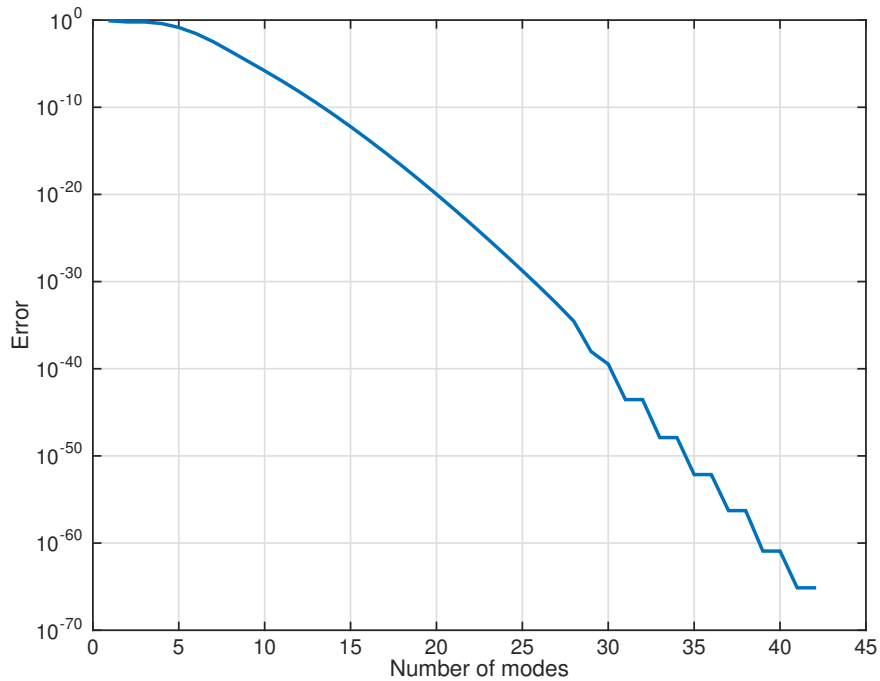

Figure 3.3: Error $\mathbb{L}^2$ of the scattering wave shown in Figure 3.14. The error compares the solution when the number of considered modes increases. The reference solution use many terms (here 200 terms).

induced by the domain decomposition method, the error induced by the absorbing boundary should not be considered. Thus, the analytical solution compared to the numerical one is not exactly (3.14). It corresponds to the solution of the same problem except for the Sommerfeld condition, which is replaced by:

$$\partial_{\mathbf{n}} u = iku \quad \text{on } \Gamma_{Inf}. \tag{3.17}$$

Using this equation, the previous mathematical reasoning remains the same except for constants C and D which become:

$$D = -C \frac{k\left(J_{\alpha-1}(kR_1) - \frac{\alpha}{kR_1} J_\alpha(kR_1)\right) - ikJ_\alpha(kR_1)}{k\left(Y_{\alpha-1}(kR_1) - \frac{\alpha}{kR_1} Y_\alpha(kR_1)\right) - ikY_\alpha(kR_1)}, \tag{3.18}$$

where R_1 is the radius of the infinite boundary Γ_{Inf}, and where the final constants E_α also change.

Thus, the interior solution becomes:

$$
\begin{aligned}
u(r,\theta) &= \sum_{\alpha=0}^{\infty} u_\alpha(r,\theta) \\
&= \sum_{\alpha=0}^{\infty} \frac{E_\alpha}{Y_{\alpha-1}(kR_1) - \frac{\alpha}{kR_1}Y_\alpha(kR_1) - iY_\alpha(kR_1)} \left[\left(Y_{\alpha-1}(kR_1) - \frac{\alpha}{kR_1}Y_\alpha(kR_1) - iY_\alpha(kR_1) \right) J_\alpha(kr) \right. \\
&\qquad \left. - \left(J_{\alpha-1}(kR_1) - \frac{\alpha}{kR_1}J_\alpha(kR_1) - iJ_\alpha(kR_1) \right) Y_\alpha(kr) \right] \cos(\alpha\theta). \quad (3.19)
\end{aligned}
$$

Finally, we discover:

$$
E_\alpha = \begin{cases} \dfrac{-J_0(kR_0)}{J_0(kR_0) - \dfrac{J_{-1}(kR_1) - \dfrac{\alpha}{kR_1}J_0(kR_1) - iJ_0(kR_1)}{Y_{-1}(kR_1) - \dfrac{\alpha}{kR_1}Y_0(kR_1) - iY_0(kR_1)} Y_0(kR_0)} & \text{if } \alpha = 0, \\[2ex] \dfrac{-2(-i)^\alpha J_\alpha(kR_0)}{J_\alpha(kR_0) - \dfrac{J_{\alpha-1}(kR_1) - \dfrac{\alpha}{kR_1}J_\alpha(kR_1) - iJ_\alpha(kR_1)}{Y_{\alpha-1}(kR_1) - \dfrac{\alpha}{kR_1}Y_\alpha(kR_1) - iY_\alpha(kR_1)} Y_\alpha(kR_0)} & \text{otherwise.} \end{cases} \quad (3.20)
$$

3.1.2 Convergence

The convergence analysis uses the following formulation:

$$
\begin{aligned}
(\Delta + k^2)u_i &= 0 \quad \text{in } \Omega_i \\
u_i &= -u_{inc} \quad \text{on } \Gamma_D \\
(\partial_{\mathbf{n}} + \mathcal{B})u_i &= 0 \quad \text{on } \Gamma_{Inf} \\
(\partial_{\mathbf{n}} + \mathcal{S})u_i &= 0 \quad \text{on } \Sigma_i
\end{aligned}
$$

where $k = 2\pi$, $\mathcal{B} = -ik$ is the classical Sommerfeld radiation condition and $\mathcal{S} = -ik + 2\pi$ is a simple zero order transmission condition.

The error committed by one, two or ten subdomains is displays in Figure 3.4. The error is measured using $\mathbb{L}^2$ relative error according to the formula:

$$
e = \sqrt{\frac{\|u_{exact} - u_i\|_2}{\|u_i\|_2}} \quad (3.21)
$$

where $\|v\|_2 = \int_\Omega v^2 dV$. The tolerance applied to the transmission boundaries is set to 10^{-4}. It corresponds to the ratio between the residual and the first residual. The convergence does

not depend on the number of subdomains. Furthermore, the observed convergence is classical for the finite elements method, namely

$$e = \mathcal{O}(h^2), \tag{3.22}$$

where e is the error and h the element size.

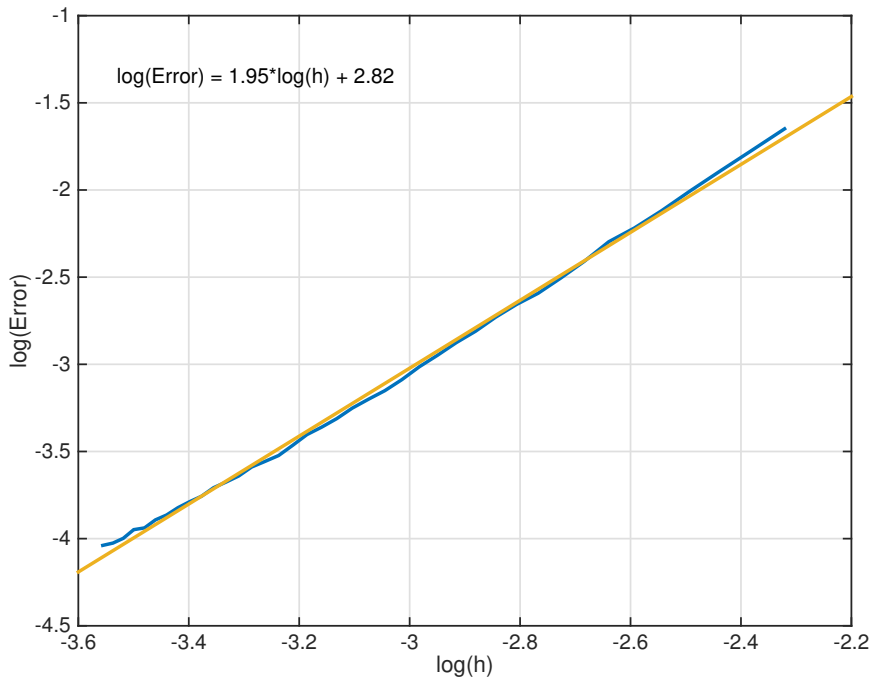

Figure 3.4: Convergence using one, two or ten subdomains, where h is the mesh size and the error is the $\mathbb{L}^2$ relative error.

To avoid measuring the error due to the approximation of the integral over elements, the initial mesh size is taking smaller enough and the number of Gauss points needed to evaluate the integral are set to the maximal value allowed by GetDP.

3.1.3 Number of iterations

Figure 3.5 illustrates the influence of the number of partitions on the speed of convergence of the optimized Schwarz algorithm applied to our test problem. As can be observed, the convergence worsens when the number of partitions increases. As Figure 3.6 appears to suggest, the number of iterations required to converge follows a linear law with respect to the number of subdomains. Experimentally, it has therefore found that:

$$Iter = \mathcal{O}(N_{sub}), \tag{3.23}$$

where $Iter$ is the number of iterations required to converge and N_{sub} the number of subdomains.

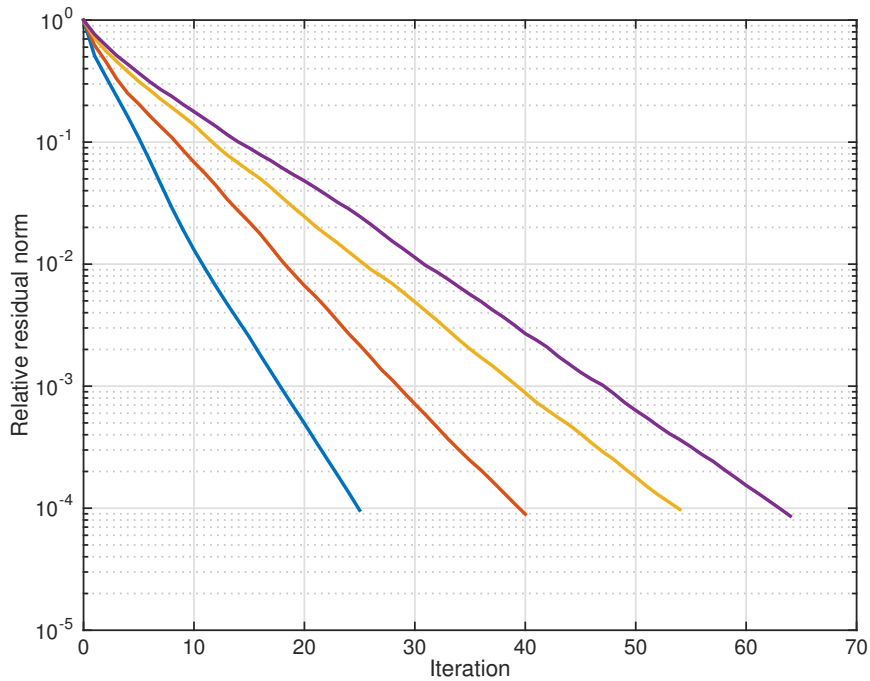

Figure 3.5: Relative residual norm in the function of the iteration for a different number of partitions applied on the same mesh. blue:2 partitions; red:10 partitions; yellow:20 partitions and purple:30 partitions.

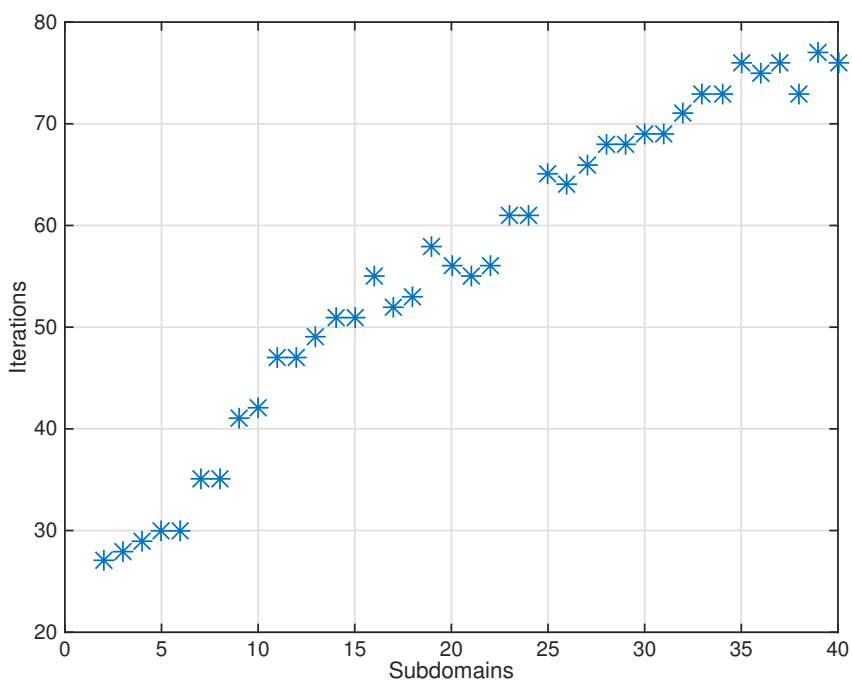

Figure 3.6: Number of iterations required to converge in function of the number of subdomains in the two-dimensional problem.

3.2 Three-dimensional problem

3.2.1 Analytical solution

The same calculations as those in the two-dimensional problem were performed, but applied to a three-dimensional problem composed of two concentric spheres. Problem 3.1 remains the same except for the Sommerfeld radiation condition, for which a three-dimensional version was used. Instead of using polar coordinates, we used spherical ones (r, θ, φ),

$$\begin{aligned} \left(\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}\right) + \frac{1}{r^2\sin^2\varphi}\frac{\partial^2}{\partial\theta^2} + \frac{1}{r^2\sin\varphi}\frac{\partial}{\partial\varphi}\left(\sin\varphi\frac{\partial}{\partial\varphi}\right) + k^2\right) u(r,\theta,\varphi) = 0 \quad \text{in, } \Omega \\ u(R_0,\theta,\varphi) = -u_{inc}(R_0,\theta,\varphi) \\ \lim_{r\to\infty} r\left(\frac{\partial}{\partial r} - ik\right) u(r,\theta,\varphi) = 0. \end{aligned} \tag{3.24}$$

where $r = \sqrt{x^2+y^2+z^2}$ and $u_{inc}(r,\theta,\varphi) = e^{-ikr\cos(\theta)\sin(\varphi)}$. This can be written as three ordinary differential equations,

$$\begin{cases} \dfrac{\partial^2\Theta}{\partial\theta^2} + \beta^2\Theta = 0, \\ \dfrac{\partial^2\Phi}{\partial\varphi^2} + \dfrac{\cos\varphi}{\sin\varphi}\dfrac{\partial\Phi}{\partial\varphi} + \left(\alpha^2 - \dfrac{\beta^2}{\sin^2\varphi}\right)\Phi = 0, \\ r^2\dfrac{\partial^2 R}{\partial r^2} + 2r\dfrac{\partial R}{\partial r} + (k^2r^2 - \alpha^2)R = 0, \end{cases} \tag{3.25}$$

if the following variable separation is used:

$$u(r,\theta,\varphi) = R(r)\Theta(\theta)\Phi(\varphi) \tag{3.26}$$

The solution for Θ is the same as in the two-dimensional problem (see (3.6)).

Let us change the variable φ by $t = \cos\varphi$ in the equation of Φ to obtain,

$$\frac{\partial}{\partial t}\left((1-t^2)\frac{\partial\Phi}{\partial t}\right) + \left(\alpha^2 - \frac{\beta^2}{1-x^2}\right)\Phi = 0, \tag{3.27}$$

which, if we express α^2 as $\gamma(\gamma+1)$ has the associated Legendre polynomials as the solution,

$$\Phi(\cos\varphi) = P_\gamma^\beta(\cos\varphi), \tag{3.28}$$

where γ is an integer and $-\gamma \le \beta \le \gamma$.

Finally, the solution to our Helmholtz problem in spherical coordinates is presented in

the following general form:

$$u(r,\theta,\varphi) = \sum_{\gamma=0}^{\infty}\sum_{\beta=-\gamma}^{\gamma} E_{\gamma,\beta} h_{\gamma}^{(1)}(kr) P_{\gamma}^{\beta}(\cos\varphi)\cos(\beta\theta), \tag{3.29}$$
$$= \sum_{\gamma=0}^{\infty}\sum_{\beta=-\gamma}^{\gamma} E_{\gamma,\beta} h_{\gamma}^{(1)}(kr) \mathbf{Y}_{\gamma}^{\beta}(\theta,\varphi),$$

where $\mathbf{Y}_{\gamma}^{\beta}$ are spherical harmonics and $h_{\alpha}^{(1)}$ is the first kind of spherical Hankel function that is the solution to the radial problem. Subsequently, the Sommerfeld radiation condition was applied.

To determine constants $E_{\alpha,n}$, the incident plane wave can be expanded in spherical harmonics as previously done,

$$u_{inc}(r,\theta,\varphi) = 4\pi \sum_{\gamma=0}^{\infty}\sum_{\beta=-\gamma}^{\gamma} i^{\gamma} j_{\gamma}(kr) \mathbf{Y}_{\gamma}^{\beta}(\hat{\mathbf{k}}) \mathbf{Y}_{\gamma}^{\beta *}(\hat{\mathbf{r}}), \tag{3.30}$$

where $i^2 = -1$, j_{γ} is the first kind of spherical Bessel function, $\mathbf{Y}_{\gamma}^{\beta *}$ is the complex conjugate of $\mathbf{Y}_{\gamma}^{\beta}$, $\hat{\mathbf{k}}$ is the unit vector pointing in the direction of the incident plane wave and $\hat{\mathbf{r}}$ is the unit vector of the position. According to the sum over the $2\gamma+1$ orthonormal spherical:

$$\sum_{\beta=-\gamma}^{\gamma} \mathbf{Y}_{\gamma}^{\beta}(\hat{\mathbf{k}}) \mathbf{Y}_{\gamma}^{\beta *}(\hat{\mathbf{r}}) = \frac{2\gamma+1}{4\pi} P_{\gamma}(\cos\theta), \tag{3.31}$$

where P_{γ} is the Legendre polynomial, the incident wave becomes:

$$u_{inc}(r,\theta,\varphi) = \sum_{\gamma=0}^{\infty} (2\gamma+1) i^{\gamma} j_{\gamma}(kr) P_{\gamma}(\cos\theta) \tag{3.32}$$

This expression allows us to find,

$$E_{\gamma,\beta} = \begin{cases} \dfrac{-(2\gamma+1) i^{\gamma} j_{\gamma}(kR_0) P_{\gamma}(\cos\theta)}{h_{\gamma}^{(1)}(kR_0)} & \text{if, } \beta = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.33}$$

Note that, as previously said for the two-dimensional case, the solution compared to the numerical solution is the one that considers the effect of the absorbing boundary condition.

3.2.2 Convergence

Figure 3.7 displays the convergence in function of the mesh size. As for the two-dimensional case, the convergence is of the order two, which is common for the finite element method. Nevertheless, it converges slower than in the two-dimensional problem.

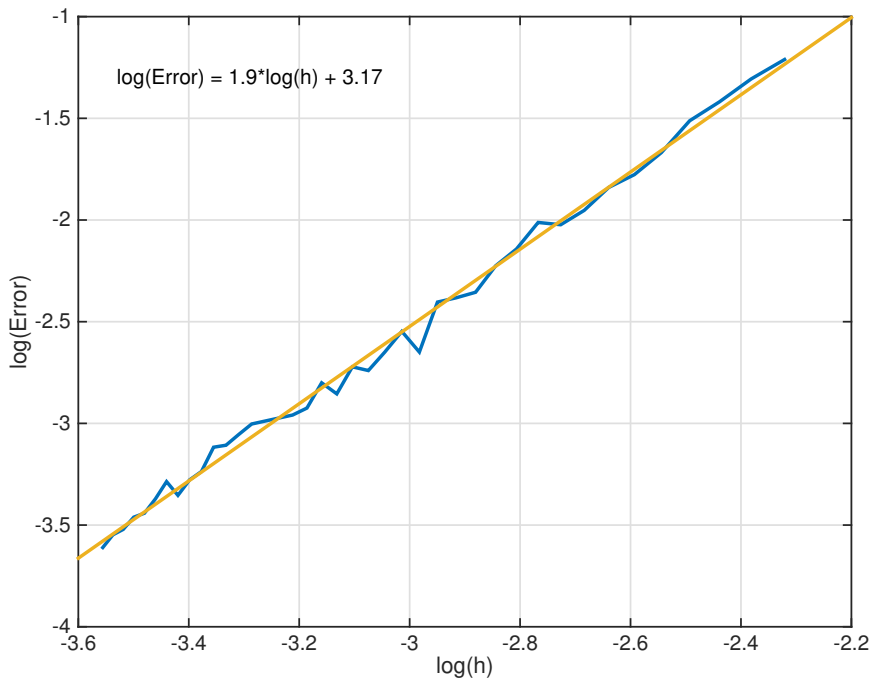

Figure 3.7: Convergence in function of the mesh size in the three-dimensional case.

3.2.3 Number of iterations

According to Figure 3.8, the influence of the number of subdomains on the Schwarz iteration in a three-dimensional problem is the same as in a two-dimensional problem. However, points higher than the straight line appear to prove that the three-dimensional case is more sensitive to the shape and the position of the partitions.

3.3 Influence of transmission conditions

The mesh partitioning tool creates irregular boundaries between the subdomains. Thus, it is necessary to determine whether this irregularity modifies the efficiency of the Schwarz algorithm. In accomplish this, two situations where analyzed. In the first situation the irregular partitioning is similar to the regular one; namely, the shape of the partitions is the same. In the second situation, the mesh partitioning tool partitions the mesh differently than does the manual partition. In all case the number of subdomains remains the same for the regular partitioning and the irregular partitioning.

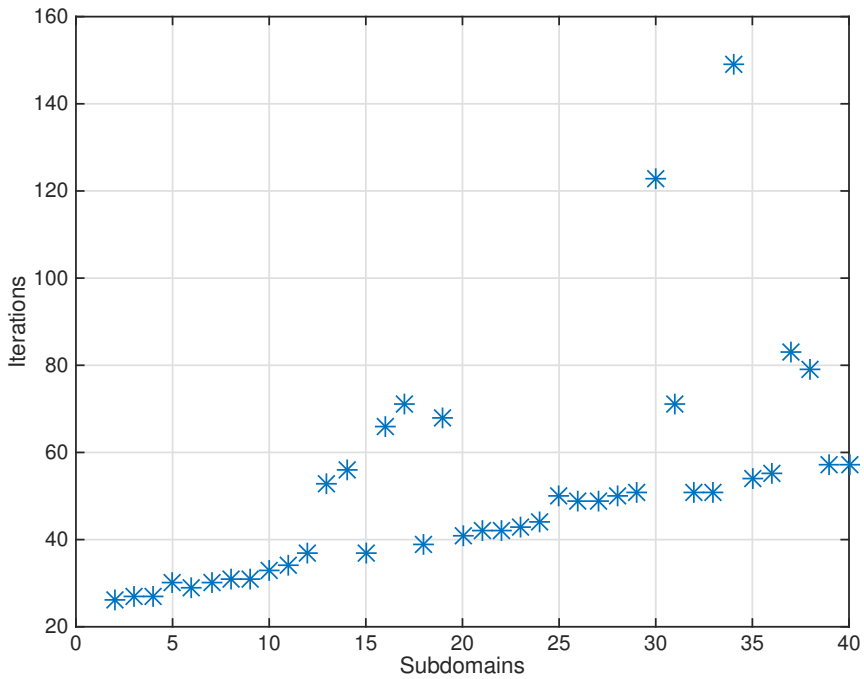

Figure 3.8: Number of iterations required to converge in the function of the number of subdomains in the three-dimensional problem.

3.3.1 Similar shape

Figures 3.9, 3.10 and 3.11 illustrate the convergence of the Schwarz iterative solver with both regular and irregular boundaries, as shown in Figures 3.12. The manual partitioning is such that a partition i, called $\mathcal{P}_i$, is defined as:

$$\mathcal{P}_i = \left\{ (r, \theta) : R_0 \leq r \leq R_1, \frac{2\pi i}{n} \leq \theta \leq \frac{2\pi(i+1)}{n} \right\}, \tag{3.34}$$

where n is the number of partitions.

When the boundaries are irregular, it can be experimentally seen that the number of iterations needed to converge is always larger than when the boundaries are regular. Furthermore, it can be seen that the transmission condition has more of an impact on a regular boundary than on an irregular one. The only one which significantly reduced the number of iterations is the second order transmission condition, shown in Figure 3.10. It is interesting to note that the Padé transmission condition, which is very efficient for regular partitioning, is inefficient in an irregular one.

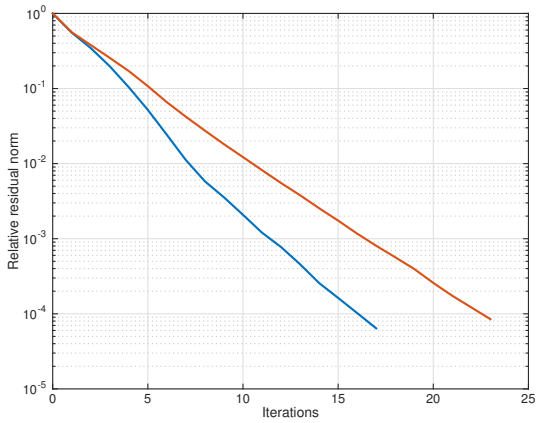

Figure 3.9: Convergence of the iterative solver with regular boundaries (blue line) and irregular ones (red line) and using the 0th order transmission condition.

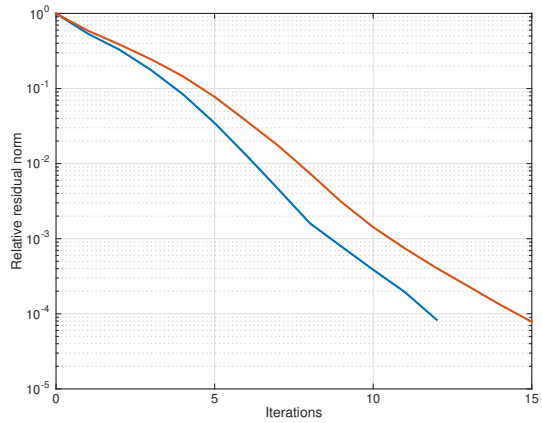

Figure 3.10: Convergence of the iterative solver with regular boundaries (blue line) and irregular ones (red line) and using the 2nd order transmission condition.

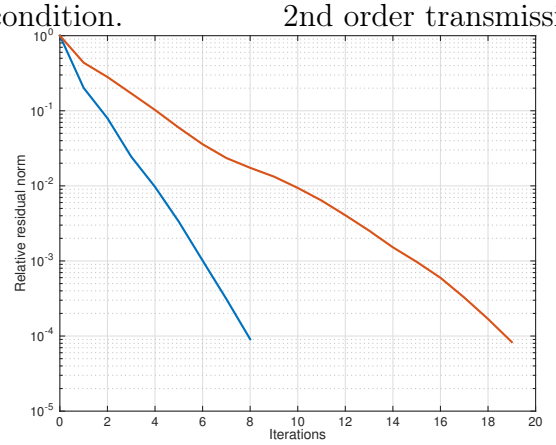

Figure 3.11: Convergence of the iterative solver with regular boundaries (blue line) and irregular ones (red line) and using the Padé transmission condition.

3.3.2 Not similar shape

Using the same geometry as in the previous case with a large number of partitions leads to irregular partitions with different shapes than what would be achieved manually. This is because the partitioning tool attempts to minimize the boundary between subdomains. Thus, for example, a partition could not have a border touching the inner circle where the Dirichlet boundary condition is applied. This can increase the number of iterations. The difference between a regular partition and an irregular partition depends on the number of partitions and the geometry of the problem. The only conclusion that we can assert is that the number of iterations is larger with irregular boundaries than with regular boundaries. Nevertheless, the difference is still acceptable.

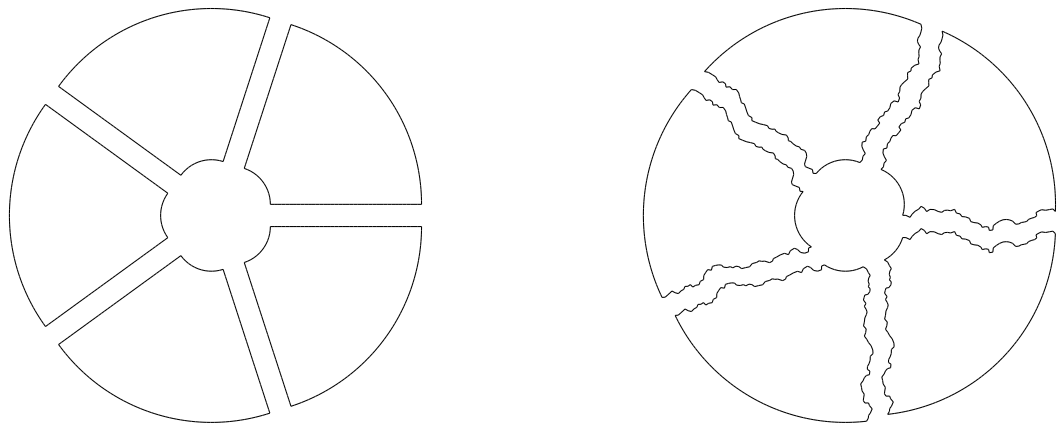

Figure 3.12: Regular and irregular partitioning.

3.4 Parallel scaling performance

We also examine the efficiency of the parallel implementation of the Schwarz algorithm. The problem studied is the same as the two-dimensional case, but with a more refined mesh. The time measured is the CPU time of the numerical Schwarz resolution. The analysis is focused on the Schwarz algorithm and therefore, the time required to conduct a post- or pre-processing was not taken into account.

Figure 3.13 displays the strong scaling associated with the two-dimensional problem with a fixed degree of freedom equal to $399,904$ and the number of processes used increased 1 one to 40.

We assume that the CPU time of execution using n processes is:

$$T_n = T_s + T_p \tag{3.35}$$

where T_s is the time to compute the sequential part of the algorithm; namely, the part of the algorithm that can be parallelized. Using p as the parallelizable fraction of the code,

$$\begin{aligned} T_s &= (1-p)T_1 \\ T_p &= p\frac{T_1}{n} \end{aligned} \tag{3.36}$$

where n is the number of processes. Knowing that $S_n = T_1/T_n$, where S_n is the speedup, we can find Amdahl's law,

$$S_n = \frac{1}{1-p+\frac{p}{n}}. \tag{3.37}$$

A fitting of the measured data allows us to determine that the parallelizable fraction p is equal to roughly 97% which is a excellent parallelization.

Figure 3.14 displays the weak scaling. The work per process remains constant when the number of processes increases. If the parallelization is perfect, this curve must be constant. However, this is not the case because only almost 97% is parallelized.

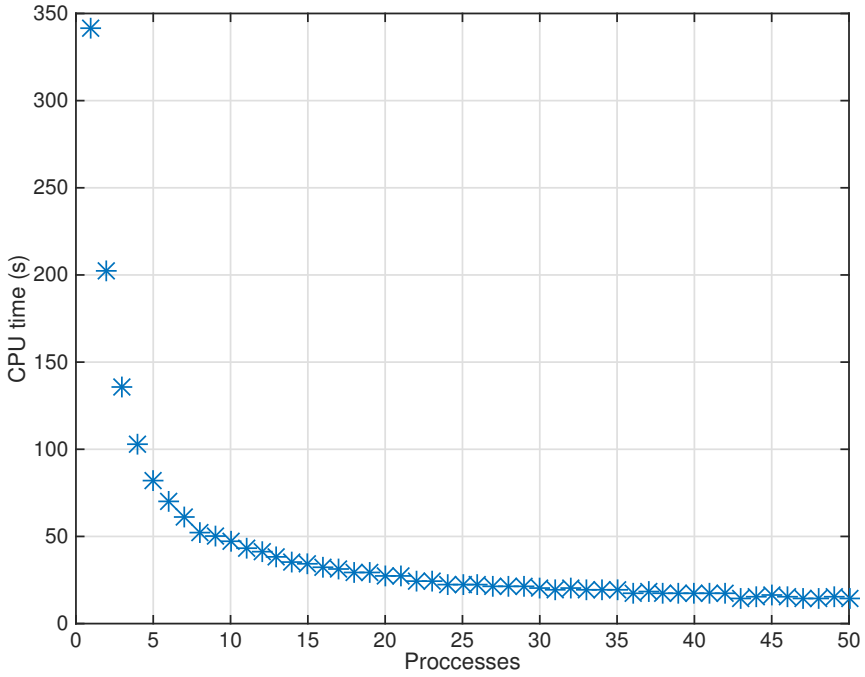

Figure 3.13: Strong scaling.

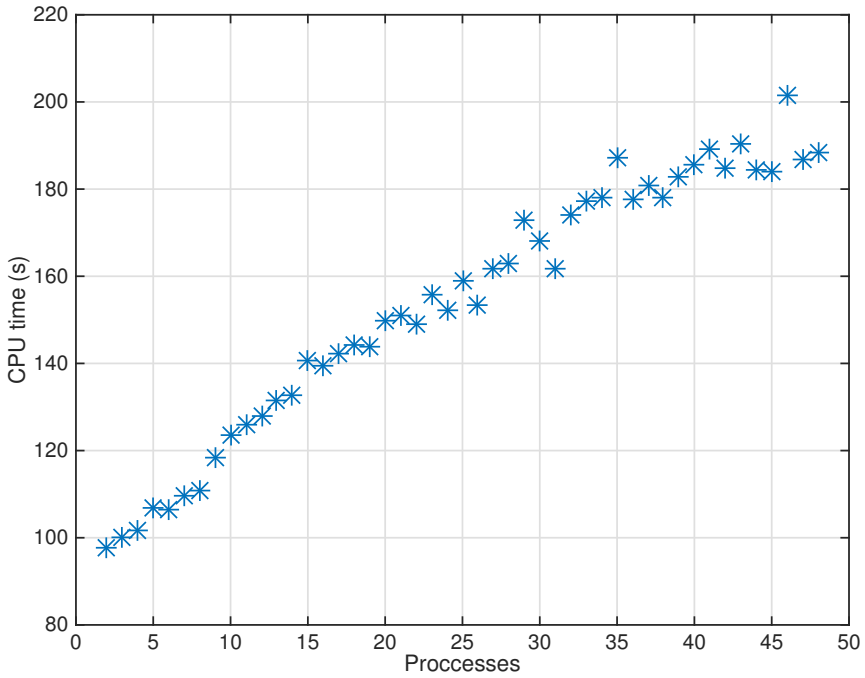

Figure 3.14: Weak scaling.

Chapter 4

Numerical simulation of p-wave in the ground

In this chapter, a numerical application of the domain decomposition Schwarz method is applied to geophysical (seismic) waves. First, this chapter reviews seismic measurement as well as the physical phenomena involved in seismic waves. Finally, numerical solutions obtained from domain decomposition are presented.

4.1 Seismic measurement

The goal of the seismic measurement and particularly, the reflective seismic measurement is to identify geological layers present in the ground using wave reflection.

Similar to all waves, when a seismic wave arrives on a surface separating two layers, a reflection occurs. This reflection depends on the physical properties of the layers. For example, the reflection of acoustic waves depends on the acoustic impedance of the two media defined as:

$$Z_{ac} = \rho c, \tag{4.1}$$

where ρ is the density and c is the speed of sound. One can find that the reflection of a normal acoustic wave follows the linear relation,

$$\begin{pmatrix} f_2 \\ g_1 \end{pmatrix} = \frac{1}{Z_1 + Z_2} \begin{pmatrix} Z_2 - Z_1 & 2Z_1 \\ 2Z_2 & Z_1 - Z_2 \end{pmatrix} \begin{pmatrix} g_2 \\ f_1 \end{pmatrix} \tag{4.2}$$

where f represents the waves moving toward the normal of the interface and g represents the waves moving in the other direction; the subscripts 1 and 2 denote the first and second media.

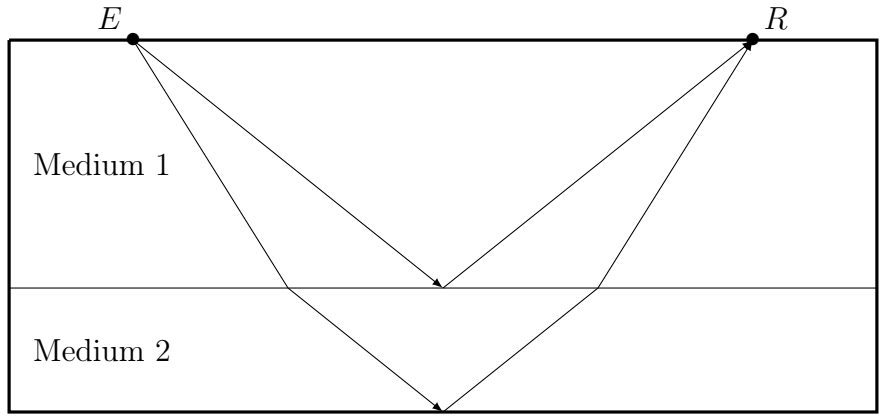

Figure 4.1: Ground measurement.

Figure 4.1 exhibits how a seismic measurement is carried. A monochrome wave is emitted at the point E on the ground surface. The wave propagates into the ground. At the interface of media 1 and 2, a part of the emitted wave is reflected and reaches the sensor in R. Another part is refracted into medium 2. As before, when this refracted wave reaches an interface it is reflected and this can be measured by the sensor.

In practice, many sensors are placed on the ground to build a complete cartography of the underground.

4.2 Physical phenomena

A seismic reflection measurement involves three types of waves:

- a wave that propagates into the underground;
- a wave that propagates in the air under the ground;
- and a wave that propagates on the ground (surface waves).

4.2.1 Underground waves

Since the present case is a three-dimensional, the particle motion has three components along the three main axes. This results in three types of underground waves (Figure 4.2):

- A primary wave or longitudinal wave that corresponds to a particle motion in the same direction as the wave.

- Two secondary waves or tangential waves that correspond to particle motions in directions perpendicular to the direction of the wave.

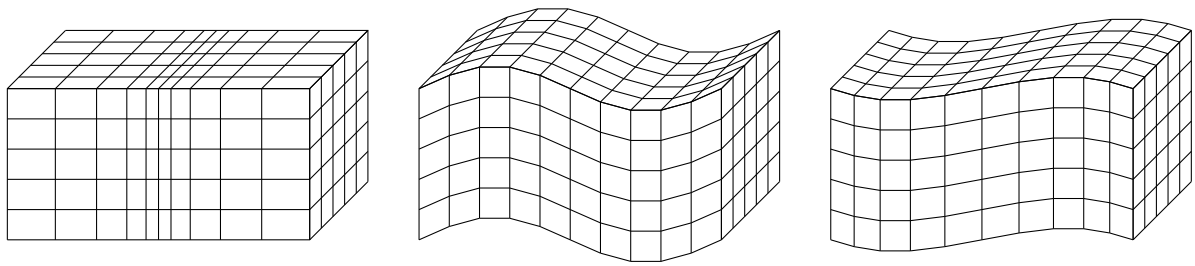

Figure 4.2: P-wave, SV-wave and SH-wave.

These three types originate from the Navier equation,

$$(\lambda + 2\mu)\vec{\nabla}\nabla \cdot \mathbf{u} - \mu\vec{\nabla} \times \vec{\nabla} \times \mathbf{u} + \mathbf{f} = \rho\frac{\partial^2 \mathbf{u}}{\partial t^2}, \tag{4.3}$$

where $\mathbf{u}$ is the displacement field, λ, μ the Lamé parameters and $\mathbf{f}$ a volume force. Using the Helmholtz theorem, $\mathbf{u}$ can be decomposed into the sum of the gradient of a scalar field ϕ and the curl of a vector field $\boldsymbol{\psi}$. The definition of $\boldsymbol{\psi}$ is not injective and thus, the Gauss condition $\vec{\nabla} \cdot \boldsymbol{\psi} = 0$ is added because it allows for a simplification. Using this decomposition, (4.3) becomes:

$$\vec{\nabla}\left(\rho\frac{\partial^2 \phi}{\partial t^2} - (\lambda + 2\mu)\Delta\phi\right) + \vec{\nabla} \times \left(\rho\frac{\partial^2 \boldsymbol{\psi}}{\partial t^2} - \mu\vec{\Delta}\boldsymbol{\psi}\right) = \mathbf{f}, \tag{4.4}$$

where $\vec{\Delta}$ is the vector Laplacian. This decomposition is valid if $\mathbf{u}$ is twice continuously differentiable. In this hypothesis, if there is no volume source, the problem can be decoupled as:

$$\begin{cases} \dfrac{\partial^2 \phi}{\partial t^2} - \dfrac{\lambda + 2\mu}{\rho}\Delta\phi = 0, \\ \dfrac{\partial^2 \boldsymbol{\psi}}{\partial t^2} - \dfrac{\mu}{\rho}\Delta\boldsymbol{\psi} = 0, \end{cases} \tag{4.5}$$

where two characteristic speeds appear $c_p = \sqrt{(\lambda + 2\mu)/\rho}$(the speed of the P-wave) and $c_s = \sqrt{\mu/\rho}$ (the speed of the S-wave). As can be observed, the S-wave is slower than the P-wave.

4.2.2 Air wave

The air wave is a classical acoustic wave. Air wave are slower than solid waves; air waves move at 340.29 m/s, while the speed of a solid wave is often larger than $1,000$ m/s.

4.2.3 Surface waves

At the surface between two media (the air and the ground), two types of waves appear. The first wave is referred to as the Rayleigh wave and the second is the Love wave. In the same medium, these waves are always slower than the P- and S-waves. Figure 4.3 represents these two waves in a basic manner.

Figure 4.3: Rayleigh and Love waves.

4.3 Numerical results

Numerical simulations were conducted on a ground of 13.52 km by 13.52 km and 4.2 km deep. This ground is composed of different layers of material with different speeds of sound which produce all phenomena observed in reflective seismic measurement, such as reflection, refraction, scattering wave, stationary wave, etc. The mesh is made by regular quadrangles (or hexahedra in three dimensions) with a 5m step.

The simulations imposed a wave of frequency equal to 50 Hz in the middle of the floor (see Figure 4.4). The absorbing boundary condition was imposed on the bottom and the borders of the domain and the Neumann condition was imposed on the top.

A mapping of speed of sound in the ground is used in this simulation. It comes from Jean Virieux, a professor at Université Joseph Fourier, Grenoble. Speeds go from 1,500 m/s to 4,482 m/s.

In this chapter, only one simulation is presented (the others can be found in Appendix D).

4.3.1 Two-dimensional simulations

This section focuses on a slice at $x = 5$ km (other slices can be found in Appendix D). Figure 4.5 presents the speed of sound distribution used to compute the wave number, according to:

$$k = \frac{\omega}{c} \tag{4.6}$$

where c is the speed of sound, and $\omega = 2\pi f$ and f represents the frequency. The corresponding particle speed distribution computed using 100 processes in 25 s is displayed in Figure 4.6. The number of iterations needed to reach a tolerance equal to 10^{-4} is 150. The mesh partitioning is illustrated in Figure 4.4.

On the top of the domain near the ground surface, reflections can be seen (as explained in Figure4.1). They are caused by the line of higher speed that probably corresponds to a thin layer of other material. Refractions in the highest speed pattern (shown in red in Figure 4.5) can also be observed.

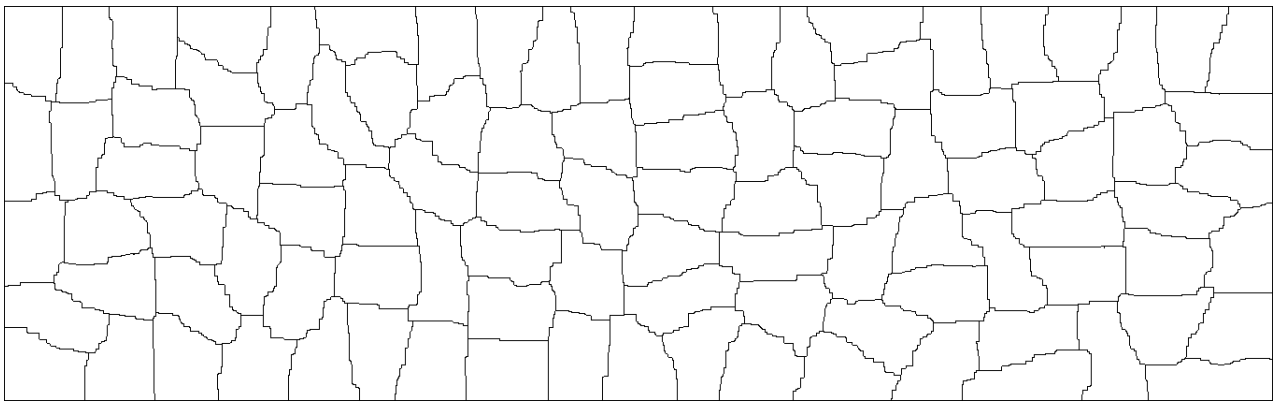

Figure 4.4: Mesh partition.

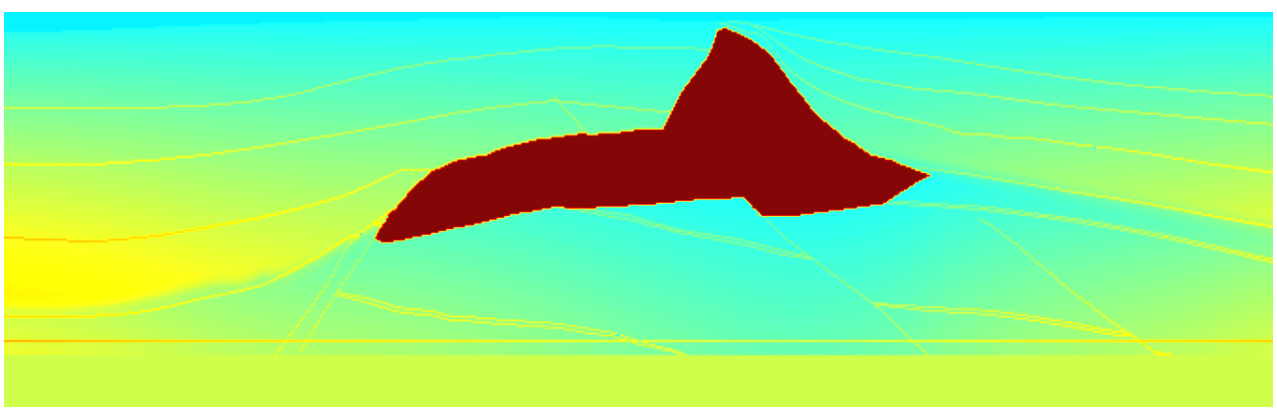

Figure 4.5: Speed of sound distribution at $x = 5$ km.

4.3.2 Three-dimensional simulation

Ideally, it would be interesting to compare the two-dimensional results with those obtained using the same frequency and the same step in a three-dimensional case. However, several problems occurred and made such a comparison impossible. We list all the issues that occurred and, to compare with computational resources, Table4.1[1] summarizes the technical

[1] `http://www.ceci-hpc.be/clusters.html`

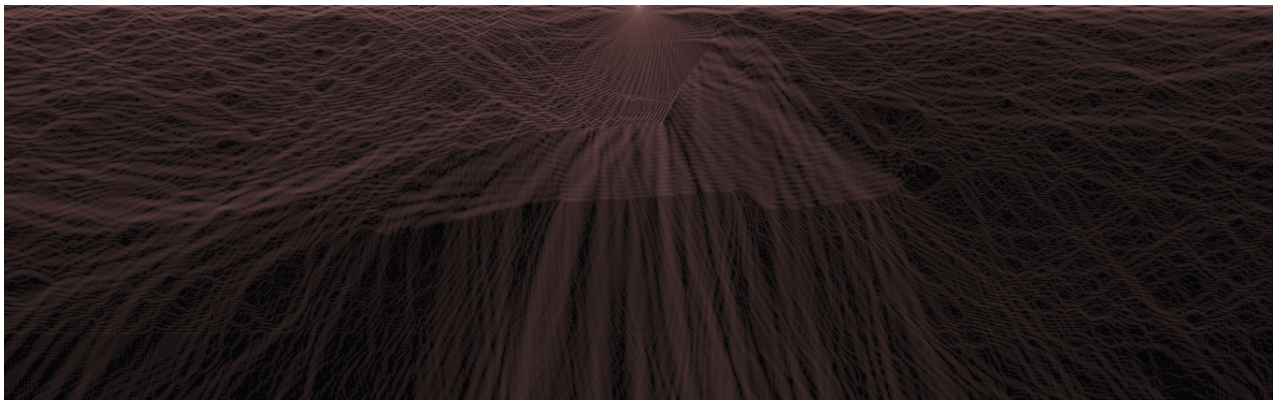

Figure 4.6: Particle speed distribution at $x = 5$ km.

characteristics of the available clusters provided by the Consortium des Équipements de Calcul Intensif (CÉCI).

Cluster's name	**CPU(s) max/job**	**RAM max/CPU**	**suitable for MPI job**
NIC4	265	4GB	Yes
Vega	1024	16GB	Yes
Hercules	48	16GB	No
Dragon1	40	16GB	No
Lemaitre2	1024	5GB	Yes
HMEM	816	4GB	No

Table 4.1: A summary of computational resources.

The first problem was the creation of the global mesh file. Using the same mesh size as in the two-dimensional case led to the creation of a mesh composed of approximately 6.1 billion regular hexahedra and roughly the same number of nodes. Computing such a mesh required a substantial amount of memory. Therefore, another approach was chosen. The mesh was computed using a mesh size of 40m, which represents approximately 959 million regular hexahedra and can be computed in ten minutes on a single core. Subsequently, this mesh is partitioned into 256 partitions by our mesh partitioning tool. Finally, small mesh files were refined using a splitting to obtain meshes of step 20m, 10m and ideally, 5m.

The second problem was the size of all mesh files using the 2.2 Gmsh mesh format. For 40m meshes, all files require 2GB, 33GB for 20m meshes and 500GB for 10m meshes. It is therefore impossible to do computations using 10m meshes or smaller because files require more place than the available on clusters.

Finally, even if the new Gmsh mesh format (which require less place to be store) is used, data needs to be charge in memory. For example, 20m meshes launched on 256 CPUs of NIC4 cluster is impossible because the 4GB/CPU is overtaken.

Thus, we decided to decrease the frequency of the study to 5Hz and compute the solution

using a 40m spacial step fro two- and three-dimensional cases. In the worst case, this spacial step involves seven points per wave length. The analysis focuses on the slice at $x = 5$ km. Figure 4.7 depicts the particle speed distribution obtained in the two-dimensional case. To prove that the spacial step is small enough, Figure 4.8 depicts the same problem size with a spacial time step of 20m. As can be observed, particle speed distributions are nearly the same in the two simulations. Thus, it can be assumed that a spacial step of 40m is enough even in the three-dimensional case.

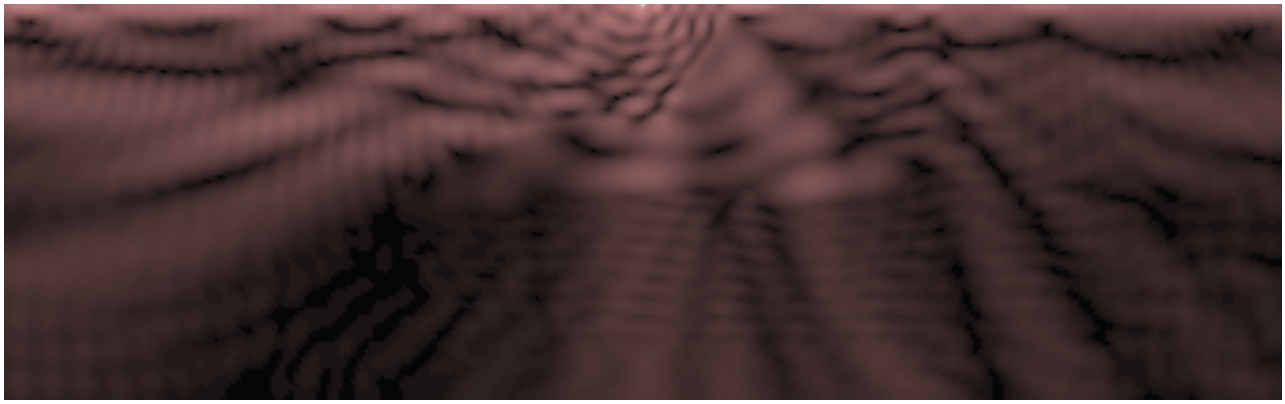

Figure 4.7: Particle speed distribution at $x = 5$ km using 5Hz and the 40 spacial step in the two-dimensional simulation.

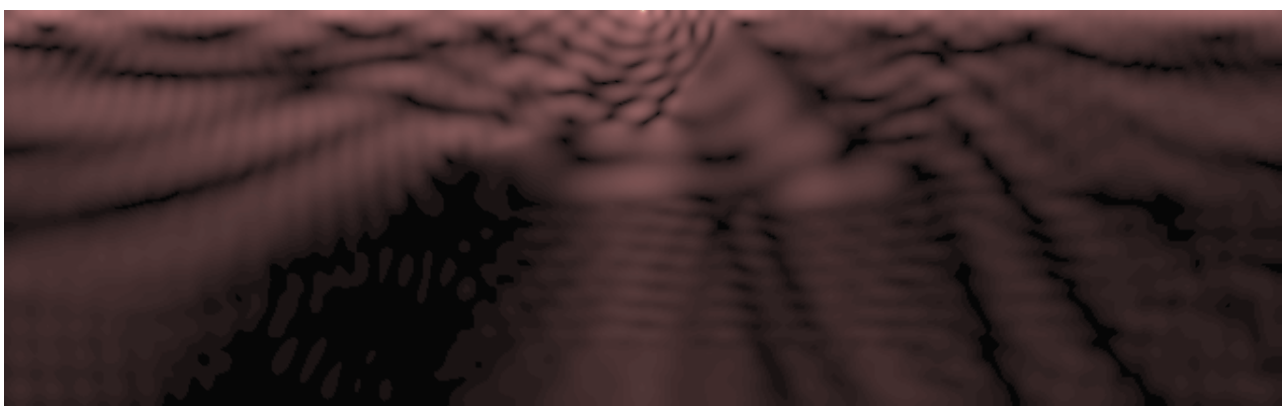

Figure 4.8: Particle speed distribution at $x = 5$ km using 5Hz and the 20 spacial step in the two-dimensional simulation.

As expected, the three-dimensional simulation is quite different (Figure 4.9), which can be explained by two simple ascertainments.

First, the problem is not exactly the extension of the two-dimensional case in three dimensions, since the excitation is not imposed in the same place. In the two-dimensional problem, it is imposed exactly at the same abscissa as the slice, while in the three-dimensional case it is imposed at a different abscissa than the slice. The source is imposed at the middle of the ground and therefore at $x = 6.76$ km, while the slice is at $x = 5$km.

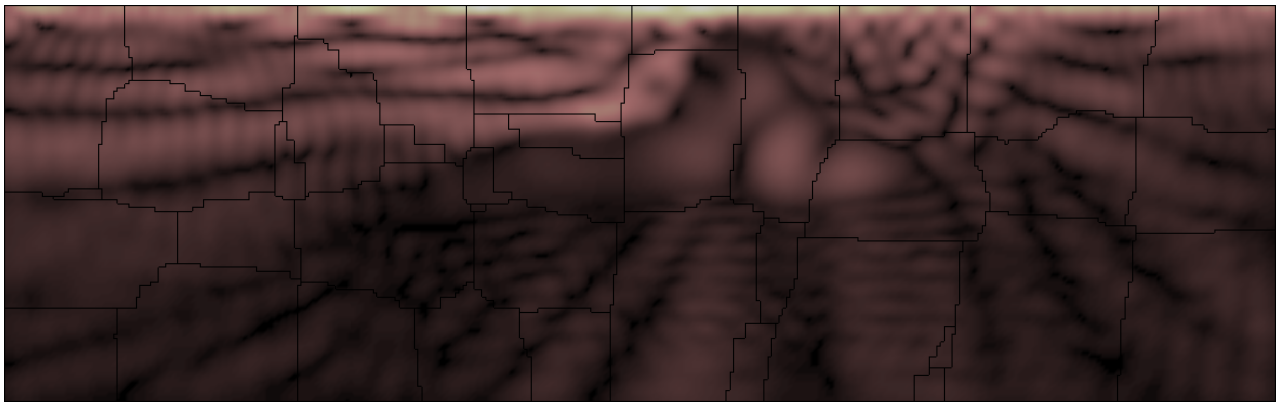

Figure 4.9: Particle speed distribution at $x = 5$ km using 5Hz and the 40 spacial step in the three-dimensional simulation.

Second, the three-dimensional simulation allows reflection in the plan of the sheet which can change the entire solution.

To conclude, two-dimensional simulations are not especially interesting in a seismic problem. The ground cannot be considered as a layer in a two-dimensional problem and therefore, an efficiency three-dimensional simulation must be built. In this context, a domain decomposition method is absolutely necessary and especially at high frequencies.

Conclusion

Domain decomposition methods and especially the parallel Schwarz algorithm applied to the Helmholtz problem were studied in this research. A parallelization of the algorithm proposed by Hermann Schwarz was the first improvemente. To minimize computations, P.L. Lions proposed another version that converges without creating overlap between the subdomains. Nevertheless, in case of the Helmholtz problem, the operator is not positive definite. This implies that the numerical implementation of such an operator is divergent if the algorithm proposed by P.L. Lions was used without modification. Specific modifications for the Helmholtz problem were presented, such as adapting the transmission condition between subdomains by tuning the transmission operator. Many studies have been and still need to be conducted to find other transmission operators and improve the convergence.

Once the mathematical problem was posed, its weak form used in the finite element modeling was presented and the Schwarz algorithm was implemented using GetDP software on a so-called GetDDM package. This package is already quite comprehensive and general. However, changes still need to be made to use high-order elements.

A partitioning tool was developed in the framework of this work to automatically partition a mesh and create a topology file used by the GetDDM package. With this tool, one does not have to worry about topology between sub-domains. A user only has to worry about the definition of his problem. A first parallel version of this tool was presented. Nevertheless, only the sequential version was used in this work since a substantial amount of work still needs to be done to make it useful and efficient. A complete redesign of the algorithm could be envisaged to make the tool more suitable for future parallelization.

The algorithm was validated for a two- and three-dimensional academic problem. The algorithm works quite well. We compared the convergence using regular partitions that were built manually, as it was done by the ACE group, and the irregular partitions were built using the partitioning tool. Irregular partitions slow down convergence without making it a problem. Nevertheless, an interesting transmission condition that exhibits excellent convergence with a regular partitioning is completely inefficient with an irregular one.

Then, a real seismic problem for which the domain decomposition method was studied involves large domain (several kilometers) and possibly high frequencies. Modelling such problems using a finite element method requires that the computations be performed on

an exceedingly refined mesh with many degrees of freedom. Thus, they are particularly well-suited to the method of decomposition of the domain, since the problem is large. Nevertheless, the method of decomposition of the domain can reduce the computation time of large problems, but not the memory used. Thus, solving large problems still requires a substantial amount of memory, which represents a limitation.

Finally, it may be interesting to apply of this method to other scientific domains, particularly biomedical applications such as the numerical simulation of different processes involved in the human body. Applying the finite volume method used in the context of fluids could lead to better simulation in fields such as meteorology or the study of oceanographic currents.

Appendix A

Propagative and evanescent modes

The two wave numbers that appear when the convergence rate is computed in Section 1.3 are: k, which is related to the wave solution and represents the traditional wave number of the solution; and the Fourier number $\tilde{k}$, which is related to the error induced by the numerical scheme. The behavior of the error is completely different if $\tilde{k}$ is greater or smaller than k. Thus, it appears natural to devise a reduced Fourier number $s = \tilde{k}/k$.

According to equations 1.26 and 1.25, the error converges in the same way that the Fourier transform of the solution, which leads to iteration n,

$$\begin{aligned} \hat{u}_1^n(x,s) &= \hat{u}_1^n(0,s)e^{\lambda(s)x} \\ \hat{u}_2^n(x,s) &= \hat{u}_2^n(0,s)e^{-\lambda(s)x}, \end{aligned} \tag{A.1}$$

where,

$$\lambda(s) = \begin{cases} k\sqrt{s^2-1} & \forall\, |s| > 1, \\ 0 & \forall\, |s| = 1, \\ ik\sqrt{1-s^2} & \forall\, |s| < 1. \end{cases} \tag{A.2}$$

As previously noted, the error in the subdomain depends only on the error that is made at the interface. The propagation of the error is expressed by the exponential term, which has a different behavior in function of the parameter λ.

A.1 Evanescent modes

The first error transmission modes discussed are the evanescent modes. They appear when the reduced Fourier number is greater than one. This leads to a real $\lambda = k\sqrt{s^2-1}$ and the exponential term becomes:

$$e^{k\sqrt{s^2-1}x}. \tag{A.3}$$

Therefore, the error propagates in the domain as a decreasing exponential equal to $\hat{u}_i^n(0, s)$ at the interface. Far from the interface, these error modes become negligible. Note that the exponential is purely real, hence this kind of error mode does not lead to a phase shift of the solution.

A.2 Propagative modes

The second error transmission mode is the propagative modes. These modes appear when the reduced Fourier number is smaller than one. This leads to a complex $\lambda = ik\sqrt{1-s^2}$ where i is the imaginary number and the exponential term becomes:

$$e^{ik\sqrt{1-s^2}x}. \tag{A.4}$$

Therefore, the error propagates in the domain as the combination of a real cosine and an imaginary sine. As for evanescent modes at the interface, the solution is equal to $\hat{u}_i^n(0, s)$, However, far from the interface the error is a complex wave function that induces an error in all the subdomains and a phase shift of the solution because the exponential term has a complex part. These kinds of error transmission modes pollute the solution more in the subdomains than in the evanescent modes.

A.3 Between propagative and evanescent modes

Between propagative and evanescent modes, when the reduced Fourier number is equal to one, λ are null and thus, the exponential term is equal to one. With this kind of error transmission mode, the error that is made at the interface is transmitted to the subdomain without a phase shift and without attenuation.

Appendix B

Absorbing boundary conditions

In this Appendix, absorbing boundary conditions (ABC) are examined. First, it is necessary to explain why the absorbing boundary conditions are needed. The problem solved is a wave propagation phenomena in a domain which extends to infinity, but numerically it is only possible to compute the solution on a finite domain. Thus, the problem needs to be truncated by adding a new boundary. This new boundary is not a physical boundary and thus, should be transparent to avoid introducing reflections. Numerically, it is not possible to completely suppress them but they can be strongly attenuated. This is the purpose of the absorbing boundary conditions with the following form:

$$(\partial_{\mathbf{n}} + \mathcal{B})u = 0, \tag{B.1}$$

where $\partial_{\mathbf{n}}$ is the derivative of u along the outer normal of the absorbing boundary and $\mathcal{B}$ is the operator introduced to avoid reflection. As explained in Section 1.4, this form is particularly interesting.

Another newer method called perfectly matched layer (PML) exists. This method does not introduce a new boundary, but an external region where an absorbing condition is imposed. This method is less general than absorbing boundary condition because their efficiency strongly depends on their parameters. This method is not be discussed here.

Numerically, to compare the different absorbing boundary conditions, the wave scattering problem seen in Figure 3.1 of Chapter 3 is used. The error shown in all figures is the relative error of norm one,

$$e = \frac{\|u_{exact} - u\|}{\|u_{exact}\|}. \tag{B.2}$$

Analytically, the reflection coefficient $\Gamma = \|u_r\|/\|u_i\|$ of an absorbing condition can be computed by introducing the sum of the incident wave u_i and the reflection wave u_r into the considered condition. To remain in the same geometry as the numerical example, it is

necessary consider the same cylindrical geometry with an incident wave equal to:

$$u_i(r,\theta) = e^{ir(k_x\cos(\theta)+k_y\sin(\theta))}. \tag{B.3}$$

Then we need to switch to a new frame attached to the absorbing boundary, with base vectors equal to its normal and tangential direction. In this frame, $k_x = k\cos(\theta+\phi)$ and $k_y = k\sin(\theta+\phi)$ with ϕ is the angle between the normal of the absorbing boundary and the direction of the wave. Figure B.1 presents this new frame.

Finally, the incident wave is:

$$\begin{aligned} u_i(r,\theta) &= e^{irk(\cos(\theta)\cos(\theta+\phi)+\sin(\theta)\sin(\theta+\phi))} \\ &= e^{irk\cos(\phi)}, \end{aligned} \tag{B.4}$$

and the reflective wave is:

$$\begin{aligned} u_r(r,\theta) &= \Gamma e^{irk(\cos(\theta)\cos(\theta+(\pi-\phi))+\sin(\theta)\sin(\theta+(\pi-\phi)))} \\ &= \Gamma e^{irk(\cos(\theta)\cos(\theta-\phi+\pi)+\sin(\theta)\sin(\theta-\phi+\pi))}, \\ &= \Gamma e^{-irk(\cos(\theta)\cos(\theta-\phi)+\sin(\theta)\sin(\theta-\phi))}, \\ &= \Gamma e^{-irk\cos(\phi)}. \end{aligned} \tag{B.5}$$

To find the reflection coefficient Γ, the total wave $u_i + u_r$ is introduced into the absorbing condition and the Γ is $\|u_r\| = \Gamma\|u_i\|$.

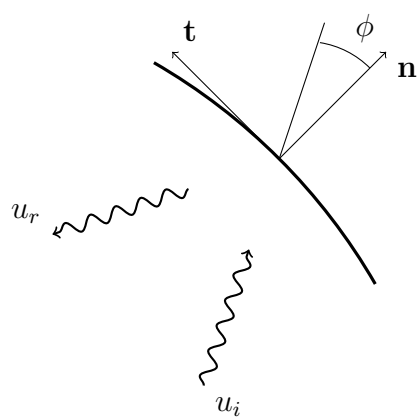

Figure B.1: New frame uses to compute the reflection coefficient.

B.1 Without ABC

To illustrate the importance of absorbing boundary conditions, let us observe what append if $\mathcal{B} = 0$ on the absorbing boundary. Figure B.2 displays the numerical solution obtained and its relative error. As can be observed, the solution is completely wrong: the $\mathbb{L}^2$ error is 238% and the maximal error is 620%. This is caused by the many reflections on the boundary which perturb the solution.

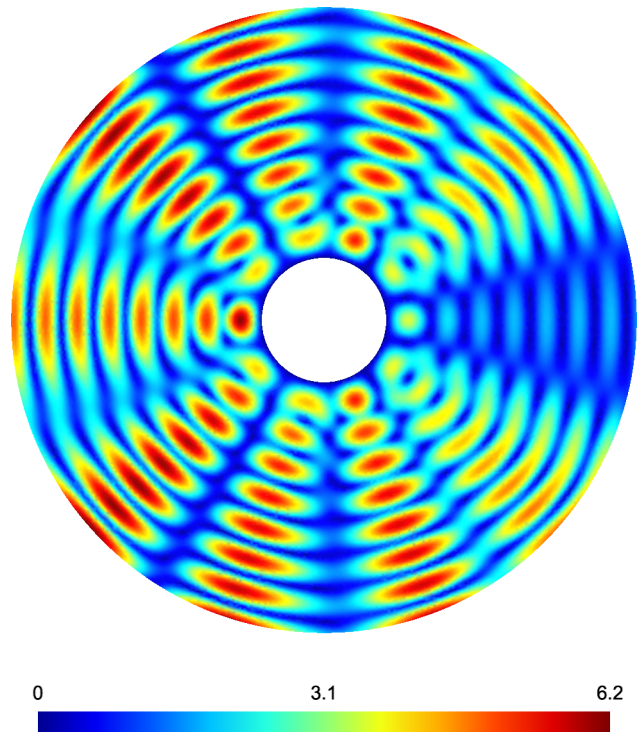

Figure B.2: Relative error without ABC.

B.2 Sommerfeld radiation condition

The simplest absorbing condition is the Sommerfeld radiation condition that was defined in 1912 by Arnold Sommerfeld. He has stated that the infinite can induce reflected wave, expressed as:

$$\lim_{r\to\infty} r^{\frac{(d-1)}{2}} \left(\partial_r u - iku\right) = 0, \tag{B.6}$$

where d is the dimension of the problem, $i^2 = -1$ and k is the wave number.

A first naive choice is to impose $\mathcal{B} = -ik$ on the absorbing boundary. This condition produces adequate if the incident wave is normal on the absorbing boundary. Otherwise, it produces some reflected waves as shown in Figure B.3. As can be seen, this solution is better than without the absorbing condition; the maximal relative error is reduced to 15% and the $\mathbb{L}^2$ error is reduced to 7.45%. This condition was used most often before 1970.

To find the reflection coefficient:

$$\begin{aligned} ik\cos(\phi)e^{ikr\cos(\phi)} - \Gamma ik\cos(\phi)e^{-ikr\cos(\phi)} &= ike^{ikr\cos(\phi)} + \Gamma ike^{-ikr\cos(\phi)} \\ \Gamma\left(\cos(\phi)+1\right)e^{-ikr\cos(\phi)} &= \left(\cos(\phi)-1\right)e^{ikr\cos(\phi)}, \end{aligned} \tag{B.7}$$

where the only possible choice to satisfy $\|u_r\| = \Gamma\|u_i\|$ is:

$$\Gamma = \left|\frac{\cos(\phi)-1}{\cos(\phi)+1}\right|. \tag{B.8}$$

Figure B.4 illustrates the variation of this reflection coefficient with the incident angle. As can be observed, the Sommerfeld condition is exact if the incident angle is normal to the boundary.

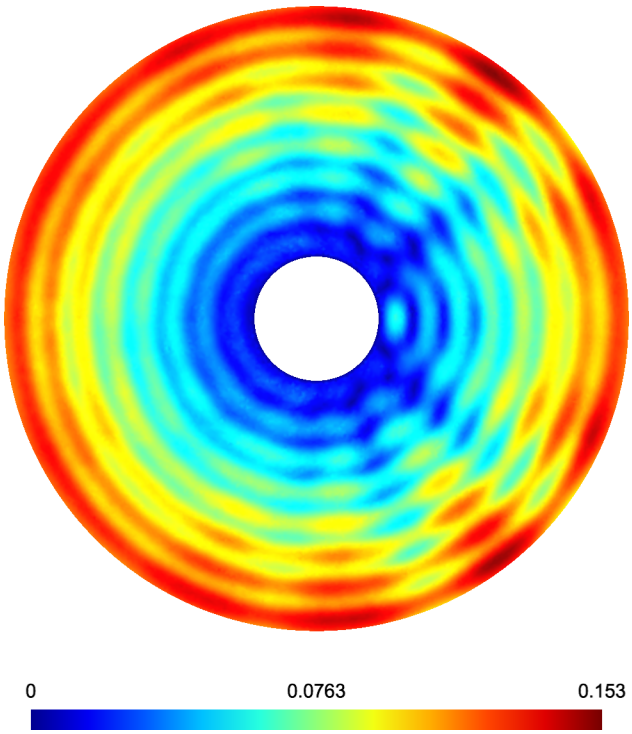

Figure B.3: Relative error induced by Sommerfeld ABC.

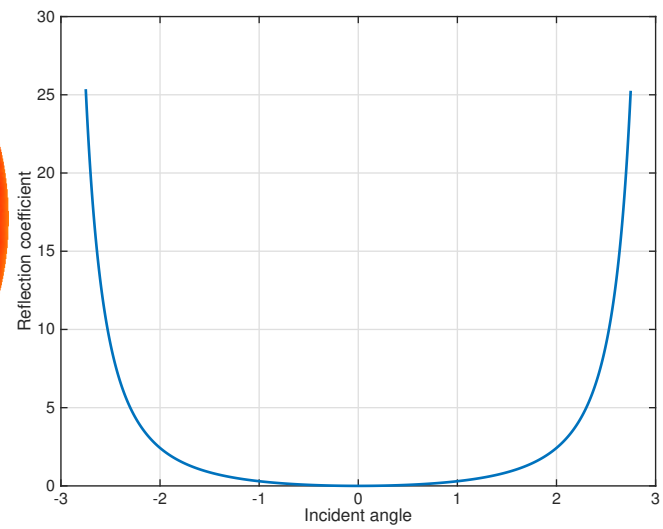

Figure B.4: Reflection coefficient of Sommerfeld ABC.

B.3 Bayliss–Turkel condition

The Bayliss–Turkel condition was commonly used between 1970 to 1980 and its second order version is currently used. It can only be applied to a circular boundary. The general two-dimensional form of this condition applied to the absorbing boundary is:

$$\left(\prod_{j=1}^{J}\left(\partial_{\mathbf{n}} - ik + \frac{4j-3}{2R}\right)\right) u = 0, \tag{B.9}$$

where J is the order of the Bayliss–Turkel condition and R is the radius of the absorbing boundary. Theoretically, the order of this method could be large, but numerically this is not the case due to the higher derivative terms that appear in the absorbing condition. Only the first and the second are used in the finite element method.

B.3.1 First order

The first order Bayliss–Turkel condition imposes $\mathcal{B} = -ik + \frac{1}{2R}$ As can be observed, the result is the same as the one obtained using the Sommerfeld condition: the maximal relative error is 15%, but the $\mathbb{L}^2$ error is smaller at 7.35%.

The reflection coefficient that can be seen in Figure B.6 is:

$$\Gamma = \left| \frac{\cos(\phi) - 1 + \dfrac{1}{2ikR}}{\cos(\phi) + 1 - \dfrac{1}{2ikR}} \right|. \tag{B.10}$$

If the product kR tends toward infinity, the first order Bayliss–Turkel becomes the same as the Sommerfeld condition. If kR is finite, it decreases the reflection for an incident wave with a larger incident angle, but creates some reflection for the normal incidence.

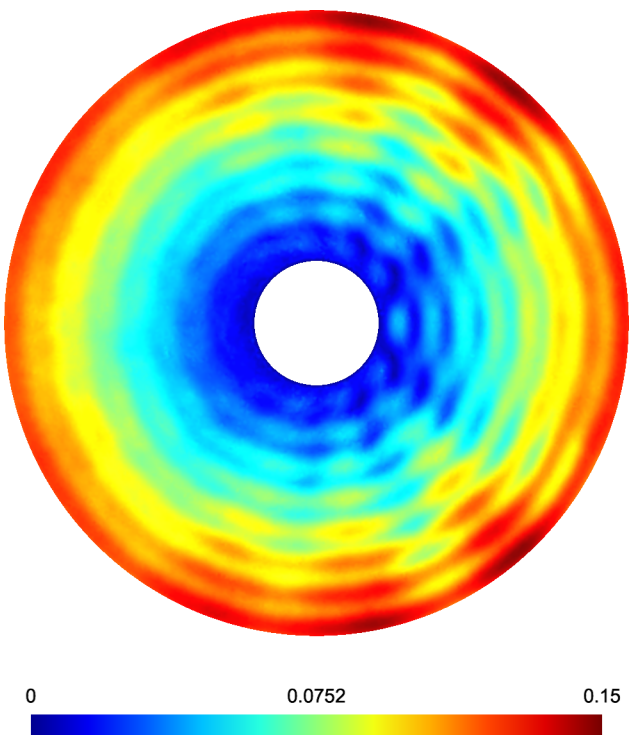

Figure B.5: Relative error induced by 1st order Bayliss–Turkel ABC.

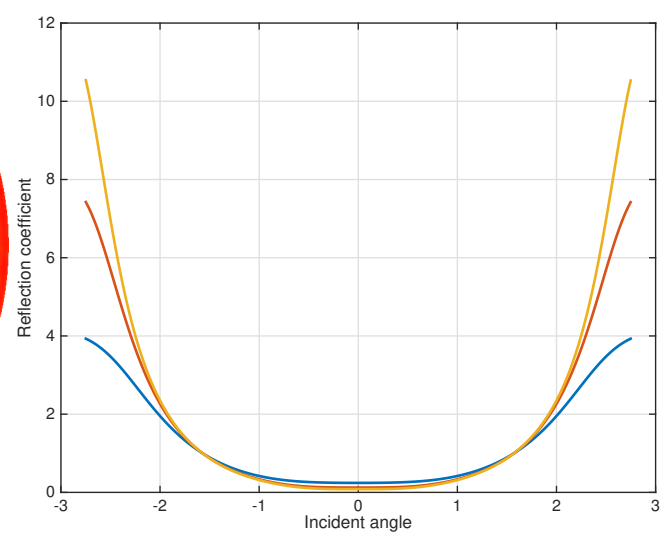

Figure B.6: Reflection coefficient of 1st order Bayliss–Turkel ABC (blue: $kR = 1$; red: $kR = 2$; yellow: $kR = 3$)

B.3.2 Second order

The main problem in the second order Bayliss–Turkel condition is expressed as:

$$\left(\partial_{\mathbf{n}} - ik + \frac{1}{2R}\right)\left(\partial_{\mathbf{n}} - ik + \frac{5}{2R}\right) u = 0 \tag{B.11}$$

in a form like Equation B.1. The following problem presents a formulation that satisfies the desired form[23]

$$\partial_{\mathbf{n}} u - iku + \alpha u + \beta \partial_{\mathbf{t}}^2 u = 0, \tag{B.12}$$

where $\mathbf{t}$ is the tangential direction of the boundary,

$$\alpha = \frac{1}{2R} - \frac{i}{8kR^2\left(1 + \frac{i}{kR}\right)} \tag{B.13}$$

and

$$\beta = -\frac{1}{2ik\left(1 + \frac{i}{kR}\right)}. \tag{B.14}$$

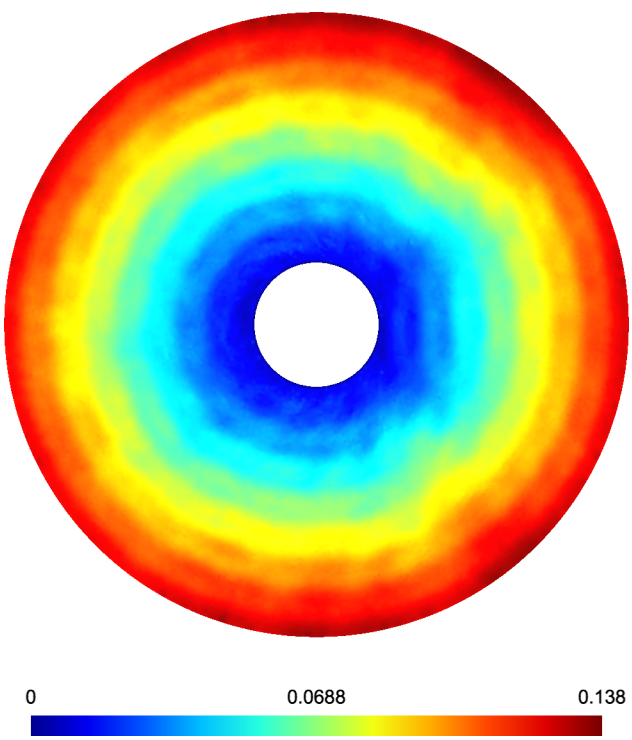

Figure B.7: Relative error induced by 2nd order Bayliss–Turkel ABC.

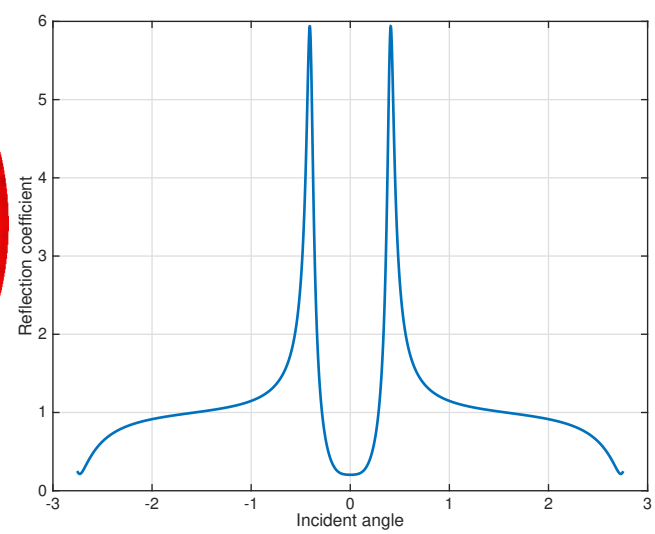

Figure B.8: Relative error of 2nd order Bayliss–Turkel ABC.

The reflection coefficient is:

$$\Gamma = \left| \frac{\cos(\phi) - 1 + \frac{\alpha i}{k} + \beta R \cos(\phi) - \beta k R^2 i \sin^2(\phi)}{\cos(\phi) + 1 - \frac{\alpha i}{k} - \beta R \cos(\phi) + \beta k R^2 i \sin^2(\phi)} \right|. \tag{B.15}$$

Figure B.8 presents the reflection coefficient using $R = 5$ and $k = 2\pi$ for the numerical simulation. The two spikes at approximately $\theta = \pm 0.4$ explains the area in Figure B.7, where the error is maximal.

Appendix C

Transmission conditions

In this Appendix, several transmission conditions (TC) applied to the non-overlapping domain decomposition Helmholtz problem are examined. Transmission conditions play an important role on the convergence of the Schwarz algorithm. This chapter complete the analysis conducted in Chapter 1 where the transmission operator $\mathcal{S}$ is defined as:

$$(\partial_{\mathbf{n}_i} + \mathcal{S})u_i^{n+1} = (\partial_{\mathbf{n}_i} + \mathcal{S})u_j^n \quad \text{on, } \Sigma_{i,j}. \tag{C.1}$$

As for the ABC this formulation is interesting because it can be directly integrated in the weak form (1.35).

C.1 Optimal transmission operator

It is possible to discover an optimal transmission operator that exactly converges in two iterations. This operator can be used for any problem of the form,

$$\begin{aligned} \mathcal{L}(e_i^{n+1}) &= 0 \quad \text{in } \Omega_i \\ e_i^{n+1} &= 0 \quad \text{on } \partial\Omega_i \cup \partial\Omega \\ \left(\frac{\partial}{\partial \mathbf{n_i}} + \mathcal{S}\right) e_i^{n+1} &= \left(\frac{\partial}{\partial \mathbf{n_i}} + \mathcal{S}\right) e_{N+1-i}^n \quad \text{on } \partial\Omega_i \cup \overline{\Omega_{N+1-i}}, \end{aligned} \tag{C.2}$$

where $\mathcal{L}$ can be any differential operator, $e_i^n = u_i^n - u$ is the error on subdomain i at iteration n and N is the number of subdomains. If an operator for which the Schwarz algorithm converges in two iterations exists, it implies that $e_i^2 = 0$, which is the case if,

$$\left(\frac{\partial}{\partial \mathbf{n_i}} + \mathcal{S}\right) e_{N+1-i}^1 = 0 \tag{C.3}$$

$$\left(-\frac{\partial}{\partial \mathbf{n_{N+1-i}}} + \mathcal{S}\right) e_{N+1-i}^1 = 0 \tag{C.4}$$

This relation is fulfilled if $\mathcal{S}$ is a new operator called the DtN (Dirichlet to Neumann) map. This operator is defined for any function u on a transmission boundary with real value by,

$$DtN_i(u) = \left.\frac{\partial v}{\partial \mathbf{n}_2}\right|_{\partial\Omega_{N+1-i}\cap\overline{\Omega}_i}, \tag{C.5}$$

with,

$$\begin{aligned} \mathcal{L}(v) &= 0 \quad \text{in, } \Omega_i \backslash \overline{\Omega}_{N+1-i}, \\ v &= 0 \quad \text{on, } \partial\Omega_i \cap \partial\Omega \\ v &= u \quad \text{on, } \partial\Omega_{N+1-i} \cap \overline{\Omega}_i. \end{aligned} \tag{C.6}$$

This operator is interesting to find and compare to other transmission operators but it is not used numerically because it is non-local and thus, computationally expensive.

C.2 Impedance transmission conditions

This family of operator is presented in Equation 1.32. This is the easier transmission condition; its general form is:

$$\mathcal{S} = -ia + b, \tag{C.7}$$

where $i^2 = -1$ and a and b are positive real numbers that can be tuned to minimize the convergence factor.

The form of its convergence factor is:

$$\rho(\tilde{k}) = \begin{cases} \left|\dfrac{\lambda(\tilde{k}) + ai - b}{\lambda(\tilde{k}) - ai + b}\right| & \forall|\tilde{k}| \geq k \\ \left|\dfrac{\lambda(\tilde{k}) - ai + b}{\lambda(\tilde{k}) + ai - b}\right| & \forall|\tilde{k}| < k \end{cases} \tag{C.8}$$

where $\lambda(\tilde{k})$ is given by Equation 1.25. It can be seen that no matter what is chosen for a or b, when the reduced Fourier number ($s = \tilde{k}/k$) is equal to one, the convergence factor will always be equal to one.

C.3 Second order impedance transmission conditions

Another family of transmission conditions uses the second order condition. This family adds a second order derivative on the tangential component of the interface, which leads to:

$$\mathcal{S} = a + b\partial_{\mathbf{t}}^2, \tag{C.9}$$

where $\mathbf{t}$ denotes the tangent direction.

Using a Fourier analyses like in Chapter 1 one can find:

$$\rho(\tilde{k}) = \begin{cases} \left| \dfrac{\lambda(\tilde{k}) - a - b\tilde{k}^2}{\lambda(\tilde{k}) + a + b\tilde{k}^2} \right| & \forall |\tilde{k}| \geq k \\ \left| \dfrac{\lambda(\tilde{k}) + a + b\tilde{k}^2}{\lambda(\tilde{k}) - a - b\tilde{k}^2} \right| & \forall |\tilde{k}| < k \end{cases} \tag{C.10}$$

Figure C.1 illustrates the convergence factor. The evanescent modes behavior is a similar manner to the one that uses the zero order impedance transmission conditions, although the propagative modes are better. Nevertheless, the convergence factor remains equal to one when $\tilde{k} = k$.

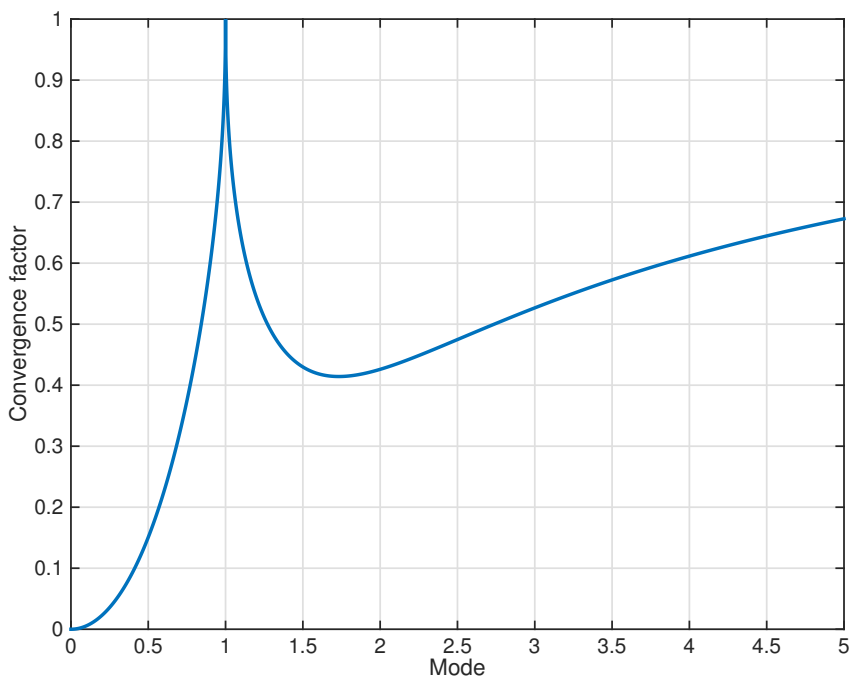

Figure C.1: Convergence factor of the problem (1.31) using the second order impedance transmission conditions with $a = -k$ and $b = 1/k$ in the case of the Helmholtz equation applied on two partitions dividing the plane.

Appendix D

Numerical simulations

D.1 Two dimensional

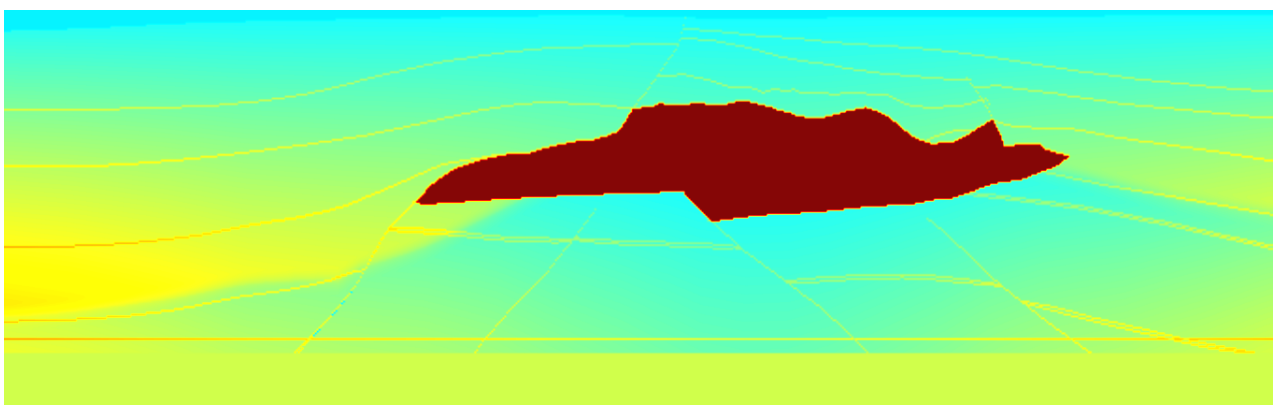

Figure D.1: Speed of sound distribution at $x = 6$ km.

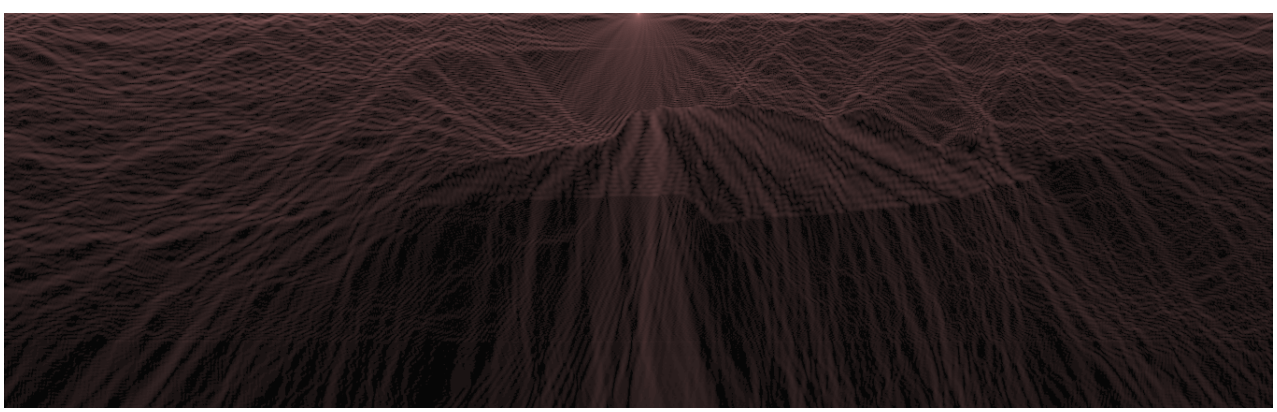

Figure D.2: Particle speed distribution at $x = 6$ km.

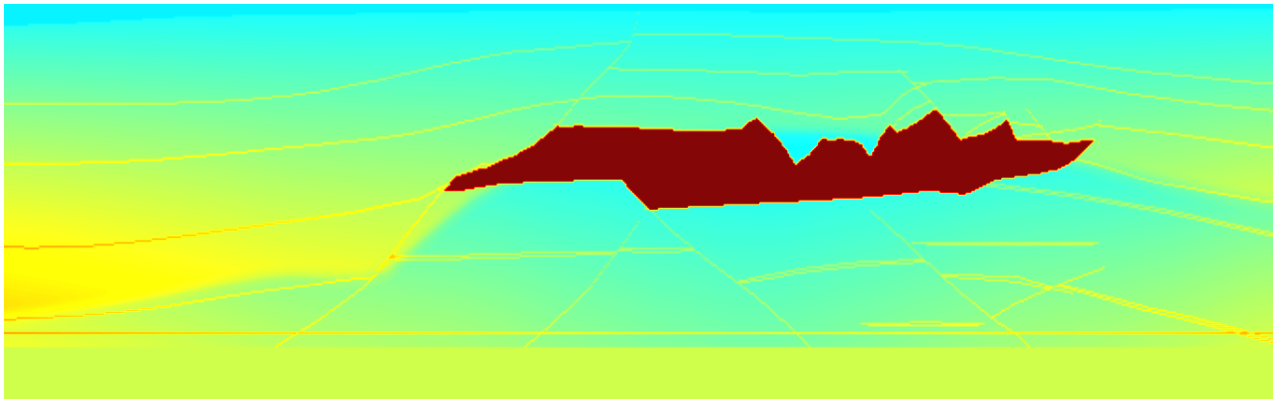

Figure D.3: Speed of sound distribution at $x = 7$ km.

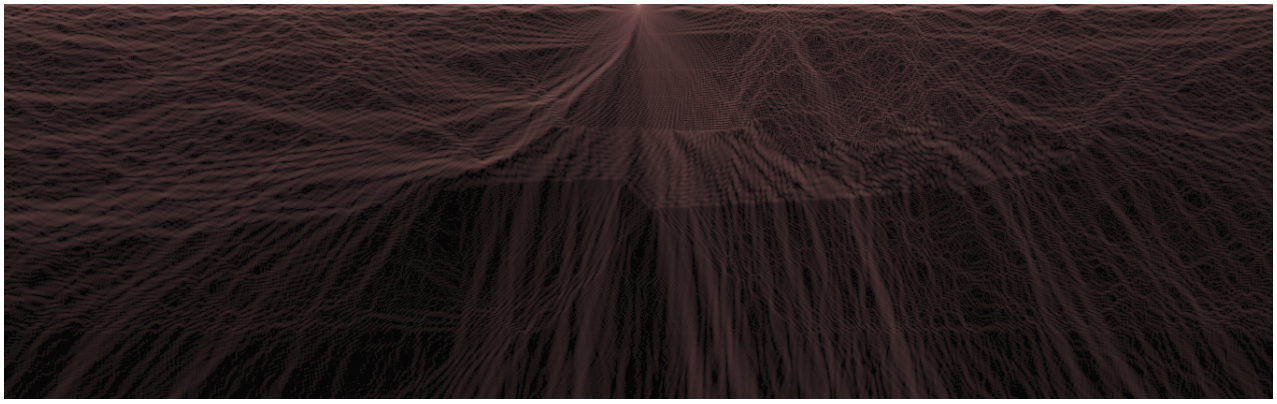

Figure D.4: Particle speed distribution at $x = 7$ km.

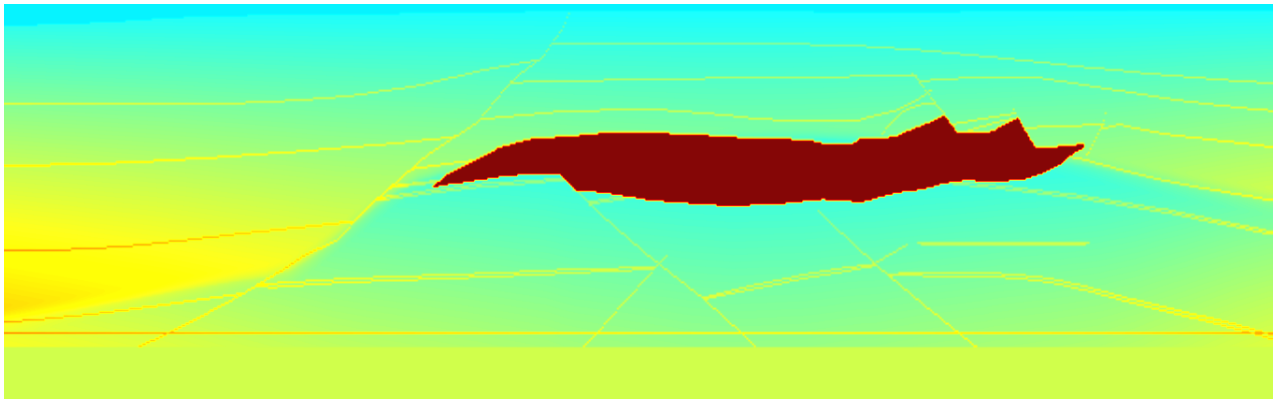

Figure D.5: Speed of sound distribution at $x = 8$ km.

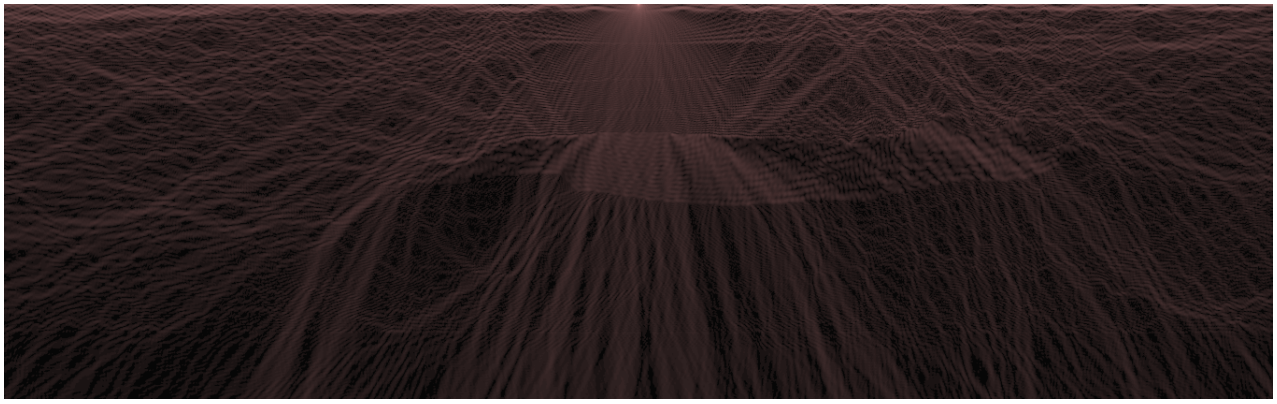

Figure D.6: Particle speed distribution at $x = 8$ km.

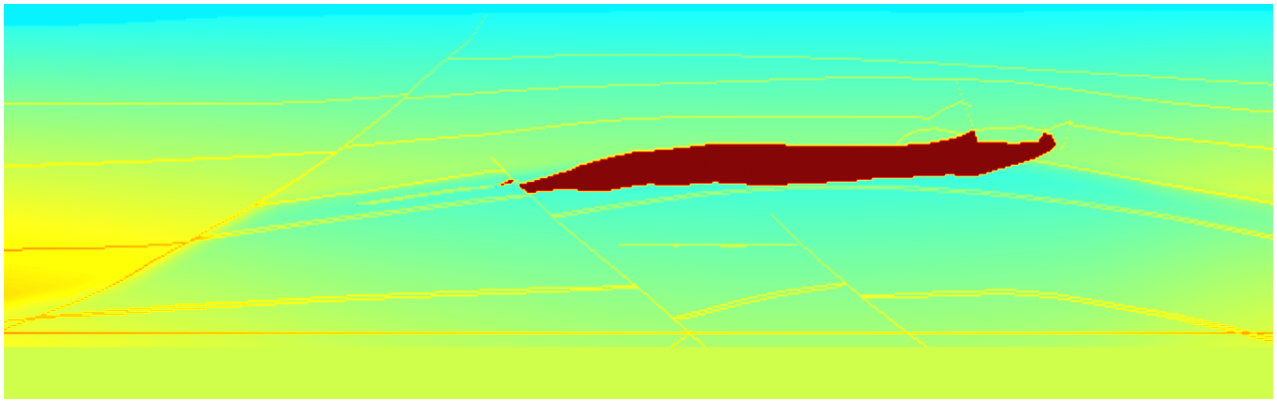

Figure D.7: Speed of sound distribution at $x = 9$ km.

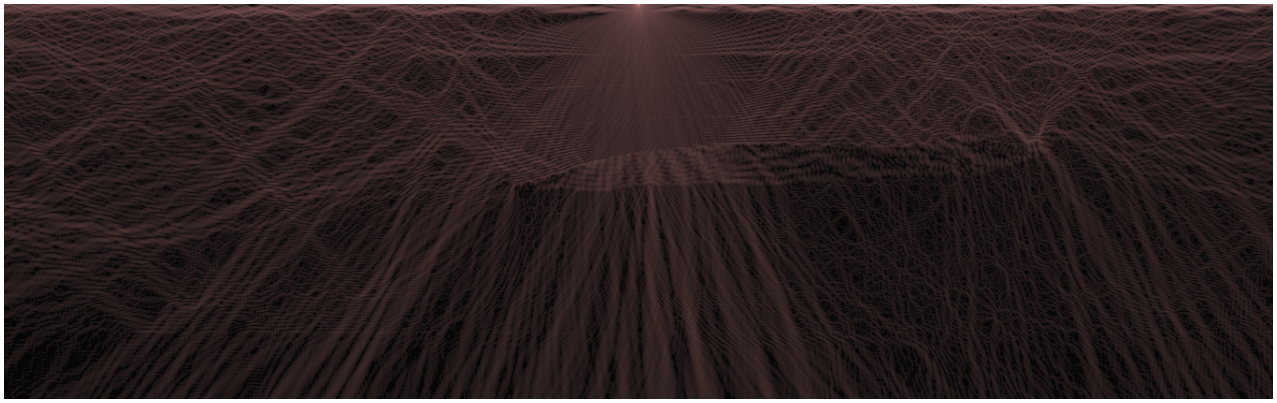

Figure D.8: Particle speed distribution at $x = 9$ km.

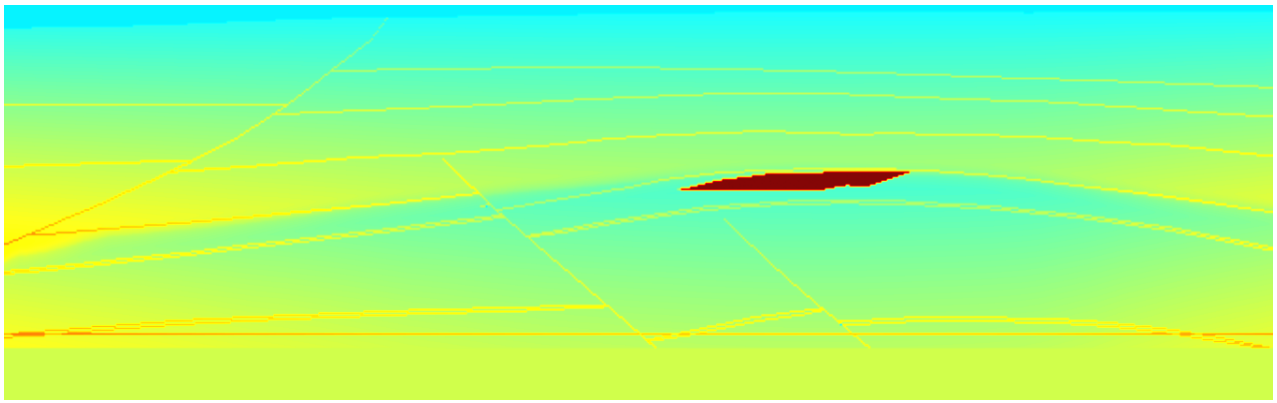

Figure D.9: Speed of sound distribution at $x = 10$ km.

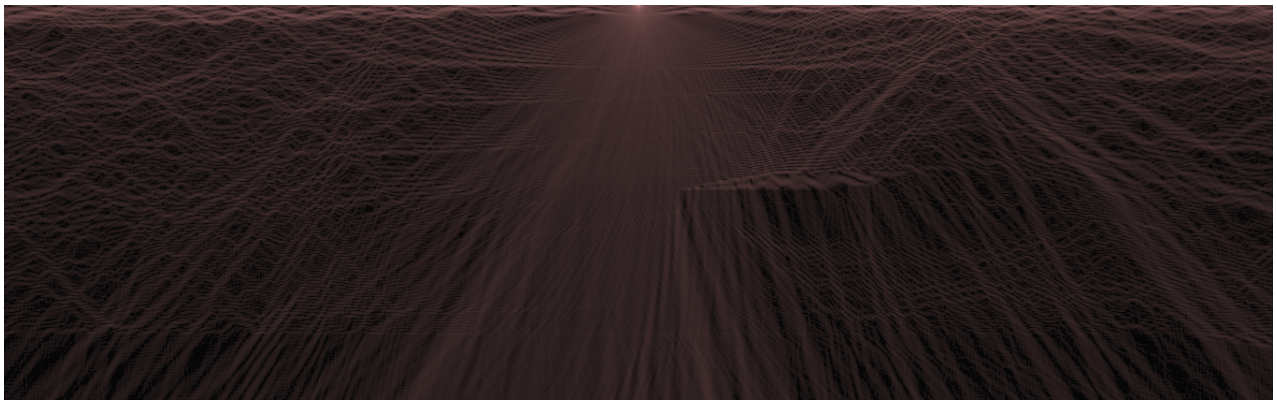

Figure D.10: Particle speed distribution at $x = 10$ km.

List of Figures

1.1 A complex geometry where the Schwarz algorithm is applied. 9

1.2 Diagram of parallel Schwarz algorithm. 10

1.3 Convergence factor of the problem (1.15) for the Helmholtz equation applied on two partitions dividing the plane ($\alpha = k$). 15

1.4 Convergence factor of the problem (1.31) using the transmission operator (1.32) with $\mathcal{S} = -ik + k$ in the case of the Helmholtz equation applied on two partitions dividing the real plane. 16

2.1 The METIS partitioning algorithm (image of George Karypis that comes from [19]). 21

2.2 Example of mesh that corresponds to the METIS mesh structure in Listing 2.1. 22

2.3 Three subdomains where the thick blue lines are the physical existing boundaries and the red ones are the partition boundaries. 25

2.4 File structure of a domain decomposition problem (the green file needs to be created by the user; the blue one contains the ready to use files and the red one contains the files generated by the partitioning tool). 33

2.5 Influence of the number of partitions on the execution time of the partitioning tool. 34

2.6 Influence of the number of nodes on the execution time of the partitioning tool. 35

3.1 Problem composed by two concentric circles. 36

3.2 Scattering wave (left) and total wave (right) corresponding to the sum of the scattering wave and the incident wave ($k = 2\pi$). 39

3.3 Error $\mathbb{L}^2$ of the scattering wave shown in Figure 3.14. The error compares the solution when the number of considered modes increases. The reference solution use many terms (here 200 terms). 40

3.4 Convergence using one, two or ten subdomains, where h is the mesh size and the error is the $\mathbb{L}^2$ relative error. 42

3.5 Relative residual norm in the function of the iteration for a different number of partitions applied on the same mesh. blue:2 partitions; red:10 partitions; yellow:20 partitions and purple:30 partitions. 43

3.6 Number of iterations required to converge in function of the number of subdomains in the two-dimensional problem. 43

3.7 Convergence in function of the mesh size in the three-dimensional case. . . . 46

3.8 Number of iterations required to converge in the function of the number of subdomains in the three-dimensional problem. 47

3.9 Convergence of the iterative solver with regular boundaries (blue line) and irregular ones (red line) and using the 0th order transmission condition. . . . 48

3.10 Convergence of the iterative solver with regular boundaries (blue line) and irregular ones (red line) and using the 2nd order transmission condition. . . . 48

3.11 Convergence of the iterative solver with regular boundaries (blue line) and irregular ones (red line) and using the Padé transmission condition. 48

3.12 Regular and irregular partitioning. 49

3.13 Strong scaling. 50

3.14 Weak scaling. 50

4.1 Ground measurement. 52

4.2 P-wave, SV-wave and SH-wave. 53

4.3 Rayleigh and Love waves. 54

4.4 Mesh partition. 55

4.5 Speed of sound distribution at $x = 5$ km. 55

4.6 Particle speed distribution at $x = 5$ km. 56

4.7 Particle speed distribution at $x = 5$ km using 5Hz and the 40 spacial step in the two-dimensional simulation. 57

4.8 Particle speed distribution at $x = 5$ km using 5Hz and the 20 spacial step in the two-dimensional simulation. 57

4.9 Particle speed distribution at $x = 5$ km using 5Hz and the 40 spacial step in the three-dimensional simulation. 58

B.1 New frame uses to compute the reflection coefficient. 64

B.2 Relative error without ABC. 65

B.3 Relative error induced by Sommerfeld ABC. 66

B.4 Reflection coefficient of Sommerfeld ABC. 66

B.5 Relative error induced by 1st order Bayliss–Turkel ABC. 67

B.6 Reflection coefficient of 1st order Bayliss–Turkel ABC (blue: $kR = 1$; red: $kR = 2$; yellow: $kR = 3$) 67

B.7 Relative error induced by 2nd order Bayliss–Turkel ABC. 68

B.8 Relative error of 2nd order Bayliss–Turkel ABC. 68

C.1 Convergence factor of the problem (1.31) using the second order impedance transmission conditions with $a = -k$ and $b = 1/k$ in the case of the Helmholtz equation applied on two partitions dividing the plane. 71

D.1 Speed of sound distribution at $x = 6$ km. 72

D.2 Particle speed distribution at $x = 6$ km. 72

D.3 Speed of sound distribution at $x = 7$ km. 73

D.4 Particle speed distribution at $x = 7$ km. 73

D.5 Speed of sound distribution at $x = 8$ km. 73

D.6 Particle speed distribution at $x = 8$ km. 74

D.7 Speed of sound distribution at $x = 9$ km. 74

D.8 Particle speed distribution at $x = 9$ km. 74

D.9 Speed of sound distribution at $x = 10$ km. 75

D.10 Particle speed distribution at $x = 10$ km. 75

Bibliography

[1] Jean-David Benamou and Bruno Desprès. A domain decomposition method for the helmholtz equation and related optimal control problems. HAL archives ouvertes, https://hal.inria.fr/inria-00073899, May 2006.

[2] Y. Boubendir, X. Antoine, and C. Geuzaine. A quasi-optimal non-overlapping domain decomposition algorithm for the helmholtz equation. *Journal of Computational Physics*, 231:262 – 280, 2012.

[3] Y. Boubendir, A. Bendali, and M. B. Fares. Coupling of a non-overlapping domain decomposition method for a nodal finite element method with a boundary element method. *International journal for numerical methodes in engineering*, 2008.

[4] Yassine Boubendir. An analysis of the bem-fem non-overlapping domain decomposition method for a scattering problem. *Journal of Computational and Applied Mathematics*, 204:282 – 291, 2007.

[5] Aliénor Burel. *Contributions à la simulation numérique en élastodynamique : découplage des ondes P et S, modèles asymptotiques pour la traversée de couches minces*. PhD thesis, Université Paris-Sud, 2014.

[6] T.F. Chan and J. Zou. Additive schwarz domain decomposition methods for elliptic problems on unstructured meshes. *Numerical algorithms*, 8:329–346, June 1994.

[7] David Colton and Rainer Kress. *Inverse Acoustic and Electromagnetic Scattering Theory*. Springer New York Heidelberg Dordrecht London, 3 edition, 2013.

[8] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. An introduction to domain decompostion methods: algorithms, theory ans parallel implementation. HAL archives ouvertes, https://hal.archives-ouvertes.fr/cel-01100932, January 2015.

[9] Guy Drijkoningen. Course materials: Introduction to reflection seismics. `https://ocw.tudelft.nl/courses/introduction-reflection-seismics/`.

[10] Patrick Dular and Christophe Geuzaine. GetDP reference manual: the documentation for GetDP, a general environment for the treatment of discrete problems. `http://getdp.info`.

[11] Martin J. Gander, Frédéric Magoulès, and Frédéric Nataf. Optimized schwarz methods without overlap for the helmholtz equation. *SIAM J. Sci. Comput.*, 24:38 – 60, 2002.

[12] Christophe Geuzaine and Jean-François Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79:1309–1331, September 2002.

[13] Dam Givoli. *Numerical Methods for Problems inInfinite Domains.* Elsevier, 1992.

[14] Dan Givoli. Computational absorbing boundaries. In Steffen Marburg and Bodo Nolte, editors, *Computational Acoustics of Noise Propagation in Fluids*, pages 145 – 166. Springer Berlin Heidelberg, 2008.

[15] Dan Givolir. High-order local non-reflecting boundary conditions: a review. *Wave Motion*, 39:319 – 326, 2004.

[16] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. *Proceedings of Super-computing*, 1998.

[17] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48:96 – 129, 1998.

[18] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20:359 – 392, 1999.

[19] George Karypis. *A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices.* Department of Computer Science and Engineering - University of Minnesota, March 2013.

[20] Matthieu Lecouvez. *Méthodes itératives de décomposition de domaine sans recouvrement avec convergence géométrique pour l'équation de Helmholtz.* PhD thesis, Université de Paris-Saclay, 2015.

[21] Frédéric Nataf. Absorbing boundary conditions and perfectly matched layers in wave propagation problems. In Ivan Graham, Ulrich Langer, Jens Melenk, and Mourad Sini, editors, *Direct and Inverse Problems in Wave Propagation and Applications*, pages 219 – 231. 2013.

[22] Omar M. Ramahi. Exact implementation of higher order bayliss–turkel absorbing boundary operators in finite-element simulation. *IEEE Microwave and guided wave letters*, 9:360 – 362, November 1998.

[23] B. Thierry, A.Vion, S. Tournier, M. El Bouajaji, D. Colignon, N. Marsic, X. Antoine, and C. Geuzaine. Getddm: an open framework for testing optimized schwarz methods for time-harmonic wave problems. *Computer Physics Communications*, 203:309–330, June 2016.

[24] Andrea Toselli and Olof Wildund. *Domain Decomposition Methods : Algorithms and Theory.* Springer, 2005.