Trefftz-DG Methods and Tent-Pitcher Algorithm for Spacetime Integration of Wave Problems

Vinduja Vasanthan

THÈSE DE DOCTORAT

Mathématiques Appliquées

Université de Pau et des Pays de l'Adour
Laboratoire de Mathématiques et de leurs Applications UMR CNRS 5142
INRIA Bordeaux Sud-Ouest Équipe-Projet Makutu

École Doctorale des Sciences Exactes et leurs Applications – ED 211

Trefftz-DG Methods and Tent-Pitcher Algorithm for Spacetime Integration of Wave Problems

–

Méthodes Trefftz-DG et algorithme Tent-Pitcher pour la résolution des équations d'ondes dans le domaine espace-temps

---

Vinduja Vasanthan

Membres du jury :

M. Reza Abedi	Associate Professor, University of Tennessee Space Institute	Examinateur
Mme Hélène Barucq	Directrice de Recherche, INRIA	Directrice
M. Julien Diaz	Directeur de Recherche, INRIA	Directeur
M. Stefano Frambati	Ingénieur de Recherche, TotalEnergies	Examinateur
M. Philippe Helluy	Professeur, Université de Strasbourg	Rapporteur
M. Alexandre Impériale	Ingénieur Chercheur, CEA Saclay	Examinateur
M. Sébastien Pernet	Maître de Recherche, Onera	Rapporteur
M. Philippe Poncet	Professeur, Université de Pau et des Pays de l'Adour	Président

Remerciements

Je souhaite commencer par remercier Hélène Barcuq et Julien Diaz, mes directeurs de thèse, sans qui cette thèse n'aurait pas pu voir le jour et qui m'ont accompagnée et conseillée tout au long de cette thèse. Merci Hélène pour ton aide pointilleuse lors de la rédaction et pour la rigueur mathématique que tu as apporté ainsi que tes nombreuses idées. Merci Julien pour ton aide informatique qui a servi lors de nos nombreuses séances de debug, ainsi que ton aide sur le plan mathématique. Merci à tous les deux de m'avoir aidée à contourner les nombreuses embûches qui se sont trouvées sur le chemin. Je remercie également Henri Calandra qui a contribué à la mise en place de ce sujet dans ses débuts et a permis à ce sujet d'exister.

Ensuite, mes remerciements vont à Philippe Helluy et Sébastien Pernet, qui ont accepté d'être rapporteurs de ma thèse et pour leurs remarques et questions pertinentes. Merci pour vos rapports exhaustifs.

Je tiens également à remercier Philippe Poncet d'avoir accepté de participer à mon jury et de l'avoir présidé.

I would like to thank Reza Abedi, whom I exchanged with during conferences and helped me through our interesting conversations and relevant questions to grasp finer notions about tent-pitching, and accepted to be part of the jury.

Merci également à Alexandre Impériale d'avoir accepté de faire partie du jury et d'avoir fait le déplacement à Pau pour assister à ma soutenance.

Merci Stefano, tout d'abord d'avoir accepté de faire partie du jury et également pour tout l'intérêt que tu as pu porter à ma thèse durant ces trois années à travers nos discussions scientifiques.

À tous les membres du jury, merci pour vos nombreuses questions qui donneront beaucoup de perspectives intéressantes à ce sujet et qui m'ont montré l'intérêt que vous avez pu y porter.

Merci à tous ceux qui ont assisté à ma soutenance que ce soit sur place ou à distance.

Ensuite, je souhaite remercier mes collèges, que ce soit les anciens doctorants, les nouveaux, les post-docs, les ingénieurs, les permanents : merci pour les moments de convivialité, nécessaires durant une thèse et bon courage à ceux qui n'ont pas fini la leur. Merci à toutes celles avec qui on discutait cuisine et celles qui, en cette fin de thèse, m'ont épaulée au travers de longues conversations (et des cocktails avec certaines), j'espère qu'elles se reconnaîtront toutes.

Merci Isabelle et Pierre, vous avez été un rayon de soleil dans cette petite rue piétonne de Pau et, je suis sûre, le serez partout où vous irez.

Merci à mes amis qui ont supporté mes plaintes nombreuses durant trois ans, qui ont assisté à ma soutenance à distance et qui ne savaient jamais comment venir me voir à Pau.

Enfin, merci à ma famille, je ne serai pas qui je suis ni où j'en suis sans elle. Merci Krishna et Daya d'avoir traversé la France pour venir à ma soutenance, c'est loin d'être la première fois que vous êtes là pour moi et probablement pas la dernière. Peu importe ce que je dis, vous savez que j'en serai toujours reconnaissante.

Last but not least, merci Jérémy pour absolument tout. Tu as vécu cette thèse avec moi, dans les hauts mais surtout les bas, et peu savent tout ce que tu as fait pour moi surtout en cette fin de thèse. Merci de m'avoir donné le sourire et de continuer à le faire.

Chapter 1

General Introduction

The numerical simulation of waves find their applications in many fields, such as mechanical engineering, medical imaging, musical instrument modeling, non-destructive testing, seismology, seismic imaging and more. This thesis falls within the context of seismic imaging, which consists in collecting underground data in order to obtain a map of the surface of the earth thanks to numerical simulations based upon them. As data are reflected waves generated by sources propagating into the domain of interest, it belongs to the family of seismic reflection techniques. The numerical study of seismic waves is thus of great importance. They are waves propagating through the earth and can be classified into body and surfaces waves. Body waves comprise direct, reflected and refracted waves and propagate through the whole medium, while the surface waves only travel on the surface. We can distinguish two body waves: the primary P-waves and the secondary S-waves and two main types of surface waves: Rayleigh waves and Love waves. P-waves are longitudinal waves, which means that they oscillate parallel to the direction of propagation. They are a particular type of elastic waves, travel faster than S-waves and can propagate through any kind of medium. S-waves are transverse waves, so they oscillate perpendicular to the direction of propagation. On the contrary to primary waves, they can only propagate in solids. The direction of oscillation of Rayleigh

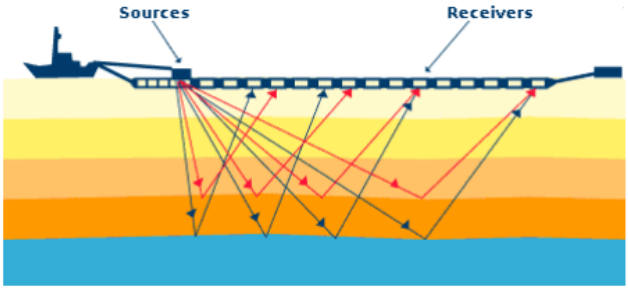


Figure 1.1: Seismic imaging[1]

[1]Source: [1] Simulation de la propagation d'ondes élastiques en domaine fréquentiel par des méthodes Galerkine discontinues, M. Bonnasse-Gahot, 2015

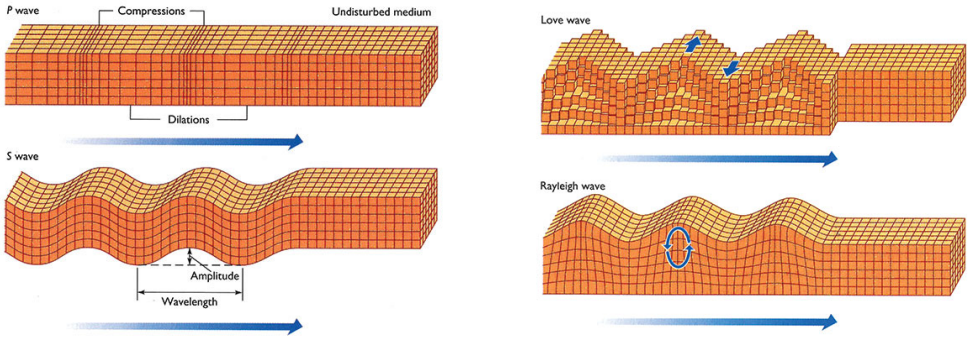


Figure 1.2: Classification of seismic waves[2]

waves is a combination of the latter two. Love waves travel in a transverse horizontal motion, so they oscillate perpendicular to the direction of propagation. Surface waves usually travel slower than body waves.

In a seismic acquisition, artificial sources emit waves towards the earth, then receivers, usually placed on the surface, record the reflected waves as can be seen in Fig. 1.1. The collected information is generally the arrival time and the amplitude of the reflected wavefields. We can construct a map of the variations of velocity in the medium by using the results of seismic imaging campaigns, that can be done in the sea or on the ground. This velocity map is also referred to as the velocity model and is more or less precise, depending on the number of sources used. To reconstruct the collected data and obtain a map of the underground along with its material characteristics, several numerical techniques have been developed, which distinguish themselves by the precision of the reconstruction. The most widely used seismic imaging methods are the *Reverse Time Migration* (RTM) and the *Full Waveform Inversion* (FWI), which both rely on the numerical resolution of wave equations. The RTM is a migration technique, which consists of three steps: (a) forward modeling of the wave field in a domain with an appropriate velocity model in order to obtain the propagated field, (b) extrapolating the obtained solution back in time by using the seismic data as an initial condition in order to obtain the back-propagated field and (c) superposing the propagated and back-propagated fields in order to obtain the image of the subsurface. The position of the reflectors is deduced from the positions where both fields coincide. More information on the RTM in the time-domain can be found in [2, 3] and in the frequency-domain in [4, 5]. The FWI is an inversion technique, which provides quantitative information on the propagation medium as the RTM retrieves qualitative information. It uses the full wavefield in order to obtain a high-resolution seismic imaging. Its implementation involves an iterative process to perform the minimization of a misfit function evaluating the difference between numerical data and observations until convergence to the propagation domain. It is a computationally intensive method based upon a local minimization procedure (*e.g.* least squares, gradient descent, steepest descent) involving the adjoint method. The

[2]Source: `https://iris.edu/`

solution of the forward problem is then at the heart of the algorithm as it describes the direct and adjoint problems. It is worth noting that as it addresses the solution of an ill-posed nonlinear inversion problem, FWI is an intensive research subject. The physical parameters are updated at each iteration until the model converges and it is crucial to dispose of accurate and affordable numerical methods for solving the forward problem. An overview of FWI by Virieux and Operto can be found in [6] and by Brittan *et al.* in [7]. It has been applied in the frequency-domain by Brossier *et al.* in [8] and in the time-domain by Pratt *et al.* in [9]. Working in the frequency domain has the advantage of easing the inversion of complex-valued parameters. Moreover, the time-harmonic solution methodology includes direct solver uses which allow multi-right-hand-side implementation. However since the resulting discrete matrix tends to be very large, it is necessary to consider frugal numerical methods as Hybridizable Discontinuous Galerkin (HDG) approximations (see [1, 10]) and even, large-scale computations are still unreachable which makes time-domain approaches good candidates to consider real applications. Time-domain approaches have indeed the capability of addressing very large domains as long as High Performance Computing is performed, and nowadays, the main challenges in this field are focused on the efficiency and performance of the forward solver, the management of 3D domains and data and the construction of velocity models.

In this thesis, we are interested in the efficiency and the performance of the forward solver. In particular, when considering the FWI, the method is sensitive to the accuracy of the propagated field in the sense that any error in the numerical simulation can transform itself into an artifact. Hence, this motivates the research of accurate and fast time-domain methods. The usual framework when solving time-domain problems consists in separating the problem in space and time. So, we discretize the spatial domain first to get a semi-discrete problem formulated with *Finite Element* methods, then we use an iterative method (such as Runge-Kutta methods) to solve it in time. This approach has been used in the framework of *Discontinuous Galerkin* (DG) and Runge-Kutta methods in [11] and is presented for Finite Elements in [12] and [13]. For linear problems this approach is proven to be successful, however for non-linear problems, it presents several disadvantages. For example, when considering a problem presenting discontinuities in the solution, such methods do not capture the discontinuities accurately and instead present oscillations near them, on the contrary to spacetime discretizations. For instance, this phenomenon is shown for the case of the impact of a one-dimensional homogeneous elastic beam against a rigid wall by Hulbert and Hughes in [14]. Another advantage of spacetime discretization over semi-discretization with time-stepping lies in the use of fully unstructured meshes in space and time. When working with semi-discrete methods, the mesh comprises cells structured in time, because the same time discretization is used for every time step. In the case of spacetime discretizations, we can work on unstructured meshes in both time and space. Hence, if there is a need to capture discontinuities, such as stresses, a defect, or interfaces, there is the possibility to refine the mesh along those discontinuities, or even adaptively refine the mesh through time following the location of the discontinuities. Moreover, spacetime methods naturally

allow local time-stepping when the time step is constrained by the space step, which is advantageous. For all these reasons, many have explored spacetime finite element methods, such as [14, 15, 16, 17, 18]. However, the question of spacetime meshing needs to be addressed and can be complex and expensive in terms of memory. In fact, spacetime problems have an additional dimension (nD+t) compared to space-only problems (nD). In particular when considering 3D domains, the computational costs and the memory consumption will sharply increase.

This work is carried out in the context of the collaborative research program Depth Imaging Partnership (DIP), which was established between the company Total Energies and the Inria project-team Makutu between 2010 and 2022. The DIP project aimed at developing high-order numerical schemes, for seismic imaging and underground structure reconstruction. Within DIP project, the numerical methods were mostly based on discontinuous finite element approximation of the wave fields, like in Discontinuous Galerkin methods, because they can take into account geometrical and physical characteristics of the domain of interest and are conducive to parallel computing [19, 20, 21]. However, DG methods suffer from a high computational cost due to the number of degrees of freedom which is doubled at the interfaces, when compared to classical Finite Element methods for example. Thus, a wide range of research is interested in reducing the cost of DG methods and, at the same time, maintaining their benefits. Having that in mind, the HDG methods have been investigated in the context of the project DIP. They consist in expressing the solution vector in terms of a Lagrange multiplier, which represents the trace of the numerical solution on the skeleton of the mesh. The number of degrees of freedom is thus reduced in comparison to DG methods, without losing its advantages [22].

In the same spirit of reducing cost and improving the precision of the solution, we consider time-domain Trefftz methods, which were introduced by Trefftz in [23] and have been investigated in [24] in the context of the project DIP, and for a spacetime acoustic wave equation in [25, 26, 27] and various other problems in [28, 29, 30]. Trefftz methods have been widely explored in the frequency-domain for hyperbolic problems in [31], for plate bending and thick plate problems in [32], for the two-dimensional Helmholtz problem and three-dimensional Maxwell problem in [33, 34]. They consist in using local solutions to the considered problem as basis functions and we call them Trefftz functions (or T-functions). Thus, the numerical solution is expected to be more precise, because characteristics of the Trefftz functions, such as the oscillatory aspect or the wavenumber, will be injected into the approximation space itself. Moreover, in the case where the system of equation is self-adjoint, the resulting variational formulation is only posed on the skeleton of the mesh as the volumic terms vanish. Hence, the number of degrees of freedom is greatly reduced compared to DG methods. So, the Trefftz method is a good candidate in the research of a faster while accurate method. However, the numerical scheme we obtain in time-domain is implicit and thus, we have a large matrix to invert and the computational cost is increased anyway. This problem is due to the fact that we work in time-domain, and so the Trefftz functions are spacetime solutions.

Hence, the method used to solve our problem naturally becomes a spacetime method and the implementation of the method requires applying a spacetime integration. In [24], E. Shishenina proposed to overcome this drawback by using the Tent-Pitcher algorithm. This algorithm was introduced by Üngör and Sheffer in [35] for spacetime hyperbolic problems, and then generalized by Erickson in [36]. It consists in meshing the spacetime domain element-by-element, and then solving the problem inside these elements, which are polygonal-based pyramids or less formally called tents. To be able to transform our implicit scheme into a locally explicit one with this algorithm, the mesh needs to follow certain constraints which are referred to as the causal constraint and the progress constraint. Under these conditions, we are able to obtain a locally explicit scheme and thus, reduce the computational cost. This algorithm has been explored in many different applications, such as for the wave equation by Richter in [37], for hyperbolic conservation laws by Lowrie *et al.* in [38], by Yin *et al.* for elastodynamics analysis in [39], by Monk and Richter for linear time dependent hyperbolic problems written as a symmetric system [40], by Miller and Haber for hyperbolic heat conduction in [41], by Gopalakrishnan *et al.* for deriving an explicit spacetime Finite Element scheme in [42], by Howard for deriving an asynchronous spacetime Discontinuous Galerkin solver in 3D+t in [43], by Moiola and Perugia in [26] and Stocker in [27] for the acoustic wave equation and by Shishenina for acoustic, elastic and elastoacoustic wave propagation in [24]. The most recent advances for the Tent-Pitcher algorithm concern the derivation of cylindrical elements to which the tents of a Tent-Pitching spacetime mesh are mapped by Gopalakrishnan in [44, 45]. The aim of this work is to be able to separate space and time and apply classical methods in space combined with high-order time-stepping methods. Moreover in the Tent-Pitcher algorithm, only the mesh front needs to be stored instead of the whole spacetime mesh, which frees us from the drawbacks of spacetime meshing without losing any advantages, such as spacetime mesh refinement. Many have explored the spacetime mesh refinement in the framework of the Tent-Pitching meshes ([46, 47, 48, 49, 50, 51]), which is a very interesting feature for nonlinear equations but also for seismic wave equations when different interfaces are present.

This thesis is composed of two main parts. The Part I introduces a Trefftz-DG solver with Tent-Pitching applied to spacetime acoustic wave equations. We first present the Trefftz-DG methods and the Tent-Pitcher algorithm and then derive the associated variational formulation. Next, we describe the implementation in details for structured meshes and its extension to unstructured meshes. Finally, we adapt it to a parallel environment. We present numerical tests and results along with comparisons with a classical Interior Penalty Discontinuous Galerkin method. The Part II addresses the question of boundary conditions, in particular by introducing Perfectly Matched Layers in the previously derived Trefftz-DG solver with Tent-Pitching. We analytically compute solutions to the acoustic wave equation with Perfectly Matched Layers and use it to solve the problem with the Trefftz-DG and Tent-Pitching method. To do so, we derive several variational formulations and then present their implementation in details, along with

numerical tests and results.

Scientific communications

- Spacetime Trefftz-DG formulation for elasto-acoustic wave propagation using Tent-Pitching meshes, ECCOMAS – WCCM, 2020
- PML applied to spacetime Trefftz-DG numerical formulation for the acoustic wave equation, ICOSAHOM, 2020
- PML applied to spacetime Trefftz-DG numerical formulation for the acoustic wave equation, WINE, 2021 (POSTER)
- PML applied to spacetime Trefftz-DG numerical formulation for the elastic wave equation, Mathias Days by TotalEnergies R&D, 2021
- Spacetime Trefftz-DG formulation for modelling wave propagation in unbounded domains, ECCOMAS, 2022
- On the construction of shape functions for spacetime Trefftz-DG formulations of wave problems with Perfectly Matched Layers, WAVES, 2022

Chapter 2

Introduction Générale

La simulation numérique des ondes trouve des applications dans de nombreux domaines, tels que l'ingénierie mécanique, l'imagerie médicale, la modélisation des instruments de musique, le contrôle non destructif, la sismologie, l'imagerie sismique et plus encore. Cette thèse s'inscrit dans le contexte de l'imagerie sismique, qui consiste en la collecte de données souterraines dans le but de cartographier la surface terrestre grâce à des méthodes numériques utilisant celles-ci. Puisque ces données sont des ondes générées par des sources se propageant et se réflechissant dans le domaine d'intérêt, ces méthodes s'inscrivent dans la famille des techniques de réflexion sismique. Ainsi, l'étude des ondes sismiques revêt une importance cruciale. Ces ondes se propagent à travers le sol et peuvent être des ondes de volume ou de surface. Les ondes de volume peuvent être directes, réfléchies, ou réfractées et se propagent dans la matière, tandis que les ondes de surface se propagent uniquement à la surface. Il existe deux types d'ondes de volume : l'onde primaire P et l'onde secondaire S, ainsi que deux principales catégories d'ondes de surface : les ondes de Rayleigh et les ondes de Love. Les ondes P sont longitudinales, elles oscillent donc parallèlement à la direction de propagation. Il s'agit d'ondes élastiques particulières, qui avancent plus rapidement que les ondes S et peuvent se propager dans tous types de milieux. Les ondes secondaires oscillent per-

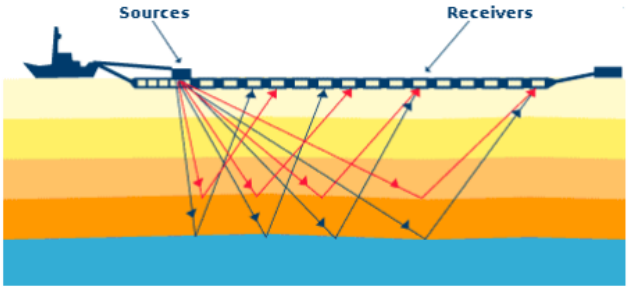


Figure 2.1: Acquisition sismique[1]

[1]Source: [1] Simulation de la propagation d'ondes élastiques en domaine fréquentiel par des méthodes Galerkine discontinues, M. Bonnasse-Gahot, 2015

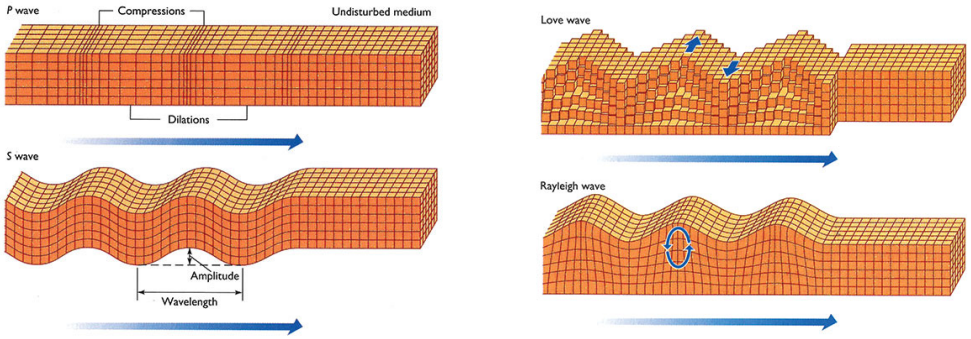


Figure 2.2: Classification des ondes sismiques[2]

pendiculairement à la direction de propagation et, contrairement aux ondes primaires, elles se propagent uniquement dans des matières solides. Le mouvement des ondes de Rayleigh est une combinaison de ces deux types d'ondes. Les ondes de Love, quant à elles, suivent un mouvement transverse horizontal, elles oscillent donc perpendiculairement à la direction de propagation. Les ondes de surface se propagent généralement plus lentement que les ondes de volume.

Lors de l'acquisition sismique, des sources artificielles émettent des ondes en direction du noyau interne, puis les ondes réfléchies sont réceptionnées par des capteurs généralement placés à la surface, comme illustré sur la Fig.2.1. Les informations collectées sont le plus souvent l'amplitude et le temps d'arrivée des ondes réfléchies. Il est alors possible de construire une carte de vitesse dans le milieu à partir des résultats de la campagne d'imagerie sismique, qui peut être menée aussi bien dans l'eau que dans le sol. Cette carte de vitesse, aussi appelée modèle de vitesse, est plus ou moins précise suivant le nombre de sources utilisées. Pour obtenir une carte du sous-sol et de ses caractéristiques physiques, plusieurs méthodes numériques ont été développées et se distinguent par la précision de la reconstruction. Les méthodes d'imagerie sismique les plus utilisées sont la *Reverse Time Migration* (RTM) et la *Full Waveform Inversion* (FWI), qui reposent toutes les deux sur la résolution numérique de l'équation des ondes. La RTM est une technique de migration profondeur qui se compose de trois étapes: (a) propagation du problème direct avec un modèle de vitesse approprié pour obtenir le champ propagé, (b) rétro-propagation de la solution obtenue en utilisant les données récoltées comme conditions initiales pour obtenir le champ rétro-propagé et (c) obtention de l'image du sous-sol par superposition des champs propagé et rétro-propagé. La position des récepteurs est déterminée à l'intersection des deux champs. De plus amples informations sur la RTM en domaine temporel peuvent être obtenues dans [2, 3], et dans [4, 5] pour le domaine fréquentiel. La FWI est une technique d'inversion qui fournit des informations quantitatives sur le milieu de propagation, au contraire de la RTM dont les informations extraites sont qualitatives. Cette technique utilise le champ d'ondes complet afin d'obtenir une image sismique à haute résolution. Sa mise en œuvre se fonde sur un procédé itératif

[2]Source: `https://iris.edu/`

permettant la minimisation d'une fonction coût, qui évalue la différence entre les données numériques et les observations jusqu'à obtenir convergence vers le domaine de propagation. Le processus calculatoire de cette méthode est intensif, et repose sur une procédure de minimisation (*e.g.* moindres carrés, descente de gradient, steepest descent) qui implique la méthode adjointe. La solution du problème direct est donc au coeur de cet algorithme, puisqu'elle décrit les problèmes direct et adjoint. Il convient de noter que la FWI fait l'objet de recherches intensives en raison du caractère mal-posé des problèmes d'inversion non-linéaires dont elle traite. Les paramètres physiques sont mis à jour à chaque étape jusqu'à convergence du modèle et il est crucial de disposer de méthodes numériques précises et à faible coût pour résoudre le problème direct. Nous retrouvons une présentation de la FWI par Virieux et Operto dans [6], ainsi que par Brittan *et al.* dans [7]. Cette méthode a été appliquée en domaine fréquentiel par Brossier *et al.* dans [8] et en domaine temporel par Pratt *et al.* dans [9]. Travailler dans le domaine fréquentiel facilite l'inversion de paramètres à valeurs complexes. De plus, la méthodologie de calcul de solution en régime harmonique intègre l'utilisation d'un solveur direct qui permet une implémentation avec multiples second membres. Cependant la matrice discrète obtenue étant généralement très grande, il est nécessaire de considérer des méthodes numériques plus économes comme les approximations Galerkine Discontinues Hybrides (HDG) (voir [1, 10]) et malgré cela, certains calculs à grande échelle restent inaccessibles, ce qui rend les approches en régime temporel attrayantes pour étudier des applications réelles. Les méthodes en domaine temporel permettent en effet de traiter de très grands domaines en tirant partie du Calcul Haute Performance et, de nos jours, les principaux défis de ce domaine portent sur l'efficacité et la performance du solveur direct, la gestion des données et des domaines en 3D ainsi que la construction des modèles de vitesse.

Dans cette thèse, nous nous intéressons à l'efficacité et la performance de la résolution du problème direct. Ces propriétés sont particulièrement importantes dans le cas de la FWI, qui est très sensible à la précision du champ propagé, dans la mesure où une faible erreur dans la simulation numérique peut se transformer en un artefact. Ainsi, cela motive la recherche de méthodes en domaine temporel plus rapides et plus précises. L'approche la plus courante pour la résolution de problèmes en domaine temporel consiste à séparer le problème en espace et en temps. Le domaine spatial est d'abord discrétisé pour obtenir un problème semi-discret formulé dans le cadre des méthodes de type *Éléments Finis*. Ensuite, une méthode itérative explicite (*e.g.* Runge-Kutta) est utilisée pour résoudre le problème en temps. Cette construction, utilisant les méthodes *Galerkine Discontinues* (DG) et Runge-Kutta, est présentée dans [11], ainsi qu'utilisant les méthodes Éléments Finis dans [12] et [13]. Cette approche s'est révélée efficace pour les problèmes linéaires, en revanche elle présente de nombreux désavantages pour les problèmes non linéaires. Par exemple, lorsque la solution du problème posé présente des discontinuités, cette approche ne parvient pas à capturer celles-ci et produit des oscillations à proximités des discontinuités, contrairement aux méthodes de discrétisation espace-temps. Ce phénomène est illustré dans le cas de l'impact d'une poutre élastique homogène en une dimension contre un mur rigide par Hulbert et Hughes dans [14]. Un

autre avantage de la discrétisation espace-temps face à la semi-discrétisation avec un pas de temps provient de l'usage d'un maillage totalement non-structuré en espace et en temps. Les méthodes de semi-discrétisation requièrent un maillage structuré en temps car la même discrétisation en temps est utilisée à tous les pas de temps. Au contraire, les méthodes de discrétisation espace-temps permettent l'utilisation d'un maillage non-structuré à la fois en espace et en temps. Ainsi, il est possible de raffiner le maillage autour des discontinuités (telles qu'un choc, un défaut ou une interface) et il est également possible de raffiner le maillage adaptativement au cours du temps, tout en suivant l'évolution des discontinuités. De plus, les méthodes espace-temps ont l'avantage de permettre naturellement l'utilisation d'un pas de temps local quand celui-ci est contraint par le pas en espace. Pour toutes ces raisons, les méthodes éléments finis espace-temps ont fait l'objet de nombreuses recherches, comme par exemple dans [14, 15, 16, 17, 18]. Néanmoins, la question de la construction de maillages espace-temps se pose en raison de sa complexité et son coût en mémoire. En effet, les problèmes espace-temps ont une dimension supplémentaire (nD+t) par rapport aux problèmes en espace uniquement (nD). En particulier dans le cas des domaines 3D, le coût de calcul et la consommation mémoire augmentent considérablement.

Ce travail a été réalisé dans le contexte du programme de recherche collaboratif Depth Imaging Partnership (DIP) entre l'entreprise TotalEnergies et l'équipe-projet Makutu de l'Inria, de 2010 à 2022. Le projet DIP porte sur le développement des schémas numériques d'ordre élevé pour l'imagerie sismique et la reconstruction de la structure des sous-sols. Ces méthodes numériques reposent principalement sur des approximations éléments finis discontinues du champ d'onde, comme dans les méthodes Galerkine Discontinu, car elles permettent de prendre en compte la géométrie et les caractéristiques physiques du domaine étudié et sont propice à la parallélisation [19, 20, 21]. Cependant, les méthodes DG souffrent d'un coût de calcul élevé en raison du nombre de degrés de liberté, qui est doublé aux interfaces par rapport aux méthodes éléments finis classiques par exemple. C'est pourquoi, réduire le coût des méthodes DG tout en conservant leurs bénéfices est un sujet de recherche actif. C'est dans cette optique qu'ont été explorées les méthodes HDG au sein du projet DIP. Dans celles-ci, nous exprimons le vecteur solution comme un multiplicateur de Lagrange, représentant la trace de la solution numérique sur le squelette du maillage. Le nombre de degrés de liberté est ainsi réduit par rapport aux méthodes DG, sans toutefois en perdre les avantages [22].

Avec ce même objectif de réduire les coûts et améliorer la précision de la solution, nous nous intéressons au méthodes de Trefftz en temps, qui ont été proposées par Trefftz dans [23] et ont été explorées dans [24] dans le contexte du projet DIP, ainsi que pour l'équation des ondes acoustiques en espace-temps dans [25, 26, 27] et pour de nombreux autres problèmes dans [28, 29, 30]. Les méthodes de Trefftz ont été largement étudiées en domaine fréquentiel pour les problèmes hyperboliques dans [31], ainsi que pour les flexions de plaques et plaques épaisses dans [32], pour le problème de Helmholtz en deux dimensions et le problème de Maxwell en trois dimensions dans [33, 34]. Leur principe consiste en l'utilisation de solutions locales du problème étudié comme fonctions de base,

dénommées fonctions de Trefftz (ou T-fonctions). Ainsi, la solution numérique devrait être plus précise, car des caractéristiques des fonctions de Trefftz, telles que l'aspect oscillatoire ou la longueur d'onde, seront injectées dans l'espace d'approximation. Dans le cas où le système d'équations est auto-adjoint, la formulation variationnelle n'est posée que sur le squelette du maillage, car les termes volumiques disparaissent. Le nombre de degrés de liberté est donc fortement réduit par rapport aux méthodes DG. C'est pourquoi les méthodes de Trefftz semblent être un bon choix pour la recherche de méthodes plus rapides, sans toutefois perdre en précision. Malheureusement, le schéma numérique obtenu en domaine temporel est implicite, par conséquent, il est nécessaire d'inverser une grande matrice et le coût du calcul augmente malgré tout. En effet, comme nous travaillons en domaine temporel, les fonctions de Trefftz sont des solutions espace-temps. Ainsi, la méthode utilisée pour résoudre le problème devient naturellement une méthode espace-temps et la mise en œuvre de la méthode nécessite d'effectuer une intégration espace-temps. Dans [24], E. Shishenina propose de pallier cet inconvénient par l'utilisation de l'algorithme Tent-Pitcher. Cet algorithme a été introduit par Üngör et Sheffer dans [35] pour les problèmes espace-temps hyperboliques, puis généralisé par Erickson dans [36]. L'algorithme consiste à mailler le domaine espace-temps élément par élément, puis à résoudre le problème dans ces éléments, qui sont des pyramides à bases polygonale aussi appelées tentes. Pour pouvoir transformer notre schéma implicite en un schéma localement explicite, le maillage doit respecter des contraintes, appelées contrainte de causalité et contrainte de progression. Dans ces conditions, nous pouvons obtenir un schéma localement explicite et ainsi réduire considérablement le temps de calcul. Cet algorithme a été appliqué à différents problèmes, comme l'équation des ondes par Richter dans [37], pour la loi de conservation hyperbolique par Lowrie *et al.* dans [38], par Yin *et al.* pour l'analyse élastodynamique dans [39], par Monk et Richter pour les problèmes hyperboliques linéaires en régime temporel sous la forme d'un système symétrique dans [40], par Miller et Haber pour l'équation de la conduction de la chaleur hyperbolique dans [41], par Gopalakrishnan *et al.* pour obtenir un schéma explicite dans le cadre éléments finis espace-temps dans [42], par Howard *et al.* pour obtenir un solveur Garlerkin Discontinu espace-temps asynchrone en 3D+t dans [43], par Moiola et Perugia dans [26] et Stocker dans [27] pour l'équation des ondes acoustiques et par Shishenina pour l'équation des ondes acoustiques, élastiques et élasto-acoustiques dans [24]. Une des avancées parmi les plus récentes pour l'algorithme de Tent-Pitching consiste en la construction d'éléments cylindriques vers lesquels sont projetées les tentes du maillage Tent-Pitching espace-temps, par Gopalakrishnan dans [44, 45]. Le but de ce travail est de pouvoir séparer la dimension spatiale de la dimension temporelle et ainsi utiliser des méthodes classiques en espace combinées à des méthodes en temps d'ordre élevé. De plus dans l'algorithme Tent-Pitcher, seul le front du maillage doit être conservé en mémoire et non l'ensemble du maillage, ce qui efface certains désavantages des maillages espace-temps tout en conservant leurs avantages, comme la possibilité de raffiner le maillage. Le raffinement du maillage Tent-Pitching espace-temps a été très étudié [46, 47, 48, 49, 50, 51], ce qui présente un intérêt pour les équations non-linéaires, mais

également pour les équations d'ondes sismiques comportant des interfaces.

Cette thèse se décompose en deux parties. La Partie I présente un solveur Trefftz-DG avec Tent-Pitching appliqué à l'équation des ondes acoustiques en espace-temps. Nous présentons d'abord la méthode de Trefftz-DG et l'algorithme de Tent-Pitcher, puis nous dérivons la formulation variationnelle associée. Ensuite, nous décrivons en détail l'implémentation pour des maillages structurés et son extension aux maillages non-structurés. Enfin, nous l'adaptons à un environnement de calcul parallèle. Nous présentons des tests numériques et des résultats que nous comparons à ceux obtenus avec une méthode Interior Penalty Discountinuous Garlerkin. La Partie II traite des conditions de bord, en particulier en introduisant des Couches Absorbantes Parfaitement Adaptées (PML) dans le solveur Trefftz-DG avec Tent-Pitching décrit précédemment. Pour ce faire, nous dérivons plusieurs formulations variationnelles et présentons en détail leurs implémentations, ainsi que des résultats et tests numériques.

Contents

List of Figures

List of Tables

Part I

Optimised Trefftz-DG

Chapter 3

Introduction

One way to study complex phenomena is to mathematically model the problem through partial differential equations and solve the obtained problem. However, for most of the equations we obtain, we do not know how to solve them analytically. Instead, we *discretize* the problem, in order to construct an approximation of the original equations. This way, we can now *numerically solve* our problem and obtain approximate solutions. Then, the question arises of the choice of discretization for the differential equations.

3.1 Finite Difference (FD) Method

Let us consider the Poisson equation with homogeneous Neumann boundary conditions to illustrate the first choice of discretization, which consists in approximating the derivatives:

$$\begin{cases} \Delta u = f\,, & \text{in } \Omega \\ u = 0\,, & \text{in } \partial\Omega\,. \end{cases} \tag{3.1}$$

where Ω is the domain of interest and $u(x, y) \in L^2(\Omega)$.

To discretize the Laplacian of u here, we consider several Taylor expansions of u as follows:

$$u(x_0 + h, y_0) = u(x_0, y_0) + h\frac{\partial u}{\partial x}(x_0, y_0) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_0, y_0) + ...$$

$$u(x_0 - h, y_0) = u(x_0, y_0) - h\frac{\partial u}{\partial x}(x_0, y_0) + \frac{h^2}{2}\frac{\partial^2 u}{\partial x^2}(x_0, y_0) + ...$$

By combining them, we obtain the following expression for the second-order derivative in x:

$$\frac{\partial^2 u}{\partial x^2}(x_0, y_0) \approx \frac{u(x_0 - h, y_0) - 2u(x_0, y_0) + u(x_0 + h, y_0)}{h^2}$$

And in the same manner, we can obtain the second-order derivative in y:

$$\frac{\partial^2 u}{\partial y^2}(x_0, y_0) \approx \frac{u(x_0, y_0 - h) - 2u(x_0, y_0) + u(x_0, y_0 + h)}{h^2}$$

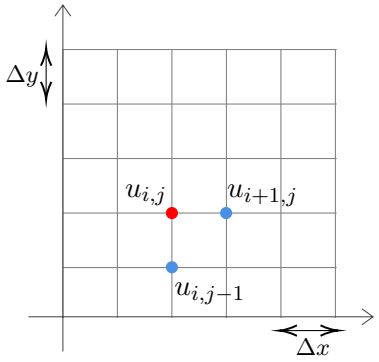


Figure 3.1: Grid in 2D

If we consider a two dimensional grid with steps Δx and Δy as pictured in Fig. 3.1, and define the solution at a point (i, j) of the grid as $u_{i,j}$, we can discretize the problem (3.1) as follows:

$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta y^2} = f\,.$$

Discretizing the problem by approximating the derivatives as above is called the *Finite Difference* method, which is a very well-known and a common approach. This method has many advantages, such as its simplicity in both its implementation and meshing process (we work on a simple Cartesian grid). They have been applied to many problems and proven to be quite efficient and robust enough. For all these reasons, this method is very popular among geophysicists, in particular for industrial cases which require seismic wave simulations. However, as can be seen quite straightforwardly, Finite Differences are not very well suited for complex topologies, as it is quite complicated to take into account discontinuities or variable grid size. Moreover, this method suffers from numerical dispersion when the grid is too coarse compared to the wavelength. It has been shown that we need ten points per wavelength in order to avoid this phenomenon, and the study of the accuracy of this method can be found in [52].

3.2 Classical Finite Element Method (FEM)

Now, let us go back to (3.1) and look at another type of discretization, that consists in approximating the solution in each subdomain of a triangulation of the domain of interest, as a linear combination of piecewise continuous functions:

$$u \approx u_h = \sum_i u_i \varphi_i$$

To do so, we write a variational formulation of the studied problem by multiplying it by a test function v chosen in an appropriate test space V, integrating it over Ω and

integrating it by parts:

Seek $u \in V$, such that for all $v \in V$, it holds true:

$$a(u, v) = l(v)$$

where

$$\begin{aligned} a(u, v) &= -\int_\Omega \nabla u \nabla v + \int_{\partial\Omega} \frac{\partial u}{\partial \mathbf{n}} v \\ l(v) &= \int_\Omega f v \,, \end{aligned} \tag{3.2}$$

where $\mathbf{n}$ is the space normal and $V = \{v \in L^2(\Omega)\,, v|_{\partial\Omega} = 0\,, \frac{\partial v}{\partial x_i} \in L^2(\Omega) \quad \forall i = 1, d\}$ with d the dimension of the problem.

Remark. *System* (3.2) *is called a variational formulation, or* ***weak form****, because the regularity needed for u is lower in this system than in the original one.*

We then introduce a conforming triangulation T_h of the domain Ω into non-overlapping elements K, where the union of all elements span the whole domain. We can then write the discrete variational formulation as follows:

Seek $u_h \in V_h$, such that for all $v_h \in V_h$, it holds true:

$$a(u_h, v_h) = l(v_h)$$

where $V_h(T_h) = \{u_h|_K \in C^0, u_h|_K \in \mathbb{P}^k\,, \forall K \in T_h\} \subset V$ and $\mathbb{P}^k$ is the space of polynomials of order less or equal to k. The well-posedness of a variational formulation is obtained if $a(u, v)$ is continuous, coercive and bilinear and $l(v)$ is continuous and linear, as required for applying the Lax-Milgram theorem.

Choosing u_h as a linear combination of the basis of V_h and choosing v_h as the basis itself leads to rewriting the variational problem as follows:

$$KU = F$$

where

$$\begin{aligned} K_{ij} &= -\int_\Omega \nabla\varphi_i \nabla\varphi_j \\ F_{ij} &= \int_\Omega f_i \varphi_j \end{aligned}$$

The matrix K and vector F consist of the assembly of analog matrices and vectors K^K and F^K defined on the elements K of T_h. Their size corresponds to the number of degrees of freedom (DoF), which is the number of nodes at which the solution is

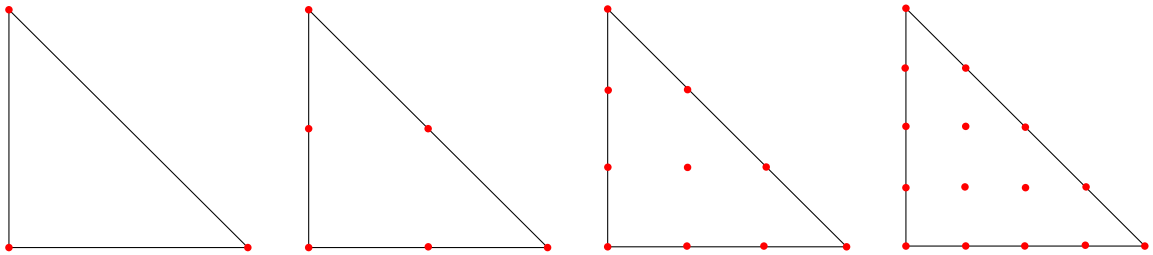


Figure 3.2: Degrees of freedom for different polynomial order

approximated. In this context, we work with $\mathbb{P}^1$ polynomials, but we can easily use higher-order polynomials which results in higher-order elements, thus more degrees of freedom (see Fig. 3.2). Instead of computing each K^K and F^K for every element K, we will compute the associated elementary quantities defined on a reference element, thus obtaining K^e and F^e. To do so, we define a mapping function $\mathcal{T}$ that transforms local coordinates into reference coordinates as shown in Fig. 3.3. This function allows us to compute K^e and F^e, then map them to local coordinates in order to obtain K^K and F^K. All is left now is to determine the basis functions φ needed to compute these quantities, which have to ensure the continuity at the interfaces leading to:

$$\varphi_i(x_j) = \delta_{ij} \, ,$$

with the symbol of Kronecker:

$$\delta_{ij} = \begin{cases} 1 & \text{if i=j} \\ 0 & \text{otherwise} \end{cases}$$

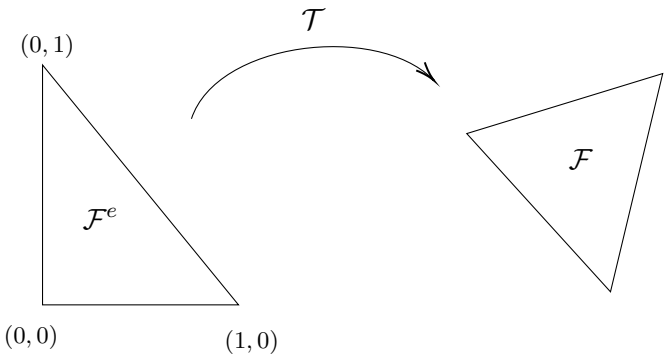


Figure 3.3: Transformation from a reference triangle to an arbitrary triangle

The method presented above is called the *Finite Element* method. It overcomes some of the disadvantages of the Finite Difference method, because the mesh is not necessarily Cartesian here, and is triangular or quadrilateral. In fact, triangular meshes fit better into complex topologies and usually one can obtain high-order resolution with this

method which is not the case with the Finite Volume method for example. However, this method also suffers from poor dispersion properties, which can be improved (modified integration rule, isogeometric elements) but has its limits. Moreover, the global matrix is sparse but is not diagonal in general (even per block), which is an important feature, in particular when considering time-dependent problems, as we would need to invert that matrix at each time step. To overcome this problem, one can use additional techniques, such as mass-lumping but this technique leads in a severe loss of convergence order. Plus, even though the mesh is better for accommodating complex geometries, Finite Element methods still impose constraints on the mesh, such as the continuity between the elements.

Remark. *Inverting an arbitrary matrix of size n using a classical algorithm can be done in $\approx n^3$ operations, i.e. $\mathcal{O}(n^3)$, while inverting a block-diagonal matrix of m blocks using the same algorithm can be done in $\approx m \times \left(\frac{n}{m}\right)^3$ operations. These results are obtained when performing a LU-decomposition on the considered matrix in order to invert it, and of course, can be greatly improved. But one can see that the costs will always be reduced when inverting a block-diagonal matrix than an arbitrary one. This is a strong feature of the method we introduce in the next section.*

3.3 Discontinuous Galerkin Method (DG)

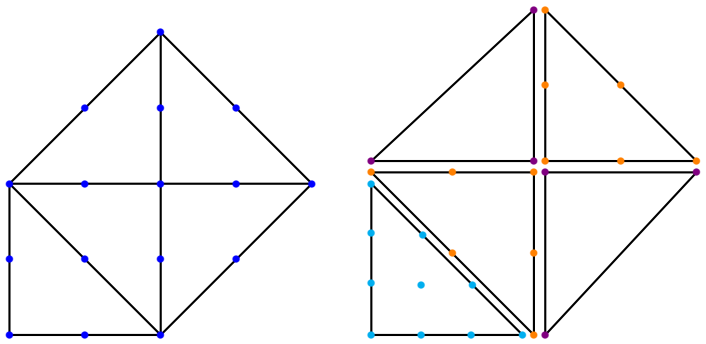
Let us now go back to (3.2), where we consider discontinuous basis functions and consider the problem on each element K and then sum the contributions on all elements, which results in:

$$-\sum_K \int_K \nabla u \nabla v + \int_{\mathcal{F}^{ext}} (\nabla u \cdot n) v + \int_{\mathcal{F}^{int}} [\![(\widehat{\nabla u} \cdot n) v]\!] = \sum \int_K f v \,,$$

where we defined two categories of boundaries: $\mathcal{F}^{int}$ and $\mathcal{F}^{ext}$, which correspond to the internal facets shared by elements and the outer boundary respectively and $[\![u]\!] = u^+ + u^-$ is the jump. Since our basis functions are discontinuous, we do not have to impose continuity at the interfaces. But, we need to find a way to connect all the elements together in order to retrieve a unique global solution, which is the role of the term $\int_{\mathcal{F}^{int}} [\![(\widehat{\nabla u} \cdot n) v]\!]$, where $\widehat{\nabla u}$ is the numerical flux approximating ∇u on the boundary.

The method presented above is called the *Discontinuous Galerkin* method. The foundations of Discontinuous Galerkin methods are attributed to Ritz [53] and Galerkin [54] and the method as we know it today was developed in the 1970s, in particular with the resolution of the problem of neutron transport by Reed and Hill on triangular meshes for first order PDEs [55]. The method was then widely used and spread among scientists following the application of DG to hyperbolic problems by Cockburn *et al.*[56]. The method was made popular by its use in the late 80s for the first time as a semi-discrete scheme with a Runge-Kutta time integration scheme [11, 57]. It also has been applied to

second-order PDEs, by rewriting them as a first-order system [58]. Many Discontinuous Galerkin formulations adapted to the considered problem exist and in particular, one can choose the numerical fluxes in many different ways, *e.g.* by adding boundary penalty terms and hence producing a different formulation. This choice of fluxes can affect the consistency, stability and conservative aspect of the scheme. A review on DG methods, the different flux choices and how they affect the scheme can be found in [21], [59] and [60].



(a) FEM (b) DG

Figure 3.4: Distribution of Degrees of Freedom

As the name indicates and as explained before, the basis functions in this method are discontinuous at the interface between two elements. Discontinuous Galerkin method can be viewed as a discontinuous Finite Element method, or a high-order Finite Volume method, or simply as a combination of both of these methods. This is a great advantage, since it can adapt to very complex geometries (*e.g.* heterogeneities). This method has many advantages, such as the fact that the solutions are piecewise smooth and discontinuous at the interfaces, the transmission conditions between elements are enforced weakly, it is conducive to hp-adaptivity, and of course, the formulation is defined locally. However, the number of degrees of freedom are doubled at the interfaces, hence leading to a higher computational cost. Some advantages and drawbacks of different numerical schemes are presented in Table 3.1.

In the end, none of these methods is perfect and we have to, either sacrifice the precision of our solution or the computational time. Our aim is to find a method in time-domain that can be fast and offer a good precision as well. Our candidate to try and achieve this purpose is the Trefftz-DG method with the Tent-Pitcher algorithm for the construction of spacetime meshes.

3.4 Trefftz Approach

Trefftz methods are numerical schemes that belong to the Finite Element family. They were first introduced by Erich Trefftz in 1926 [23], and consist in using local solutions of the studied problem as basis and test functions. Thus, the idea behind the Trefftz

	FD	FV	FEM	DG
Implementation	Very easy	Easy	Complex	Complex
Unstructured meshes	Not supported	Supported	Supported under constraints	Very well supported
Stiffness matrix	Tridiagonal or triangular	Mesh-dependent	Sparse	Block-diagonal
Number of DoF	Low	Low	Low	High
Conservative	Not necessarily	Natural	Not necessarily	Locally
High-order	Yes	No	Yes	Yes
Adaptive hp-refinement	Yes	No	Yes under constraints	Yes
Explicit semi-discrete scheme	Yes	Yes	No	Yes

Table 3.1: Comparison of different numerical methods

approach is to use the knowledge we have of the definition of the solutions and then inject it in our problem. A general definition given by Herrera [61] would be the following one: *Given a region of an Euclidean space of some partitions of that region, a "Trefftz Method" is any procedure for solving boundary value problems of partial differential equations or systems of such equations, on such region, using solutions of that differential equation or its adjoint, defined in its subregions.*

Trefftz methods were first applied to structural mechanics problems (*e.g.* calculations of stress of beam). Trefftz formulations can be split into two categories: direct and indirect methods. The difference lies in the expression of the global solution and thus, the choice of trial and test functions. In indirect Trefftz formulations, the solution is approximated by a linear combination of Trefftz functions, *i.e.* local solutions of the considered PDE. Whereas in direct Trefftz formulations, we use the weighted residual method and rewrite the governing equations as boundary integral equations. Here, the test functions (*i.e.* weighted functions) are required to be solutions to the problem of interest. Some of the most well-known Trefftz methods are the Trefftz-Discontinuous Galerkin method (TDG), introduced in a general framework by Gabard in [31] for time-harmonic hyperbolic problems. Another popular Trefftz method is the Ultra-Weak Variational Formulation (UWVF), introduced by Cessenat and Després in [33, 34]. Trefftz methods have been used in a wide range of applications, such as mechanics and fluid dynamics: plate bending and thick plate problems [62], heat conduction problems [32], elasticity

problems in [63], wave propagation problems and in particular, the first-order spacetime acoustic wave equation is first introduced in [25]. A larger panel of Trefftz methods is reviewed in [64] and [65].

Since we have to compute local solutions to each studied problem beforehand to use as basis, numerous Trefftz functions exist. The basis functions that are the most used in the litterature are plane waves, generalized harmonic polynomials (*i.e.* polynomial solution to the Laplace equation) and fundamental solutions.

Remark. *If we have two sets of Trefftz functions (i.e. two kinds of solutions) for a given problem, we could use any one of them as basis. Moreover, any one of them could be used with any kind of Trefftz variational formulation of the considered problem. This means that if we had a Trefftz-DG formulation and a UWVF formulation, any Trefftz basis could be used for any of the two formulations of the same problem. The only prerequisite is for the basis functions to be solutions of the considered problem.*

Most of the mathematical modeling problems cannot be solved without dividing our domain of interest in subregions. This division is necessary, because otherwise we obtain ill-conditioned mass matrices and in the case of Trefftz methods, we also need a sufficient number of linearly independent basis functions in order to avoid obtaining an ill-conditioned matrix. Moreover, as explained before, it is not always possible to find analytical solutions on the entire domain. Thus, it is necessary to partition the domain and compute numerical solutions, which involves continuity conditions along the interfaces.
In contrast to classical numerical methods, Trefftz methods draw attention for two main reasons:

- a lower number of degrees of freedom is needed,
- physical characteristics of the solution are incorporated into the discrete solution of the problem (oscillatory aspect, wave number, ...).

When using a variational formulation, having local solutions of the considered problem as basis functions generally leads to the volumic terms to vanish if the considered problem is self-adjoint, which is the reason for the lower number of degrees of freedom. In that sense, Trefftz methods resemble Boundary Element methods, which consist in rewriting a PDE as boundary integral equations and thus, only involve elements on the boundary of the considered domain. But compared to them, in Trefftz methods, computing singular integrals (which is often complex) is not necessary.

However, Trefftz methods suffer from ill-conditioning due to the high linear dependence between the basis functions and the fact that we need to know solutions to the considered problem beforehand. This latter can be overcome by using *Quasi-Trefftz* methods, which consist in using approximate solutions as basis functions and are currently increasing in popularity. This idea was introduced for Trefftz-DG methods applied to wave problems in the form of Generalized Plane Waves by Imbert-Gérard and Desprès in [66].

3.5 Plan of Part I

In this thesis, we consider the Trefftz-DG method for transient wave propagation problems, in particular first-order acoustic wave equations, which is a subject that has raised the interest of many, such as [24], [25], [26], [27]. We consider the problem accompanied by the Tent-Pitcher algorithm for spacetime mesh construction in a parallel environment. Thus, the plan of the first part of this thesis is as follows:

- Chapter 4 **Trefftz-DG Method for Wave equations**: in this chapter, we present the considered acoustic wave equation and show its well-posedness. Then, we explain how the Trefftz-DG method is derived, along with the presentation of the considered Trefftz basis functions, which are exact polynomials in our case. Finally, we will present the drawbacks of this method, which encouraged us to find an alternative, which is the use of the Tent-Pitcher algorithm in the Trefftz-DG framework.

- Chapter 5 **Tent-Pitcher algorithm**: in this chapter, we present the alternative found to overcome some of the drawbacks of the previously introduced Trefftz-DG formulation, which is the Tent-Pitcher algorithm. The general algorithm is given in details, along with some historical background. After which, we present the algorithms we specifically used in 1D+time and 2D+time, for structured and unstructured meshes. Then, we present a variational formulation on Tent-Pitched meshes, with demonstration that the variational formulation is well-posed.

- Chapter 6 **Implementation**: in this chapter, we explain in details each task we carried out and how it was done. The different tasks we implemented were: the Trefftz-DG method with Tent-Pitching for structured meshes, then for unstructured meshes and both of these cases in a HPC-environment.

Chapter 4

Trefftz-DG Method for Wave equations

Waves are everywhere around us and there are many types of them. But before describing them, let us first define a wave:

Waves

A wave is a disturbance that propagates gradually and originates from a vibration. A wave can be characterized by its speed c, its frequency f, its angular frequency $\omega = 2\pi f$, its wavelength λ, its wave number $k = {}^{2\pi}/_{\lambda}$.

From this definition, we can highlight three characteristics of a wave:

- the source: it is the initial disturbance that causes the propagation,
- the direction of propagation: it describes how the wave travels and its orientation,
- the medium: it is the environment in which the wave propagates.

We can distinguish two categories of waves based on the medium in which they propagate. **Mechanical waves** need a medium to travel in, so they cannot propagate in vacuum. The medium can be various, such as the air, water, gas, a solid and the speed at which the wave travels will depend on the medium. Examples of mechanical waves are water waves, sound waves or seismic waves as seen in the General Introduction. On the contrary, **electromagnetic waves** do not need a medium to propagate in, thus can also travel in vacuum. Examples of electromagnetic waves are infrared waves, radio waves, x-ray waves, light waves, etc.

If we categorize mechanical waves based on their direction of propagation, we obtain three types of waves: longitudinal waves, transverse waves and surface waves. A longitudinal wave oscillates parallel to the direction of propagation, while a transverse wave travels perpendicular to the direction of propagation. Surface waves, on the other

hand, propagate on the surface of the domain and their amplitude decreases as they travel further from the surface. These three types of waves are illustrated in Fig. 4.1. In the category of seismic waves, for example, we saw that there are four types of waves: primary waves, secondary waves, Rayleigh waves and Love waves. The first two are body waves: the primary P-wave is a longitudinal wave, whereas the secondary S-wave is a transverse one. Rayleigh and Love waves are surface waves.

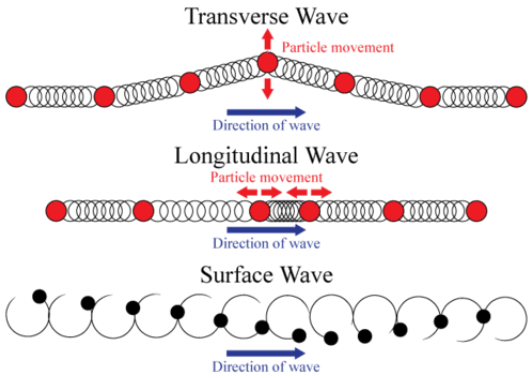


Figure 4.1: Longitudinal, transverse and surface waves[1]

We can also distinguish progressive waves from stationary waves. A wave is progressive when the front continuously advances in a direction and the amplitude decreases as the front travels further (in 1D, the amplitude does not decrease and stays the same). Whereas a stationary wave does not advance and stays at a fixed position and its amplitude varies. These two types of waves are illustrated in Fig. 4.2 for the one-dimensional case. The vibration of the string of a violin when being stroked is a stationary wave. Acoustic waves and seismic waves are examples of progressive waves.

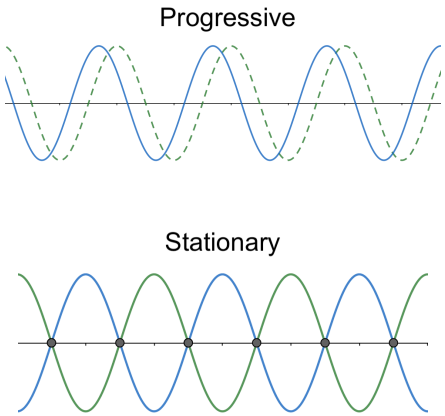


Figure 4.2: Progressive and stationary waves in 1D

[1]Source: CK-12 Foundation

4.1 Acoustic wave equation

Acoustic waves fall in the category of mechanical waves and travel by transfer of energy with a characteristic velocity, which depends on the medium in which they propagate.

To derive the acoustic wave equations, one needs to use three laws: the equation of state, the conservation of mass and the conservation of momentum. Since we are working with the linear acoustic wave equation, we will linearize these laws. Let us consider a small volume of fluid dV.

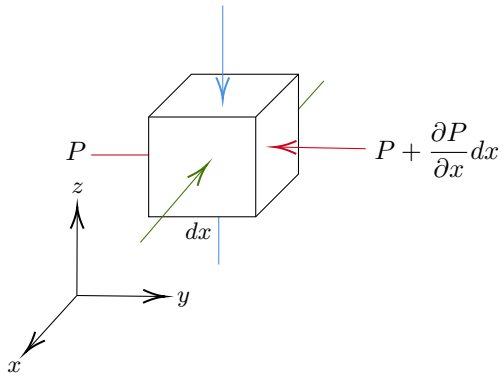


Figure 4.3: Pressure applied to small volume of fluid

We consider the sum of pressure forces as illustrated on Fig. 4.3, the acceleration as the time derivative of the velocity and using Newton's second law, we obtain the following:

$$\rho_0 \frac{\partial \mathbf{v}}{\partial t} = -\nabla P \,,$$

where P is the pressure, $\mathbf{v}$ represents the velocity field and ρ_0 is the equilibrium density. The linear continuity equation is written as follows:

$$\frac{\partial s}{\partial t} + \nabla \cdot \mathbf{v} = 0 \,,$$

with $s = \dfrac{\rho - \rho_0}{\rho}$, where ρ represents the instantaneous density and s is assumed to be very small. For fluids other than an ideal gas, the state equation can be represented by a Taylor's expansion:

$$P \approx P_0 + \Big(\frac{\partial P}{\partial \rho}\Big)_{\rho_0} (\rho - \rho_0) + \frac{1}{2}\Big(\frac{\partial^2 P}{\partial \rho^2}\Big)_{\rho_0} (\rho - \rho_0)^2 + ...$$

where P_0 is the equilibrium pressure. Since we are considering a small volume and assuming that the fluctuations are small, only the lowest order term in $(\rho - \rho_0)$ needs to be considered. Hence, we obtain for the pressure p:

$$p = P - P_0 \approx \mathcal{B}\frac{(\rho - \rho_0)}{\rho_0}$$

where $\mathcal{B} = \rho_0 \Big(\frac{\partial P}{\partial \rho}\Big)_{\rho_0} = \rho_0 c^2$, with $\mathcal{B}$, the adiabatic bulk modulus and c, the speed of sound. Injecting this into the linear continuity equation, we finally obtain:

$$\frac{1}{\rho_0 c^2}\frac{\partial p}{\partial t} + \nabla \cdot \mathbf{v} = 0$$

Thus, leading to the linear first order acoustic wave equation, which can be written:

$$\begin{cases} \dfrac{1}{c^2\rho}\dfrac{\partial p}{\partial t} + \nabla \cdot \mathbf{v} = f\,, \\ \\ \quad \rho\dfrac{\partial \mathbf{v}}{\partial t} + \nabla p = 0\,. \end{cases} \tag{4.1}$$

To obtain the second order wave equation from these, one needs to derive the second equation of (4.1) with respect to $\mathbf{x}$ and the first equation of (4.1) with respect to t, which results in:

$$\begin{gathered} \frac{1}{c^2\rho}\frac{\partial^2 p}{\partial t^2} + \nabla \cdot \frac{\partial \mathbf{v}}{\partial t} = 0\,, \\ \\ \rho \nabla \cdot \frac{\partial \mathbf{v}}{\partial t} + \Delta p = 0\,. \end{gathered} \tag{4.2}$$

Now, injecting inside the first equation of (4.2) the expression we find for $\nabla \cdot \dfrac{\partial \mathbf{v}}{\partial t}$ in the second equation of (4.2), we finally obtain the second order wave equation as follows:

$$\frac{\partial^2 p}{\partial t^2} - c^2 \Delta p = 0\,.$$

In this framework, we will work with the first order acoustic wave equation, with a second member f, initial conditions p_0 and $\mathbf{v}_0$ and Neumann boundary condition g_N:

$$\begin{cases} \dfrac{1}{c^2\rho}\dfrac{\partial p}{\partial t} + \nabla \cdot \mathbf{v} = f & \text{in } \Omega\,, \\ \\ \rho\dfrac{\partial \mathbf{v}}{\partial t} + \nabla p = 0 & \text{in } \Omega\,, \\ \\ \mathbf{v}(\cdot, 0) = \mathbf{v}_0,\ p(\cdot, 0) = p_0 & \text{in } \mathcal{D} \times \{0\}\,, \\ \\ \mathbf{v} \cdot \mathbf{n} = g_N & \text{in } \partial\mathcal{D} \times \mathcal{I}\,. \end{cases} \tag{4.3}$$

with

- $\mathbf{x} \in \mathbb{R}^d$, the vector of space coordinates
- $t \in \mathbb{R}^+$, the time coordinate,
- $p = p(\mathbf{x}, t) \in L^2(\Omega)$, the pressure,
- $\mathbf{v}(\mathbf{x}, t) = (v_x, v_y) \in (L^2(\Omega))^2$, the velocity,
- $f = f(\mathbf{x}, t) \in L^2(\Omega)$, the source term,
- T, the final time,

$\mathcal{D} \subset \mathbb{R}^d$, the spatial domain, c, the wave speed,

$\mathcal{I} = [0, T] \subset \mathbb{R}^+$, the time interval,

$\Omega = \mathcal{D} \times \mathcal{I}$, the spacetime domain, ρ, the density.

4.1.1 Well-posedness

To prove the well-posedness of the Cauchy problem (4.3), we follow the framework of the book [67] by Kreiss and Lorenz.

First, we need to introduce $P(i\omega)$, which is obtained by substituting $\dfrac{\partial}{\partial x_j}$ for $i\omega_j$, and is called the *symbol* of the differential operator $P(\partial/\partial \mathbf{x})$ and we consider an arbitrary Cauchy problem under the following form:

$$\begin{aligned} \frac{\partial u}{\partial t} &= P(\frac{\partial}{\partial \mathbf{x}})u\,, \qquad \mathbf{x} \in \mathbb{R}^d\,, t \in \mathbb{R}^+ \\ u(\mathbf{x}, 0) &= u_0(\mathbf{x})\,, \qquad \mathbf{x} \in \mathbb{R}^d\,, t = 0 \end{aligned} \tag{4.4}$$

We can now introduce the definition of well-posedness for (4.4) in terms of the symbol as follows:

Well-posedness

The Cauchy problem (4.4) is well-posed, if $\exists \alpha\,, K$ constants, such that:

$$|e^{P(iw)t}| \leq K e^{\alpha t} \qquad \forall t \geq 0\,, \forall \omega \in \mathbb{R}^d\,,$$

where $e^{P(iw)t}$ is an exponential matrix and $|\cdot|$ its module.

Following the concepts of hyperbolicity in [67], we have the following definitions:

Concepts of hyperbolicity

The Cauchy problem (4.4) is called

- *weakly hyperbolic* if $\forall \omega \in \mathbb{R}^d$, all eigenvalues of $P(i\omega)$ are purely imaginary,
- *strongly hyperbolic* if the problem is well-posed,
- *strictly hyperbolic* if $\forall \omega \in \mathbb{R}^d\,, \omega \neq 0$, all eigenvalues of $P(i\omega)$ are purely imaginary and distinct.

Moreover, the following theorem holds and is demonstrated in [67]:

Concepts of hyperbolicity

Theorem 4.1. A strictly hyperbolic problem is strongly hyperbolic.

Hence, showing that our system is strictly hyperbolic suffices to show that it is well posed. Let $U = \begin{pmatrix} v_x \\ v_y \\ p \end{pmatrix}$. The first order acoustic wave equations can be rewritten as:

$$\begin{cases} \dfrac{1}{c^2\rho}\dfrac{\partial U}{\partial t} = P(\partial/\partial \mathbf{x})U & \text{in } \Omega\,, \\ U(\mathbf{x},0) = U_0(\mathbf{x}) & \text{in } \mathcal{D}\,. \end{cases} \qquad \begin{matrix} \mathbf{x} \in \mathbb{R}^2 \\ \mathcal{I} = [0,T] \end{matrix}$$

where $P(\partial/\partial \mathbf{x}) = \begin{pmatrix} 0 & 0 & -\partial/\partial x \\ 0 & 0 & -\partial/\partial y \\ -\partial/\partial x & -\partial/\partial y & 0 \end{pmatrix}$, thus $P(i\omega) = \begin{pmatrix} 0 & 0 & i\omega_x \\ 0 & 0 & i\omega_y \\ i\omega_x & i\omega_y & 0 \end{pmatrix}$. To find the eigenvalues of the symbol, we search for the roots of the characteristic polynomial by solving $|P(i\omega - \lambda)| = 0$, which results in:

$$|P(i\omega - \lambda)| = -\lambda^3 - \lambda(\omega_x^2 + \omega_y^2) = -\lambda(\lambda^2 + |\omega|^2) = 0\,.$$

We obtain three *purely imaginary and distinct* eigenvalues:

$$\lambda_1 = 0 \qquad \lambda_2 = i|\omega| \qquad \lambda_3 = -i|\omega|$$

According to the concepts of hyperbolicity previously introduced, **the first order acoustic system is strictly hyperbolic, so strongly hyperbolic and thus, well-posed**.

4.2 Trefftz-DG Method

As presented in the previous chapter, there are many types of Trefftz formulations. In our case, we decided to work with the Discontinuous Galerkin discretization.

As explained in the introduction, we will consider a variational formulation (or weak form) of (4.3) in order to apply a numerical method and solve the problem in a Finite Element framework. To obtain the weak formulation, we will proceed in a classical manner and multiply our equations by wisely chosen test functions, after which we will integrate them by parts.

First of all, we need to discretize Ω. Let T_h be a triangulation of Ω composed of non-overlapping spacetime elements K, with the following properties:

- $T_h = \cup K$,
- $K \in D \times I$,
- $\forall K$, K is not degenerate.

In our case, we work with conforming meshes, such as the ones introduced in the previous chapter in the Finite Element framework, thus K is a polyhedron and $\forall K \neq K'$, $K \cap K'$ is either empty, or contains one single node or an entire edge. Let us also introduce the approximate wavefields p_h and $\mathbf{v}_h$, where:

$$\lim_{h\to 0} \|p - p_h\|_V = 0,$$
$$\lim_{h\to 0} \|\mathbf{v} - \mathbf{v}_h\|_{\mathbf{V}} = 0.$$

V and $\mathbf{V}$ are the functional spaces in which the solutions are seek and we will explicit them in the following subsection. We assume that the physical parameters are constant per element.

4.3 Functional spaces

To construct suitable functional spaces for variationnally solving (4.3), we consider a local variational formulation on a spacetime element K. For sake of simplicity, we will omit h and keep the notations as p and $\mathbf{v}$ from now on. We thus begin with the following expressions:

$$\begin{aligned} \frac{1}{c^2\rho}\int_K \frac{\partial p}{\partial t} q + \int_K \nabla\cdot\mathbf{v} q &= \int_K f q \\ \rho\int_K \frac{\partial \mathbf{v}}{\partial t}\cdot\mathbf{w} + \int_K \nabla p\cdot\mathbf{w} &= 0 \end{aligned} \tag{4.5}$$

Here, $(q, \mathbf{w}) \in L^2(\Omega) \times (L^2(\Omega))^2$ are the test functions and their functional spaces will be defined later on.

We want our integrals in (4.5) to be finite, *i.e.*, we want:

$$\frac{\partial p}{\partial t} \in L^2(K),\ \nabla\cdot\mathbf{v} \in L^2(K),\ \frac{\partial \mathbf{v}}{\partial t} \in (L^2(K))^d,\ \nabla p \in (L^2(K))^d$$

Hence, we introduce V and $\mathbf{V}$:

$$V = \Big\{ p \in L^2(\Omega) \mid p \in H^1(K) \Big\}$$
$$\mathbf{V} = \Big\{ \mathbf{v} \in (L^2(\Omega))^d \mid \nabla\cdot\mathbf{v} \in L^2(K)\,, \frac{\partial \mathbf{v}}{\partial t} \in (L^2(K))^d \Big\}$$

where $H^1(K) = \Big\{ \Phi \in L^2(K)\,, \nabla\Phi \in (L^2(K))^d\,, \frac{\partial \Phi}{\partial t} \in L^2(K) \Big\}$.
As a consequence, (4.5) is well-defined if we assume $p \in V$, $\mathbf{v} \in \mathbf{V}$.

Now that we have our functional spaces, we can move on to the next step, which consists in integrating (4.5) by parts in order to derive its weak formulation.

4.4 Variational Formulation

Following the above section and summing all equations of (4.5), we obtain:

$$
\begin{aligned}
&-\int_K \frac{1}{c^2\rho} p \frac{\partial q}{\partial t} + \mathbf{v}\cdot\nabla q + \rho\mathbf{v}\cdot\frac{\partial \mathbf{w}}{\partial t} + p\nabla\cdot\mathbf{w} \\
&+\int_{\partial K} \frac{1}{c^2\rho} pqn_t + \mathbf{v}q\cdot\mathbf{n} + \rho\mathbf{v}\cdot\mathbf{w}n_t + p\mathbf{w}\cdot\mathbf{n} \\
&=\int_K fq
\end{aligned}
\tag{4.6}
$$

where $\mathbf{n}$ corresponds to the space normal and n_t corresponds to the time normal. To complete our DG discretization, we need to properly choose the numerical traces of p and $\mathbf{v}$, which are their respective values on ∂K and consist in making the elements communicate. From now on, the traces (also called fluxes) obtained from the integration by parts of the space derivatives will be denoted $\hat{p}$ and $\hat{\mathbf{v}}$, and the ones obtained from the integration by parts of the time derivatives will be denoted $\check{p}$ and $\check{\mathbf{v}}$.
Before defining $\hat{p}$, $\hat{\mathbf{v}}$, $\check{p}$ and $\check{\mathbf{v}}$, we are going to decompose ∂K into various types of boundaries and define them on each of these new frontiers.

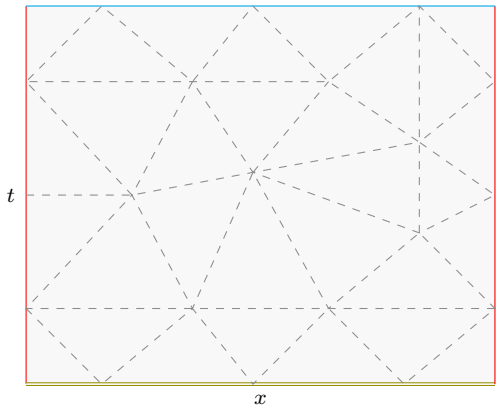


Figure 4.4: Spacetime triangulation of Ω in 1D+t

- $\mathcal{F}^e$ represents the internal element faces,
- $\mathcal{F}^D$ represents the domain boundary faces $\partial D \times [0,T]$,
- $\mathcal{F}^0$ represents the initial time faces $D \times \{0\}$,
- $\mathcal{F}^T$ represents the final time faces $D \times \{T\}$.

We also introduce additional notations, similar to the ones in standard DG methods, needed to define the numerical traces. We define the average $\{\!\{\cdot\}\!\}$, the jump along the space normal $[\![\cdot]\!]_x$ and the jump along the time normal $[\![\cdot]\!]_t$ between two elements K^+ and K^- for a piecewise-continuous scalar p and vector field $\mathbf{v}$:

$$\{\!\{p\}\!\} = \frac{1}{2}(p^+ + p^-)\,, \qquad \{\!\{\mathbf{v}\}\!\} = \frac{1}{2}(\mathbf{v}^+ + \mathbf{v}^-)\,,$$
$$[\![p]\!]_x = p^+\mathbf{n}^+ + p^-\mathbf{n}^-\,, \qquad [\![\mathbf{v}]\!]_x = \mathbf{v}^+ \cdot \mathbf{n}^+ + \mathbf{v}^- \cdot \mathbf{n}^-\,,$$
$$[\![p]\!]_t = p^+ n_t^+ + p^- n_t^-\,, \qquad [\![\mathbf{v}]\!]_t = \mathbf{v}^+ n_t^+ + \mathbf{v}^- n_t^-\,.$$

K^+ and K^- represent two neighboring elements that share one common face. Thus, p^+ embodies the value of p in K^+ and p^- embodies the value of p in K^-, likewise for v^+ and v^-. The normals $\mathbf{n}^+$ and $\mathbf{n}^-$ depict the outgoing space normals of K^+ and K^- respectively, as can be seen in Figure 4.5 and the same goes for n_t^+ and n_t^-, the time normals.

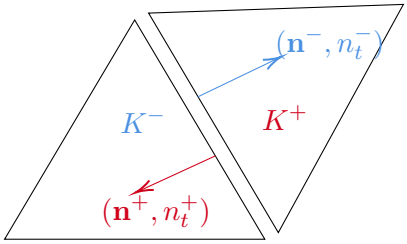


Figure 4.5: Interface between two neighboring mesh cells

4.4.1 Numerical Fluxes

As is explained in [56], the choice of these numerical traces is crucial, as it can affect the stability, consistency and accuracy of the method and even the convergence to the exact solution in some cases.

As explained in details by Hesthaven in [21], in order to construct a numerical method, one has to decide on an approximate solution. Two natural questions arise on how to approximate the solution and in which sense will this approximation satisfy the initial problem.

As in [21], let us consider the one-dimensional scalar conservation law as follows:

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} = g, \quad x \in \Omega\,,$$

with appropriate initial and boundary conditions on $\partial\Omega$. Here, $f(u)$ is the flux and $g(x,t)$ a forcing function. We introduce D, a tessellation of Ω, such that $D^k = [x^k, x^{k+1}]$. In each element, we assume that we can express the solution u as a linear combination of Lagrange polynomials and form the approximation u_h (and the same applies for the flux f). The idea here, is that we want the approximation u_h to be as close as possible to the solution u, thus, we want the residual $R_h = \frac{\partial u_h^k}{\partial t} + \frac{\partial f_h^k}{\partial x} - g_h^k$ to vanish in each cell D^k. We introduce a space of discontinuous test functions V_h and impose that R_h is

orthogonal to all test functions in V_h, which gives us:

$$\int_{D^K}(\frac{\partial u_h^k}{\partial t}+\frac{\partial f_h^k}{\partial x}-g_h^k)\phi^k=0\,,\quad \forall\phi^k\in V_h\,,$$

where ϕ^k are the Lagrange polynomials. However, this problem is posed locally and there is no way to recover the global solution yet. Let us perform an integration by parts on $\int_{D^K}\frac{\partial f_h^k}{\partial x}\phi^k$ which results in:

$$\int_{D^K}(\frac{\partial u_h^k}{\partial t}-f_h^k\frac{\partial \phi^k}{\partial x}-g_h^k)\phi=-[f_h^k\phi^k]_{x^k}^{x^{k+1}}\,,\quad \forall\phi^k\in V_h\,.$$

The term on the right hand side is simply what connects all elements together, and is nothing more than what we introduced as the jump. Indeed, element D^k and element D^{k+1} are linked by their endpoint x^{k+1}, and this additional term requires the evaluation of the flux at x^{k+1} for both of them. Hence, let us denote the numerical flux as f^*, which is the only value we need at the interface, in order to make the elements communicate:

$$\int_{D^K}(\frac{\partial u_h^k}{\partial t}-f_h^k\frac{\partial \phi^k}{\partial x}-g_h^k)\phi=-[f^*\phi^k]_{x^k}^{x^{k+1}}\,,\quad \forall\phi^k\in V_h\,, \tag{4.7}$$

We understand here the role of the fluxes we always introduce in DG schemes, they serve the purpose to retrieve the global solution by connecting the elements together with appropriate conditions. As described later in [21], it is not the only purpose it serves. The role of the flux is also to guarantee stability of the formulation. Indeed, if the chosen approximation is a good one, (4.7) will be small. But if the approximation is not done well enough, (4.7) will be big and either result in a more dissipative DG method or in an unstable method, as is very well explained and detailed in [56]. In order to compensate for these drawbacks, the jump is often penalized in order to ensure stability, among other properties and the flux is usually written in the following form: $f^*=\{\!\{u_h\}\!\}+\gamma[\![f_h(u_h)]\!]+\tau\{\!\{u_h\}\!\}$, where γ and τ are penalty terms, varying between 0 and 1.

In this framework, we choose the numerical traces like the classical DG ones, proven to give good results. Thus, let us set the numerical fluxes as follows:

$$\begin{pmatrix}\hat{\mathbf{v}}\cdot\mathbf{n}\\ \hat{p}\end{pmatrix}=\begin{pmatrix}\{\!\{\mathbf{v}\cdot\mathbf{n}\}\!\}+\beta_1[\![p]\!]_x\\ \{\!\{p\}\!\}+\alpha_1[\![\mathbf{v}]\!]_x\end{pmatrix}\quad\text{on }\mathcal{F}^e\,,$$

$$\begin{pmatrix}\breve{\mathbf{v}}\\ \breve{p}\end{pmatrix}=\begin{pmatrix}\{\!\{\mathbf{v}\}\!\}+\alpha_2[\![\mathbf{v}]\!]_t\\ \{\!\{p\}\!\}+\beta_2[\![p]\!]_t\end{pmatrix}\quad\text{on }\mathcal{F}^e\,,$$

$$\begin{pmatrix}\hat{\mathbf{v}}\cdot\mathbf{n}\\ \hat{p}\end{pmatrix}=\begin{pmatrix}g_N\\ p+\alpha_1(\mathbf{v}\cdot\mathbf{n}_x-g_N)\end{pmatrix}\text{on }\mathcal{F}^D\,,$$

$$\begin{pmatrix}\breve{\mathbf{v}}\\ \breve{p}\end{pmatrix} = \begin{pmatrix}1/2(\mathbf{v}^0+\mathbf{v}) + \alpha_2(\mathbf{v}^0-\mathbf{v})\\ 1/2(p^0+p) + \beta_2(p^0-p)\end{pmatrix} \text{on } \mathcal{F}^0\,,$$

$$\begin{pmatrix}\breve{\mathbf{v}}\\ \breve{p}\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ p\end{pmatrix} \qquad \text{on } \mathcal{F}^T\,.$$

We can notice the presence of penalty parameters α_1, α_2, β_1 and β_2. As mentioned before, the choice of the fluxes can be crucial and so is the choice of these parameters. Moreover, it was shown in [24], that the choice of non-zero parameters actually improves the accuracy and convergence of the scheme. We use the most common parameters used for classical DG methods, however, other possible choices have been explored by many. Some of these choices are reviewed in [21, 60]. Let us recall that g_N is the Neumann boundary condition function and $\mathbf{v}^0$ and p^0 are the initial conditions, *i.e.*, they give the values of $\mathbf{v}$ and p at initial time $t = 0$. To obtain our variational formulation, we are going to inject the above defined numerical fluxes inside (4.6) and sum the integrals over all elements $K \in T_h$. Hence, here is our DG variational formulation:

Seek $(\mathbf{v}, p) \in \mathbf{V} \times V$*, such that for all* $(\mathbf{w}, q) \in \mathbf{V} \times V$*, it holds true:*

$$\mathcal{A}_{dg}(\mathbf{v}, p; \mathbf{w}, q) = l_{dg}(\mathbf{w}, q)$$

where

$$\begin{aligned}
\mathcal{A}_{dg}(\mathbf{v}, p; \mathbf{w}, q) := &- \sum_{K\in T_h} \int_K p\Big(\frac{1}{c^2\rho}\frac{\partial q}{\partial t} + \operatorname{div}\mathbf{w}\Big) + \mathbf{v}\cdot\Big(\rho\frac{\partial \mathbf{w}}{\partial t} + \nabla q\Big)\\
&+ \int_{\mathcal{F}^e} \frac{1}{c^2\rho}\{\!\{p\}\!\}[\![q]\!]_t + \rho\{\!\{\mathbf{v}\}\!\}\cdot[\![\mathbf{w}]\!]_t + \{\!\{\mathbf{v}\}\!\}\cdot[\![q]\!]_x + \{\!\{p\}\!\}[\![\mathbf{w}]\!]_x\\
&+ \int_{\mathcal{F}^e} \frac{\beta_2}{c^2\rho}[\![p]\!]_t[\![q]\!]_t + \rho\alpha_2[\![\mathbf{v}]\!]_t\cdot[\![\mathbf{w}]\!]_t + \beta_1[\![p]\!]_x\cdot[\![q]\!]_x + \alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x\\
&+ \int_{\mathcal{F}^T} \frac{1}{c^2\rho}pq + \rho\mathbf{v}\cdot\mathbf{w}\\
&+ \int_{\mathcal{F}^0} (0.5-\beta_2)\frac{1}{c^2\rho}pq + (0.5-\alpha_2)\rho\mathbf{v}\cdot\mathbf{w}\\
&+ \int_{\mathcal{F}^D} p\mathbf{w}\cdot\mathbf{n}_x + \alpha_1(\mathbf{v}\cdot\mathbf{n}_x)(\mathbf{w}\cdot\mathbf{n}_x)\\
l_{dg}(\mathbf{w}, q) := &\sum_{K\in T_h}\int_K fq\\
&- \int_{\mathcal{F}^0} (0.5+\beta_2)\frac{1}{c^2\rho}p^0q + (0.5+\alpha_2)\rho\mathbf{v}^0\cdot\mathbf{w}\\
&+ \int_{\mathcal{F}^D} \alpha_1 g_N(\mathbf{w}\cdot\mathbf{n}_x) - qg_N
\end{aligned}$$

Now that we have brought in all the DG-aspects of our method, it remains to introduce the Trefftz space in which the basis functions will be chosen. As has already been

explained before, the idea of the Trefftz method is to take solutions as basis functions. Thus, our global Trefftz space can be defined as follows:

$$\mathbf{T}(T_h) = \Big\{ (\mathbf{v}, p) \in \mathbf{V} \times V, \text{ such that } \frac{1}{c^2\rho}\frac{\partial p}{\partial t} + \nabla \cdot \mathbf{v} = 0 \text{ and } \rho\frac{\partial \mathbf{v}}{\partial t} + \nabla p = 0 \quad \forall K \in T_h \Big\}$$

Taking $(q, \mathbf{w})$ in $\mathbf{T}(T_h)$ leads to all volumic terms vanishing from (4.4.1). For simplicity, from now on we will consider a homogeneous acoustic wave equation with homogeneous Neumann boundary conditions, *i.e.* $f = 0$ and $g_N = 0$. **So, here is, at last, our Trefftz-DG variational formulation for the acoustic system:**

Trefftz-DG Variational Formulation

Seek $(\mathbf{v}, p) \in \mathbf{T}(T_h)$, *such that for all* $(\mathbf{w}, q) \in \mathbf{T}(T_h)$, *it holds true:*

$$\mathcal{A}_{tdg}(\mathbf{v}, p; \mathbf{w}, q) = l_{tdg}(\mathbf{w}, q)$$

where

$$\begin{aligned}
\mathcal{A}_{tdg}(\mathbf{v}, p; \mathbf{w}, q) := & \int_{\mathcal{F}^e} \frac{1}{c^2\rho}\{\!\{p\}\!\}[\![q]\!]_t + \rho\{\!\{\mathbf{v}\}\!\} \cdot [\![\mathbf{w}]\!]_t + \{\!\{\mathbf{v}\}\!\} \cdot [\![q]\!]_x + \{\!\{p\}\!\}[\![\mathbf{w}]\!]_x \\
& + \int_{\mathcal{F}^e} \frac{\beta_2}{c^2\rho}[\![p]\!]_t[\![q]\!]_t + \rho\alpha_2[\![\mathbf{v}]\!]_t \cdot [\![\mathbf{w}]\!]_t + \beta_1[\![p]\!]_x \cdot [\![q]\!]_x + \alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x \\
& + \int_{\mathcal{F}^T} \frac{1}{c^2\rho}pq + \rho\mathbf{v} \cdot \mathbf{w} \\
& + \int_{\mathcal{F}^0} (0.5 - \beta_2)\frac{1}{c^2\rho}pq + (0.5 - \alpha_2)\rho\mathbf{v} \cdot \mathbf{w} \\
& + \int_{\mathcal{F}^D} p\mathbf{w} \cdot \mathbf{n} + \alpha_1(\mathbf{v} \cdot \mathbf{n}_x)(\mathbf{w} \cdot \mathbf{n}_x) \\
l_{tdg}(\mathbf{w}, q) := & - \int_{\mathcal{F}^0} (0.5 + \beta_2)\frac{1}{c^2\rho}p^0q + (0.5 + \alpha_2)\rho\mathbf{v}^0 \cdot \mathbf{w} \\
& + \int_{\mathcal{F}^D} \alpha_1 g_N(\mathbf{w} \cdot \mathbf{n}_x) - qg_N
\end{aligned}$$

We can notice here that all volumic terms have disappeared from the variational formulation, which is now only posed on the boundaries of the domain.

4.4.2 Wellposedness of our problem

To establish uniqueness and existence of a solution of a weak variational form, one uses the Lax-Milgram theorem, which can be written as follows:

Lax-Milgram

Theorem 4.2. Let $(H, ||\cdot||)$ be a Hilbert space. Let $\mathcal{A}$ be a bilinear form such that:

- $\exists M > 0$, $\forall u, v$, $\mathcal{A}(u,v) \leq M||u|| \cdot ||v|| \Leftrightarrow \mathcal{A}$ is continuous,
- $\exists \alpha > 0$, $\forall u$, $\mathcal{A}(u,u) \geq \alpha||u||^2 \Leftrightarrow \mathcal{A}$ is coercive.

Let l be a linear continuous form.
There exists a unique $u \in H$ such that $\forall v \in H$, $\mathcal{A}(u,v) = l(v)$.

Hence, to establish the well-posedness of our variational formulation, we need to show that our bilinear form $\mathcal{A}(u,v)$ is continuous and coercive. To do so, let us first define the following quantities:

$$\begin{aligned}
|||\mathbf{w},q|||_{tdg} =& \left\|\alpha_1^{1/2}[\![\mathbf{w}]\!]_x\right\|^2_{L^2(\mathcal{F}^e)} + \left\|\beta_1^{1/2}[\![q]\!]_x\right\|^2_{L^2(\mathcal{F}^e)} \\
&+ \left\|\alpha_2^{1/2}\rho^{1/2}[\![\mathbf{w}]\!]_t\right\|^2_{L^2(\mathcal{F}^e)} + \left\|\beta_2^{1/2}(\frac{1}{c^2\rho})^{1/2}[\![q]\!]_t\right\|^2_{L^2(\mathcal{F}^e)} \\
&+\frac{1}{2}\left\|(\frac{1}{c^2\rho})^{1/2}q\right\|^2_{L^2(\mathcal{F}^T)} + \frac{1}{2}\left\|\rho^{1/2}\mathbf{w}\right\|^2_{L^2(\mathcal{F}^T)} \\
&+ \left\|\beta_2^{1/2}(\frac{1}{c^2\rho})^{1/2}q\right\|^2_{L^2(\mathcal{F}^0)} + \left\|\alpha_2^{1/2}\rho^{1/2}\mathbf{w}\right\|^2_{L^2(\mathcal{F}^0)} \\
&+ \left\|\alpha_1^{1/2}(\mathbf{w}\cdot\mathbf{n})\right\|^2_{L^2(\mathcal{F}^D)}
\end{aligned}$$

$$\begin{aligned}
|||\mathbf{w},q|||_{tdg^+} =& |||\mathbf{w},q|||^2_{tdg} \\
&+ \left\|\alpha_1^{-1/2}\{\!\{q\}\!\}\right\|^2_{L^2(\mathcal{F}^e)} + \left\|\beta_1^{-1/2}\{\!\{\mathbf{w}\}\!\}\right\|^2_{L^2(\mathcal{F}^e)} \\
&+ \left\|\alpha_2^{-1/2}\{\!\{\mathbf{w}\}\!\}\right\|^2_{L^2(\mathcal{F}^e)} + \left\|\beta_2^{-1/2}\{\!\{q\}\!\}\right\|^2_{L^2(\mathcal{F}^e)} \\
&+ \left\|(\frac{1}{2\beta_2}+1)^{1/2}(\frac{1}{c^2\rho})^{1/2}q\right\|^2_{L^2(\mathcal{F}^0)} + \left\|(\frac{1}{2\alpha_2}+1)^{1/2}\rho^{1/2}\mathbf{w}\right\|^2_{L^2(\mathcal{F}^0)} \\
&+ \left\|\alpha_1^{-1/2}q\right\|^2_{L^2(\mathcal{F}^D)}
\end{aligned}$$

We can show that the semi-norms $|||\mathbf{w},q|||_{tp}$ and $|||\mathbf{w},q|||_{tp^+}$ define norms on the discrete

space $\mathbf{T}(\mathcal{T}_h)$. We then have the following:

$$\begin{aligned}\mathcal{A}(\mathbf{w},q;\mathbf{w},q) = &\left\|\alpha_1^{1/2}[\![\mathbf{w}]\!]_x\right\|^2_{L^2(\mathcal{F}^{int})} + \left\|\beta_1^{1/2}[\![q]\!]_x\right\|^2_{L^2(\mathcal{F}^{int})} + \left\|\alpha_1^{1/2}(\mathbf{w}\cdot\mathbf{n})\right\|^2_{L^2(\mathcal{F}^{ext})}\\ &+ \left\|\beta_2^{1/2}(\frac{1}{c^2\rho})^{1/2}q|n_t|^{1/2}\right\|^2_{L^2(\mathcal{F}^{in})} + \left\|\alpha_2^{1/2}(\rho)^{1/2}w_x|n_t|^{1/2}\right\|^2_{L^2(\mathcal{F}^{in})}\\ &+ \int_{\mathcal{F}^{out}} \frac{1}{2n_t}\Big[\frac{1}{c^2\rho}(qn_t)^2 + \rho(w_xn_t)\cdot(w_xn_t) + 2(qn_t)(w_x\cdot\mathbf{n})\Big]ds\end{aligned}$$

Thus, the following result holds true:

Coercivity

Theorem 4.3. The following holds true $\forall(\mathbf{w},q)\in\mathbf{T}(K)$:

$$\mathcal{A}(\mathbf{w},q;\mathbf{w},q) \geq |||\mathbf{w},q|||^2_{tp}$$

Thus, $\mathcal{A}$ is a coercive bilinear form.

To prove continuity, we sum the norms of all terms in the bilinear and linear form and apply the Cauchy-Schwartz inequality, which leads to the following:

Continuity

Theorem 4.4. The following holds true $\forall(\mathbf{v},p),(\mathbf{w},q)\in\mathbf{T}(K)$:

$$\begin{aligned}\mathcal{A}_{tp}(\mathbf{v},p;\mathbf{w},q) \leq& C_1^{tp}|||\mathbf{v},p|||_{tp}|||\mathbf{w},q|||_{tp}\,,\\ l_{tp}(\mathbf{w},q) \leq& \Bigg[\left\|(\frac{1}{2\beta_2}+1)^{1/2}(\frac{1}{c^2\rho})^{1/2}q\right\|^2_{L^2(\mathcal{F}^{in})}\\ &+ \left\|(\frac{1}{2\alpha_2}+1)^{1/2}(\rho)^{1/2}\mathbf{w}\right\|^2_{L^2(\mathcal{F}^{in})}\Bigg]^{1/2}|||\mathbf{w},q|||_{tp}\end{aligned}$$

Thus, $\mathcal{A}_{tp}$ is a continuous bilinear form and l_{tp} is a continuous linear form.

Hence, since all conditions of the Lax-Milgram theorem are met, the variational formulation has a unique solution.

The Trefftz-DG method is dissipative as it has been shown by Moiola and Perugia in [68]. To do so, energy estimates $\mathcal{E}$ are calculated on two faces $\Sigma_1(\mathbf{x}, f_{\Sigma_1}(\mathbf{x}))$ and $\Sigma_2(\mathbf{x}, f_{\Sigma_2}(\mathbf{x}))$, with f_Σ a Lipschitz-continuous function whose Lipschitz constant $L < 1/c$ and $f_{\Sigma_1} \leq f_{\Sigma_2}$. It is then shown that the following inequality between the two energy estimates holds: $\mathcal{E}(\Sigma_1) < \mathcal{E}(\Sigma_2)$.

Moreover, the norms we have introduced are mesh-dependent. Moiola and Perugia also establish error bounds with mesh-independent norms in [68].

4.5 Polynomial basis functions

We have not yet defined the discrete Trefftz space in which our basis is taken. As explained before, Trefftz functions need to be solutions to the governing equations, but other than that, the choice of basis is quite flexible. For time-harmonic problems, commonly used Trefftz basis are plane waves and generalized harmonic polynomials. Trefftz functions for non-stationary problems have been developed by many and for multiple frameworks, such as the parabolic equation, the wave equation, the beam vibration, etc. These Trefftz functions are surveyed in [69] by Grysa and Maciejewska. In this framework, we use spacetime polynomials that are solutions to the acoustic wave equation.

There are many ways to find polynomial solutions to the wave equation. One of them is described by Maçiag in [70], and consists in finding a generating function for wave polynomials

$$g = e^{i(ax+by+dt)} ,$$

which satisfies the second-order wave equation if $d^2 = a^2 + b^2$. Then, g is expanded into Taylor series, which results in:

$$g = \sum_{n=0}^{\infty} \sum_{k=0}^{n} \sum_{l=0}^{n-k} S_{n-k-l,k,l}(x,y,t) a^{n-k-l} b^k d^l ,$$

where $S_{n-k-l,k,l}(x,y,t)$ are spacetime polynomials. Replacing d^2 by $a^2 + b^2$, we obtain:

$$g = \sum_{n=0}^{\infty} \sum_{k=0}^{n} \sum_{\substack{i=0 \\ i<2}}^{n-k} R_{n-k-l,k,l}(x,y,t) a^{n-k-l} b^k d^l .$$

Each imaginary and real part of R satisfies the wave equation and thus forms the polynomial basis for the wave equation. Using this idea, Shishenina ([24]) extended this for the acoustic wave equation, which resulted in a spacetime wave polynomial basis for the first-order acoustic wave equation.

Another way of deriving spacetime polynomial solutions is shown in [27] and [26]. A spacetime polynomial $R(\mathbf{x},t) = \sum a_{k,\alpha} \mathbf{x}^{\alpha} t^k$ is searched as to satisfy the second-order wave equation. Hence, the expression is injected into the wave equation and equal power terms are collected.

In this framework, we use the polynomials obtained using the first method as described in [24], and they can be found in Appendix A.

Hence, our global and local discrete Trefftz space can be written as follows:

$$\mathbb{T}^p(T_h) = \Big\{ (\mathbf{v},p) \in \mathbf{T}(T_h), (\mathbf{v},p) \in \mathbb{P}^p(K)^d \times \mathbb{P}^p(K) , \quad \forall K \in T_h \Big\}$$

$$\mathbb{T}^p(K) = \Big\{(\mathbf{v}, p) \in \mathbf{T}(K), (\mathbf{v}, p) \in \mathbb{P}^p(K)^d \times \mathbb{P}^p(K)\Big\}$$

where $\mathbb{P}^p$ is the set of polynomials presented in Appendix A of degree p and smaller, for $p = 1, 2, 3$.

4.6 Drawbacks

When applying the previously presented Trefftz-DG method to solving the acoustic wave equation, one is faced with some limitations. Indeed, as explained in [24], we end up with an implicit scheme and a huge sparse matrix to invert. This situation is not ideal and the computational costs are high. To overcome this, another presented approach is the computation in time slabs, which results in a block-diagonal matrix, which is more ideal than the previous sparse matrix and its inverse matrix is approximated. In fact, we can see in Fig. 4.6 that this approach (denoted $TDG2D_{ai}$ and depicted in orange, where ai stands for approximate inverse) leads to a better computational time than the first approach (denoted $TDG2D_{ei}$ and depicted in gray, where ei stands for exact inverse). But the computational costs arising from this method are still high and a new approach needs to be found.

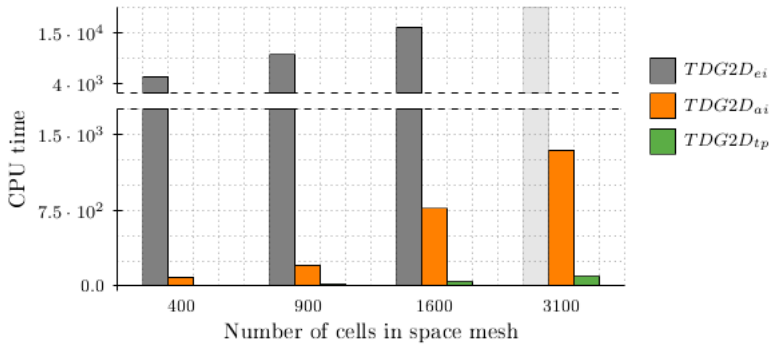


Figure 4.6: Time and memory limitations[2]

In the next chapter, we propose to solve the problem element-by-element, under some constraints on the mesh. This stems directly from the hyperbolic nature of the equations and boils down to applying the Tent-Pitcher algorithm to our problem. It has already been explored in [24] for acoustic, elastic and elasto-acoustic wave equations on structured meshes and in [26] and [27] for the acoustic wave equation on unstructured meshes.

[2]Source: [24] Space-Time Discretization of Elasto-Acoustic Wave Equation in Polynomial Trefftz-DG Bases, E. Shishenina, 2018

Chapter 5

Tent-Pitcher Algorithm

Using Finite Element approaches to solve time-dependent problems is classical and has been widely explored and most of them consist in using a semi-discretization method. This means that first, the space domain is discretized using Finite Elements, from which we obtain a set of ordinary differential equations in time and then, we discretize the obtained equations using a time-stepping method. Hence, we separate the problem in two, discretizing the spatial problem first then discretizing the temporal problem second. It is also possible to adopt the idea of discretizing the time derivatives first and then the space operators, leading to the so-called modified equation technique (see for instance [71] and the references therein).

In the case of Trefftz methods applied to transient problems, if we want to fully exploit the potential of Trefftz formulations (*i.e.* the vanishing of volumic terms in the variational formulation), we need to introduce spacetime solutions to the considered problem. Thus, we cannot separate the problem in space and time and perform a semi-discretization in space, followed by a time-stepping method. Indeed, the Trefftz-DG method is based on the idea of using approximation spaces made of discontinuous functions which are defined element-by-element as local solutions to the considered problem. If we consider the Helmholtz equation or more generally wave equations in harmonic regime coupled with simple boundary conditions (Dirichlet, Neumann, first order absorbing conditions), the associated variational problem is simplified by involving only integral terms defined on the skeleton of the mesh, *i.e.*, on the edges or faces of the elements, and even more remarkably without involving any differential operator. This property is an inheritance of the reciprocity property of the wave equations, a condition itself inherited from the divergence formula. In the time domain, we can obtain the same type of formulation and for that, we must absolutely write a variational formulation fully in space and time. Indeed, we can then exploit the fact that the test functions are solutions to the problem considered in each of the elements and thus reduce ourselves to a problem posed on the skeleton of the mesh. Moreover, spacetime methods naturally allow local time-stepping when the time step is constrained by the space step, which is

advantageous.

The Trefftz-DG method applied to wave equations has been carried out by Shishenina ([24]) followed by Stocker ([27]) and it turns out that a direct solution of the variational problem faces the difficulty of having an implicit scheme in time. This is a serious drawback for applying this method to seismic imaging problems which are so memory intensive that only explicit time schemes can be considered. It is therefore important to find an integration method that makes the solution calculation explicit. In her thesis, E. Shishenina proposed to apply a sort of time discretization which amounts to solve the problem in slices for a given time step so that the scheme is explicit. But methods proceeding by layers in time suffer from a global time step, which is constrained by the smallest element in the initial space mesh, making them costly. This leads us to consider other options, such as the more appealing element-by-element approach. In [24] and [27], the Tent-Pitcher algorithm is considered and so do we.

The wave equations are hyperbolic equations, which means that their solution propagates at finite speed. In other words, the wave field is defined at a point in spacetime by the values of the wave field within a domain constrained by the values of the physical parameters (typically the speed of propagation) and this domain is defined by respecting the associated causality principle.

This property stems directly from Huygens' principle, which explains the fact that light propagates in a straight line. To demonstrate this statement, one needs to analyze the following setup: a short signal is emitted from a point $O(x, y, t_0)$ (a shock) and is perceived at a random point $A(x, y, t_2)$ after some time T, which is determined by the ratio between the distance OA separating the two points and the wave propagation speed c. Christian Huygens considered a middle point between O and A and the state of this new point could entirely be determined thanks to the state of the initial perturbation point. Then, the state of A could be determined using the new intermediate state. We summarize this idea in the same way as in [72] as follows:

Huygens' Principle

Let t_0 be the moment when the initial perturbation occurs, t_1 an intermediate moment and t_2 the final moment, at which we want to know the produced effect from the shock.

A. To deduce the state at t_2 from the known state at t_0, we can first determine the state at t_1 and from it, deduce the one at t_2.

B. If the initial shock is located in the neighborhood of a point O, it will only impact a neighborhood of a sphere S of center O and radius $c(t_1 - t_0)$, where c is the propagation speed.

C. At t_2, the initial perturbation can be substituted by a series of perturbations at t_1 and properly distributed on the surface of S.

Not only is Huygen's principle fascinating in itself and constitutes the foundation of the wave theory of light, but it also raised many disagreements within the scientific community which contributed greatly to the field. One can find much more details on this subject in many works, one of which is by Hadamard[72]. This last one presents the principle in a very interesting manner, along with historical facts concerning this matter.

We can see that Huygen's principle perfectly depicts the above mentioned fact that if the initial data are compactly supported, then the solution will also have a compact support. This property, typical of hyperbolic equations, has led to the idea of considering causal spacetime meshes composed of tents (triangles in 1d, pyramids with polygonal base in 2d) whose construction is dynamic and orchestrated by a time advance that is not necessarily the same in each tent. This approach is known as the Tent-Pitcher algorithm and is combined with a numerical scheme. In this thesis, and as in most works using Tent-Pitching, we consider a discontinuous Galerkin method which is conducive to parallel computation and very efficient for high order approximations. The solution is computed at each point of the tent and the values of the solution at the edges of the tent are used as initial conditions to perform the computations in the neighboring tents that follow in time. This method preserves the parallelization potential of DG methods, since many tents can be solved independently. It was proposed in 2000 by Üngör and Sheffer [35]. The height of each tent represents the time step and can be different for each tent. Local-time stepping is thus possible which contributes to ensure the stability of explicit integration while providing a lower computational cost.

In this chapter, we introduce the Tent-Pitcher algorithm and construct the variational formulation for the wave equation that is actually solved later.

5.1 Domains of dependence and influence

In this section, we consider the 1D acoustic wave equation to illustrate its main properties allowing the implementation of the Tent-Pitcher algorithm, especially the finite wave propagation speed. We consider the second-order wave equation in one dimension, which can be written:

$$\begin{aligned}
&\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \frac{\partial^2 p}{\partial x^2} = 0\,, \qquad x \in \mathbb{R}\,, t \in \mathbb{R}^+ \\
&p(\cdot, 0) = p_0\,, \\
&\frac{\partial p}{\partial t}(\cdot, 0) = p_1\,.
\end{aligned}$$

The fields p_0 and p_1 represent the solution at time $t = 0$, so they are our initial solutions. A general solution of this equation is given by the d'Alembert formula:

$$p(x,t) = \frac{1}{2}(p_0(x-ct) + p_0(x+ct)) + \frac{1}{2c}\int_{x-ct}^{x+ct} p_1(s)ds$$

According to this formula, we can see that if the support of p_0 and p_1 is in the interval $[-R; R]$, then the support of the solution is in the interval $[-R - ct; R + ct]$, which depicts the property of finite propagation speed. This formula also indicates that at a point $M = (x, t)$, the solution is fully determined from the knowledge of the initial data.

To illustrate these notions, let us consider a case as depicted in Fig. 5.1, where p_0 and p_1 have a compact support, *i.e.*,

$$\begin{aligned}
p_0, p_1 &\neq 0 \quad \text{in } [a, b] \\
p_0, p_1 &= 0 \quad \text{elsewhere.}
\end{aligned}$$

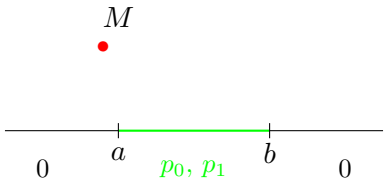
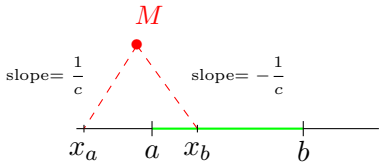
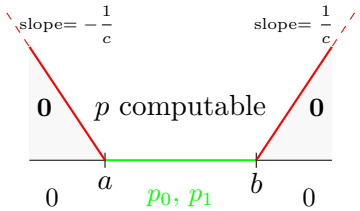


Figure 5.1: The initial condition vanishes outside $[a, b]$

In this case, we want to determine the solution p at M and so, we start by drawing the lines along the characteristic varieties of the wave equation. Here, this means that we draw the lines of slope $\pm\, 1/c$ passing through the point M, as depicted in Fig. 5.2a.



(a) Domain of dependence



(b) Domain of influence

Figure 5.2: Case 1

The domain bounded by the dashed red lines is the **domain of dependence** of the point M. According to d'Alembert's formula, if the initial datum is known in the interval $[x - ct, x + ct]$, then one can compute the solution p at the point (x, t). In this case, $x_a = x - ct\,, x_b = x + ct$ and $[x_a, x_b] = [x_a, a] \cup [a, x_b]$. By the definition of the support of p_0 and p_1, they equal zero in $[x_a, a]$, whereas in $[a, x_b]$, they are non-zero functions. Hence, the solution $p(x, t)$ can be fully determined.

Thus, since the initial datum is known everywhere (either equal to zero or to a non-null function), the solution can be determined at all points by following the same method. By doing so, we obtain that the solution is non-zero in the interior domain bounded by the red lines in Fig. 5.2b and zero elsewhere. This domain represents the **domain of influence** of $[a, b]$.

Hence, Fig. 5.3 represents the domains of dependence and of influence of a point M (we refer the reader to [73] for a nice introduction of the domains of dependence and influence).

Remark. *The solution at point M is fully determined by the solution in its domain of dependence. However, knowing the solution at M is only necessary and not sufficient for finding the solution in its domain of influence. The domain of influence of M regroups all the points that will be influenced by the solution at M in the process of finding their own solution.*

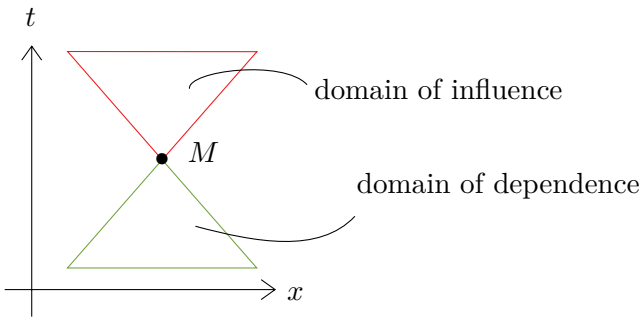
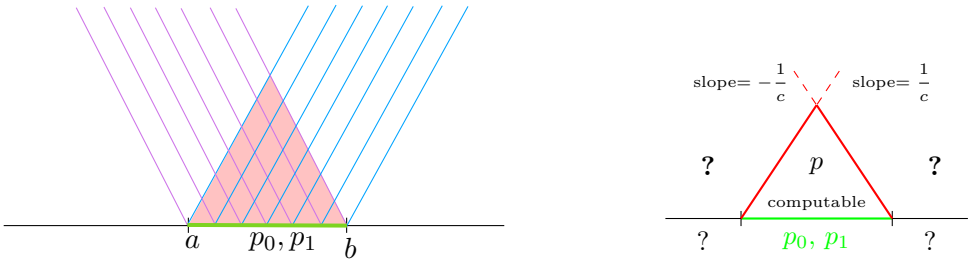


Figure 5.3: Domains of dependence and influence

Let us now assume that p_0 and p_1 are only defined in the interval $[a, b]$ and unknown elsewhere. Following the previous methodology, to find the domain in which the solution is computable knowing p_0 and p_1 in the interval $[a, b]$, we will draw the domains of influence of all points in $[a, b]$ and take their intersection as depicted in Fig. 5.4a. By doing so, we obtain the domain bounded by the red lines in Fig. 5.4b, which is the domain of dependence of the point (x, t).

Thus, this depicts perfectly the idea conveyed by d'Alembert formula. If we want to know the solution p at a point (x, t), we need to know the initial data at the bottom of the triangle bounded by the lines $x - ct$, $x + ct$ and $t = 0$. Taking this the other



(a) Intersection of domains of influence of points in $[a, b]$

(b) Domain of dependence

Figure 5.4: Case 2

way around, **if p_0 and p_1 are known in an interval $[a, b]$, we can determine the solution $p(x, t)$ at every point (x, t) in the domain formed by the lines $x - ct$, $x + ct$ and the segment $[a, b]$.**

If we have a 1D mesh with several cells, we will proceed in the same manner for all of them and progress step-by-step using this notion of domain of dependence, provided that we know the initial solution for all cells, which is the core of Tent-Pitcher algorithm. Since the mesh's construction depends on the wave propagation speed c, we call it a *causal* mesh. Indeed, a mesh cell at time t cannot be constructed before its neighboring mesh cells at time $t - 1$. Hence, **to guarantee causality of a Tent-Pitching mesh, all faces of a tent need to respect the following:** $c|\mathbf{n}| \geq n_t$**,** called the *causality constraint*. This constraint is explained in the following.

The mesh cells in a Tent-Pitcher algorithm will be called tents from now on. To fix notations for the rest of the thesis, let us separate the boundaries of a tent in two categories:

- the initial faces where the solution is known will be referred to as *inflow faces*. For example in Fig. 5.4b, the green cell is an inflow face, since we know the solution on it and use it to compute the solution in the whole tent,
- the rest of the boundaries will be referred to as *outflow faces*. In this example, the lines of slope $\pm\frac{1}{c}$ of the tent are outflow faces, on which we will compute the solution.

In practice, there exist other kinds of tents with additional types of boundaries. We will explicit them at a later stage. It is also interesting to notice that the outflow faces become inflow faces for the next layer of elements.

5.2 Tent-Pitcher algorithm

The Tent-Pitcher algorithm originates from the previously described ideas and so, was introduced for spacetime hyperbolic problems by Alper Üngör and Alla Sheffer in [35]. As explained above, the idea of this algorithm is to construct a causal mesh which respects the wave propagation speed.

The idea of element-wise resolution is not new and has already been explored in [38], but the proposed algorithm is based on an initial spatial quadrilateral mesh, which makes it harder to fit complex geometries. The algorithms proposed by Üngör *et al.* [35] deal with fully triangular unstructured spatial meshes, which are better-suited to complex topologies. We will explicit the algorithm of Lowrie *et al.*[38] in section 5.4.1, as it is the one used by Shishenina [24] that we mostly worked with in order to have a prototype.

Many spacetime meshing algorithms are based on a time-layer strategy (see [74] and the references therein), but the Tent-Pitcher algorithm in [35] does not rely on layers and advances the mesh element-by-element. Generally, methods proceeding by layers in time suffer from a global time step, which is constrained by the smallest element in the initial space mesh, which is the reason why an element-by-element resolution algorithm such as the Tent-Pitching one is appealing. Let us explain how it works. The outer facets of the mesh are referred to as a *front* and as in [35] an angle function α is introduced, depicted in Fig. 5.5. For every point M of the mesh, α is determined by the boundaries of the domain of influence of M. To ensure the causality of the mesh, each point M has to respect the cone constraint. This means that the dihedral angles of each face of the mesh with respect to the spatial domain need to be less or equal to α or equally, $c|\mathbf{n}| \leq n_t$. Indeed, this is equivalent to saying that the slope of each facet of the mesh has to be less than $1/c$, and we have that $\alpha = \arctan {|\mathbf{n}|}/{n_t} \leq \arctan 1/c$.

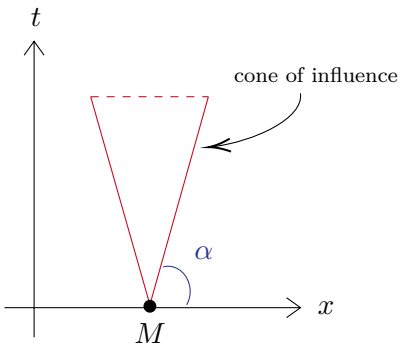


Figure 5.5: Angle function α

The general meshing and resolution algorithm is described in Algorithm 1. It consists of an advancing front algorithm, which uses a Tent-Pitcher algorithm. It is used to numerically solve a hyperbolic problem.

First, let us define some terms:

- tentpole: the vertical line that connects a point M to its projection M' advanced in time. $M' = M + \Delta t$, where Δt is the height of the tentpole,
- quality value: a value attributed to each node, based on the quality of a potential tent on it,
- priority queue: structure for storing the nodes ordered by their quality value.

Algorithm 1: Meshing and resolution algorithm

- construct initial space mesh D at time $t = 0$;

while $t \leq T$ **do**

  while *domain D is not covered* **do**

    - advance front using Tent-Pitcher algorithm;
      - ◇ select a point to advance in time;
      - ◇ construct the tent by computing its tentpole height;
      - ◇ recompute the tents and their quality value;
      - ◇ add the new tent to the priority queue;
    - solve numerical problem in newly created tents;
    - update cone constraint;

  end

  - $D \leftarrow$ front

end

Remark. *We call the tentpole height "Δt", because when we extrude a point by the chosen height, we actually advance it in time by that quantity. Thus, the tentpole height is the time-step Δt of the method. Notice that the time-steps are determined for each tent, thus Tent-Pitcher algorithms naturally allow local time-stepping.*

Let us explain some parts of this algorithm, particularly the computation of the tentpole height Δt and the selection of a pitch point.

The computation of Δt is a crucial part, since it ensures the causality of the mesh. Let M be the point we want to advance in time. Let us call $\texttt{star}(M)$ all the points connected to M and $\texttt{link}(M)$ all the edges whose endpoints are in $\texttt{star}(M)$, as shown in Fig. 5.6. Each edge $e \in \texttt{link}(M)$ has a cone plane (plane that respects the cone constraints of all points in e) that intersects with the tentpole. To find Δt, we choose the lowest intersection point between all cone planes and the tentpole.

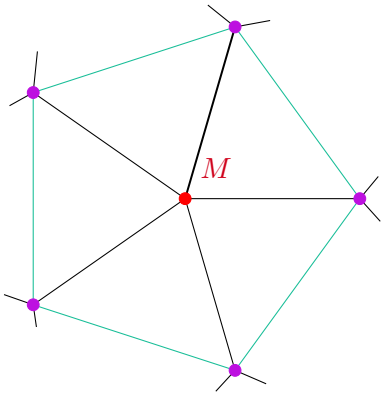


Figure 5.6: The nodes in purple represent the set `star`(M) and the edges in cyan represent the set `link`(M)

To select a point to advance in time, there are several rules and strategies. First, we have to ensure causality and then, we can choose a point as to optimize the mesh quality. To do so, we have various options such as choosing the lowest node, maximizing the tent volume... They are all presented in [35] and in our case, we use the lowest node method to choose our point to advance in time; this will be presented in a later section.

In [36], Erickson *et al.* extended the algorithm of [35] to any simplicial space meshes. Indeed, in the original Tent-Pitcher algorithm, the mesh can be fully triangular but each triangle has to be acute, in order to guarantee progress of the mesh until any desired time. In [36], an additional constraint is imposed to maintain progress of the mesh, and thus use Tent-Pitching for any kind of simplicial mesh. The new constraint is called the *progress constraint*. Consider a triangular face pqr, with p the point we want to pitch, qr the remaining vertices of the triangle, the time components of each node of the face $t(p), t(q), t(r)$, with $t(p) \leq t(q) \leq t(r)$ and the distance w_p from point p to its opposite side $[qr]$. The idea of the progress constraint is to make sure that the chosen pitch-point p can always be advanced above the middle vertex q without violating the cone constraint. Hence, the progress constraint can be described with the following inequality:

$$t'(p) \leq t(r) + (1 - \varepsilon)w_p$$

where $\varepsilon \in [0, 1/2]$ is a fixed constant and $t'(p)$ is the time at which point p will be advanced. When choosing a point to pitch, we need to ensure that the progress constraint is respected for all facets connected to the point in question.

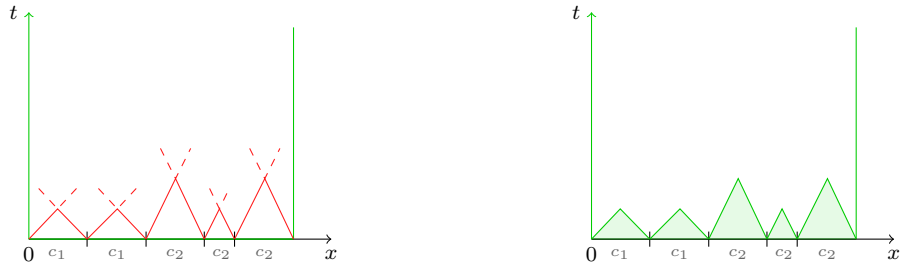
In [44], Gopalakrishnan *et al.* introduce a new class of methods called Mapped Tent-Pitching. The goal is to map the tents to a cylindrical element, to be able to separate space and time and apply classical methods in space combined with high-order time-stepping methods. The reason to do so is to obtain a fully explicit method, even within a tent.

In [75], Abedi *et al.* studied the case of elastodynamic wave propagation. When shocks, cracks or fractures are involved, the problem becomes multi-scale. This multi-scale property induces a need for mesh refinement, to be able to capture small-scale phenomena. However, static refinement is not optimal, since it will be constrained by the smallest scale and will result in high computational costs and other issues (dispersion, ...). In this paper, the Tent-Pitcher algorithm is used to allow non-uniform mesh refinement in space and time and avoid the above mentioned issues. The authors improved the Tent-Pitcher algorithm and present an adaptive and parallel version up to 3D+time. Since we are working in the homogeneous case, we do not have these issues, hence use the regular algorithm.

5.3 Tent-Pitching in 1D+time

In this section, we would like to explain how we construct the mesh in 1D+time with the Tent-Pitcher algorithm. We place ourselves in the setting of the resolution of an acoustic wave equation with a Trefftz-DG method in 1D+t.

We have a 2D graph where the horizontal axis represents space x and the vertical axis represents time t. We have five space cells; two of them have a wave speed c_1 and the remaining three have a wave speed c_2. The grid size is not constant. The color green means that the solution is known in the corresponding region. Here, we will proceed in the same manner as in Section 5.1, where we explained the notions of domains of dependence and influence based on the d'Alembert formula. So, we will use the center of the cells as pitch points, unlike the works seen previously, where vertices of the space mesh were pitched. From now on, when a line is depicted in green on a figure, it means that the solution is known at all points on the line.



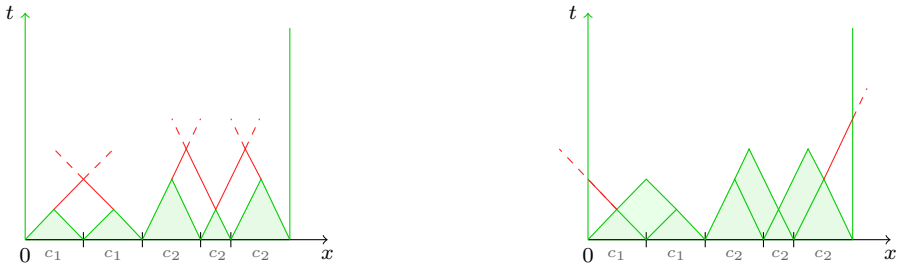
(a) Construction of the tents

(b) Computation of the solution

Figure 5.7: First layer of the mesh

In Fig. 5.7a, we can see that several lines are already green. Indeed, the solution is known at $t = 0$ (initial solution) and also on the domain boundaries (boundary conditions).

As previously, we construct the red tents, which are obtained by intersecting the domains of influence of each point in the cells at $t = 0$. The red lines bounding each tent have a slope of $\pm\frac{1}{c_1}$ or $\pm\frac{1}{c_2}$, depending on the velocity in the corresponding space cell. Thus, we obtain our first layer of tents and, as depicted in Fig. 5.7b, we can compute the solution in each of them and depict them in green from now on.



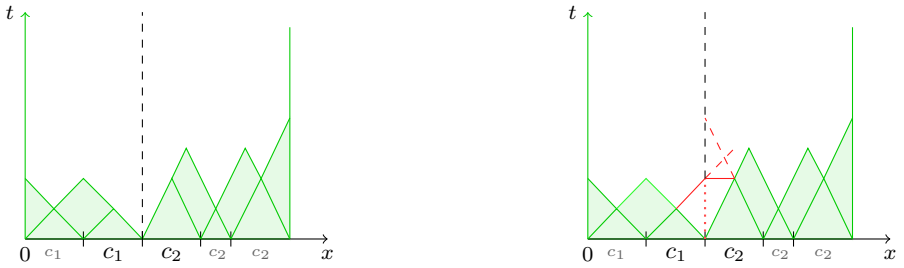
(a) Construction of classical tents

(b) Construction of boundary tents

Figure 5.8: Second layer of the mesh

For the second layer, the previous outflow faces are now inflow faces for the new tents. They are constructed in the same way, by drawing lines of slope $\pm\frac{1}{c_1}$ or $\pm\frac{1}{c_2}$ as shown in Fig. 5.8a.

On the boundary of the domain, it works in the same way, except that we will only have one line to draw. The second line is not needed since the boundary itself closes the tent (see Fig. 5.8b). In this case, we have one inflow face and one boundary face and in both cases, the solution is known. So, we can proceed as before and compute the solution inside these tents as well.



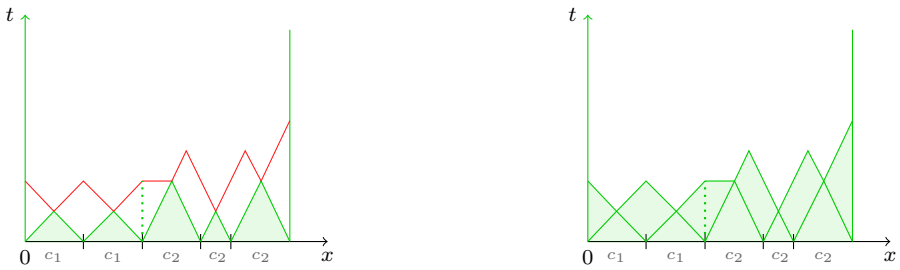
(a) Interface of the domain

(b) Construction of the heterogeneous cell

Figure 5.9: Heterogeneity

Now, let us see how to deal with the heterogeneous cell. As we can see in Fig. 5.9b, the second and third cells have very different-sized tents, induced by their respective wave

speed. It can be seen in Fig. 5.9a that the domain is separated into two subdomains by a vertical dashed interface and each of them has its own wave propagation speed, that we consider unvarying in time. In fact, since we work in a Trefftz environment, we need to consider the fact that we use homogeneous solutions as basis, hence each part of the domain will have its own set of basis. Hence, if we draw the lines along the characteristic varieties of the equation for the heterogeneous cell, one line will intersect with the interface before the other one does (see Fig. 5.9b). So, we obtain two possible heights for our tent; we need to choose the smallest one in order to respect the causality. In fact, if we chose the highest height, the tent would violate the cone constraint in its left part, *i.e.*, $c|\mathbf{n}| > n_t$.



(a) Construction of the tents

(b) Computation of the solution

Figure 5.10: Second layer of the mesh

Thus, we have constructed our second layer and we can compute the solution inside all these tents, using the solution computed in the first layer as initial solution, as can be seen in Fig. 5.10b. We will keep constructing tents until we reach or exceed a time T.

In the advantages of the Tent-Pitcher algorithm, we mentioned that it is very conducive to parallel computing. Indeed, we can clearly see in Fig. 5.11 that two patches can be computed independently.

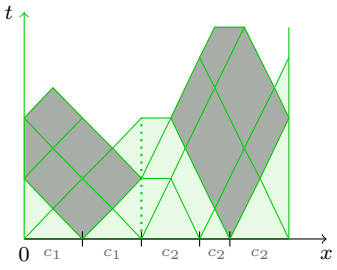


Figure 5.11: The grey spacetime patches can be computed independently from each other

5.4 Tent-Pitching in 2D+time

Now that we have explained Tent-Pitcher algorithm in 1D+time in our framework, let us see the process in 2D+time. Actually, the process is very similar and could be considered as an extension of the one dimensional case, but it is not straightforward to visualize how the mesh is constructed and what it looks like. That is why we will detail the 2D case as well.

We will give the details of two different algorithms. In fact, the first version of our code for the 2D+time problem works with a structured mesh and it is based on the same algorithm as in [38]. Whereas the second version works with an unstructured mesh and is similar to [36]. In practice, we are still following the same main steps for both of these algorithms:

- choose a point to pitch,
- construct the new cell by extruding the said point by Δt,
- compute the solution,
- interpolate the solution on the outflow faces.

And we repeat this process until all points reach or exceed a chosen time T.

The changes between the structured and the unstructured mesh mainly operate on the choice of the pitch-point and the expression of Δt, which is the tent-pole height. In fact, for structured meshes we will pitch the center of a cell, whereas for unstructured meshes, we will choose a vertex of the mesh to be pitched. The reason why we extrude the center of the cell for structured meshes, is because this gives us a way to completely control the created elements in our mesh and hence, introduce an appropriate reference element and be sure to keep a structured mesh at each step of its dynamical construction. This would be more difficult to achieve if we choose a node of the mesh as a pitch-point. In the unstructured case, we extrude nodes of the mesh, which leads to bigger tents compared to the tents we would obtain if we extruded the center of the mesh facets. Moreover and more importantly, since the underlying mesh is unstructured, if we selected cell centers as pitch points at the beginning, we could not continue following this concept for the next layers, since the lowest points in time would lie on edges and/or nodes of the mesh. Hence, it is better to choose nodes of the mesh since it is a choice that we can reproduce for every tent and is more adapted to unstructured meshes.

Let us now see the details of the Tent-Pitcher algorithm for both types of meshes.

5.4.1 Structured mesh

In the structured case, the mesh is a grid in space and then we extrude pyramids, tetrahedra and octahedra to form our mesh as we go. These are the only types of element we

will have; we can also consider the octahedron as two pyramids, which reduces to only having pyramids and tetrahedra. Thus, we can see that our algorithm for structured meshes is the same as the one by Lowrie *et al.* in [38]. Let us explain in detail how it is implemented.

Our initial spatial mesh is a grid as depicted in Fig. 5.12. We are going to refer to the faces where the solution is known as *inflow* faces and the faces where we need to compute the solution as *outflow* faces, in the same manner as in the 1D+time case. So here, the space grid cells are inflow faces. In this algorithm, we choose the center of each cell to be our pitch points, which means that the center of each quadrilateral will be extruded. By doing so, we obtain pyramids, depicted in dark gray in Fig. 5.13. The height at which we are going to extrude our points depends on the wave propagation speed. Actually, the faces of the pyramids need to respect a certain slope: $\pm\frac{1}{c}$, *i.e.* each face has to respect the following: $c|\mathbf{n}| \geq n_t$. This is the cone constraint, which ensures causality of the mesh. Since we are working with 2D facets, it is more accurate to speak of angles than slope, as is done by Üngör *et al.* in [35].

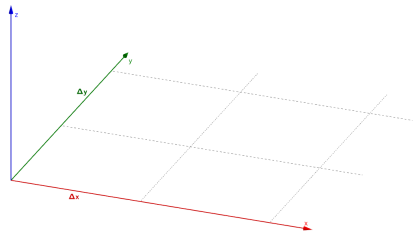


Figure 5.12: Initial 2D space grid

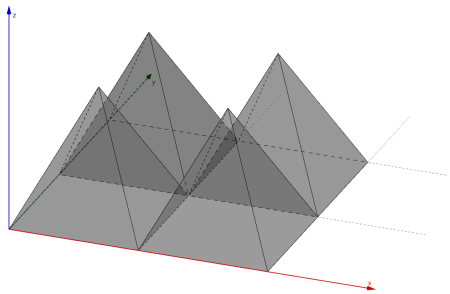


Figure 5.13: First layer of pyramids

After constructing the causal pyramid, we can compute the solution inside. To do so, we only need to know the solution on the inflow face, hence on the corresponding grid cell.

Remark. *Notice that we can treat each pyramid simultaneously, i.e., construct the element and solve the problem in it at the same time as it is done for another element, because each of them only depends on its respective initial data. We can see here why the Tent-Pitcher algorithm is said to be very conducive to parallel computing.*

Once we know the solution in the pyramid, we can interpolate it to obtain the solution on the outflow faces. This will serve as initial solution for the next elements. The next two steps consist in joining the pyramids' summits to form two kinds of tetrahedra: *vertical* tetrahedra (depicted in green in Fig. 5.17) and *horizontal* tetrahedra (depicted in blue in Fig. 5.15). Each tetrahedron will use one face from each two neighbor pyramids

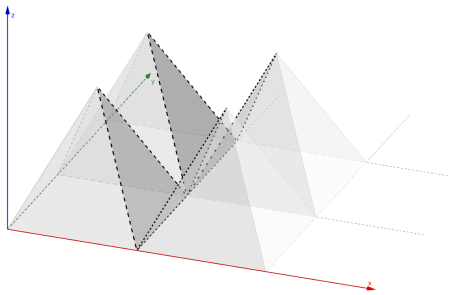


Figure 5.14: Inflow faces of horizontal tetrahedra

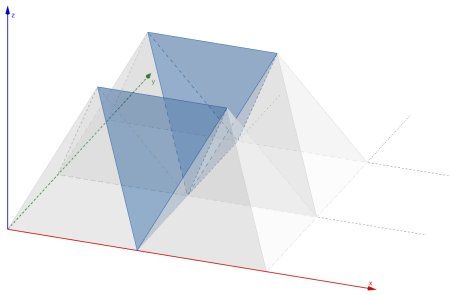


Figure 5.15: Second layer of horizontal tetrahedra

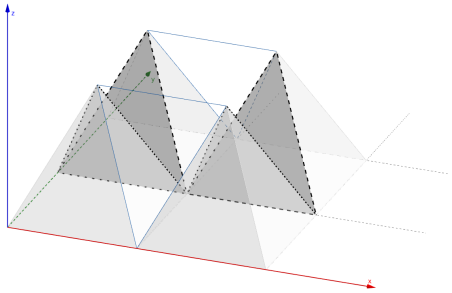


Figure 5.16: Inflow faces of vertical tetrahedra

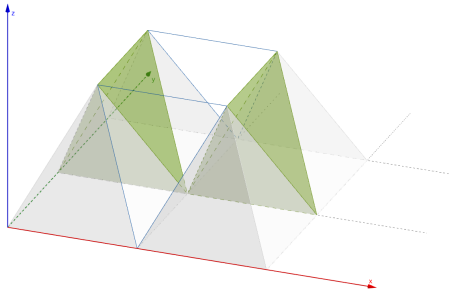


Figure 5.17: Second layer of vertical tetrahedra

as base and the solution on those faces will be the new initial solution. In other words, the pyramids' outflow faces become the tetrahedra's inflow faces, which is illustrated in Figs. 5.14 and 5.16. Two additional faces are constructed to form the tetrahedra. Since they are made by joining two summits of the previous pyramids, we can see that they have the same slope, hence respecting the causality. Once we have the solutions inside the blue and green tetrahedra, we can proceed in the same manner as above and interpolate them on the outflow faces. Using them as initial solution, we can solve the problem inside the red octahedron. As before, the outflow faces of the tetrahedra become inflow faces for the octahedra, as can be seen in Fig. 5.18 and the height of the octahedron is chosen as to respect the causality constraint.

We proceed in the same way on the whole domain, and at this point, we end up with several octahedra side by side. Actually, we are in the same setup as the initial pyramid layer, only shifted by half a cell in both directions. This is the reason why we said that the octahedra could also be considered as two pyramids; the first one would be upside down and would give us a flat layer which is the initial grid shifted by half a cell in both directions. So, we can go through the same steps again and join the octahedra summits

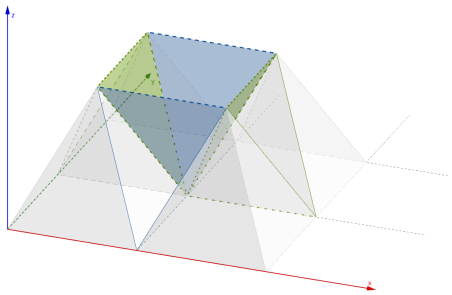


Figure 5.18: Inflow faces of the octahedra

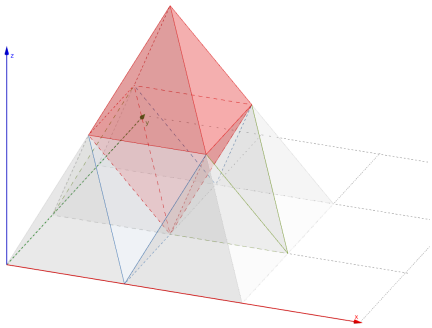


Figure 5.19: Third layer of octahedra

to form tetrahedra. We also get two kinds of tetrahedra (vertical and horizontal) but again, they are shifted by a half-cell in both directions. Then, we can form a new layer of octahedra, which will be placed in the same spatial position as the original pyramids. We can consider this as one layer that will be repeated exactly through time. Thus, our structured Tent-Pitching mesh is made by repeating these steps all over again, on the whole domain, until we reach a desired time T.

Concerning the boundaries of the space-time domain, we will have to slightly modify these elements by splitting them so they will fit the domain, as depicted in Fig. 5.20.

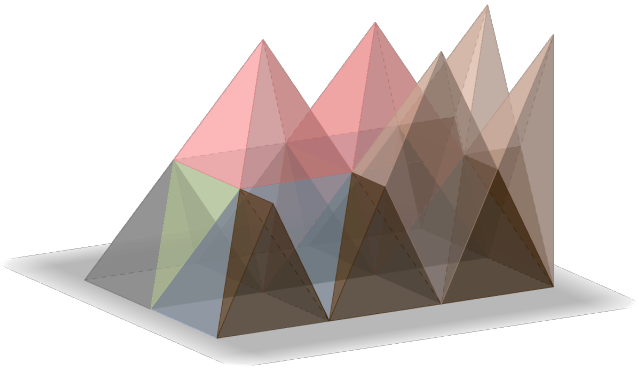


Figure 5.20: Splitted elements

5.4.2 Unstructured mesh

Working on structured meshes is ideal to gain time and to test a prototype without having to worry about mesh-induced bugs. But in real-life problems, unstructured meshes are best-suited. Indeed, it is easier to fit complex topologies and capture small details when using unstructured meshes. That is why we developed a Tent-Pitching framework for unstructured meshes. This work has also been done in [26] and [27] for the first-order acoustic wave equation. It also has been applied to other types of hyperbolic systems, as

mentioned above [44, 46]. However, the process is not the same as the previous one and so, we will now present the second algorithm. In the unstructured case, the initial space mesh is a triangular mesh. The main change is the choice of the point we will advance in time. In the previous algorithm, we chose center of cells as pitch points. Here, we will work more similarly to the algorithms presented in 5.2; we will choose a vertex of the mesh as the point that will be advanced in time.

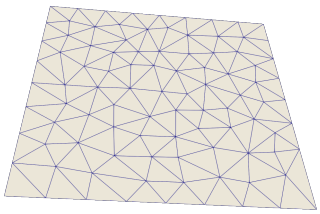


Figure 5.21: Initial space mesh at $t = t_0$

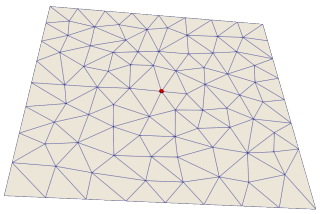


Figure 5.22: Choice of pitch point

In Fig. 5.21, we see the initial space mesh and the difference with the previous case is very clear. Here we have triangles as explained before. Then, we will choose a point to advance in time, depicted in red in Fig. 5.22.

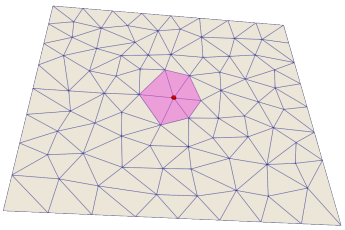


Figure 5.23: Inflow faces of the tent

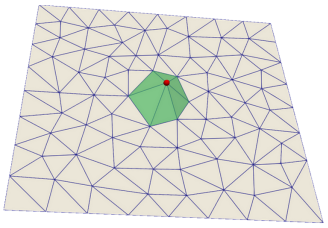


Figure 5.24: Outflow faces of the tent

Once the point is chosen, we are going to extrude it. We will create a new set of facets, that are obtained by connecting the extruded point to all the points connected to it, *i.e.*, all the points in its `star`, defined page 55 (see Fig. 5.24). Now that the tent is constructed, we can compute the solution inside. The purple faces in Fig. 5.23 are the inflow faces of the tent. The green faces in Fig. 5.24 are the outflow faces. We proceed in the same manner for the whole space domain. To choose the pitch-point, we use the lowest node technique. This means that we will choose the point which is at a minimum in time and visually speaking, it is the lowest (or one of the lowest) point of the mesh front. When having several points that fill this criterion, we add another constraint, which states that the sum of the time components of `star`(node) has to be minimal as well. In fact, if we do not impose for `star`(node) to respect this last constraint, we obtain meshes resembling the one in Fig. 5.27. As we can see, the first node to be pitched is the point $A(x_A, y_A, t_A)$ depicted in green on the figure, and then the

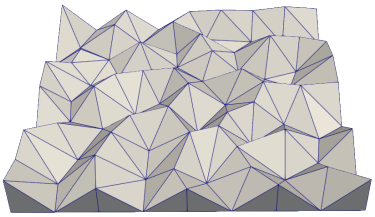


Figure 5.25: Construction of tents until space domain is covered

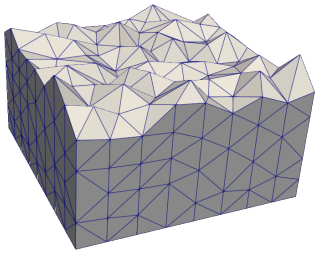


Figure 5.26: Several layers later

purple point $B(x_B, y_B, t_B)$ is selected to be pitched, since it is a lowest node. However, it violates the constraint on its `star`. Indeed, one of the points of `star`(B) is A, *i.e.*, $A \in$ `star`(B), hence the sum of the time components of star(B) is not a minimum, and is actually equal to t_A. This results in the solution of the second tent depending on the solution of the first tent, whereas we could have constructed another independent tent. Finally, we keep constructing tents and solving the problem until reaching or exceeding T and obtain a mesh such as the one depicted in Fig. 5.26.

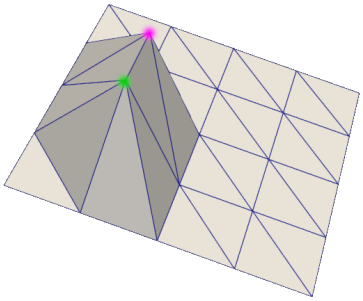


Figure 5.27: Choice of node to pitch with respect to `star`(node)

Now that we have presented the meshing process, we can derive a new variational formulation adapted to the Tent-Pitcher algorithm and its elementwise computation.

5.5 Variational Formulation

To express the variational formulation for the Trefftz-DG method enriched with the Tent-Pitcher algorithm, we will proceed in the same manner as in Chapter Trefftz-DG Method for Wave equations. We multiply the system (4.3) by appropriate test functions and then integrate the equations by parts, which results in equations (4.6). Then, we need to determine the numerical fluxes and to do so, we will define again a set of boundaries, adapted to the Tent-Pitching framework, on which the different traces will exist.

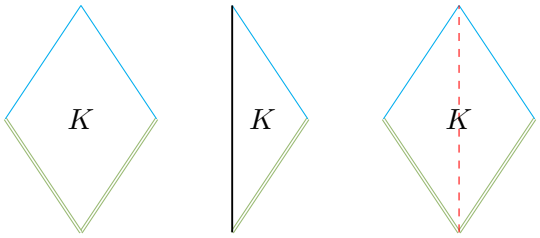


Figure 5.28: Classification of boundaries in Tent-Pitching

—— $\mathcal{F}^{out}$ represents the outflow faces,

$\mathcal{F}^{in}$ represents the inflow faces,

—— $\mathcal{F}^{ext}$ represents the domain boundary faces,

- - - - $\mathcal{F}^{int}$ represents the internal boundary faces for heterogeneous domains.

As previously, let us now define the numerical fluxes as follows:

$$\begin{pmatrix} v \\ p \end{pmatrix} = \begin{pmatrix} \{\!\{v\}\!\} + \beta_1 [\![p]\!]_x \\ \{\!\{p\}\!\} + \alpha_1 [\![v]\!]_x \end{pmatrix} \qquad \text{on } \mathcal{F}^{int},$$

$$\begin{pmatrix} v \\ p \end{pmatrix} = \begin{pmatrix} v \\ p \end{pmatrix} \qquad \text{on } \mathcal{F}^{out},$$

$$\begin{pmatrix} v \\ p \end{pmatrix} = \begin{pmatrix} (0.5 - \alpha_2)v + (0.5 + \alpha_2)v_0 \\ (0.5 - \beta_2)p + (0.5 + \beta_2)p_0 \end{pmatrix} \qquad \text{on } \mathcal{F}^{in},$$

$$\begin{pmatrix} v \cdot n_x \\ p \end{pmatrix} = \begin{pmatrix} g \\ p + \alpha_1(v \cdot n_x - g) \end{pmatrix} \qquad \text{on } \mathcal{F}^{ext},$$

Hence, using these flux definitions and taking our basis and test functions in Trefftz spaces, we obtain the following Variational Formulation:

Trefftz-DG Tent-Pitching Variational Formulation

Seek $(\mathbf{v}, p) \in \mathbf{T}(T_h)$*, such that for all* $(\mathbf{w}, q) \in \mathbf{T}(T_h)$*, it holds true:*

$$\mathcal{A}_{tp}(\mathbf{v}, p; \mathbf{w}, q) = l_{tp}(\mathbf{w}, q)$$

where

$$\begin{aligned}
\mathcal{A}_{tp}(\mathbf{v}, p; \mathbf{w}, q) := & \int_{\mathcal{F}^{out}} (\frac{1}{c^2\rho} pq + \rho \mathbf{v} \cdot \mathbf{w}) n_t + \mathbf{v}q \cdot \mathbf{n} + p\mathbf{w} \cdot \mathbf{n} \, \mathrm{ds} \\
& + \int_{\mathcal{F}^{in}} (0.5 - \alpha)\rho \mathbf{v} \cdot \mathbf{w} + (0.5 - \beta)\frac{1}{c^2\rho} pq \, \mathrm{ds} \\
& + 0.5 \int_{\mathcal{F}^{in}} \mathbf{v}q \cdot \mathbf{n} + p\mathbf{w} \cdot \mathbf{n} \, \mathrm{ds} \\
& + \int_{\mathcal{F}^{ext}} p\mathbf{w} \cdot \mathbf{n} + \alpha(\mathbf{v} \cdot \mathbf{n}_x)(\mathbf{w} \cdot \mathbf{n}_x) \, \mathrm{ds} \\
& + \int_{\mathcal{F}^{int}} \{\!\{\mathbf{v}\}\!\} \cdot [\![q]\!]_x + \beta_1 [\![p]\!]_x \cdot [\![q]\!]_x + \{\!\{p\}\!\} [\![\mathbf{w}]\!]_x + \alpha_1 [\![\mathbf{v}]\!]_x [\![\mathbf{w}]\!]_x \, \mathrm{ds} \\
l_{tp}(\mathbf{w}, q) := & \int_{\mathcal{F}^{in}} - (0.5 + \alpha) \left(\rho \mathbf{v}^{in} \cdot \mathbf{w}\right) n_t - (0.5 + \beta) \left(\frac{1}{c^2\rho} p^{in} q\right) n_t \, \mathrm{ds} \\
& - 0.5 \int_{\mathcal{F}^{in}} p^{in}\mathbf{w} \cdot \mathbf{n} + \mathbf{v}^{in} q \cdot \mathbf{n} \, \mathrm{ds}
\end{aligned} \tag{5.1}$$

We can see that the variational formulation enriched with Tent-Pitching is different from the Trefftz-DG formulation. In fact, we could think that the Tent-Pitching formulation would reduce to writing the Trefftz-DG variational formulation for one cell, but it is not the case. We actually need to adapt the fluxes, since here, the outflow faces are the equivalent of the previous final-time faces but are not parallel to the space axis, in the same way as the inflow faces are the equivalent of the previous initial time faces. Hence, the fluxes need to be adjusted to a Tent-Pitching cell.

5.6 Properties of the scheme

The well-posedness of this Trefftz-DG Tent-Pitching scheme has been established by Shishenina in [24] and by Moiola and Perugia in [68]. We recall here the main results.

In the same manner as in Chapter Trefftz-DG Method for Wave equations, to establish uniqueness and existence of a solution of a weak variational form, we use the previously introduced Lax-Milgram theorem. Hence, to establish the well-posedness of our variational formulation, we need to show that our bilinear form $\mathcal{A}(u, v)$ is continuous

and coercive. To do so, let us first define the following quantities:

$$
\begin{aligned}
|||\mathbf{w},q|||_{tp} = \Big[&\frac{1-\theta}{2}\left\|(\frac{1}{c^2\rho})^{1/2}q(n_t)^{1/2}\right\|^2_{L^2(\mathcal{F}^{out})} + \frac{1-\theta}{2}\left\|(\rho)^{1/2}\mathbf{w}(n_t)^{1/2}\right\|^2_{L^2(\mathcal{F}^{out})} \\
&+ \frac{\theta}{2}\left\|(\frac{1}{c^2\rho})^{1/2}q(n_t)^{1/2} + (\rho)^{1/2}\mathbf{w}(n_t)^{1/2}\right\|^2_{L^2(\mathcal{F}^{out})} \\
&+ \left\|\beta_2^{1/2}(\frac{1}{c^2\rho})^{1/2}q|n_t|^{1/2}\right\|^2_{L^2(\mathcal{F}^{in})} + \left\|\alpha_2^{1/2}(\rho)^{1/2}w_x|n_t|^{1/2}\right\|^2_{L^2(\mathcal{F}^{in})} \\
&+ \left\|\alpha_1^{1/2}(\mathbf{w}\cdot\mathbf{n})\right\|^2_{L^2(\mathcal{F}^{ext})} + \left\|\alpha_1^{1/2}[\![\mathbf{w}]\!]_x\right\|^2_{L^2(\mathcal{F}^{int})} + \left\|\beta_1^{1/2}[\![q]\!]_x\right\|^2_{L^2(\mathcal{F}^{int})}\Big]^{1/2}
\end{aligned}
$$

$$
\begin{aligned}
|||\mathbf{w},q|||_{tp^+} = \Big[&|||\mathbf{w},q|||_{tp} \\
&+ \frac{1-\theta}{2}\left\|(\frac{1}{c^2\rho})^{1/2}q(n_t)^{1/2}\right\|^2_{L^2(\mathcal{F}^{out})} + \frac{1-\theta}{2}\left\|(\rho)^{1/2}\mathbf{w}(n_t)^{1/2}\right\|^2_{L^2(\mathcal{F}^{out})} \\
&+ \left\|(\frac{1}{2\beta_2}+1)^{1/2}(\frac{1}{c^2\rho})^{1/2}q\right\|^2_{L^2(\mathcal{F}^{in})} + \left\|(\frac{1}{2\alpha_2}+1)^{1/2}(\rho)^{1/2}\mathbf{w}\right\|^2_{L^2(\mathcal{F}^{in})} \\
&+ \left\|\alpha_1^{-1/2}q\right\|^2_{L^2(\mathcal{F}^{ext})} + \left\|\beta_1^{-1/2}\{\!\{\mathbf{w}\}\!\}\right\|^2_{L^2(\mathcal{F}^{int})} + \left\|\alpha_1^{-1/2}\{\!\{q\}\!\}\right\|^2_{L^2(\mathcal{F}^{int})}\Big]^{1/2}
\end{aligned}
$$

The function θ is a piecewise-constant function, such that $\theta \equiv \frac{c|\mathbf{n}|}{n_t}$. We can see that $\theta \in [0,1)$. The semi-norms $|||\mathbf{w},q|||_{tp}$ and $|||\mathbf{w},q|||_{tp^+}$ define norms on $\mathbf{T}(\mathcal{T}_h)$. We then have the following:

$$
\begin{aligned}
\mathcal{A}_{tp}(\mathbf{w},q;\mathbf{w},q) = &\left\|\alpha_1^{1/2}[\![\mathbf{w}]\!]_x\right\|^2_{L^2(\mathcal{F}^{int})} + \left\|\beta_1^{1/2}[\![q]\!]_x\right\|^2_{L^2(\mathcal{F}^{int})} + \left\|\alpha_1^{1/2}(\mathbf{w}\cdot\mathbf{n})\right\|^2_{L^2(\mathcal{F}^{ext})} \\
&+ \left\|\beta_2^{1/2}(\frac{1}{c^2\rho})^{1/2}q|n_t|^{1/2}\right\|^2_{L^2(\mathcal{F}^{in})} + \left\|\alpha_2^{1/2}(\rho)^{1/2}w_x|n_t|^{1/2}\right\|^2_{L^2(\mathcal{F}^{in})} \\
&+ \int_{\mathcal{F}^{out}} \frac{1}{2n_t}\Big[\frac{1}{c^2\rho}(qn_t)^2 + \rho(w_xn_t)\cdot(w_xn_t) + 2(qn_t)(w_x\cdot\mathbf{n})\Big]ds
\end{aligned}
$$

Thus, the following result holds true:

Coercivity

Theorem 5.1. The following holds true $\forall(\mathbf{w},q) \in \mathbf{T}(T_h)$:

$$\mathcal{A}_{tp}(\mathbf{w},q;\mathbf{w},q) \geq |||\mathbf{w},q|||^2_{tp}$$

Thus, $\mathcal{A}_{tp}$ is a coercive bilinear form.

To prove continuity, we sum the norms of all terms in the bilinear and linear form and apply the Cauchy-Schwartz inequality, which leads to the following:

Continuity

Theorem 5.2. The following holds true $\forall(\mathbf{v},p),(\mathbf{w},q)\in\mathbf{T}(T_h)$:

$$\begin{aligned}\mathcal{A}_{tp}(\mathbf{v},p;\mathbf{w},q)\leq& C_1^{tp}|||\mathbf{v},p|||_{tp}|||\mathbf{w},q|||_{tp}\,,\\ l_{tp}(\mathbf{w},q)\leq&\Bigg[\left\|(\frac{1}{2\beta_2}+1)^{1/2}(\frac{1}{c^2\rho})^{1/2}q\right\|^2_{L^2(\mathcal{F}^{in})}\\ &+\left\|(\frac{1}{2\alpha_2}+1)^{1/2}(\rho)^{1/2}\mathbf{w}\right\|^2_{L^2(\mathcal{F}^{in})}\Bigg]^{1/2}|||\mathbf{w},q|||_{tp}\end{aligned}$$

Thus, $\mathcal{A}_{tp}$ is a continuous bilinear form and l_{tp} is a continuous linear form.

Hence, since all conditions of the Lax-Milgram theorem are met, the variational formulation has a unique solution.

5.7 Variational formulation C

In Part II of the thesis, we show that we struggle to obtain good results with the previous variational formulation, which encouraged us to look for another one. We present here one of the variational formulations we tried out and that presented correct results, which has the advantage to only be posed on the inflow boundaries of a tent, leading to a reduced computational cost. Thus, it is an interesting formulation even in the framework of the current Part I of the thesis.

To derive another variational formulation, let us go back to the variational formulation (5.1), but without taking our test functions in a Trefftz space but in a regular space $\mathbf{V}\times V$, which means that the volumic terms remain, as follows:

Seek $(\mathbf{v},p)\in\mathbf{T}$*, such that for all* $(\mathbf{w},q)\in\mathbf{V}\times V$*, it holds true:*

$$\mathcal{A}(\mathbf{v},p;\mathbf{w},q)=l(\mathbf{w},q)$$

where

$$\begin{aligned}\mathcal{A}(\mathbf{v},p;\mathbf{w},q):=&-\sum_{K\in T_h}\int_K p\Big(\frac{1}{c^2\rho}\frac{\partial q}{\partial t}+\operatorname{div}\mathbf{w}\Big)+\mathbf{v}\cdot\Big(\rho\frac{\partial\mathbf{w}}{\partial t}+\nabla q\Big)\\ &+\int_{\mathcal{F}^{out}}(\frac{1}{c^2\rho}pq+\rho\mathbf{v}\cdot\mathbf{w})n_t+\mathbf{v}q\cdot\mathbf{n}+p\mathbf{w}\cdot\mathbf{n}+\int_{\mathcal{F}^{ext}}p\mathbf{w}\cdot\mathbf{n}+\alpha(\mathbf{v}\cdot\mathbf{n}_x)(\mathbf{w}\cdot\mathbf{n}_x)\\ &+\int_{\mathcal{F}^{in}}(0.5-\alpha)\rho\mathbf{v}\cdot\mathbf{w}+(0.5-\beta)\frac{1}{c^2\rho}pq+0.5\int_{\mathcal{F}^{in}}\mathbf{v}q\cdot\mathbf{n}+p\mathbf{w}\cdot\mathbf{n}\\ &+\int_{\mathcal{F}^{int}}\{\!\{\mathbf{v}\}\!\}\cdot[\![q]\!]_x+\beta_1[\![p]\!]_x\cdot[\![q]\!]_x+\{\!\{p\}\!\}[\![\mathbf{w}]\!]_x+\alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x\end{aligned}$$

$$l(\mathbf{w},q) := \int_{\mathcal{F}^{in}} -(0.5+\alpha)\left(\rho\mathbf{v}^{in}\cdot\mathbf{w}\right)n_t - (0.5+\beta)\left(\frac{1}{c^2\rho}p^{in}q\right)n_t$$
$$-0.5\int_{\mathcal{F}^{in}} p^{in}\mathbf{w}\cdot\mathbf{n} + \mathbf{v}^{in}q\cdot\mathbf{n}$$

Now, let us integrate the volumic term by parts again, which means that we obtain a vanishing volumic term and a new boundary integral. Indeed, when integrating by parts once more, since $(\mathbf{v},p)$ is taken in a Trefftz space, *i.e.*, is solution to the equations of interest, the volumic integral vanishes. This results in:

$$\begin{aligned}\mathcal{A}(\mathbf{v},p;\mathbf{w},q) := -\sum_{\boldsymbol{K}\in\boldsymbol{\mathcal{T}_h}}\int_{\boldsymbol{\partial K}}&\frac{1}{c^2\rho}pqn_t + \mathbf{v}q\cdot\mathbf{n} + \rho\mathbf{v}\cdot\mathbf{w}n_t + p\mathbf{w}\cdot\mathbf{n}\\
&+\int_{\mathcal{F}^{out}}(\frac{1}{c^2\rho}pq + \rho\mathbf{v}\cdot\mathbf{w})n_t + \mathbf{v}q\cdot\mathbf{n} + p\mathbf{w}\cdot\mathbf{n}\\
&+\int_{\mathcal{F}^{in}}(0.5-\alpha)\rho\mathbf{v}\cdot\mathbf{w} + (0.5-\beta)\frac{1}{c^2\rho}pq\\
&+0.5\int_{\mathcal{F}^{in}}\mathbf{v}q\cdot\mathbf{n} + p\mathbf{w}\cdot\mathbf{n}\\
&+\int_{\mathcal{F}^{ext}}p\mathbf{w}\cdot\mathbf{n} + \alpha(\mathbf{v}\cdot\mathbf{n}_x)(\mathbf{w}\cdot\mathbf{n}_x)\\
&+\int_{\mathcal{F}^{int}}\{\!\{\mathbf{v}\}\!\}\cdot[\![q]\!]_x + \beta_1[\![p]\!]_x\cdot[\![q]\!]_x + \{\!\{p\}\!\}[\![\mathbf{w}]\!]_x + \alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x\end{aligned} \tag{5.2}$$

Now, we decompose the obtained boundary integral (depicted in bold in (5.2)) into several kinds of frontiers: outflow boundaries, inflow boundaries, internal faces and external boundaries. We inject the expressions of the fluxes for the outflow frontiers only, which gives the following:

$$\begin{aligned}\mathcal{A}(\mathbf{v},p;\mathbf{w},q) := &-\int_{\mathcal{F}^{in}}(0.5+\alpha)\rho\mathbf{v}\cdot\mathbf{w}n_t + (0.5+\beta)\frac{1}{c^2\rho}pqn_t\\
&-0.5\int_{\mathcal{F}^{in}}\mathbf{v}q\cdot\mathbf{n} + p\mathbf{w}\cdot\mathbf{n}\\
&+\int_{\mathcal{F}^{ext}}\alpha(\mathbf{v}\cdot\mathbf{n}_x)(\mathbf{w}\cdot\mathbf{n}_x)\\
&+\int_{\mathcal{F}^{int}}[\![\mathbf{v}]\!]_x\{\!\{q\}\!\} + \beta_1[\![p]\!]_x\cdot[\![q]\!]_x + [\![p]\!]_x\cdot\{\!\{\mathbf{w}\}\!\} + \alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x\end{aligned}$$

We can see that the outflow integrals cancel each other out, and we finally obtain the following variational formulation, that we denote *Variational Formulation C*:

Trefftz-DG Variational Formulation C with Tent-Pitching

Seek $(\mathbf{v}, p) \in \mathbf{T}$*, such that for all* $(\mathbf{w}, q) \in \mathbf{V} \times V$*, it holds true:*

$$\mathcal{A}(\mathbf{v}, p; \mathbf{w}, q) = l(\mathbf{w}, q)$$

where

$$\begin{aligned}
\mathcal{A}(\mathbf{v}, p; \mathbf{w}, q) := & \int_{\mathcal{F}^{in}} (0.5+\alpha)\rho\mathbf{v}\cdot\mathbf{w} + (0.5+\beta)\frac{1}{c^2\rho}pq \\
& + 0.5\int_{\mathcal{F}^{in}} \mathbf{v}q\cdot\mathbf{n} + p\mathbf{w}\cdot\mathbf{n} \\
& + \int_{\mathcal{F}^{ext}} \alpha(\mathbf{v}\cdot\mathbf{n}_x)(\mathbf{w}\cdot\mathbf{n}_x) \\
& + \int_{\mathcal{F}^{int}} [\![\mathbf{v}]\!]_x \{\!\{q\}\!\} + \beta_1 [\![p]\!]_x \cdot [\![q]\!]_x + [\![p]\!]_x \cdot \{\!\{\mathbf{w}\}\!\} + \alpha_1 [\![\mathbf{v}]\!]_x [\![\mathbf{w}]\!]_x
\end{aligned}$$

$$\begin{aligned}
l(\mathbf{w}, q) := & \int_{\mathcal{F}^{in}} (0.5+\alpha)\left(\rho\mathbf{v}^{in}\cdot\mathbf{w}\right) n_t + (0.5+\beta)\left(\frac{1}{c^2\rho}p^{in}q\right) n_t \\
& + 0.5\int_{\mathcal{F}^{in}} p^{in}\mathbf{w}\cdot\mathbf{n} + \mathbf{v}^{in}q\cdot\mathbf{n}
\end{aligned}$$

Notice that the resulting formulation is only posed on the inflow boundaries and that only the basis functions are taken in the Trefftz space $\mathbf{T}$. This means that this variational formulation is valid for any kind of test function in $\mathbf{V} \times V$.

Now that we have introduced our different variational formulations, the next chapter will be devoted to explaining their implementation in details and present numerical results. The implementation consists in several tasks, following the different cases presented in the present and previous chapters: Trefftz-DG method with Tent-Pitching on structured meshes, then on unstructured meshes and in a HPC-environment afterwards.

Chapter 6

Implementation

In this chapter, we explain in details how the implementation has been done, in order for our algorithms to be fully reproducible. We accomplished several tasks and their ultimate goal was to improve the Trefftz Discontinuous Galerkin Tent-Pitching 2D+time solver. This latter has been implemented by Elvira Shishenina ([24]) for the acoustic, elastic and elastoacoustic wave equations in Matlab/Octave for structured meshes. It has also been implemented by Paul Stocker [27] for the acoustic wave equation on unstructured meshes with Thread-Based Parallelism, which is performed on machines with shared memory. Since this thesis falls within the context of the DIP project (presented in the General Introduction), the foreseen applications were geophysical applications for industrial cases. Hence, the considered domains are very large, thus shared memory parallelism is not adapted and it is better to use distributed memory parallelism.

We were provided with the aforementioned Matlab/Octave code and used it as a base to carry out our tasks, which we will specify now.

Task 1: Extension to Fortran

As explained, we work on a Matlab/Octave code that contains a Trefftz-DG solver with Tent-Pitching on structured meshes. Although Matlab/Octave is a good choice for quick prototyping, the execution is generally slower and less efficient in a parallel environment compared to Fortran. In fact, Fortran is HPC-friendly and allows us to integrate many tools (METIS, Lapack, MPI...) more easily. Hence, our first task consists in porting the code to Fortran, having in mind to illustrate the potential of the method on industrial cases.

Task 2: Extension to unstructured meshes

The second task consists in extending the solver to unstructured meshes. Working on a structured mesh has many advantages, such as avoiding mesh-induced errors and having a faster execution, but it is not adequate for complex/non-square geometries, as it is the case in geophysics. Hence, we adapt our methods to triangular meshes, in

particular we need to design the Tent-Pitcher algorithm accordingly. To do so, we will mostly base ourselves on the algorithm introduced by Erickson *et al.* [36].

Task 3: Parallelism with MPI for structured case

In seismic applications, like the ones presented in the General Introduction, we often deal with very large domains and depending on the level of details we want, the computational time can become very large. This explains the need for the third task, which consists in parallelizing the code using MPI. In order to have a prototype, we parallelized the solver with structured meshes first.

Task 4: Parallelism with MPI for unstructured case

This task is similar to the previous one. It also consists in parallelizing the solver using MPI, but on unstructured meshes. In practice, the task is not carried out in the same way whether working on structured or unstructured meshes and ends up being easier on structured meshes. Which is why this consists of a task in itself and deserves an explanation as well.

Tests & Comparison

This last section serves the purpose of testing all these developments. We will test the accuracy, convergence, scalability of the different codes and then compare them to an IPDG solver in time domain.

6.1 Task 1: Extension to Fortran

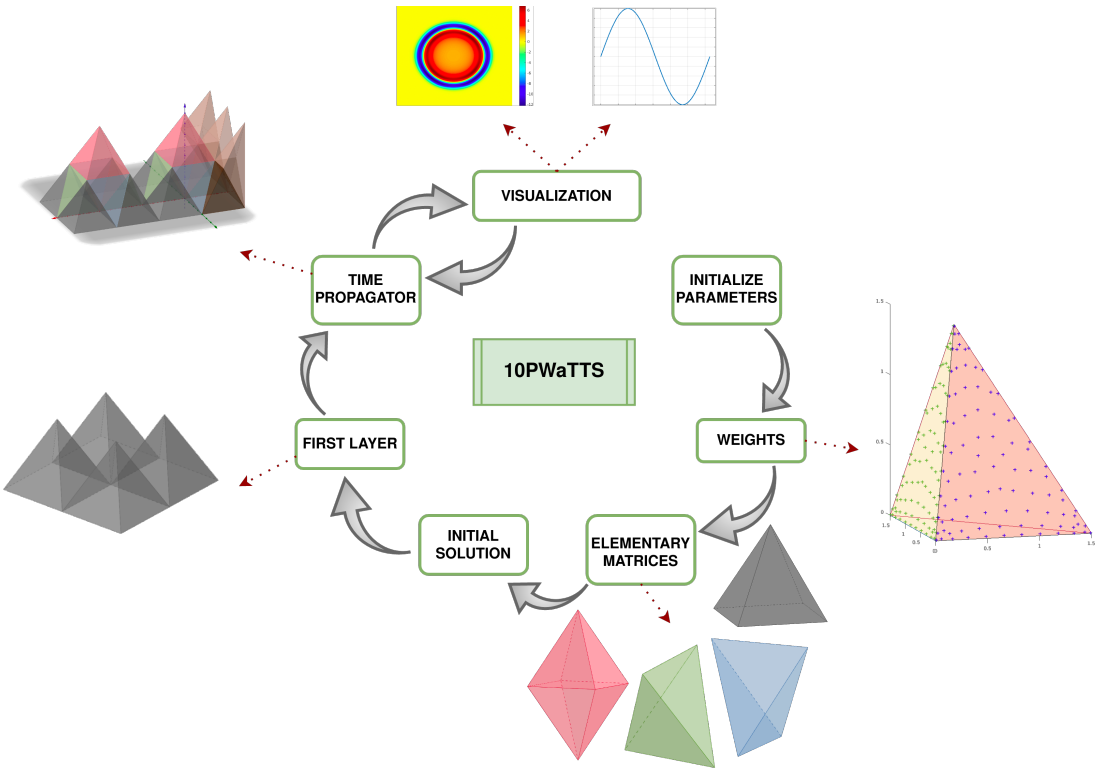


Figure 6.1: Flowchart

Our first task is to implement the already existing Trefftz-DG Tent-Pitching solver in Fortran. To do so, we created a new Git repository named `10PWaTTS` and constructed our code following the flowchart 6.1. In order to build the program, we used `CMake` to automate the compilation and linking process, which facilitates the use of external libraries, such as MPI, METIS, Lapack, etc. This makes the compilation process independent from one's computer setup, hence allowing to share a code more easily. An example of `CMake` file can be found in Appendix C.

Each cell of the flowchart corresponds to a `Fortran SUBROUTINE` or `MODULE`, and we can see the order in which each of them is called. In the following, each task will have its own flowchart and we will use the flowchart 6.1 as a basis, to have an overview of the major differences that occur between each version of the solver.

6.1.1 Parameters & weights

This first step comprises the initialization of all necessary parameters such as the:

- domain sizes L_x, L_y and final time L_t,

- number of cells in each direction, n_x, n_y and thus, the space steps Δx, Δy,
- time step Δt, which we obtain through the Tent-Pitcher algorithm,
- order of approximation of basis functions, which are polynomials here,
- physical parameters, such as the wave propagation speed c and the density ρ,
- boundary conditions, which can be homogeneous Neumann conditions or first order absorbing boundary conditions,
- penalty parameters α and β.

As we have explained in the previous chapters, we need to introduce a weak form in order to solve our problem numerically, which involves integrals. Hence, we need a way to perform those integration numerically and we will use the Gauss quadrature to do so. The idea here is to make a judicious choice of integration points and weights, such that we can approximate I as follows:

$$I = \int_a^b f(\mathbf{x})dx = \sum_{i=1}^{n} w_i f(\mathbf{x}_i) + E \approx \sum_{i=1}^{n} w_i f(\mathbf{x}_i).$$

The points x_i and the weights w_i are chosen in order to minimize the error E and thus, maximize the accuracy. These integration points and weights are well-known, so we simply initialize them to be able to perform our various integrations. They are obtained by mapping the points from a reference triangle to an arbitrary one, thanks to a mapping function. The quadrature we use is of order 21 and the weights and points can be found in Appendix B and are obtained following the framework by Zhang in [76] [1].

In our case, we need the integration formula for 2D triangles. Indeed, we solve the 2D+time problem but since we use Trefftz functions, we only need to perform the computations on the boundary faces of our elements, which are triangles. Fig. 6.2 shows the distribution of integration points on the triangular boundary faces of a tetrahedron.

In order to use the 2D integration formula on a surface defined in spacetime, we need to use a mapping function $\mathcal{T}$ and more details are provided in the equation (6.3).

6.1.2 Elementary matrices

Equation (5.1) can be rewritten as the following system:

$$M \begin{pmatrix} p \\ \mathbf{v} \end{pmatrix} = K \begin{pmatrix} p^{in} \\ \mathbf{v}^{in} \end{pmatrix}$$

[1] Their source code, and in particular the quadrature implementation, can be found in `http://lsec.cc.ac.cn/phg/index_en.htm`

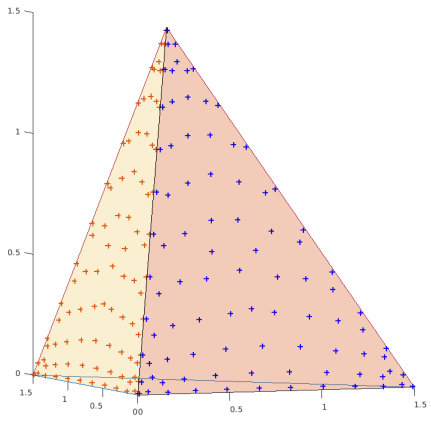


Figure 6.2: Gaussian quadrature points on a tetrahedron

Or equivalently as:

$$MU^n = KU^{n-1} \tag{6.1}$$

with

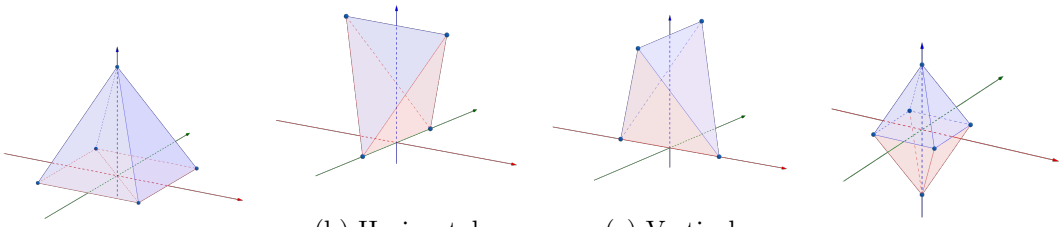
$$\begin{aligned} M_{ij} &= \int_{\mathcal{F}^{out}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \phi_j^{\mathbf{v}} \cdot \phi_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \phi_i^{\mathbf{w}} + \phi_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\ &\quad + \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \phi_j^{\mathbf{v}} \cdot \phi_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \phi_i^{\mathbf{w}} + \phi_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\ &= M^{out} + M^{in} \\ K_{ij} &= (1-\gamma) \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \phi_j^{\mathbf{v}} \cdot \phi_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \phi_i^{\mathbf{w}} + \phi_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\ M &= \begin{pmatrix} M_{11} & \dots & M_{1N} \\ \vdots & \ddots & \vdots \\ M_{N1} & \dots & M_{NN} \end{pmatrix} \qquad K = \begin{pmatrix} K_{11} & \dots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \dots & K_{NN} \end{pmatrix} \end{aligned} \tag{6.2}$$

where $N = \text{dof}$, is the number of degrees of freedom considered in the problem.

The vector U^n is the solution vector at time $t = n\Delta t$ and U^{n-1} corresponds to the solution at the previous time step. The functions ϕ are the basis functions associated with p, $\mathbf{v}$, q and $\mathbf{w}$ and as discussed before, we use polynomial basis functions in this case. The matrix M^{out} corresponds to the integral in M posed on $\mathcal{F}^{out}$, which are the outflow boundaries as explained in Chapter Tent-Pitcher Algorithm. Whereas M^{in} corresponds to the one posed on $\mathcal{F}^{in}$, which are the inflow boundaries.

The mesh we work with is a structured mesh composed of four types of elements: pyramids, vertical tetrahedra, horizontal tetrahedra and octahedra, which are the same as the ones presented in Section 5.4.1. They are all the same throughout the mesh, only their location varies. Hence, we will use these four elements as reference elements

and compute the elementary matrices for each of them. The reference elements are constructed such that $x \in [-\frac{\Delta x}{2}, \frac{\Delta x}{2}]$, $y \in [-\frac{\Delta y}{2}, \frac{\Delta y}{2}]$ and $t \in [-\frac{\Delta t}{2}, \frac{\Delta t}{2}]$ and are depicted in Fig. 6.3.



(a) Pyramidal reference element

(b) Horizontal tetrahedral reference element

(c) Vertical tetrahedral reference element

(d) Octahedral reference element

Figure 6.3: Reference elements

As discussed before, since we are working with Trefftz polynomials, we only need to perform the various computations on the skeleton of the mesh. Here, all the elements' boundaries are triangles, or can be decomposed into two triangles.

So, we will break down the computation of the elementary matrices into two steps:

1. Compute M and K on a triangle defined in 3D using a mapping function $\mathcal{T}$ to transform 2D coordinates to 3D coordinates: we will call them M^f and K^f,
2. Compute M^f and K^f for every face of each reference elements and then sum each matrix on each face of a reference element to obtain its corresponding elementary matrix. We will add the suffix $-p$, $-ht$, $-vt$ and $-o$ to the matrices corresponding to the pyramid, horizontal tetrahedron, vertical tetrahedron and octahedron respectively.

First step: The first step consists in defining a transformation $\mathcal{T}$ between a triangle $\mathcal{F}$ and its spatial projection $\mathcal{F}^e$, in order to be able to compute M^f and K^f in function of M^e and K^e respectively. This process is illustrated in Fig. 6.4.

Let $\mathcal{T}$ be a bijective linear function, such that:

$$(x, y, t) = \mathcal{T}(\xi, \eta, 0) \tag{6.3}$$

where $(x, y, t) \in \mathcal{F}$ and $(\xi, \eta, 0) \in \mathcal{F}^e$. In other words, $\mathcal{T}$ transforms the reference coordinates $(\xi, \eta, 0)$ into real coordinates (x, y, t). The basis functions in the reference element are such that:

$$\phi^e = \phi \circ \mathcal{T}(\xi, \eta, 0)$$

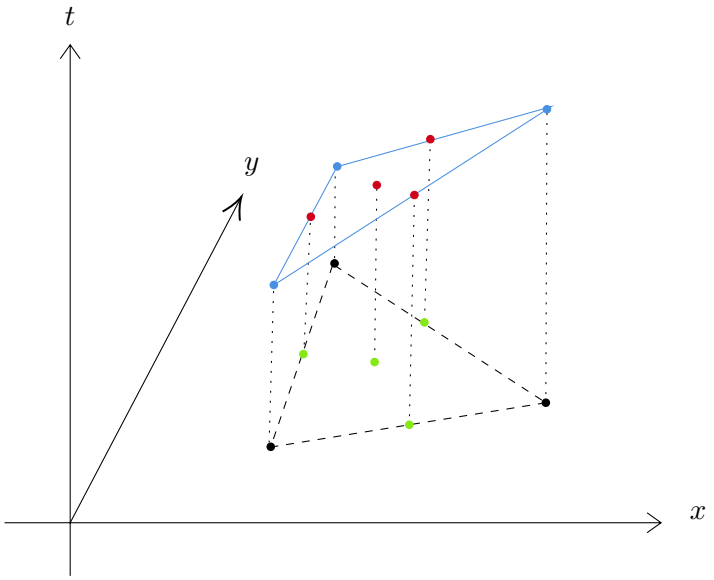


Figure 6.4: Transformation from a 3D triangle (depicted in blue) to a 2D triangle

where the set $\{\phi\}$ represents the set of basis on $\mathcal{F}$. From this, we can deduce that $\phi = \phi^e \circ \mathcal{F}^{-1}$, and knowing the formula for a change of variable in an integral, we obtain that:

$$\int \phi \, dxdydt = \int \phi^e \det(J_{\mathcal{F}}) d\xi d\eta$$

where $J_{\mathcal{F}}$ is the Jacobian matrix of the transform. Since we are working with triangles, $\det(J_{\mathcal{F}}) = 2|\mathcal{F}|$, where $|\mathcal{F}|$ is the area of the triangle. Hence,

$$M^f = 2|\mathcal{F}|M^e \quad K^f = 2|\mathcal{F}|K^e$$

where

$$\begin{aligned} M^e_{ij} &= \int_{\mathcal{F}^{out}} \left(\frac{1}{c^2\rho} \phi^{pe}_j \phi^{qe}_i + \rho \phi^{\mathbf{v}e}_j \cdot \phi^{\mathbf{w}e}_i \right) n_t + \left(\phi^{pe}_j \phi^{\mathbf{w}e}_i + \phi^{\mathbf{v}e}_j \phi^{qe}_i \right) \cdot \mathbf{n} \\ &+ \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi^{pe}_j \phi^{qe}_i + \rho \phi^{\mathbf{v}e}_j \cdot \phi^{\mathbf{w}e}_i \right) n_t + \left(\phi^{pe}_j \phi^{\mathbf{w}e}_i + \phi^{\mathbf{v}e}_j \phi^{qe}_i \right) \cdot \mathbf{n} \\ K^e_{ij} &= (1-\gamma) \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi^{pe}_j \phi^{qe}_i + \rho \phi^{\mathbf{v}e}_j \cdot \phi^{\mathbf{w}e}_i \right) n_t + \left(\phi^{pe}_j \phi^{\mathbf{w}e}_i + \phi^{\mathbf{v}e}_j \phi^{qe}_i \right) \cdot \mathbf{n} \end{aligned}$$

$$M^e = \begin{pmatrix} M^e_{11} & \dots & M^e_{1N} \\ \vdots & \ddots & \vdots \\ M^e_{N1} & \dots & M^e_{NN} \end{pmatrix} \qquad K^e = \begin{pmatrix} K^e_{11} & \dots & K^e_{1N} \\ \vdots & \ddots & \vdots \\ K^e_{N1} & \dots & K^e_{NN} \end{pmatrix}$$

and $N = \text{dof}$, is the number of degrees of freedom considered in the problem.

Let us notice that, since the mesh is structured and composed of identical elements overall the domain, all elements are made of the same set of triangles. Thus the surface of a given facet will be identical for every pyramids of the mesh, and likewise for the tetrahedra and octahedra.

Second step: We can now compute M^f and K^f on each face of the pyramid, tetrahedra and octahedron and sum them on their respective elements to obtain the elementary matrices.

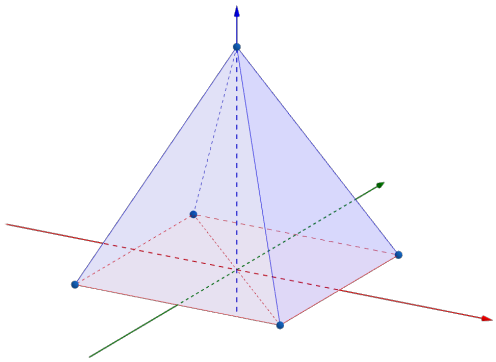


Figure 6.5: Reference pyramid

For the pyramid, we have four outflow triangles, which are depicted in blue in Fig. 6.5. The square base of the pyramid is cut into two triangles, depicted in red. Thus, M^p is obtained by summing $(M^f)^{out}$ on the four outflow faces and $(M^f)^{in}$ on the two base triangles. K^p is only defined on the inflow faces, as can be seen from expressions (6.2), hence we sum K^f on the inflow triangles only:

$$M^p = \sum_{i=1}^{2}(M_i^f)^{in} + \sum_{i=1}^{4}(M_i^f)^{out} \quad K^p = \sum_{i=1}^{2} K_i^f$$

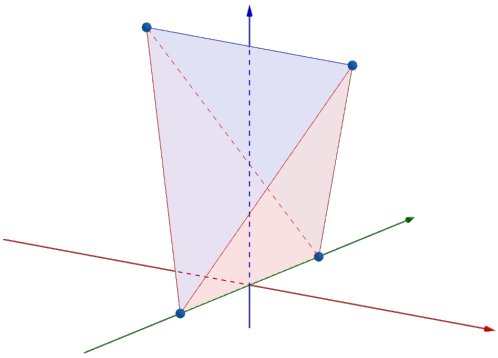


Figure 6.6: Reference horizontal tetrahedra

For both the horizontal and vertical tetrahedra, we have two outflow faces and two inflow faces, depicted in blue and red respectively on Fig. 6.6. We proceed in the same manner as for the pyramid in order to obtain M^{ht}, K^{ht}, M^{vt} and K^{vt}:

$$M^{ht} = \sum_{i=1}^{2}(M_i^f)^{in} + \sum_{i=1}^{2}(M_i^f)^{out} \quad K^{ht} = \sum_{i=1}^{2} K_i^f$$

$$M^{vt} = \sum_{i=1}^{2}(M_i^f)^{in} + \sum_{i=1}^{2}(M_i^f)^{out} \quad K^{vt} = \sum_{i=1}^{2} K_i^f$$

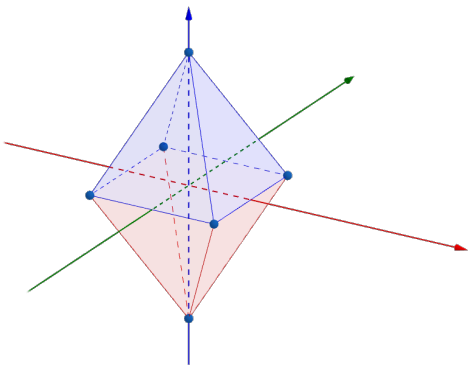


Figure 6.7: Reference octahedron

For the octahedron, we have four outflow faces and four inflow faces, depicted in blue and red respectively on Fig. 6.7. We proceed in the same manner as for the previous elements in order to obtain M^o and K^o:

$$M^o = \sum_{i=1}^{4}(M_i^f)^{in} + \sum_{i=1}^{4}(M_i^f)^{out} \quad K^o = \sum_{i=1}^{4} K_i^f$$

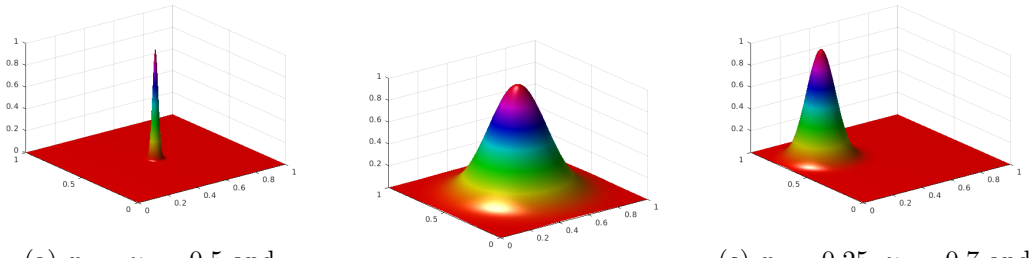
6.1.3 Initial solution and first layer

Now that we have our elementary matrices, we can start solving our actual problem. First of all, we need to compute the solution on the initial space mesh, which is a grid in our case. We use a Gaussian function as initial solution, defined as follows:

$$f(x,y) = exp\left(-2\pi^2 \frac{(x-x_0)^2 + (y-y_0)^2}{r_0^2}\right)$$

with (x_0, y_0) the centre of the source and r_0 its radius.

In Fig. 6.8, we can see a 2D representation of this function and how the center and the radius affect its shape.



(a) $x_0 = y_0 = 0.5$ and $r_0 = 0.1$

(b) $x_0 = y_0 = 0.5$ and $r_0 = 1$

(c) $x_0 = 0.25$, $y_0 = 0.7$ and $r_0 = 0.1$

Figure 6.8: Gaussian function

Once we have the initial solution, we can advance our elements in time, *i.e.*, "pitch" our tents. To do so, we will proceed as Lowrie *et al.* ([38]) and advance the centre of the grid cells in time. This leaves us with pyramids across the mesh as depicted in Fig. 6.9.

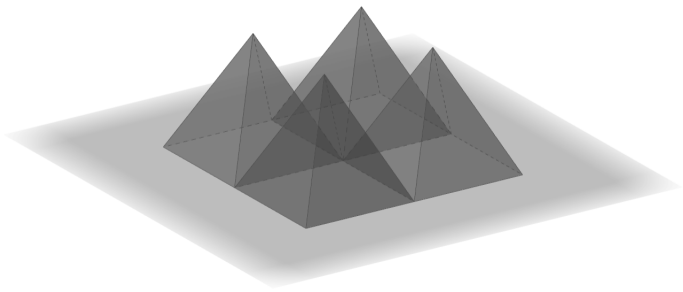


Figure 6.9: First layer of pyramids

As explained in Chapter Tent-Pitcher Algorithm, the height at which the point will be advanced depends on the wave propagation speed. Here, the height is given by $\Delta t = \frac{\min(\Delta x, \Delta y)}{2c}$.

Now that we have constructed the pyramids, we can compute the solutions inside them. Indeed, the aforementioned initial solution acts as the inflow solution U^{n-1} at the

previous time step and we can compute M and K, hence we can compute U^n following (6.1).

Let us notice that, since the mesh is structured and composed of the same elements everywhere, $|\mathcal{F}|$ will be identical for one same $\mathcal{F}$ across all pyramids of the mesh, and likewise for the tetrahedra and octahedra.

Now, we only need to propagate the solution through time. To do so, we proceed in the exact same manner as in Subsection 5.4.1.

6.1.4 External Boundaries

We explained that we can split some of the elements on the boundaries so that they can fit the initial spatial domain. When doing this, we will obtain vertical faces on which we can apply boundary conditions. In case of homogeneous Neumann boundary conditions, the only change operates on the matrix M of the boundary element, which becomes:

$$\begin{aligned} M^{\text{Neumann}} = &\int_{\mathcal{F}^{out}} \left(\frac{1}{c^2\rho} \phi_p \phi_q + \rho \phi_{\mathbf{v}} \cdot \phi_{\mathbf{w}} \right) n_t + (\phi_p \phi_{\mathbf{w}} + \phi_{\mathbf{v}} \phi_q) \cdot \mathbf{n} \\ &+ \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_p \phi_q + \rho \phi_{\mathbf{v}} \cdot \phi_{\mathbf{w}} \right) n_t + (\phi_p \phi_{\mathbf{w}} + \phi_{\mathbf{v}} \phi_q) \cdot \mathbf{n} \\ &+ \int_{\mathcal{F}^{ext}} \phi_p \phi_{\mathbf{w}} \cdot \mathbf{n} + \gamma (\phi_{\mathbf{v}} \cdot \mathbf{n})(\phi_{\mathbf{w}} \cdot \mathbf{n}) \end{aligned}$$

A first order absorbing boundary condition for the acoustic wave equation is written $\dfrac{\partial p}{\partial n} = \dfrac{1}{c} \dfrac{\partial p}{\partial t}$, thus, we obtain:

$$\begin{aligned} M^{\text{ABC}} = &\int_{\mathcal{F}^{out}} \left(\frac{1}{c^2\rho} \phi_p \phi_q + \rho \phi_{\mathbf{v}} \cdot \phi_{\mathbf{w}} \right) n_t + (\phi_p \phi_{\mathbf{w}} + \phi_{\mathbf{v}} \phi_q) \cdot \mathbf{n} \\ &+ \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_p \phi_q + \rho \phi_{\mathbf{v}} \cdot \phi_{\mathbf{w}} \right) n_t + (\phi_p \phi_{\mathbf{w}} + \phi_{\mathbf{v}} \phi_q) \cdot \mathbf{n} \\ &+ \int_{\mathcal{F}^{ext}} (\phi_{\mathbf{v}} \cdot \mathbf{n})(\phi_{\mathbf{w}} \cdot \mathbf{n}) + \phi_{\mathbf{v}} \phi_q \cdot \mathbf{n} \end{aligned}$$

6.1.5 Visualization and Results

To visualize our results, we use ParaView[2], which is an open-source visualization and data analysis software, able to run on a HPC cluster as well as on a standalone computer. It can handle multiple input data files and is conducive to handling the visualization in a parallel setup. In order to use ParaView, we need to provide it with files in a specific format, called VTK files. They can be either binary or XML files, which we need to write from our program each time we want to save or visualize our results. The format we used is the XML one and below is an example of VTK file we use.

[2] Ahrens, James, Geveci, Berk, Law, Charles, ParaView: An End-User Tool for Large Data Visualization, Visualization Handbook, Elsevier, 2005, ISBN-13: 978-0123875822

VTK file example for ParaView

```
<?xml version="1.0"?>
<VTKFile type="UnstructuredGrid" version="0.1" byte_order="BigEndian">
  <UnstructuredGrid>
    <Piece NumberOfPoints="       nPoints" NumberOfCells="        nCells">

      <Points>
        <DataArray type="Float32" NumberOfComponents="3" Name="Point" format="ascii">
            List of points  x y z
        </DataArray>
      </Points>

      <Cells>
        <DataArray type="Int32" Name="connectivity" format="ascii">
            List of elements, where each of them is represented by the index of their
            vertices e.g.  10  15  9  represents a triangular element whose vertices
            are the 10th, 15th and 9th node in the previous list of points
        </DataArray>
        <DataArray type="Int32" Name="types" format="ascii">
            List giving the type of each element from 1 to 16 for linear cells
            e.g.  5 for triangles - 7 for polygons - 9 for quads - 10 for tetrahedra
        </DataArray>
        <DataArray type="Int32" Name="offsets" format="ascii">
            List of cumulative number of vertices as we go through the elements
        </DataArray>
      </Cells>

      The solution can be given in each cell or at each node:
      <CellData Scalars="name of vector">
        <DataArray type="Float64" Name="name of field" format="ascii">
            Values of the solution in each cell.
        </DataArray>
      </CellData>
      <PointData Scalars="name of vector">
        <DataArray type="Float64" Name="name of field" format="ascii">
            Values of the solution at each node.
        </DataArray>
      </PointData>

    </Piece>
  </UnstructuredGrid>
</VTKFile>
```

Results

We solve our problem in a domain of size $L_x = L_y = 1$, discretized with two meshes (Figs. 6.10–6.11): one containing $n_x = n_y = 100$ cells in each direction and the second one containing $n_x = n_y = 200$ cells in each direction. We visualize the pressure and the velocity fields at different time steps and let the program run until it reaches $t = 1s$. We

use periodic boundary conditions and a Gaussian source function centered at $x = 0.5$ and $y = 0.5$. We use $\mathbb{P}^3$ polynomial basis functions. The results are presented in Tables 6.1 and 6.2. We observe that the solutions propagate properly and we obtain similar results to those in [24], which were validated by comparison with analytical solutions. The accuracy of these solutions will be discussed in more details in section 6.5.

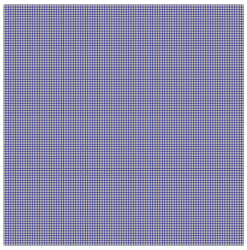


Figure 6.10: Structured mesh with 10 000 elements

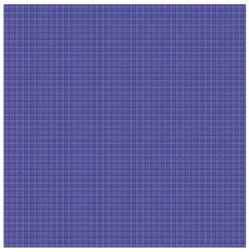


Figure 6.11: Structured mesh with 40 000 elements

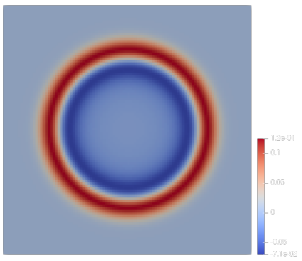
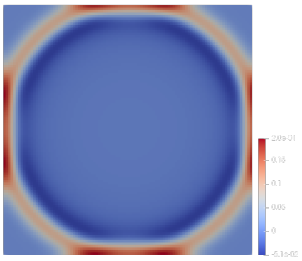
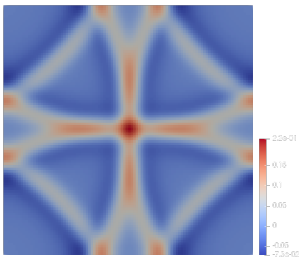
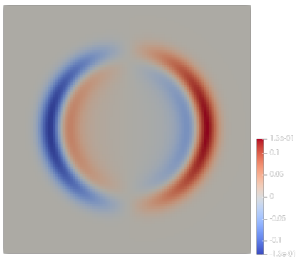
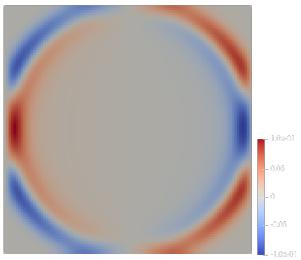
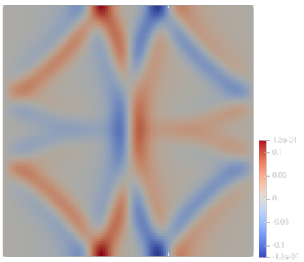
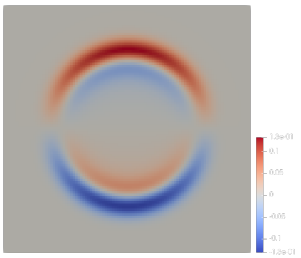
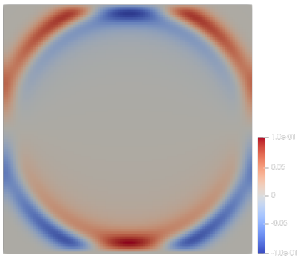
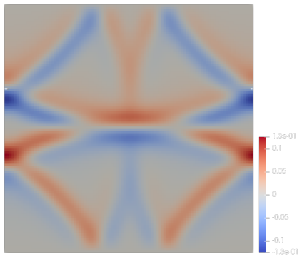
	$t = 0.3s$	$t = 0.5s$	$t = 1s$
p			
v_x			
v_y			

Table 6.1: Acoustic pressure and velocity field on 10 000 elements

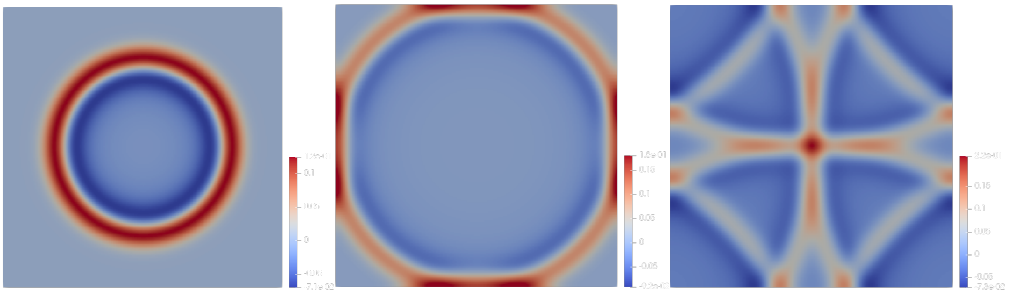
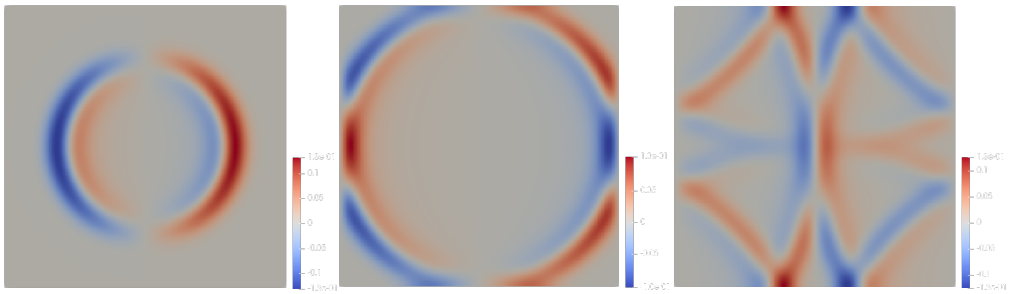
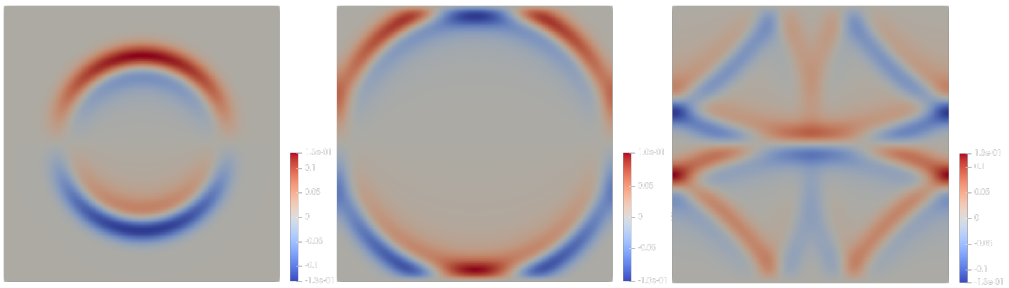
	$t = 0.3s$	$t = 0.5s$	$t = 1s$
p			
v_x			
v_y			

Table 6.2: Acoustic pressure and velocity field on 40 000 elements

6.2 Task 2: Extension to Unstructured Meshes

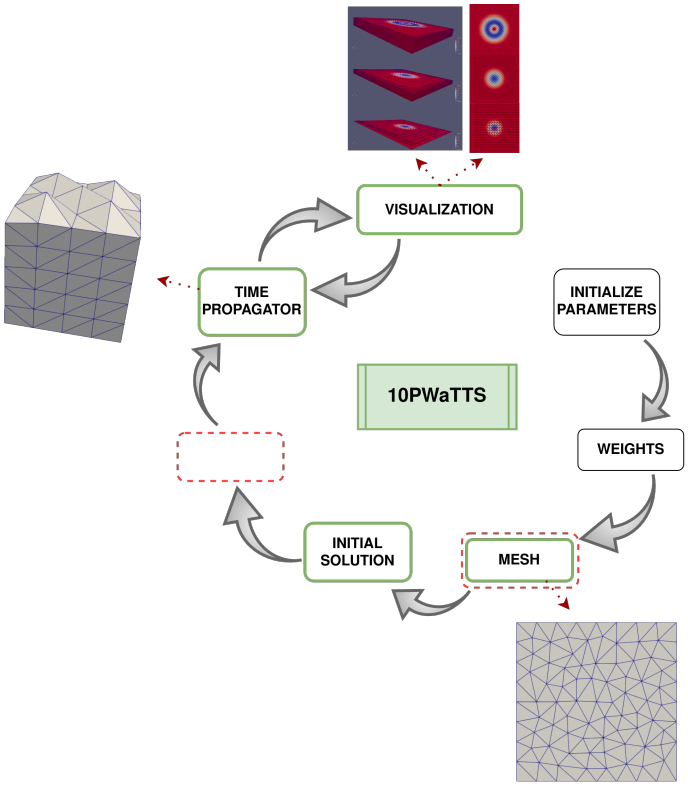


Figure 6.12: Flowchart for unstructured meshes

In this section, we aim to explicit the implementation process of extending the Trefftz-DG Tent-Pitching solver to unstructured meshes. As one can see in the flowchart 6.12, several bricks of the code are different from the structured case. Indeed, as seen in Chapter Tent-Pitcher Algorithm, the construction of a Tent-Pitching mesh in the unstructured case differs greatly from the structured case, hence differing in the implementation as well. The points we will explicit are the generation of the initial spatial mesh, the choice of a node to advance in time, the height at which it will be advanced, *i.e.*, the computation of the local time step Δt_K and the propagation in time, along with the computation of the matrices.

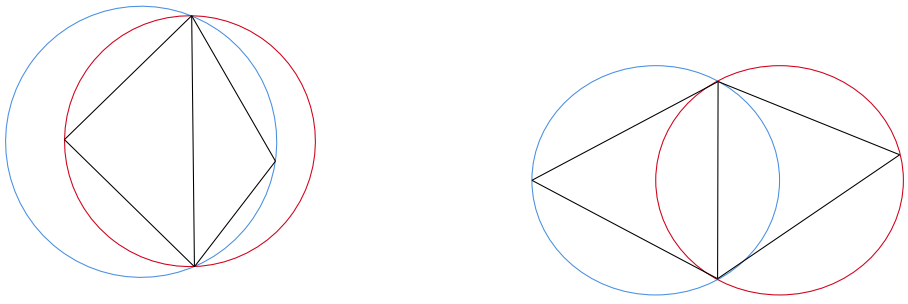
6.2.1 Mesh

The first point to discuss is the mesh. Indeed, working with an "unstructured mesh" means here that the spatial initial mesh is unstructured. In the previous case, the initial mesh was a grid and we constructed it very easily at the beginning of the program. Here, we use a meshing software called "Triangle" [3] to generate simplicial unstructured mesh.

[3] Jonathan Richard Shewchuk, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, in "Applied Computational Geometry: Towards Geometric Engineering" (Ming C. Lin and

Triangle is a mesh generating C program developed by Jonathan Richard Shewchuk. It can generate *exact Delaunay triangulations, constrained Delaunay triangulations, conforming Delaunay triangulations, Voronoi diagrams and triangular meshes.*

A triangulation is said to be Delaunay when the circumscribed circle of each simplex does not contain any other mesh node. This property is illustrated in Figs. 6.13a and 6.13b. It has the property of being unique and brings the advantage of avoiding flat elements, by forcing them to be as equiangular as possible. Also, the mesh we obtain is independent of the order of the nodes and obtaining such a triangulation is rather simple for convex domains. Indeed, as we can see in Fig. 6.13, when two triangles do not fit the Delaunay critera, one can simply flip the common edge and obtain a Delaunay triangulation.



(a) Not Delaunay (b) Delaunay

Figure 6.13: Delaunay triangulation

In practice, we provide Triangle with a `.poly` file where we describe the domain that we would like to mesh, and it looks like the following:

Dinesh Manocha, editors), volume 1148 of Lecture Notes in Computer Science, pages 203-222, Springer-Verlag, Berlin, May 1996. (From the First ACM Workshop on Applied Computational Geometry.)

`mesh.poly`

```
# Number of vertices - Dimension - Attributes - Boundary markers
4 2 0 0 # A box with four points in 2D, no attributes, no element markers
# List of vertices:
1   0 0
2   0 1
3   1 0
4   1 1

# Number of segments - Boundary markers
4 0
# List of segments:
1   1 2
2   2 4
3   4 3
4   3 1

# Holes
0
```

Then, all one needs to do is to launch the following command `./triangle -[commands] mesh.poly`, where `-[commands]` allow many different settings, such as constraining the angles, the area of the elements, activating boundary markers, etc. In the end, Triangle provides us with several output files which list all elements, edges and nodes of the produced mesh, as follows:
`mesh.1.ele`

```
# Number of elements - Nodes per element - Boundary markers
1577 3 0    #here, 1577 triangles thus 3 nodes per element and no boundary markers

# List of elements:
1  117   356   114     # element n°1 has three vertices 117, 356 and 114
.
.
.
```

`mesh.1.node`

```
# Number of nodes - Dimension - Attributes - Boundary markers
839 2 0 1    # here, 839 nodes in 2D, no attributes, boundary marker 1

# List of nodes:
1  0.5  0.5   0  # node n°1 has coordinates (0.5,0.5) and no boundary marker

2  0.0  1.0   1  # node n°2 has coordinates (0.0,1.0) and is on the boundary
.
.
.
```

`mesh.1.edge`

```
# Number of edges - Boundary markers
2415 1    #here, 2415, boundary marker 1

# List of edges:
1  117  356  0  # edge n°1 is delimited by nodes 117 and 356 and is not on the
                # boundary
2  133   6   1  # edge n°2 is delimited by nodes 133 and 6 and has boundary
.               # marker 1
.
.
```

The mesh is described through these files, but can also be graphically displayed using the display program Show Me *via* command line `showme mesh.1.ele` as shown in Fig. 6.14.

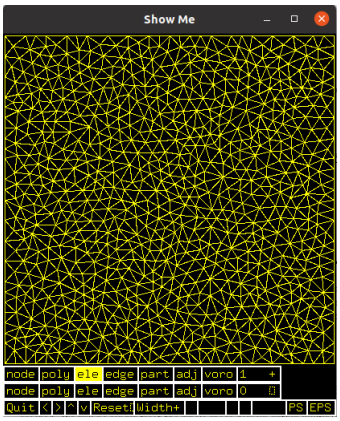


Figure 6.14: Display program for Meshes Show Me

Using such an external mesh generator also means that we have to parse the obtained files in order to get the information we need. In practice, the `mesh.1.ele` file gives us the array of elements, which are triangles. The `mesh.1.node` file is parsed to obtain the array of nodes with their coordinates. The `mesh.1.edge` file is parsed to obtain the array of edges. Additional files can be obtained, such as the neighbors of a triangle, but we do not need them in our case.

In order to carry out the Tent-Pitcher algorithm, we need additional connectivity matrices, that we construct upstream. They consist in:

- `tab_co_nodes[i]`: for a given node i, this array gives us the list of nodes connected to i; thus, this is `star[i]` (see Fig. 6.15) (see Fig. 6.15),
- `tab_co_node_ele[i]`: for a given node i, this array gives us the list of facets connected to i (see Fig. 6.15),

- `tab_node_edge[i]`: for a given node i, this array gives us the list of edges enveloping i; thus, this is `link[i]` (see Fig. 6.15).

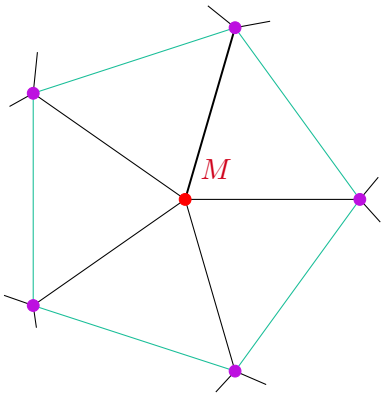


Figure 6.15: `star` of, `link` of and facets connected to a point M

6.2.2 Time propagator

The time propagation loop is also changed. Previously, we first computed elementary matrices, then the solution in the first layer of pyramids and finally repeated the construction of the tents and calculation of solution in the remaining layers until a desired time is reached. We will no longer treat the whole space domain within multiple layers, from now on we choose a point to pitch, construct the corresponding tent and compute all necessary matrices inside. Hence, for each tent we proceed in this manner and advance the front.

Choice of a Tent-Pitching point

While presenting the Tent-Pitcher algorithm in Chapter 5, we mentioned the fact that we would be choosing the point to pitch by searching for a minimum in time of the mesh front. First, to explain and justify this choice, let us see which points we can not pitch and why. In [35], Üngör and Sheffer define illegal base nodes, in order to prevent the construction of tents violating the causality constraint or having an empty or near-empty volume, as follows:

Illegal base node

A mesh node M is called an illegal base node to be advanced in time if there exists a point $Q \in$ `star`(M), where $t(Q) < t(M)$ and $t()$ represents the time component

Actually, it is explained that this definition is over-restrictive and can be relaxed but in our case, we follow this over-restrictive definition and only pitch points that are not illegal according to this definition. Now that we know what kind of points we can not pitch, let us see which points we can pitch and how to choose them. There are several

ways to choose a node to advance in time, as we have explained in Chapter Tent-Pitcher Algorithm, which are:

- lowest apex: this choice suggests pitching a tent with the lowest apex,
- optimal face angle: in this case, we want the angle between a face and the space domain β to be as close as possible to the cone constraint α. The tent which would be the best according to this criterion is chosen to be pitched,
- maximal tent volume: we choose a tent to pitch based on its volume, which we want to maximize,
- lowest node: the lowest node of the mesh is always legal and consists of one of the minimal nodes in time.

As we can see, the reason why there are many ways to decide on which node to choose is because it affects the mesh quality. In our case, we choose the lowest node, which is always legal, and disregards the mesh quality.

We impose an additional constraint on pitch point if there are more than one minimum in time. If there were two minima in time, we choose the one with the "lowest" `link`. In Fig.6.16, we represent such a case. There are two nodes M and N minimal in time, however, $\sum_{i=1}^{N_l^N} \texttt{link}(N) < \sum_{i=1}^{N_l^M} \texttt{link}(M)$. Hence, in this case, we choose N to be advanced in time. Let us notice that we could also choose one of the points depicted in orange by comparing their `link` to `link`(N).

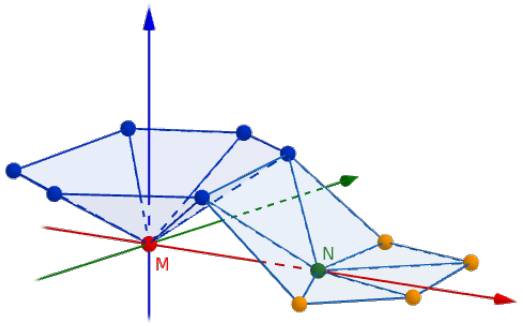
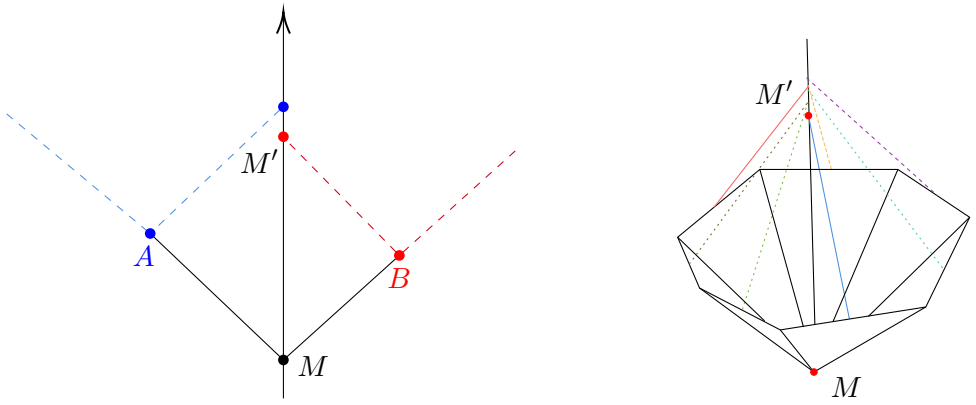


Figure 6.16: Constraint on the `link`

Now that we know which point to advance in time, we need to determine at which height it will be extruded. To do so, we will lean on the causal constraint. In fact, we want all of the new faces that will be constructed to respect the causality constraint. To do so, we compute the domain of influence of every facet connected to the pitch point M, intersect them, and choose the minimal tentpole height. If anything else than the

minimum is chosen, the causality constraint would be broken. We can see the process on Fig.6.17 for a 1d tent and then for a 2d tent.



(a) 1D+t (b) 2D+t

Figure 6.17: Choice of tentpole height $\Delta t = M'_t - M_t$, where the domain delimited by the blue dashed lines is the cone of influence of A and the one delimited by the red dashed lines is the cone of influence of B

Computation of matrices

Now we will address the question of matrices. Once Δt_K is determined, we construct the tent by adding new faces connecting M' to `link`(M) as shown in Fig. 6.18. Now that the tent is constructed, we can compute the solution inside.

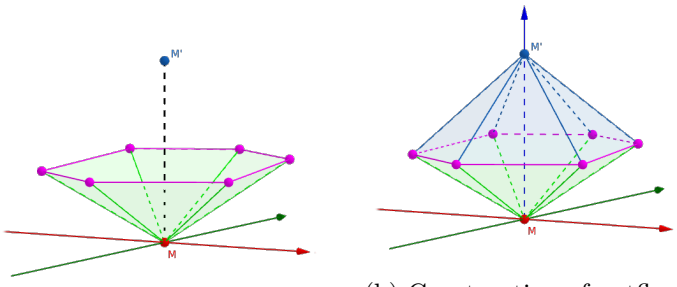
Let us recall some notations from the previous chapter:

- M^e, K^e matrices defined on an elementary triangle
- M^f, K^f matrices defined on a random triangle

For the structured case, M^f and K^f were identical throughout the mesh. However here, they vary depending on the considered tent. In fact, the area of the triangle f will differ for every facet, since the mesh is fully unstructured. Using these matrices, we ultimately want to construct M^t and K^t, which represent the matrices on a tent, in order to solve

$$M^t U^n = K^t U^{n-1}\,, \tag{6.4}$$

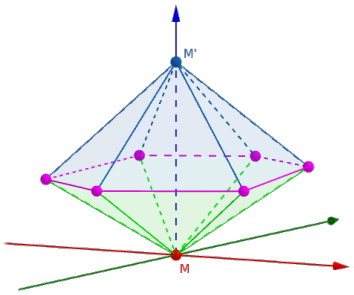
where U^n represents the solution vector at current time step and U^{n-1} represents the solution vector at previous time step.



(a) Choice of pitch-point

(b) Construction of outflow faces

Figure 6.18: Construction of a tent



Taking the scenario in Fig. 6.18 as example, if we pitch the point M depicted in red in Fig. 6.18 and would like to compute the matrix M^t, we have six inflow faces. Thus, M^t is obtained by summing $(M^f)^{out}$ on the six outflow faces and $(M^f)^{in}$ on the six inflow faces. K^t is only defined on the inflow faces, hence we sum K^f on the inflow facets only.

$$M^t = \sum_{i=1}^{6} (M_i^f)^{in} + \sum_{i=1}^{6} (M_i^f)^{out} \qquad K^t = \sum_{i=1}^{6} K_i^f$$

To generalize this idea, notice that N_s gives the number of edges connected to a point, which is equivalent to the number of simplexes connected to a point. Hence:

$$M^t = \sum_{i=1}^{N_s(M)} (M_i^f)^{in} + (M_i^f)^{out} \qquad K^t = \sum_{i=1}^{N_s(M)} K_i^f$$

Then, for each tent, we need to invert M^t in order to obtain U^n, which is the solution of the problem inside the constructed tent. In order to visualize the solution, we need to interpolate U^n on the outflow faces of the tent.

6.2.3 Visualization & Results

We test our code on several meshes, in order to verify that our implementation of the Tent-Pitcher algorithm on unstructured meshes works properly. We test it on four meshes: a bicycle seat and a leaf (provided by the website of Triangle[4]) and two squares. The two first meshes illustrate how the algorithm naturally adapts to very unstructured meshes and very different-sized cells. We can see that some tents are very small, while others are larger. The Tent-Pitcher algorithm handles such cases by naturally performing a local time-stepping method.

[4] `https://people.sc.fsu.edu/~jburkardt/data/triangle_files/triangle_files.html`

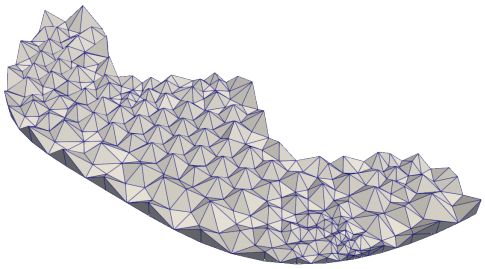


Figure 6.19: After the construction of 150 tents

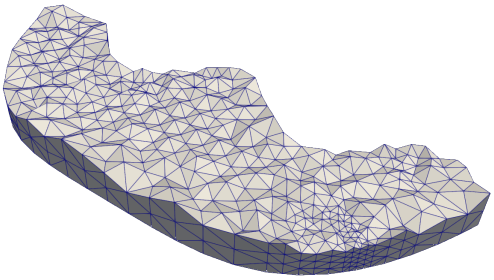


Figure 6.20: After the construction of 1050 tents

Figure 6.21: Tent-Pitching mesh of a bicyle seat

The first mesh represents a bicycle seat, as can be seen in Fig. (6.21).

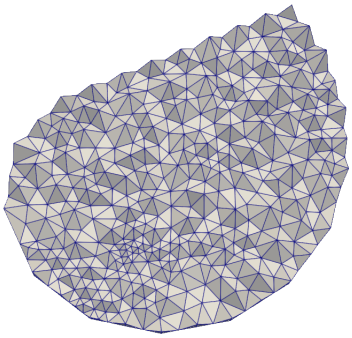


Figure 6.22: After the construction of 150 tents

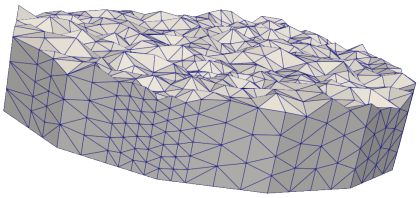


Figure 6.23: After the construction of 1950 tents

Figure 6.24: Tent-Pitching mesh of a leaf

The second mesh represents a leaf, as can be seen in Fig. (6.24). This one is particularly interesting, because we can see the structure of the mesh and how small and large cells cohabit on the boundaries of the spacetime mesh in Fig. (6.23).

Results

We solve our problem in a domain of size $L_x = L_y = 1$, discretized with two square meshes (Fig. 6.25–6.26): one containing 159 elements and the second one containing 1577 elements. Since we are validating the sequential code, we work with smaller meshes. We visualize the pressure and the velocity fields at different time steps and let the program run until it reaches $t = 1s$. We test the code with either Neumann boundary conditions

or absorbing boundary conditions. We use Gaussian initial function centered at $x = 0.5$ and $y = 0.5$ and $\mathbb{P}^3$ polynomial basis functions. The solutions presented in Tables 6.3–6.5 propagate properly and seem to converge to the solutions obtained in the structured case shown previously. The convergence and accuracy of the obtained solutions will be further discussed in Section 6.6.

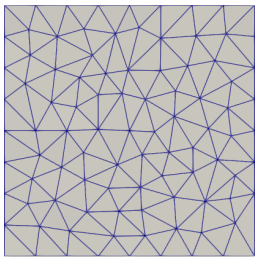


Figure 6.25: Coarse unstructured mesh

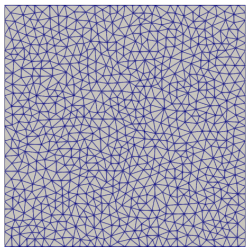


Figure 6.26: Refined unstructured mesh

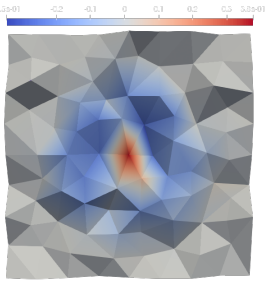
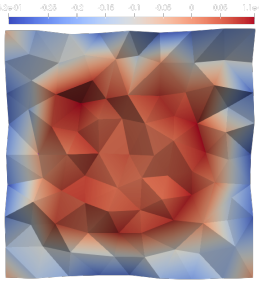
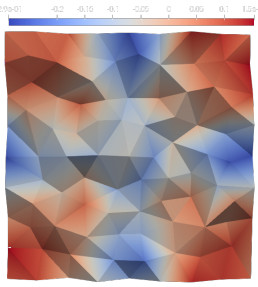
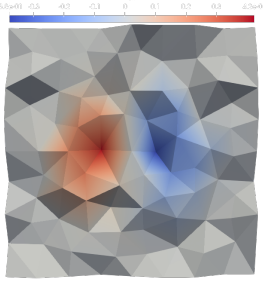
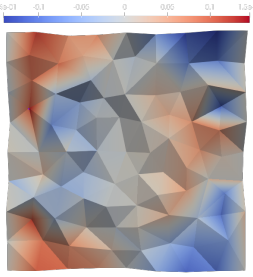
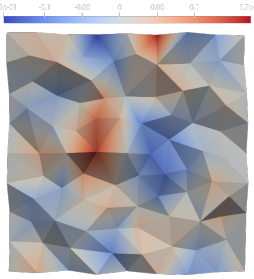
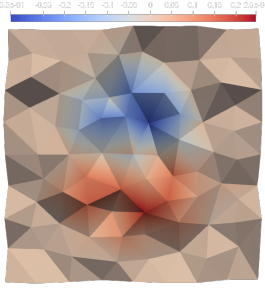
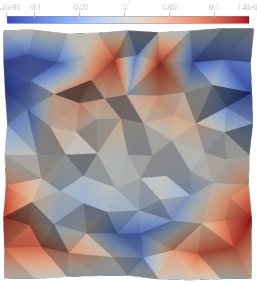
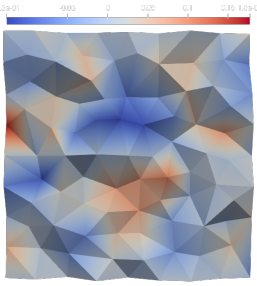
	$t = 0.15s$	$t = 0.5s$	$t = 1s$
p			
v_x			
v_y			

Table 6.3: Acoustic pressure and velocity fields on coarse unstructured mesh with homogeneous Neumann conditions

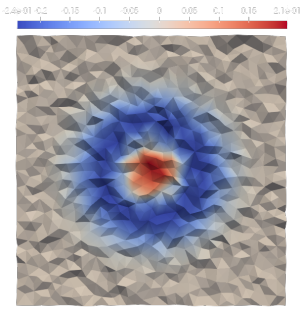
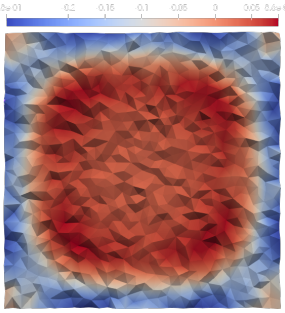
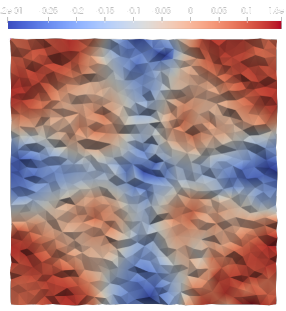
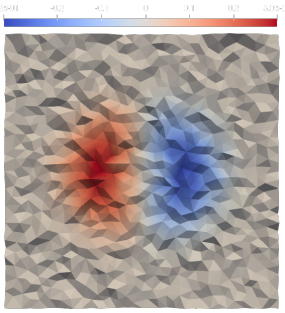
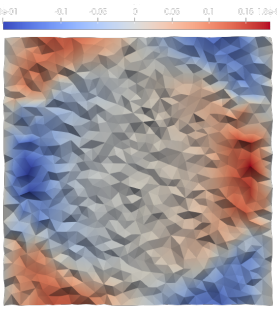
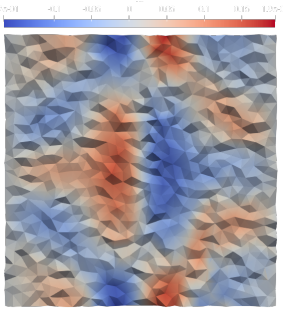
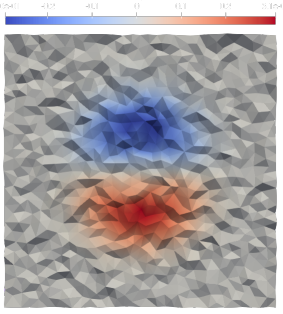
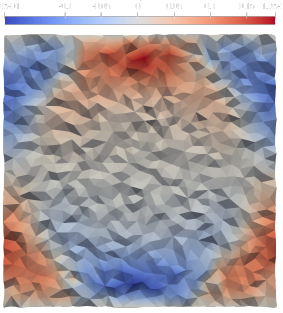
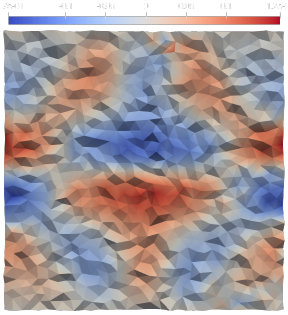
	$t = 0.15s$	$t = 0.5s$	$t = 1s$
p			
v_x			
v_y			

Table 6.4: Acoustic pressure and velocity fields on refined unstructured mesh with homogeneous Neumann conditions

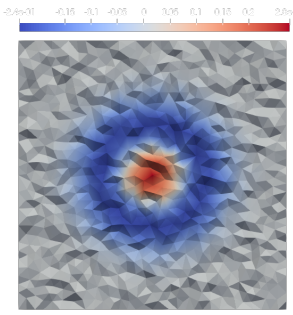
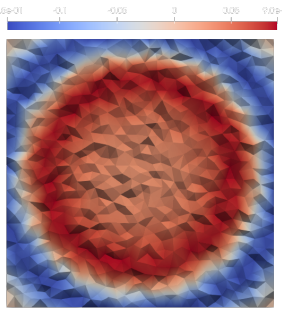
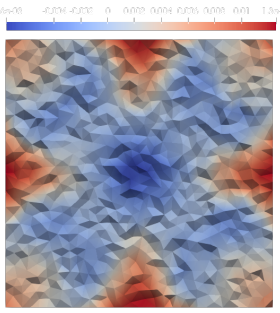
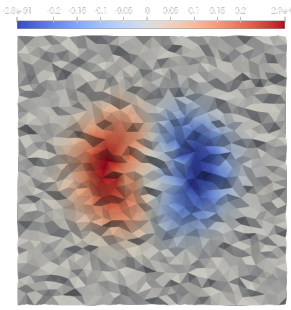
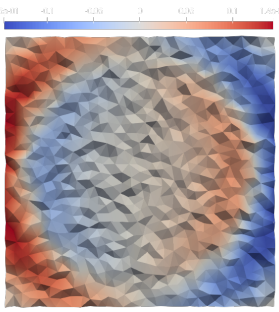
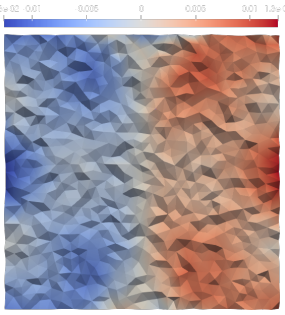
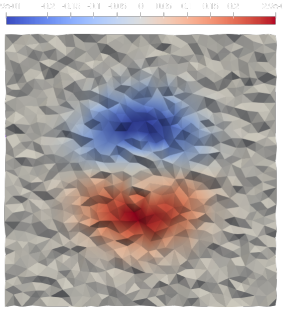
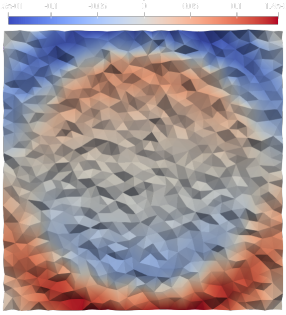
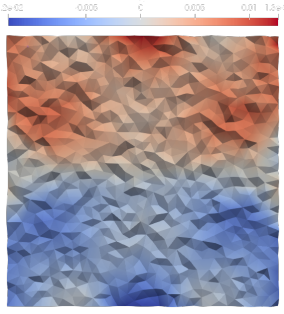
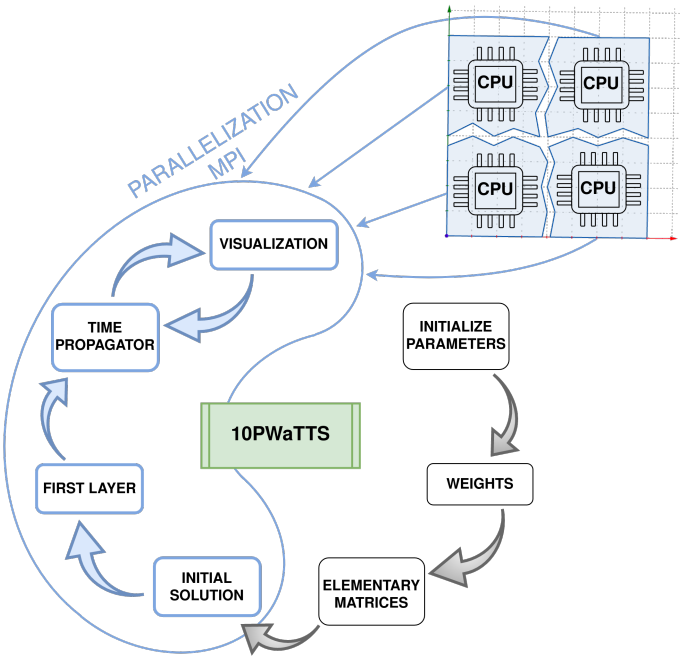
	$t = 0.15s$	$t = 0.5s$	$t = 1s$
p			
v_x			
v_y			

Table 6.5: Acoustic pressure and velocity fields on refined unstructured mesh with ABCs

6.3 Task 3: Parallelizing the structured case



6.3.1 High Performance Computing

High Performance Computing allows to both solve larger complex problems and to solve them faster, by using supercomputers or computer clusters. In some cases, using such environments to solve problems goes faster than on a regular computer, but in some others, it simply makes it possible to solve the problem, because a regular computer may not support it. Some algorithms are more fit to be parallelized than others.

There are three means to parallelize a piece of code, each coming with its own pros and cons:

- threading (OpenMP),
- distributed memory (MPI),
- GPU computing.

6.3.2 MPI

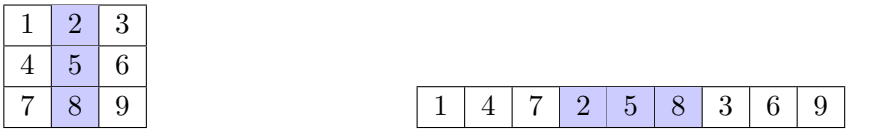
Message Passing Interface is a standard for HPC allowing to pass messages between distinct computers or inside a multiprocessor computer. It is a standard of communication for nodes executing parallel programs on distributed memory systems. It defines a library of functions that can be used in C, C++ and Fortran.

When a program using MPI is launched, a *communicator* is set and it designs a set of processes that can communicate with each other. Each process is given a *rank* starting from zero, in order to be identified when sending or receiving messages. There are several types of communications, such as:

- **Point-to-point communication**: describes communications between two processes in one communicator. There are two types of them: *blocking* and *non-blocking*. *Blocking* means that a process will wait until the message is received or sent before performing the remaining tasks. Thus, if a send is initiated somewhere, there needs to be a receive. And analogously, *non-blocking* means that the process does not wait and moves on to its next task after performing its part of the communication. The functions used in order to carry out blocking communications are `MPI_Send` `MPI_Recv` and `MPI_SendRecv` and their equivalent for non-blocking communications are `MPI_iSend` `MPI_iRecv` and `MPI_iSendRecv`,
- **Collective communications**: This type of communication implies all processes in one communicator. This means that one process can send information to all other processes and it can be performed in multiple manners depending on the information we want to send. One way, is to send the same piece of information to everyone and it is done through `MPI_Bcast`. A second way, is to scatter an array for example to the other processes. It means that a distinct chunk of the array will be distributed to all processes, this is done through `MPI_Scatter`. A last type of collective communication is when one wants to gather distinct information from every process into one array; this is the reverse of the previous function and is done through `MPI_Gather`.

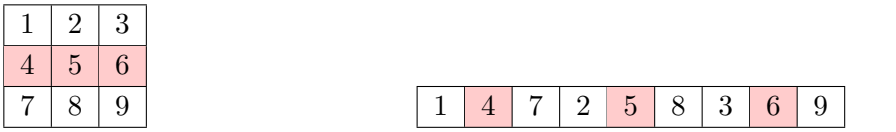
Basic types such as `integer` or `real` are included in the MPI standard, but it is also possible to create a type. Types are necessary during communications, because we need to specify the type of the sent data. It can be convenient to create a type in order to carry out a certain communication. For example, if one wants to send a submatrix or a vector, it is not possible with the standard types and functions but a submatrix or vector type could be created to carry this out. The only important thing to keep in mind when creating types is the layout in memory. For example, if we want to send a column of a matrix $A(m, n)$ to another process, we create a type and we need to specify how many elements to consider and at which location to start considering them. Thus, we just need to give the size m of the column and pass the information that we start counting at $A(1, 1)$. This will result in sending the first column of matrix A, because the elements are stored column-wise in the memory in Fortran. This is called a *contiguous type*. But if we want to send the first row of the same matrix, we need to specify a displacement. Hence, we pass the information that we send n elements, starting from $A(1, 1)$ every m elements. This is called a *vector type* and these notions are illustrated in Figs. 6.27 and 6.28.

Remark. *Contiguous types and vector types are inverted in Fortran and C, because the*



(a) Column of a matrix (b) Memory layout of the column in Fortran

Figure 6.27: Contiguous type



(a) Row of a matrix (b) Memory layout of the row in Fortran

Figure 6.28: Vector type

memory in Fortran is arranged following columns whereas in C it is arranged following lines.

Among the many functionalities found in the `MPI` library, we are interested in five of them in order to parallelize the structured case, and those are:

- `MPI_Dims_create`: given the number of nodes in the meshgrid and of dimensions, creates a division of processors in a Cartesian grid. Hence, the output is a vector denoted `dims` which gives us the number of processes in the x- and y-direction,
- `MPI_Cart_create`: creates a new Cartesian communicator to which topology information has been attached, which is the output of the previous function,
- `MPI_Cart_rank`: given Cartesian coordinates, determines process rank in communicator,
- `MPI_Cart_coords`: given rank, determines process coordinates in Cartesian topology,
- `MPI_Cart_shift`: returns the ranks of the neighboring processes in the given direction at the displacement d of the actual process. It is simply a wrapper of `MPI_Cart_coords` and `MPI_Cart_shift`.

	-1	-1	
-1	$(0,1)$	$(1,1)$	-1
-1	$(0,0)$	$(1,0)$	-1
	-1	-1	

(a) Coordinates of process without periodicity

	$(0,0)$	$(1,0)$	
-1	$(0,1)$	$(1,1)$	-1
-1	$(0,0)$	$(1,0)$	-1
	$(0,1)$	$(1,1)$	

(b) Coordinates of process with periodicity in the y direction

	-1	-1	
$(1,1)$	$(0,1)$	$(1,1)$	$(0,1)$
$(1,0)$	$(0,0)$	$(1,0)$	$(0,0)$
	-1	-1	

(c) Coordinates of process with periodicity in the x direction

	$(0,0)$	$(1,0)$	
$(1,1)$	$(0,1)$	$(1,1)$	$(0,1)$
$(1,0)$	$(0,0)$	$(1,0)$	$(0,0)$
	$(0,1)$	$(1,1)$	

(d) Coordinates of process with periodicity in both directions

Figure 6.29: Structured Tent-Pitching process seen from above

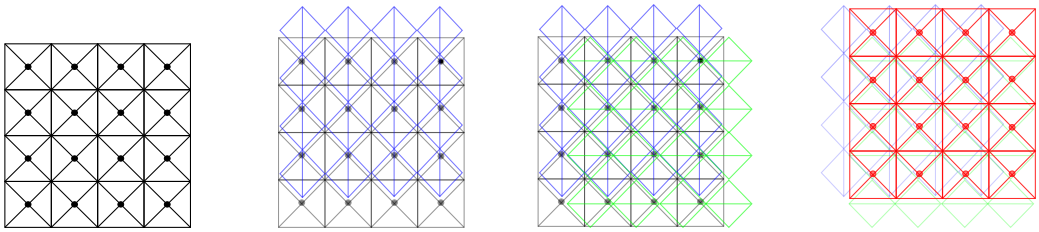
In fact, our initial space mesh is a grid, thus owns a cartesian topology. This is why we wanted to use `MPI_Cart_create` to create a cartesian communicator. As seen before, the first communicator attaches a rank to each process. Thanks to this new Cartesian communicator, a process will not only be identified by a rank but also with coordinates, as depicted in Fig.6.29. This allows us to identify the neighboring processes very easily and in fact, if we denote as (x, y) the Cartesian coordinates of a process, obtained through `MPI_Cart_coords`, we have:

- its left neighbor at coordinates $(x-1, y)$,
- its right neighbor at coordinates $(x+1, y)$,
- its bottom neighbor at coordinates $(x, y-1)$,
- its top neighbor at coordinates $(x, y+1)$.

And each of their rank can be obtained through the function `MPI_Cart_rank`. Or, one can use `MPI_Cart_shift`, which gathers a call to `MPI_Cart_coords` and `MPI_Cart_rank`, and allows to get the neighbors at a specified distance. Moreover, with this Cartesian topology, it is also very easy to set periodic boundary conditions. The period argument in `MPI_Cart_create` is a vector of size of the problem (so, it is a vector of size two here) which specifies if the grid is periodic (one) or not (zero) in each dimension. Hence, since we consider periodic media in both directions, $\texttt{period} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

We are parallelizing the code in space and the spatial blocks stay the same throughout time. This is a very efficient parallelization for large domains, but it would be even more interesting to combine MPI and thread parallelizing. This way, we would be using the fact that all the tents can be computed independently, and the ratio between usage of resources and timing would be at its best.

6.3.3 Communications

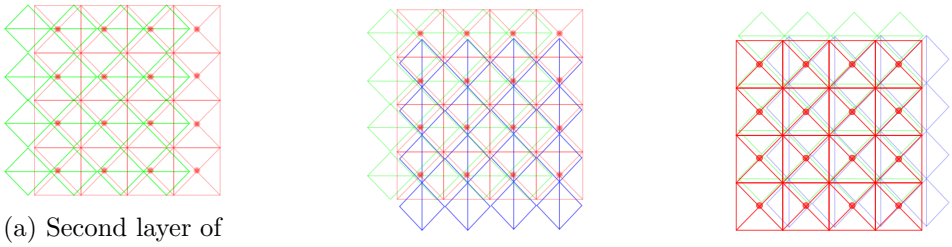


(a) First layer of pyramids

(b) First layer of vertical tetrahedra (blue)

(c) First layer of horizontal tetrahedra (green)

(d) First layer of octahedra (red)



(a) Second layer of horizontal tetrahedra (green)

(b) Second layer of vertical tetrahedra (blue)

(c) Second layer of octahedra (red)

Figure 6.31: Structured Tent-Pitching process seen from above

In the structured case, we are completely in control as to which information needs to be sent and to whom, because we know the exact layout of the mesh. Let us consider a domain separated in four seen from above as in Fig.6.31. The gray points represent the summit of the pyramids, the horizontal tetrahedra are depicted in green and the vertical tetrahedra are depicted in blue, while the octahedra are represented by red squares and their summit in red. We can see on this figure what was explained before; the elements are shifted by half a cell. Hence, there will be an overlap of some elements between the processes, which indicates us that each process needs to communicate information to its neighbors, in order for each of them to carry out their own computations.

Let us consider only one process and its corresponding subdomain as in Fig.6.32. We can see in Figs. 6.30b and 6.30c that in order to compute the last column of horizontal tetrahedra, this process needs to know the solution on some pyramids from its right neighbor. In the same way, to compute the solution on its first row of vertical tetrahedra, this process needs to know the solution on some pyramids from its top neighbor. As this lack of information applies to all processes, the left and bottom neighbor need the solution on certain pyramids from this process as well.

Communication n°1: Hence, every process will send the solution on its last column of pyramids to its left neighbor and the solution on its first row to its bottom neighbor,

as can be seen in Fig. 6.32.

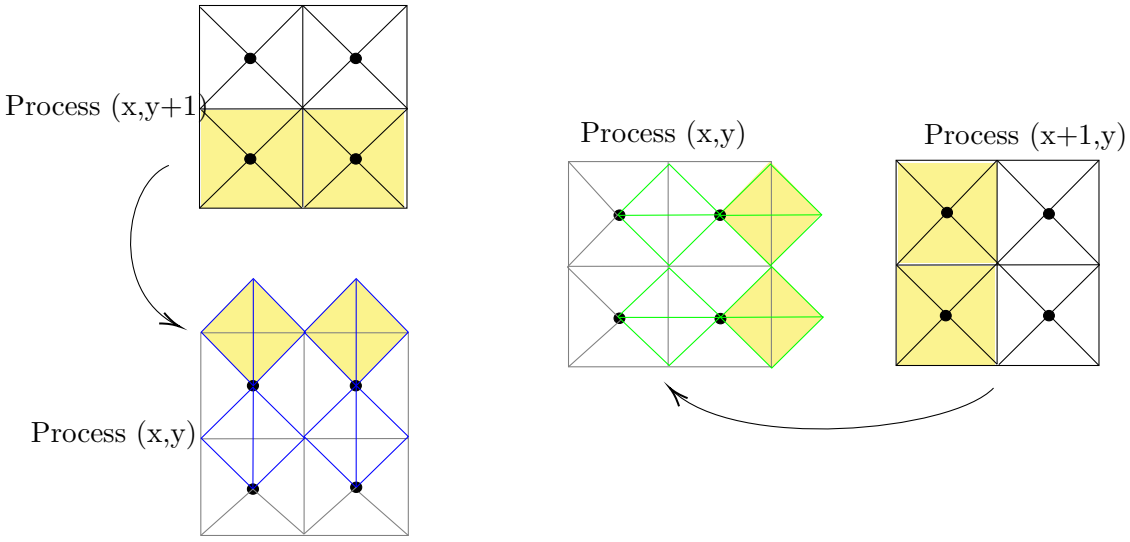


Figure 6.32: Communications n°1

In the same manner, we see in Fig. 6.30d that to compute the solution in the first row and last column of octahedra, we need to know the solution on some vertical tetrahedra from the right neighbor and horizontal tetrahedra from the top neighbor.
Communication n°2: Hence, every process will send the solution on its first column of vertical tetrahedra to its left neighbor and the solution on its last first of horizontal tetrahedra to its bottom neighbor as illustrated in Fig. 6.33.

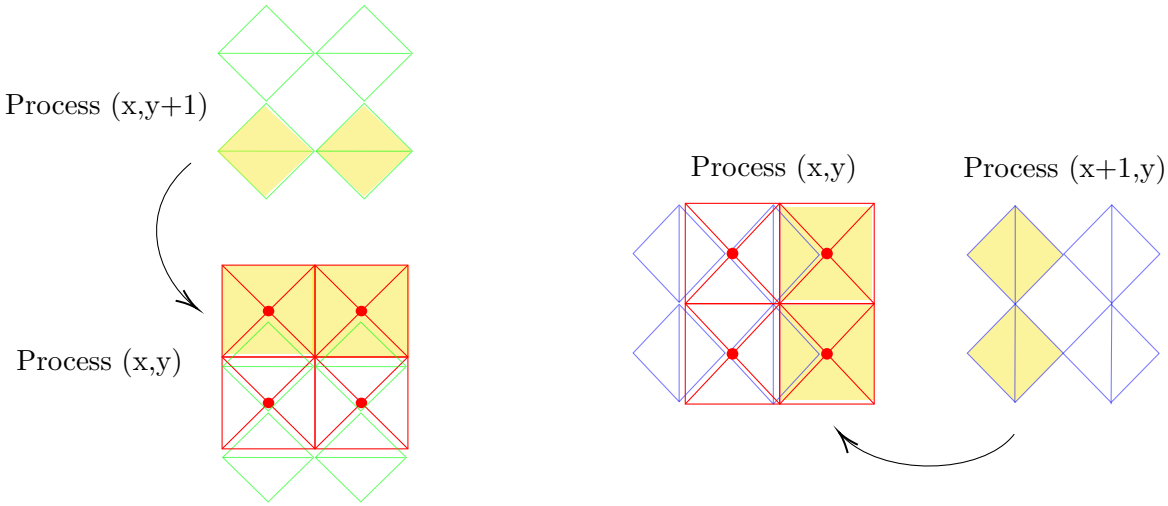


Figure 6.33: Communications n°2

As explained when presenting the structured Tent-Pitcher algorithm and as shown in Figs. 6.31b and 6.31a, we now need to form, again, a layer of horizontal and vertical

tetrahedra by connecting the octahedra's summits.
Communication n°3: Hence, every process will send the solution on last column and first row of octahedra to its top and right neighbor respectively, as illustrated in Fig. 6.34.

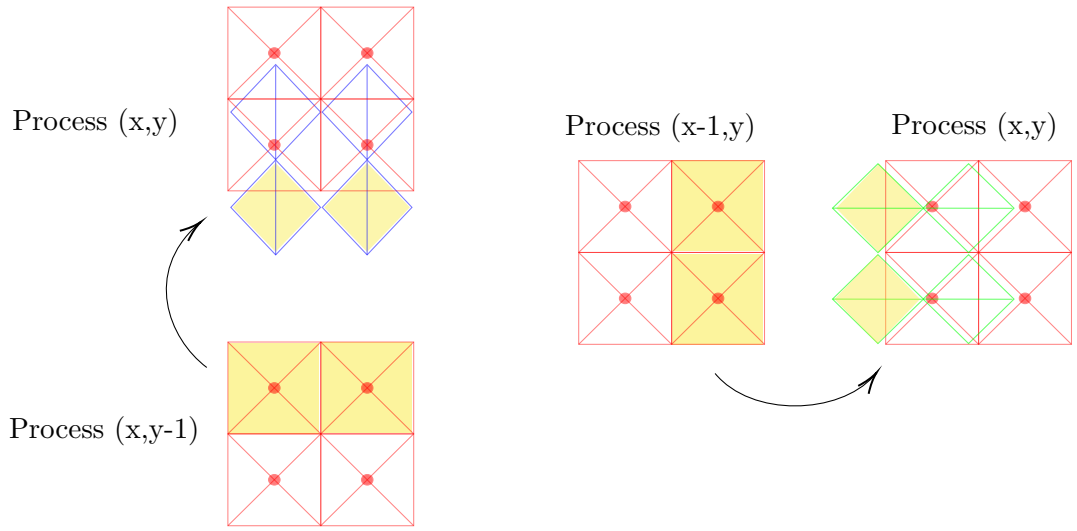


Figure 6.34: Communications n°3

We repeat all three of these communications at each new set of layers using `MPI_SendRecv`. Since we are using periodic boundary conditions and that we set our `MPI_Cart` in a periodic manner as well, the communications on the boundary processes will be performed following Fig. 6.29d.

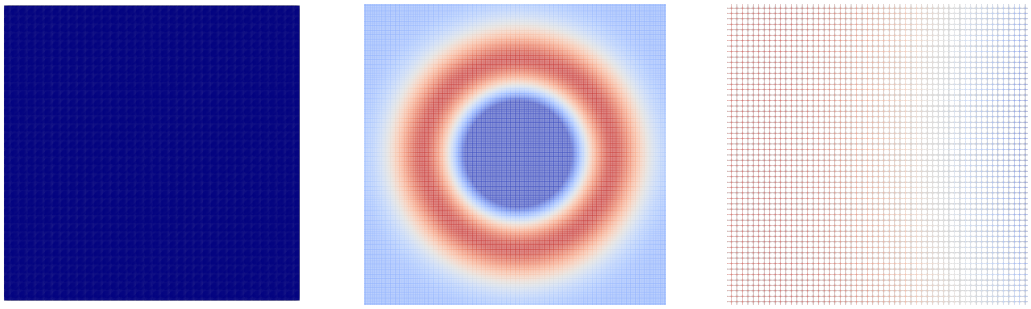
6.3.4 Visualization & Results

We solve our problem in a domain of size $L_x = L_y = 1$, discretized with a very large mesh of one million elements. This mesh is depicted in Fig. 6.35. Such tests are too big to be done on a laptop, hence we run our tests on the PlaFRIM[5] platform, which is an HPC cluster located in Bordeaux and is operated by Inria, Labri, IMB. We visualize the pressure and the velocity fields at different time steps and let the program run until it reaches $t = 1s$. We use periodic boundary conditions and a Gaussian source function centered at $x = 0.5$ and $y = 0.5$ and $\mathbb{P}^3$ polynomial basis functions. The results presented in Table 6.6 propagate properly and are similar to the previously obtained results with the sequential code. The tests are performed on a machine Zonda, which has the following specifications:

- 2x 32-core AMD Zen2 EPYC 7452 @ 2.35 GHz,

[5]Experiments presented in this section were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr).

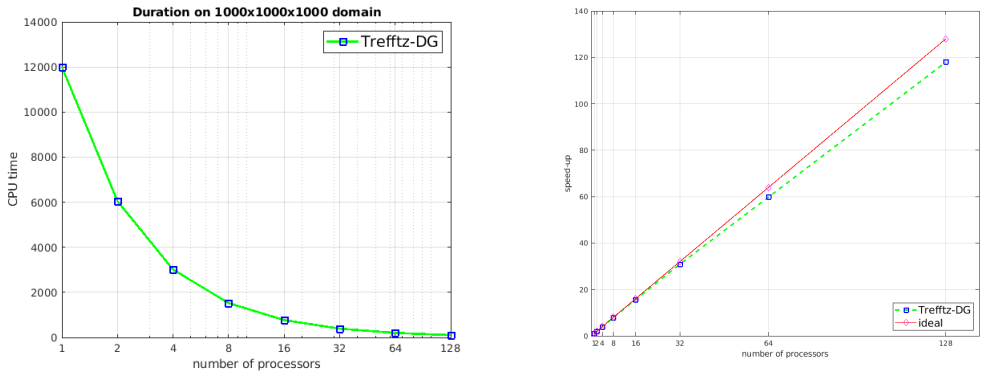
- 256 GB (4 GB/core) @ 3200 MT/s,
- 10 Gbit/s Ethernet,
- storage over 10G Ethernet.



(a) Mesh (b) Zoom x2 on mesh (c) Zoom x4 on mesh

Figure 6.35: Structured mesh with one million elements

In the following table, we present the time spent relative to the number of processes (MPI nodes). The time is obtained using the function `CPU_time` and is the mean on all processes. We measure the time starting from the very beginning after the `MPI_init` to the very end, before the `MPI_finalize` once the desired time $t = 1s$ is reached by all tents in the domain. The time spent on writing ParaView files at every hundred time steps for the visualization is also included. The performance of the code is illustrated in Table 6.7 and Figs 6.36 by representing the duration and the speed-up of the code.



(a) CPU time in function of the number of MPI nodes (b) Speed-up in function of the number of MPI nodes

Figure 6.36: Performance of the parallelized Trefftz-DG solver with Tent-Pitching

When increasing the number of MPI nodes n, the CPU time should ideally be equal to $t_n = \frac{t_1}{n}$, where t_1 is the CPU time spent with one process and t_n is the CPU time

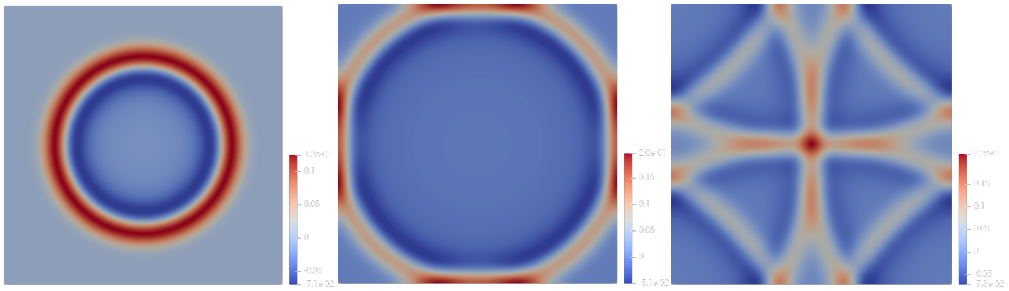
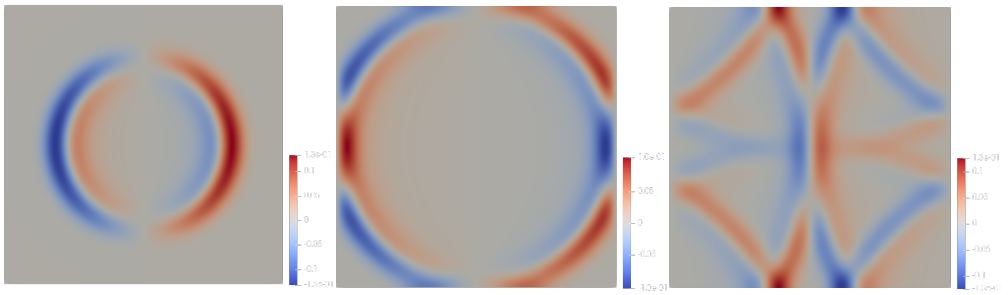
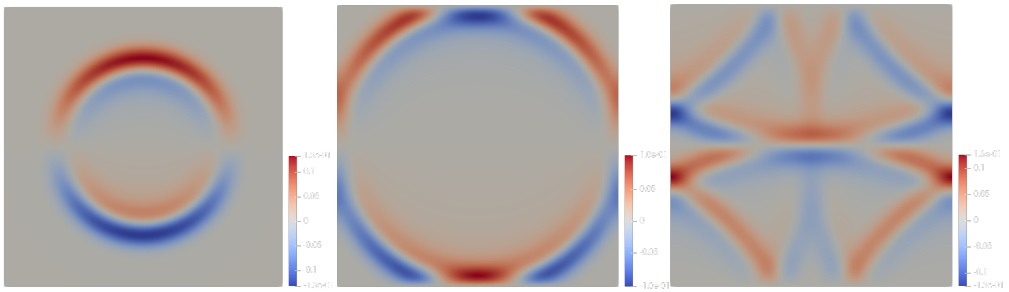
	$t = 0.3s$	$t = 0.5s$	$t = 1s$
p			
v_x			
v_y			

Table 6.6: Acoustic pressure and velocity fields computed with one million elements

processes	1	2	4	8	16	32	64	128
CPU time (s)	12003	6050	3012	1525	770	389	201	102
CPU time (h, m, s)	3h20m	1h41m	50m	25m	13m	6m30s	3m20s	1m42s

Table 6.7: Duration of the simulation in CPU time (in seconds, then in hours/minutes/seconds)

spent with n processes. A code scales properly when it is the closest to the ideal $y = n$ and to represent this, we draw the line $y = \frac{t_1}{t_n}$. This line represents the speedup of our code and the ideal speedup is represented in red in 6.36b by the line $y = n$. We can see that the line representing the Trefftz-DG solver is very close to the ideal line, which means that it has a good scaling.

6.4 Task 4: Parallelizing the unstructured case

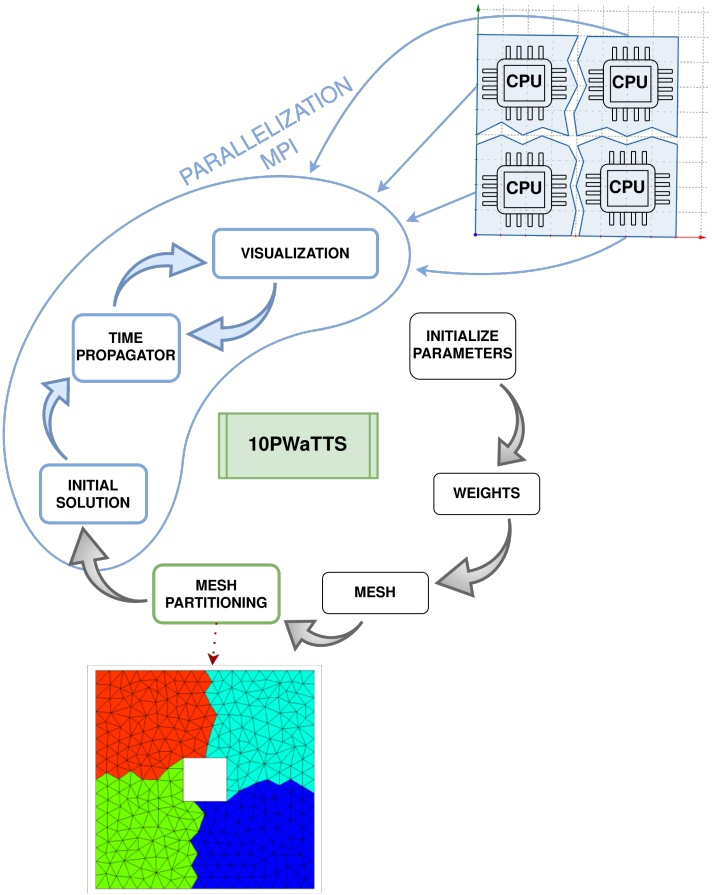


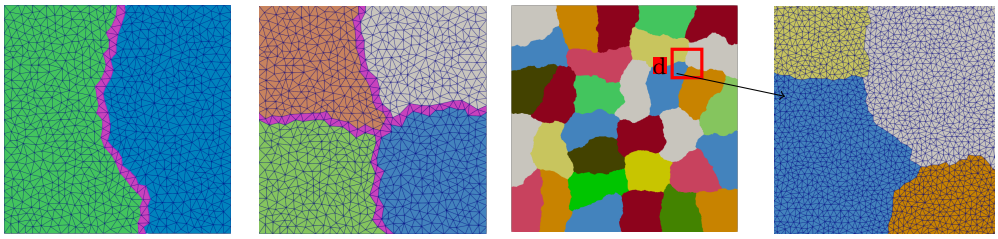
Figure 6.37: Flowchart: Parallelization of unstructured meshes

In this section, we intend to explain the process of parallelizing the Trefftz-DG solver applied to unstructured meshes. As we have seen in the last section, parallelizing the structured case was very specific to the Cartesian topology we had. Thus, we have here an additional brick which deals with the partitioning of the unstructured initial space mesh, which will be the object of the first subsection. Then, we will address the question of communications in the second subsection, which will be very different from the structured case. Finally, we will present some results in the third subsection.

6.4.1 Mesh Partitioning

The idea here, is to distribute the work evenly between all processes. Hence, we need to distribute the mesh to the processes, as evenly as possible, and then each process will perform the computations on its submesh. In this case, the initial space mesh is unstructured and partitioning it is not as easy as partitioning a simple grid, hence we use a software to partition the mesh, such as METIS and SCOTCH. We will be using METIS.

METIS[6] is a software developed by George Karypis and Kumar Vipin for graph partitioning. The partition of the mesh relies on Graph Partitioning, which consists in dividing a graph into k-parts, which satisfy some constraints. METIS can be used as a command line tool, but there are also C and Fortran interface to the software. Here, we use the Fortran function `METIS_PartMeshNodal` from the corresponding interface to carry out our mesh partitioning. On Fig. 6.38, we can see several mesh partitions we obtain, for two, four and thirty-two processes. This last partition was performed on a very large mesh of one million elements, and the detail can be seen on Subfig. 6.38d.



(a) Partition of a mesh in 2 parts with the overlap depicted in purple

(b) Partition of a mesh in 4 parts with the overlap depicted in purple

(c) Partition of a mesh in 32 parts

(d) Zoom on mesh 6.38c

Figure 6.38: Partitioning of the domain using METIS

Once we have partitioned the space domain with METIS, we need to parse the output it gives us and construct all the structures we need and then send them to each process. METIS provides us with a simple array `npart` of size of the number of elements $nEle$, with the rank of the process to which the element belongs as illustrated below:

[6]Karypis, George & Kumar, Vipin. (1997). METIS—A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes and Computing Fill-Reducing Ordering of Sparse Matrices.

$$\texttt{npart} = \begin{pmatrix} 0 & \text{element 1 belongs to process 0} \\ 1 & \text{element 2 belongs to process 1} \\ 3 & \text{element 3 belongs to process 3} \\ 0 & \text{element 4 belongs to process 0} \\ & \vdots \\ 2 & \text{element nEle belongs to process 2} \end{pmatrix}$$

Using this output, we construct the matrices of elements, of nodes and of edges for each process and then send them via MPI communications.

We also need to consider the communications between each process. In Figs. 6.38, we can see some elements depicted in purple. These elements consist of the overlap between domains. In fact, since we pitch our nodes, we decided to partition the nodes of the domain and not the elements and this leads in an overlap of facets at the interfaces. So, a subdomain is composed of

- interior facets and ghost facets (that constitute the overlap),
- interior nodes, boundary nodes and ghost nodes (which are the vertices of the ghost elements that belong to the neighboring process),
- interior edges, boundary edges and ghost edges.

Thus, we need to communicate to each process how many of these elements it has, along with a way to distinguish them. We also need to know to which neighboring process the ghost elements belong to. We construct all appropriate structures for this data and then use the `MPI_Scatterv` function to communicate them to the other processes. Since we are sending chunks of matrices, we need to define several new types here: vector type to send a specific number of rows to the processes.

6.4.2 Communications

Once the mesh is partitioned and that all individual information is sent to every process, each of them can start its computations, that we break down as summarized in Algorithm 2 and as follows.

Each process computes the initial solution on its portion of the mesh and then starts its time propagation, by choosing a point to advance in time in each submesh. If the chosen point is not on the boundary, we perform the computations as usual: we construct the tent and compute the matrices M^t and K^t in order to find U^n as introduced in (6.4). Now, if the pitch-point is on the boundary, there are several communications to perform. First, we need to notify the neighboring process that we are pitching this node (so, send its number) so that the neighbor will not try to pitch it at the same moment. Then,

Algorithm 2: Computation flow in one process

```
• calculation of initial solution on the initial subdomain of the process;
while t ≤ T do
    • choose a point to advance in time;
    • compute matrices M^t and K^t;
    if the chosen point is on the boundary of the subdomain then
        ◇ perform communications;
    end
    • visualize the results;
end
```

we need to send the Δt_K and U^n to the neighbor, so that it can update those values on the corresponding element faces and node. We need to do that for each boundary pitch-point, hence producing a lot of little communications. The sending side is clear but what about the receiving side? As explained, when using MPI communications, if there is a sending of information, there needs to be a receipt. However, in this scenario, a process does not know when it might be receiving information. Let us say that process A is treating an interior node and process B is also treating an interior node. There is no need to exchange information here. Now, say that process A found a boundary node as lowest node. A will send information to B, but B does not know what A is doing, hence does not expect anything to be sent. This is why here, we use `MPI_IProbe`. It is an MPI function that allows receiving information dynamically. As the name says, the process will probe the incoming of a message from A, and if there is a matching send from A, B will receive it. Otherwise, nothing is done.

There are other ways to deal with the problem we have. One solution mentioned by Abedi *et al.* in [47] also carried out in a Tent-Pitching framework, is to dedicate a process to handling the mesh and mesh-related decisions. Hence, if A wants to send information concerning a boundary node to B, it will first tell the mesh-handling process which node is being treated and A will inquire at the beginning of each step to see if any of its boundary nodes are occupied. Hence, B can now send the information and A knows it will receive it. This method is more complex to implement and since our priority was to develop a proof of concept, we focused on the simpler parallel implementation. The implementation of the method proposed by Abedi *et al.* [47] is an interesting perspective to this PhD research project.

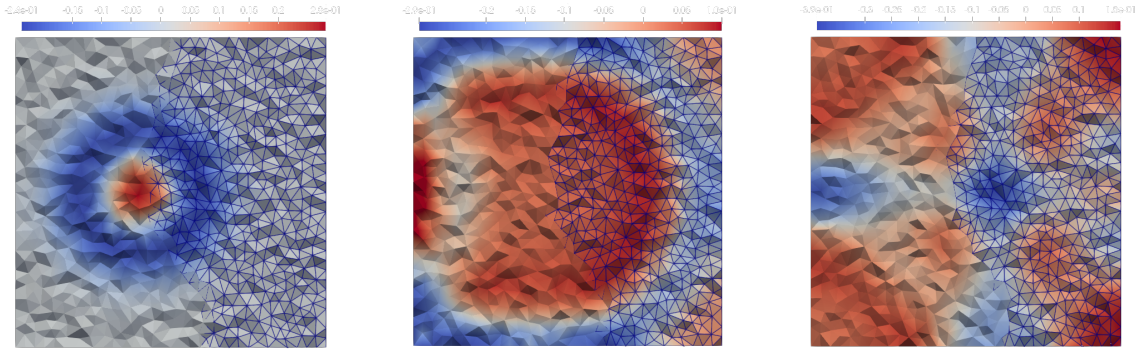
6.4.3 Visualization & Results

We solve our problem in a domain of size $L_x = L_y = 1$, discretized with the same mesh as in the previous section, which contains 1577 elements. We visualize the pressure field at different time steps and let the program run until it reaches $t = 1s$. We use Neumann

boundary conditions and we use Gaussian initial function centered at $x = 0.4$ and $y = 0.5$ and $\mathbb{P}^3$ polynomial basis functions. We tested our code with two MPI processes. We performed the tests on a laptop with the following specifications:

- Intel Core i7-8650U CPU @ 1.90GHz × 8,
- Mesa Intel® UHD Graphics 620 (KBL GT2),
- 15,5 GiB memory,
- 10 Gbit/s Ethernet,
- 1,0 TB storage.

The two MPI nodes have each been given a subdomain. In Fig. 6.39, the difference between the subdomains has been highlighted by depicting the mesh in one of them and not in the other. We can see that the acoustic pressure propagates as the solution obtained with the sequential code and seems to converge to the solution computed on structured meshes.



(a) Pressure at $t = 0.15s$ (b) Pressure at $t = 0.5s$ (c) Pressure at $t = 1s$

Figure 6.39: Acoustic pressure obtained with two MPI nodes

Running this simulation on a rather small mesh (only 1577 elements) takes approximately forty minutes, which is very slow. The solver with structured meshes only takes a few seconds to run a simulation for an equivalent number of elements. We believe that the code for unstructured meshes is not optimized enough and could produce better results with more developments to optimize it. Thus, we do not provide additional time results on multiple MPI nodes because we would have to fully optimize our code first. Due to time constraints, this work could not be completed.

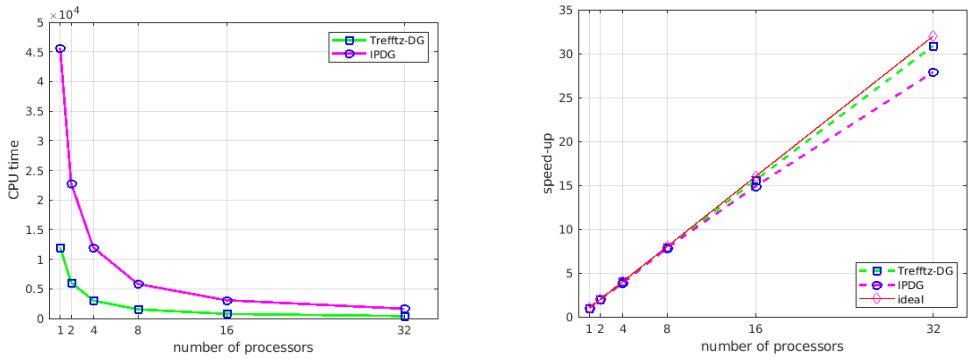
6.5 Comparison between IPDG and Trefftz-DG solvers on structured meshes

We compare the performance of our Trefftz-DG+Tent-Pitching solver to the performance of the IPDG solver of Hou10ni[7] [77], which is a software developed by project-team Makutu at Inria. We solve our problem in a domain of size $L_x = L_y = 1$, discretized with a very large mesh of one million elements. We run our tests on the PlaFRIM[8] platform, as before. We measure the CPU time once we reach the final time $t = 1s$. We observe that the Trefftz-DG solver on structured meshes is faster than the IPDG solver (see Table 6.8). Moreover, the Trefftz-DG solver scales better than the IPDG solver. In fact, when increasing the number of MPI nodes n, the CPU time should ideally be equal to $t_n = \frac{t_1}{n}$, where t_1 is the CPU time spent with 1 process and t_n is the CPU time spent with n processes. A code scales properly when it is the closest to this ideal. In Fig. 6.40, this ideal line is represented in red. And we can see that the Trefftz-DG solver is the closest to this line when compared with the IPDG solver.

Next, we compare the precision of the numerical solutions obtained with both solvers

MPI nodes	1	2	4	8	16	32
Trefftz-DG	12003	6050	3012	1525	770	389
IPDG	64583	32177	16889	8235	4336	2315

Table 6.8: Comparison of CPU time (s) of both solvers



(a) CPU time in function of the number of MPI nodes

(b) Speed-up in function of the number of MPI nodes

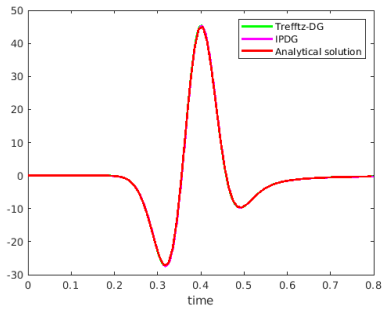
Figure 6.40: Performance of the parallelized Trefftz-DG solver with Tent-Pitching

through seismograms. We place ourselves in the same test case as previously, except

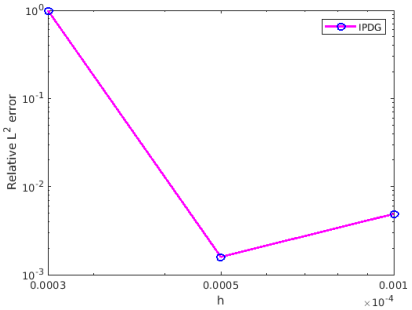
[7] `https://gitlab.inria.fr/hou10ni/hou10ni_dt`

[8] Experiments presented in this section were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see https://www.plafrim.fr).

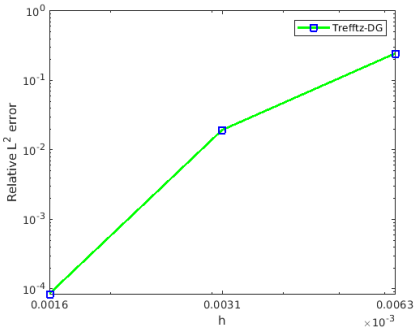
that we use a source term instead of an initial condition and compare the obtained solutions with an analytical solution computed with *Gar6more2D*[78]. We can see in the seismograms in Fig. 6.41a that the numerical solutions are very close to the analytical one and we cannot distinguish them. To further test the accuracy of each method, we compute the relative L^2-error between the numerical and the analytical solutions as we diminish the mesh cell size (see Figs. 6.41b6.41c) and we observe that the Trefftz-DG numerical solution converges faster to the analytical solution when compared with the IPDG numerical solution.



(a) Comparison of seismograms for exact and numerical pressure on structured meshes



(b) Relative L^2-error between IPDG solution and analytical solution



(c) Relative L^2-error between Trefftz-DG solution and analytical solution

Figure 6.41: Tests on the accuracy of the solutions

6.6 Tests on the Trefftz-DG solver on unstructured meshes

As done in the structured case, we compare the performance of the Trefftz-DG solver on unstructured Tent-Pitching meshes with the IPDG solver. To do so, we started with comparing the time spent on 1 process for a mesh composed of 1577 elements, and we obtained that the IPDG solver takes a few seconds to run the simulation for 1 second, whereas the Trefftz-DG solver takes approximately 40 minutes. This time, our code is

much slower than the IPDG one and since the difference in time is so big, we do not present further time results. We have already mentioned that our code on unstructured mesh deserves a thorough optimization work. In particular, it is important to note that we do not have a reference matrix and therefore we have to compute the M and K matrices for each tent at each time step. It is clear that this method induces a very high computational load.

Next, we compare the precision of the numerical solutions obtained with both solvers through seismograms. We place ourselves in the same test case as previously, except that we use a source term instead of an initial condition. We perform simulations for three meshes: the first one with ≈700 elements, the second one with ≈2000 elements and the third one with ≈3000 elements. We can see in Fig. 6.42 that as we refine the mesh, the numerical solution converges to the analytical solution. Although we expect the same results as for the structured case, we unfortunately could not test the accuracy of the solver on very refined meshes. Actually, the presented simulations took between 45 minutes and ≈ 4 hours, which does not encourage to consider finer meshes. Anyway,

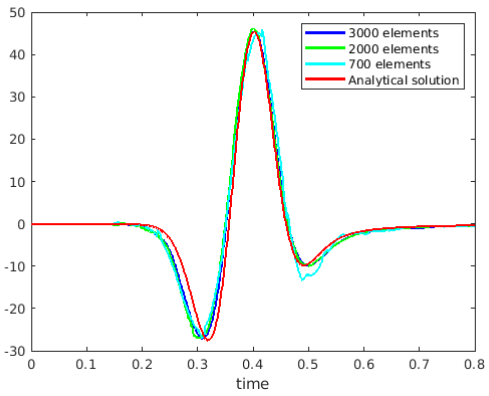


Figure 6.42: Comparison of seismograms for exact and numerical pressure on unstructured meshes

these results illustrate the accuracy of the numerical method and define a proof of concept in favor of optimization to further analyze the performance of the method.

Part II

Perfectly Matched Layers

Chapter 7

Introduction

The reflections induced by a wave hitting the boundaries of a computational domain are a recurring issue in numerical simulations for wave equations. In practice, those reflections are minimized by setting on the boundaries absorbing boundary conditions or by introducing absorbing layers surrounding the domain under study, also called Perfectly Matched Layers. A Perfectly Matched Layer (PML) is an artificial absorbing layer that surrounds the computational domain. It was introduced in 1994 by Bérenger [79] for electromagnetic waves and in theory, it is proved to absorb without any reflection. The wave enters the layer and is strongly absorbed in such a way that its amplitude decays exponentially. Hence, even if the wave reflects at the external boundary of the absorbing layer, the reflected wave is sufficiently attenuated to have no impact on the simulated field when it is back-propagated in the domain of interest. Since the pioneering work of Bérenger, this methods quickly increased in popularity and has been extended to many applications [80, 81, 82, 83, 84, 85, 86]. Error estimates and convergence analysis have been derived for the PMLs applied to different types of problem [87, 88, 89, 90, 91]. We refer to Bérenger's formulation of the PML as the split-field PML. It was then shown that the PML can be interpreted as a complex space coordinate stretching [92, 93] and this remark led to the unsplit PML, also called the Uniaxial PML, which is well-posed on the contrary to the only weakly well-posed split formulation [94]. The Uniaxial PML consists in using a layer of diagonally anisotropic absorbing material, does not involve any modifications on the wave equations and aims at rendering an easier implementation than the original formulation. The Convolutional Perfectly Matched Layer is an improved formulation of the classical PML and in particular, it was introduced to handle the case of grazing incidence, for which the reflection coefficient is not zero and even very large. It was introduced by Roden and Gedney [95] for Maxwell equations and then by Komatitsch and Martin in [84] for the seismic wave equation. Many other formulations of Perfectly Matched Layers exist and are adapted to different kinds of problems, some of them are reviewed and compared in [96].

In the following of this part of the dissertation, we address the idea of introducing

PMLs into Trefftz formulations of wave equations by focusing on the acoustic wave equations and the formulations introduced in Part I. We consider the problem accompanied by the Tent-Pitcher algorithm for spacetime mesh construction on structured meshes. The plan of the second part of this thesis is as follows:

- Chapter 8 **Perfectly Matched Layers with Trefftz-DG Methods**: in this chapter, we present the considered acoustic wave equation with PML in the y-direction and analyze the PML, in particular verify the perfectly matched criterion. Then, we explain how the Trefftz-DG method is derived in this case and introduce multiple variational formulations and their drawbacks and advantages. The construction of basis functions is a key step in the implementation of the Trefftz method. Hence we address the question of constructing polynomial basis functions for the PML formulation of the acoustic wave equation we consider. It turns out that it is not possible to construct polynomials both in time and space, which leads us to consider other basis functions in the next chapter.

- Chapter 9 **Green's functions**: in this chapter, we analytically compute Green's functions for the acoustic wave equation with PML using the Cagniard-De Hoop method. We derive these solutions for PMLs with absorption in the y-direction and also in both directions. We present several possible choices for the basis functions using the computed Green's functions in order to compare the robustness of the associated approximations.

- Chapter 10 **Implementation**: in this chapter, we explain in details how the PML is implemented and display numerical results. We present several cases: the case of a PML in y-direction, the case of PMLs in both directions and the coupling of polynomials in the domain of interest and Green's functions in the PML. This feature turns out to be very interesting since Green's functions require a sensitive tuning related to the use of source-points that are not obvious to locate and numerate.

7.1 Overview on the implementation of PMLs in a classical numerical method

To fix the idea on the PML formulation we intend to consider, let us focus on the acoustic wave equation. As formerly indicated, one widely spread implementation of PML into wave problems reduces to applying a change of variable to the direction in which one

wishes to absorb the wave. Let us recall the first order acoustic wave equation:

$$\begin{cases} \rho\dfrac{\partial v_x}{\partial t} + \dfrac{\partial p}{\partial x} = 0\,, \\ \rho\dfrac{\partial v_y}{\partial t} + \dfrac{\partial p}{\partial y} = 0\,, \\ \dfrac{1}{c^2\rho}\dfrac{\partial p}{\partial t} + \dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y} = 0\,. \end{cases}$$

Following Bérenger [79], we decompose p into $p^x + p^y$ and add absorption in the x- and y-direction in the form of continuous damping functions $\sigma_y(y)$ and $\sigma_x(x)$:

$$\begin{cases} \rho\dfrac{\partial v_x}{\partial t} + \boldsymbol{\sigma_x v_x} + \dfrac{\partial p}{\partial x} = 0\,, \\ \rho\dfrac{\partial v_y}{\partial t} + \boldsymbol{\sigma_y v_y} + \dfrac{\partial p}{\partial y} = 0\,, \\ \dfrac{1}{c^2\rho}\dfrac{\partial p^x}{\partial t} + \boldsymbol{\sigma_x p^x} + \dfrac{\partial v_x}{\partial x} = 0\,, \\ \dfrac{1}{c^2\rho}\dfrac{\partial p^y}{\partial t} + \boldsymbol{\sigma_y p^y} + \dfrac{\partial v_y}{\partial y} = 0\,, \\ p = p^x + p^y\,. \end{cases} \tag{7.1}$$

The system of equations (7.1) is called the split-form of the PML acoustic wave equations, which is the original formulation proposed by Bérenger for Maxwell equations [79]. In order to get rid of p^x and p^y, which are artificial variables with no physical meaning and increase the size of the system, we derive the equations (7.1) with respect to time and retrieve the unsplit-form of the PML acoustic wave equations:

$$\begin{cases} \rho(\dfrac{\partial}{\partial t} + \sigma_x)\dfrac{\partial v_x}{\partial t} + \dfrac{\partial}{\partial t}\dfrac{\partial p}{\partial x} = 0\,, \\ \rho(\dfrac{\partial}{\partial t} + \sigma_y)\dfrac{\partial v_y}{\partial t} + \dfrac{\partial}{\partial t}\dfrac{\partial p}{\partial y} = 0\,, \\ \dfrac{1}{c^2\rho}(\dfrac{\partial}{\partial t} + \sigma_x)\dfrac{\partial p^x}{\partial t} + \dfrac{\partial}{\partial t}\dfrac{\partial v_x}{\partial x} = 0\,, \\ \dfrac{1}{c^2\rho}(\dfrac{\partial}{\partial t} + \sigma_y)\dfrac{\partial p^y}{\partial t} + \dfrac{\partial}{\partial t}\dfrac{\partial v_y}{\partial y} = 0\,. \end{cases} \tag{7.2}$$

We multiply the first and third equations of (7.2) by $(\frac{\partial}{\partial t}+\sigma_x)^{-1}$ and the second and fourth equations by $(\frac{\partial}{\partial t}+\sigma_y)^{-1}$:

$$\begin{cases} \rho\frac{\partial v_x}{\partial t}+(\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}+\boldsymbol{\sigma_x})^{\boldsymbol{-1}}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}\frac{\partial}{\partial t}\frac{\partial p}{\partial x}=0\,, \\ \rho\frac{\partial v_y}{\partial t}+(\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}+\boldsymbol{\sigma_y})^{\boldsymbol{-1}}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}\frac{\partial p}{\partial y}=0\,, \\ \frac{1}{c^2\rho}\frac{\partial p^x}{\partial t}+(\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}+\boldsymbol{\sigma_x})^{\boldsymbol{-1}}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}\frac{\partial v_x}{\partial x}=0\,, \\ \frac{1}{c^2\rho}\frac{\partial p^y}{\partial t}+(\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}+\boldsymbol{\sigma_y})^{\boldsymbol{-1}}\frac{\boldsymbol{\partial}}{\boldsymbol{\partial t}}\frac{\partial v_y}{\partial y}=0\,. \end{cases} \tag{7.3}$$

Now, summing the third and fourth equation of (7.3) and using the fact that $p=p^x+p^y$, we obtain:

$$\begin{cases} \rho\frac{\partial v_x}{\partial t}+(\frac{\partial}{\partial t}+\sigma_x)^{-1}\frac{\partial p}{\partial x}=0\,, \\ \rho\frac{\partial v_y}{\partial t}+(\frac{\partial}{\partial t}+\sigma_y)^{-1}\frac{\partial}{\partial t}\frac{\partial p}{\partial y}=0\,, \\ \frac{1}{c^2\rho}\frac{\partial p}{\partial t}+(\frac{\partial}{\partial t}+\sigma_x)^{-1}\frac{\partial v_x}{\partial x}+(\frac{\partial}{\partial t}+\sigma_y)^{-1}\frac{\partial}{\partial t}\frac{\partial v_y}{\partial y}=0\,. \end{cases} \tag{7.4}$$

Thus, we can see that adding PML to the problem reduces to applying the following variable stretch

$$\frac{\partial}{\partial x}\hookrightarrow(\frac{\partial}{\partial t}+\sigma_x)^{-1}\frac{\partial}{\partial t}\frac{\partial}{\partial x}\,,\quad \frac{\partial}{\partial y}\hookrightarrow(\frac{\partial}{\partial t}+\sigma_y)^{-1}\frac{\partial}{\partial t}\frac{\partial}{\partial y}$$

Looking at (7.4) and assuming that the initial data at $t=0$ are null, we can see that when $\sigma_x=\sigma_y=0$, we retrieve the acoustic wave equations without PML. Hence, denoting the PML parallel to the y-axis as Ω_x^{PML} and the PML parallel to the x-axis as Ω_y^{PML}, the damping functions are usually assumed as follows:

$$\begin{aligned} &\sigma_x>0\,,\quad (x,y)\in\Omega_x^{\mathrm{PML}}\,,\\ &\sigma_x=0\,,\quad (x,y)\in\Omega\setminus\Omega_x^{\mathrm{PML}}\,,\\ &\sigma_y>0\,,\quad (x,y)\in\Omega_y^{\mathrm{PML}}\,,\\ &\sigma_y=0\,,\quad (x,y)\in\Omega\setminus\Omega_y^{\mathrm{PML}}\,. \end{aligned}$$

This means that we do not derive different systems of equations for the domain and the PMLs, but we rather use the same equations (7.4) everywhere with varying damping functions depending on whether we are in the PMLs or not. Many works have been devoted to the optimization of the damping function, which aim at reducing the reflections, increasing the absorption and optimizing the PML width. In fact, even though the layers are perfectly matched in theory, in practice the discretization always induces reflections, more particularly at the domain-PML interface when the damping function

becomes very large. Using a Finite Difference method, Collino *et al.* proposed in [93] a method to optimize the value of σ in each cell of the grid, but this optimization is strongly dependent on the numerical schemes and on the physical parameters. A more general definition of the damping function has been proposed in [85] by choosing σ as $\sigma(x) := \sigma_0(x - x_0)^2$. This definition has been used by many authors (see for instance [97]). In [98], Bermúdez *et al.* have proposed to consider a damping function whose integral is unbounded. In [99], Modave *et al.* have compared the different choices using various numerical methods (finite differences, finite volumes, continuous finite elements and discontinuous finite elements) and have concluded that the unbounded choice was the most efficient. Here, we use the most simple choice and consider the case of a damping coefficient constant in the layer.

In order to illustrate how PMLs work with a standard numerical method, we solve (7.4) with a staggered Finite Difference Time Domain method and present the results we obtain in Table 7.1. We consider a domain of size $L_x = L_y = 1$. We choose the damping functions as follows: $\sigma_x(x) = \dfrac{d_x}{w}\sigma_x^{\max}, \sigma_y(y) = \dfrac{d_y}{w}\sigma_y^{\max}$, where w is the width of the PML, $\sigma^{\max}$ is the maximum absorption and d_m is the distance from m to the inner PML boundary. Thus, the absorption will be very small near the domain-PML interface and very large towards the boundary of the domain. In Table 7.1, we present the obtained results. The first row of the table represents the pressure of the acoustic wave equation solved with the FDTD without considering any PML with homogeneous Neumann boundary conditions. We can see the reflections at the boundaries. The second row of the table presents a domain surrounded with absorbing layers of width 0.2 and a maximal damping of $\sigma^{\max} = \sigma_x^{\max} = \sigma_y^{\max} = 50$. We can see that the wave is absorbed as it goes through the layer, however we see here that even though the absorbing layers are perfectly matched in theory, in practice discretization induces reflections as can be seen in the results. Finally, the third row presents a domain surrounded with absorbing layers of width 0.1 and a maximal damping of $\sigma^{\max} = 100$. These results help us understand how the PML works in practice and we observe that it acts like a sponge absorbing the wave as it propagates through the layer.

In the following, we focus on introducing PMLs in the Trefftz-DG solver. We will see that this work raises several new questions at the mathematical level which transforms the implementation of PMLs into a much more technical task.

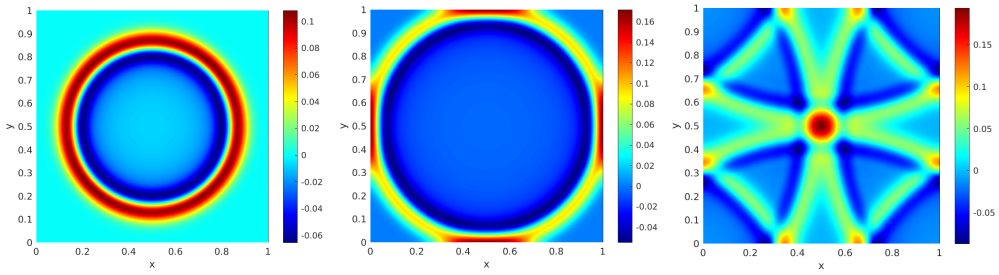
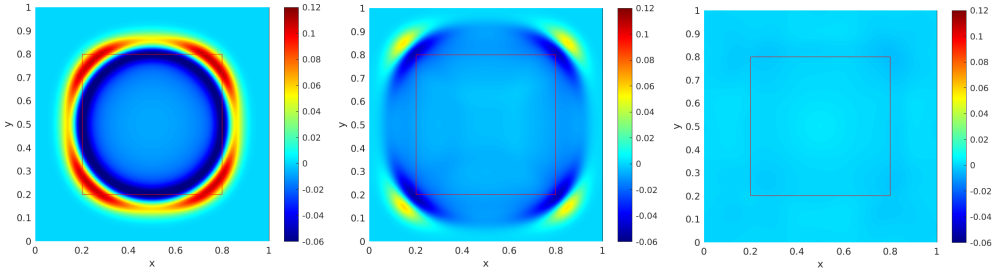
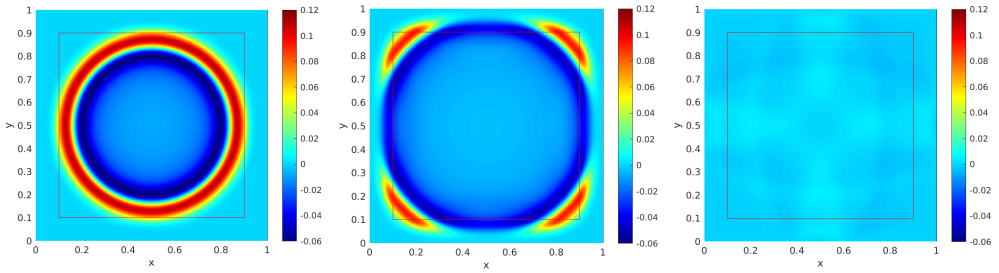
w , $\sigma^{\max}$	
0 , 0	
0.2 , 50	
0.1 , 100	

Table 7.1: Pressure for varying width and $\sigma^{\max}$

Chapter 8

Perfectly Matched Layers with Trefftz-DG methods

In this chapter, we formulate the Trefftz approximation of the acoustic system considered in Chapter Trefftz-DG Method for Wave equations when adding Perfectly Matched Layers to truncate the computational domain. Here again, the formulation includes the Tent-Pitcher algorithm. First, we consider one absorbing layer parallel to the x-axis as shown in Fig. 8.1.

We will first introduce the acoustic wave equation with one PML in the y-direction and derive the corresponding formulation along with the appropriate functional spaces. Then, we will present some other variational formulations motivated by the observations we made when performing numerical experiments. Actually, it turns out that depending on the variational formulation we choose, the solution is not guaranteed to converge to the exact solution.

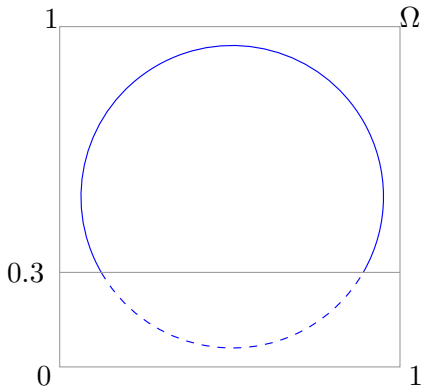


Figure 8.1: Considered domain with one PML

We proceed in the same manner is previously, except that we only consider a PML in the y-direction here. Thus, considering (7.4) with $\sigma_x = 0$, we obtain the acoustic wave

equations with a PML in the y-direction:

$$\begin{cases} \rho\dfrac{\partial v_x}{\partial t} + \dfrac{\partial p}{\partial x} = 0\,, \\ \rho\dfrac{\partial v_y}{\partial t} + (\dfrac{\partial}{\partial t} + \sigma_y)^{-1}\dfrac{\partial}{\partial t}\dfrac{\partial p}{\partial y} = 0\,, \\ \dfrac{1}{c^2\rho}\dfrac{\partial p}{\partial t} + \dfrac{\partial v_x}{\partial x} + (\dfrac{\partial}{\partial t} + \sigma_y)^{-1}\dfrac{\partial}{\partial t}\dfrac{\partial v_y}{\partial y} = 0\,. \end{cases} \tag{8.1}$$

After multiplying the second and third equations by $(\dfrac{\partial}{\partial t} + \sigma_y)$ in order to avoid having differential operators in the denominator, we obtain the acoustic wave equations with PML in the y-direction, that we will be working with:

$$\begin{cases} \rho\dfrac{\partial v_x}{\partial t} + \dfrac{\partial p}{\partial x} = 0\,, \\ \rho(\dfrac{\partial}{\partial t} + \sigma_y)\dfrac{\partial v_y}{\partial t} + \dfrac{\partial}{\partial t}\dfrac{\partial p}{\partial y} = 0\,, \\ \dfrac{1}{c^2\rho}(\dfrac{\partial}{\partial t} + \sigma_y)\dfrac{\partial p}{\partial t} + (\dfrac{\partial}{\partial t} + \sigma_y)\dfrac{\partial v_x}{\partial x} + \dfrac{\partial}{\partial t}\dfrac{\partial v_y}{\partial y} = 0\,. \end{cases} \tag{8.2}$$

8.1 Perfect transmission

To show that our layers are perfectly matched, we need to verify that there are no reflections at the interface between the domain and the PML, hence having a perfect transmission of the solution. This is one of the important characteristics of such absorbing layers.

Bérenger in [79] and Halpern *et al.* in [91] analyze the PML in other frameworks and the perfectly matched character is proved based on the calculation of the reflection coefficient. Here, to verify that our layers are perfectly matched, we will also show that reflection coefficients are null, but we will carry it out by computing Green's functions at the interface between a domain and its PML, instead of plane waves as in [79, 91]. To do so, let us consider a domain with a PML defined in the bottom half-plane for simplicity as in Fig. 8.2, although the approach and the result are the same if we consider a differently positioned PML. We will use the following decomposition of the solution:

$$\begin{aligned} p &= p^i + p^r\,, \quad && y > 0 \\ p &= p^t\,, && y < 0 \end{aligned}$$

Hence, we will write the Green's functions for the incident wave p^i, the reflected wave p^r and the transmitted wave p^t we would have at the interface, and finally show that the reflection coefficient is null.

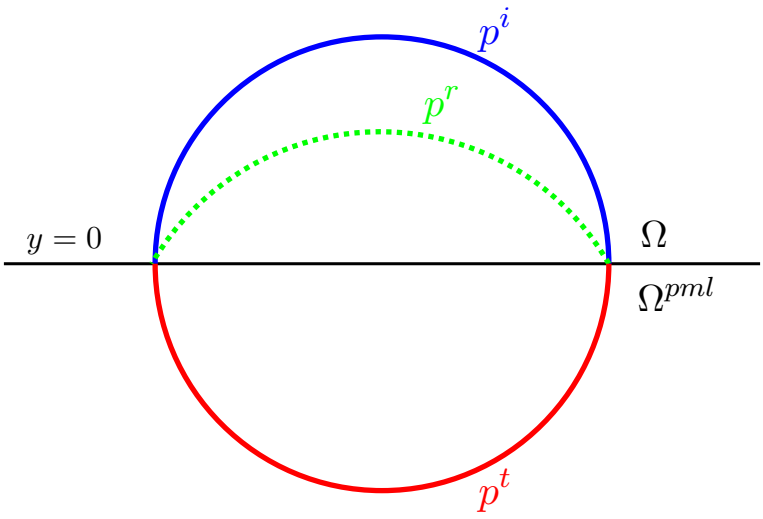


Figure 8.2: Incident, reflected and transmitted wavefronts

Let us consider the second-order wave equations. We assume that the incident wave is generated by a point source at a height h. The three waves satify the following three wave equations and two transmissions conditions at the interface $y = 0$:

$$\begin{cases} \dfrac{1}{c^2}\dfrac{\partial^2 p^i}{\partial t^2} - \dfrac{\partial^2 p^i}{\partial x^2} - \dfrac{\partial^2 p^i}{\partial y^2} = \delta(x)\delta(y-h)\delta(t)\,, & y > 0\,, \\ \dfrac{1}{c^2}\dfrac{\partial^2 p^r}{\partial t^2} - \dfrac{\partial^2 p^r}{\partial x^2} - \dfrac{\partial^2 p^r}{\partial y^2} = 0\,, & y > 0\,, \\ \dfrac{1}{c^2}\dfrac{\partial^2 p^t}{\partial t^2} - \dfrac{\partial^2 p^t}{\partial x^2} - \left(\dfrac{\partial t}{\partial t + \sigma_y}\right)^2 \dfrac{\partial^2 p^t}{\partial y^2} = 0\,, & y < 0\,, \\ p^i + p^r = p^t\,, & y = 0\,, \\ \dfrac{\partial p^i}{\partial y} + \dfrac{\partial p^r}{\partial y} = \dfrac{\partial t}{\partial t + \sigma_y}\dfrac{\partial p^t}{\partial y}\,, & y = 0\,. \end{cases} \tag{8.3}$$

Here, the transmitted wave is located in the PML, which explains the change of variable depicted in the third and fifth equation of (8.3), whereas the incident and reflected waves are located in the domain of interest.

Then, we perform a Laplace transform on p:

$$\mathcal{L}[p](s) = \tilde{p}(x, y, s)$$

and a partial Fourier transform along x on $\tilde{p}$:

$$\mathcal{F}_x[\tilde{p}](k) = \hat{p}(k, y, s)$$

which gives us the following system:

$$\begin{cases} -\dfrac{\partial^2 \hat{p}^i}{\partial y^2} + (k^2 + \dfrac{s^2}{c^2})\hat{p}^i = \delta(y-h)\,, & y > 0\,, \\ -\dfrac{\partial^2 \hat{p}^r}{\partial y^2} + (k^2 + \dfrac{s^2}{c^2})\hat{p}^r = 0\,, & y > 0\,, \\ -\left(\dfrac{s}{s+\sigma_y}\right)^2 \dfrac{\partial^2 \hat{p}^t}{\partial y^2} + (k^2 + \dfrac{s^2}{c^2})\hat{p}^t = 0\,, & y < 0\,, \\ \hat{p}^i + \hat{p}^r = \hat{p}^t\,, & y = 0\,, \\ \dfrac{\partial \hat{p}^i}{\partial y} + \dfrac{\partial \hat{p}^r}{\partial y} = \dfrac{s}{s+\sigma_y}\dfrac{\partial \hat{p}^t}{\partial y}\,, & y = 0\,. \end{cases}$$

The solutions to these equations can be computed explicitly and are given by:

$$\begin{cases} \hat{p}^i(k,y,s) = \dfrac{e^{-|y-h|(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} \\ \hat{p}^r(k,y,s) = \mathcal{R}(k,s)e^{-y\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} \\ \hat{p}^t(k,y,s) = \mathcal{T}(k,s)e^{y(1+\frac{\sigma_y}{s})\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} \end{cases}$$

$\mathcal{R}$ and $\mathcal{T}$ will be called the reflection and transmission coefficient, respectively. As we are considering the problem at the interface between the domain of interest and the PML, we have $y - h < 0$. Thus, the incident wave solution can be rewritten as follows:

$$\hat{p}^i(k,y,s) = \frac{e^{(y-h)(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}}$$

We now inject these solutions into the transmission conditions, which are the fourth and fifth equation of (8.3), which results in:

$$p^i + p^r = p^t \iff \frac{e^{-h(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} + \mathcal{R}(k,s) = \mathcal{T}(k,s)\,,$$

$$\frac{\partial p^i}{\partial y} + \frac{\partial p^r}{\partial y} = \frac{\partial t}{\partial t + \sigma_y}\frac{\partial p^t}{\partial y} \iff \frac{e^{-h(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2} - \mathcal{R}(k,s)\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}} = \mathcal{T}(k,s)\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}} \tag{8.4}$$

Notice that equations (8.4) are evaluated at $y = 0$, since the interface is located there. Hence, we obtain:

$$\mathcal{R} = 0\,,$$
$$\mathcal{T} = \frac{e^{-h(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}}\,.$$

Thus, we showed that there is no reflection between the domain and the PML for the pressure computed as a solution to the second-order acoustic wave equation with PML. Regarding the velocity, we have the following equation for the reflected field:

$$\rho \frac{\partial \mathbf{v}^r}{\partial t} + \nabla p^r = 0$$

If we perform a Laplace transform followed by a partial Fourier transform along x to $\mathbf{v}$ and p, we obtain the following:

$$\begin{cases} \rho s \widehat{v_x} - ik\hat{p} = 0\,, \\ \rho s \widehat{v_y} + \dfrac{\partial \hat{p}}{\partial y} = 0\,. \end{cases}$$

Since we verified that $\mathcal{R} = 0$, as a consequence we obtain $\mathbf{v}^r = 0$ as well. Thus, we showed that our absorbing layer is indeed perfectly matched, as we retrieve zero reflections at the interface between the domain and the PML, for all wavefields.

Remark. *The absorbing layers are indeed perfectly matched when considering the continuous equations. However, when we turn to numerical computations, we introduce approximations of the solutions. This means that there can be reflections, although not strong ones, since we assume the approximations are close enough to the continuous solutions. Moreover, the remaining reflections can be minimized depending on the choice of the damping function $\sigma_y(y)$.*

8.2 Variational formulation without auxiliary variables

The system (8.2) does not involve any additional variables, hence it will be called the system without auxiliary variables. It is worth noting that it is quite frequent that PML formulations involve auxiliary variables, which can increase the associated computational costs. Before determining our PML system's variational formulation, we need to define new functional spaces and a Trefftz space.

Functional spaces

By operating in the same way as in Chapter Trefftz-DG Method for Wave equations, we consider a local variational formulation on a spacetime element K:

$$\begin{aligned} \frac{1}{c^2\rho}\int_K (\frac{\partial}{\partial t} + \sigma_y)\frac{\partial p}{\partial t}\boldsymbol{q} + \int_K (\frac{\partial}{\partial t} + \sigma_y)\frac{\partial v_x}{\partial x}\boldsymbol{q} + \int_K \frac{\partial}{\partial t}\frac{\partial v_y}{\partial y}\boldsymbol{q} &= \int_K f\boldsymbol{q} \\ \rho\int_K \frac{\partial v_x}{\partial t}\boldsymbol{w_x} + \int_K \frac{\partial p}{\partial x}\boldsymbol{w_x} &= 0 \\ \rho\int_K (\frac{\partial}{\partial t} + \sigma_y)\frac{\partial v_y}{\partial t}\boldsymbol{w_y} + \int_K \frac{\partial}{\partial t}\frac{\partial p}{\partial y}\boldsymbol{w_y} &= 0 \end{aligned} \tag{8.5}$$

Here, $(q, \mathbf{w})$ are test functions and their functional spaces will be defined in the following.

The minimal requirement to get a well-defined problem is to ensure that each integral in (8.5) is finite. This leads to searching for solutions such that:

$$\frac{\partial p}{\partial t} \in L^2(K)\,,\ \frac{\partial^2 p}{\partial t^2} \in L^2(K)\,,\ \frac{\partial p}{\partial x} \in L^2(K)\,,\ \frac{\partial^2 p}{\partial t \partial y} \in L^2(K)\,,$$
$$\frac{\partial \mathbf{v}}{\partial t} \in (L^2(K))^d\,,\ \frac{\partial}{\partial t}\operatorname{div}\mathbf{v} \in L^2(K)\,,\ \frac{\partial^2 v_y}{\partial t^2} \in L^2(K)\,.$$

We can observe that compared to the original system, the PML system assumes that the solution is more regular. Actually, we consider that the time derivative of the solution satisfies the same regularity properties as the wavefield solutions to the system without PML. Hence, we introduce V and $\mathbf{V}$:

$$V = \Big\{ p \in L^2(\Omega) \mid p \in H^2(K) \Big\}$$
$$\mathbf{V} = \Big\{ \mathbf{v} \in (L^2(\Omega))^d \mid \mathbf{v} \in (H^2(K))^d \Big\}$$
$$H^2(K) = \Big\{ \Phi \in L^2(K), \partial^\alpha \Phi \in L^2(K), \forall \alpha \in \mathbb{N}^d, |\alpha| \leq 2 \Big\}$$

As a consequence, (8.5) is well-defined if we assume $p \in V$ and $\mathbf{v} \in \mathbf{V}$.

Now that we properly defined the functional spaces, it remains to introduce the Trefftz space in which the basis functions will be chosen. As has already been explained before, the idea of the Trefftz method is to take particular solutions as basis functions. Here, the equations are different than previously, hence the Trefftz space will also differ. According to the system we want to solve, our Trefftz space can be defined as:

$$\mathbf{T} = \Big\{ (\mathbf{v}, p) \in \mathbf{V} \times V, \text{ such that } (\mathbf{v}, p) \text{ is solution to (8.2)} \}$$

In the following, we present different variational formulations that we have experienced numerically and provided us with different solutions.

8.2.1 Variational formulation A

To obtain the variational formulation, let us sum all equations of (8.5) and integrate by parts:

$$\begin{aligned}
&\int_K \left[\rho(\frac{\partial}{\partial t} - \sigma_y)\frac{\partial w_y}{\partial t} + \frac{\partial}{\partial t}\frac{\partial q}{\partial y}\right] v_y - \left[\rho\frac{\partial w_x}{\partial t} - (\frac{\partial}{\partial t} - \sigma_y)\frac{\partial q}{\partial x}\right] v_x \\
&\quad + \left[\frac{1}{c^2\rho}(\frac{\partial}{\partial t} - \sigma_y)\frac{\partial q}{\partial t} - \frac{\partial w_x}{\partial x} + \frac{\partial}{\partial t}\frac{\partial w_y}{\partial y}\right] p \\
&+ \int_{\partial K} -\rho v_y \frac{\partial w_y}{\partial t} n_t + \rho(\frac{\partial}{\partial t} + \sigma_y) v_y w_y n_t + \frac{\partial p}{\partial t} w_y n_y - p\frac{\partial w_y}{\partial y} n_t \\
&+ \int_{\partial K} \rho v_x w_x n_t + p w_x n_x \\
&+ \int_{\partial K} \frac{1}{c^2\rho}(\frac{\partial}{\partial t} + \sigma_y) p q n_t - \frac{1}{c^2\rho} p \frac{\partial q}{\partial t} n_t - \mathbf{v}\cdot\nabla q\, n_t + (\frac{\partial}{\partial t} + \sigma_y) v_x q n_x + \frac{\partial v_y}{\partial t} q n_y = 0
\end{aligned}$$

One of the strengths of Trefftz formulations for wave equations is that the volumic terms vanish, because we take $(q, \mathbf{w})$ in $\mathbf{T}$, *i.e.*, the test functions are local solutions to the problem of interest and we can apply a sort of reciprocity principle. However, here we can see that it is not the case, because the PML formulation introduces second-order time derivatives, which results in modifying the expressions involving the test functions. In a perfect world, we would have a plus sign before σ_y and then, according to the definition of the Trefftz space $\mathbf{T}$, the volumic terms would vanish. We are recovering the fact that when considering wave equations with attenuation, the reciprocity principle does not apply. In fact for the PML formulation, in order to keep a formulation only defined on the skeleton of the mesh, we should consider test functions satisfying the following problem:

$$\begin{cases} \rho \dfrac{\partial w_x}{\partial t} - (\dfrac{\partial}{\partial t} - \sigma_y) \dfrac{\partial q}{\partial x} = 0 \,, \\ \rho \dfrac{\partial w_y}{\partial t} + (\dfrac{\partial}{\partial t} - \sigma_y)^{-1} \dfrac{\partial}{\partial t} \dfrac{\partial q}{\partial y} = 0 \,, \\ \dfrac{1}{c^2 \rho} \dfrac{\partial q}{\partial t} - (\dfrac{\partial}{\partial t} - \sigma_y)^{-1} \dfrac{\partial w_x}{\partial x} + (\dfrac{\partial}{\partial t} - \sigma_y)^{-1} \dfrac{\partial}{\partial t} \dfrac{\partial w_y}{\partial y} = 0 \,. \end{cases}$$

In order to simplify these equations as much as possible, we introduce a change of variable $w_x = -(\partial_t - \sigma_y)\tilde{w_x}$. By doing so, we obtain a system of equations that is closer to the initial one:

$$\begin{cases} \rho \dfrac{\partial \tilde{w_x}}{\partial t} + \dfrac{\partial q}{\partial x} = 0 \,, \\ \rho \dfrac{\partial w_y}{\partial t} + (\dfrac{\partial}{\partial t} - \sigma_y)^{-1} \dfrac{\partial}{\partial t} \dfrac{\partial q}{\partial y} = 0 \,, \\ \dfrac{1}{c^2 \rho} \dfrac{\partial q}{\partial t} + \dfrac{\partial \tilde{w_x}}{\partial x} + (\dfrac{\partial}{\partial t} - \sigma_y)^{-1} \dfrac{\partial}{\partial t} \dfrac{\partial w_y}{\partial y} = 0 \,, \\ w_x = -(\partial_t - \sigma_y)\widetilde{w_x} \,. \end{cases} \tag{8.6}$$

Following this, let us then define the suitable test function space:

$$\tilde{\mathbf{T}}(T_h) = \Big\{ (\mathbf{w}, q) \in \mathbf{V} \times V, \text{ such that } (\mathbf{w}, q) \text{ is solution to (8.6)} \Big\}$$

Hence, the generalized local Trefftz formulation of the PML problem (8.2) reads:

Generalized Trefftz Variational Formulation

Seek $(\mathbf{v}, p) \in \mathbf{T}$*, such that for all* $(\mathbf{w}, q) \in \tilde{\mathbf{T}}$ *and for all* K*, it holds true:*

$$\begin{aligned}
&\int_{\partial K} -\rho v_y \frac{\partial w_y}{\partial t} n_t + \rho(\frac{\partial}{\partial t} + \sigma_y) v_y w_y n_t + \frac{\partial p}{\partial t} w_y n_y - p \frac{\partial w_y}{\partial y} n_t \\
&+ \int_{\partial K} \rho v_x w_x n_t + p w_x n_x \\
&+ \int_{\partial K} \frac{1}{c^2 \rho} (\frac{\partial}{\partial t} + \sigma_y) p q n_t - \frac{1}{c^2 \rho} p \frac{\partial q}{\partial t} n_t - \mathbf{v} \cdot \nabla q \, n_t + (\frac{\partial}{\partial t} + \sigma_y) v_x q n_x + \frac{\partial v_y}{\partial t} q n_y \\
&= 0
\end{aligned} \tag{8.7}$$

Notice that the volumic terms have indeed vanished and we are left with boundary integrals.

Numerical fluxes

Now that we have defined the local Trefftz variational formulation, we move on to the discretization of the domain Ω. Let T_h be a triangulation of Ω composed of non-overlapping spacetime elements K. To complete our DG discretization, it remains to define the numerical fluxes, which connect the elements together and are chosen as follows:

$$\begin{aligned}
\mathbf{v} &= \mathbf{v} && \text{on } \mathcal{F}^{out}, \\
\begin{pmatrix} \hat{\mathbf{v}} \\ \dfrac{\partial \check{\mathbf{v}}}{\partial t} \end{pmatrix} &= \begin{pmatrix} \{\!\{\mathbf{v}\}\!\} + \beta_1 [\![p]\!]_{\mathbf{x}} \\ \left\{\!\!\left\{ \dfrac{\partial \mathbf{v}}{\partial t} \right\}\!\!\right\} + \beta_3 \left[\!\!\left[\dfrac{\partial p}{\partial t} \right]\!\!\right]_{\mathbf{x}} \end{pmatrix} && \text{on } \mathcal{F}^{int}, \\
\begin{pmatrix} \mathbf{v} \\ \dfrac{\partial \hat{\mathbf{v}}}{\partial t} \end{pmatrix} &= \begin{pmatrix} (0.5 - \alpha_2)\mathbf{v} + (0.5 + \alpha_2)\mathbf{v}^0 \\ (0.5 - \alpha_4) \dfrac{\partial \mathbf{v}}{\partial t} + (0.5 + \alpha_4) \dfrac{\partial \mathbf{v}^0}{\partial t} \end{pmatrix} && \text{on } \mathcal{F}^{in}, \\
\begin{pmatrix} \hat{\mathbf{v}} \cdot \mathbf{n}_{\mathbf{x}} \\ \dfrac{\partial \check{\mathbf{v}}}{\partial t} \cdot \mathbf{n}_{\mathbf{x}} \end{pmatrix} &= \begin{pmatrix} g_D \\ \dfrac{\partial g_D}{\partial t} \end{pmatrix} && \text{on } \mathcal{F}^{ext},
\end{aligned}$$

$$
\begin{aligned}
p &= p && \text{on } \mathcal{F}^{out}, \\
\begin{pmatrix} \hat{p} \\ \dfrac{\partial \check{p}}{\partial t} \end{pmatrix} &= \begin{pmatrix} \{\!\{p\}\!\} + \alpha_1 [\![\mathbf{v}]\!]_{\mathbf{x}} \\ \left\{\!\!\left\{ \dfrac{\partial p}{\partial t} \right\}\!\!\right\} + \alpha_3 \left[\!\!\left[\dfrac{\partial \mathbf{v}}{\partial t} \right]\!\!\right]_{\mathbf{x}} \end{pmatrix} && \text{on } \mathcal{F}^{int}, \\
\begin{pmatrix} p \\ \dfrac{\partial \hat{p}}{\partial t} \end{pmatrix} &= \begin{pmatrix} (0.5-\beta_2)p + (0.5+\beta_2)p^0 \\ (0.5-\beta_4)\dfrac{\partial p}{\partial t} + (0.5+\beta_4)\dfrac{\partial p^0}{\partial t} \end{pmatrix} && \text{on } \mathcal{F}^{in}, \\
\begin{pmatrix} \hat{p} \\ \dfrac{\partial \check{p}}{\partial t} \end{pmatrix} &= \begin{pmatrix} p + \alpha_1(\mathbf{v}\cdot\mathbf{n}_{\mathbf{x}} - g_D) \\ \dfrac{\partial p}{\partial t} + \alpha_3 \left(\dfrac{\partial \mathbf{v}}{\partial t}\cdot\mathbf{n}_{\mathbf{x}} - \dfrac{\partial g_D}{\partial t} \right) \end{pmatrix} && \text{on } \mathcal{F}^{ext}.
\end{aligned}
$$

where α_1, α_2, α_3 and α_4 are penalty coefficients, taken between 0 and 1.

Since we are working with Tent-Pitching meshes, we can see that the fluxes are defined on a Tent-Pitching cell as introduced in Part I. Indeed, a Tent-Pitching cell is composed of an inflow and outflow boundary, and in some cases can have an external and/or an internal boundary. In classical DG, the considered boundaries differ from those in Tent-Pitching and so do the fluxes. We then inject these fluxes into (8.7) and hence obtain the following **Trefftz-DG PML variational formulation on Tent-Pitched meshes**, that we will more simply refer to as "Variational formulation A":

PML Trefftz-DG Variational Formulation with Tent-Pitching

Seek $(\mathbf{v}, p) \in \mathbf{T}$, *such that for all* $(\mathbf{w}, q) \in \tilde{\mathbf{T}}$, *it holds true:*

$$\mathcal{A}(\mathbf{v}, p; \mathbf{w}, q) = l(\mathbf{w}, q)$$

where

$$
\begin{aligned}
l(\mathbf{w}, q) := &- \int_{\mathcal{F}^{in}} (0.5+\alpha)\Big(- \rho {v_y}^{in} \frac{\partial w_y}{\partial t} + \rho(\frac{\partial}{\partial t} + \sigma_y){v_y}^{in} w_y - p^{in}\frac{\partial w_y}{\partial y} + \rho {v_x}^{in} w_x \Big) n_t \\
&+ (0.5+\beta)\Big(\frac{1}{c^2\rho}(\frac{\partial}{\partial t} + \sigma_y) p^{in} q - \frac{1}{c^2\rho} p^{in} \frac{\partial q}{\partial t} - \mathbf{v}^{in}\cdot\nabla q \Big) n_t \\
&- \frac{1}{2}\int_{\mathcal{F}^{in}} \frac{\partial p^{in}}{\partial t} w_y n_y + p^{in} w_x n_x + (\frac{\partial \mathbf{v}^{in}}{\partial t}\cdot\mathbf{n}) q + \sigma_y {v_x}^{in} q n_x \,.
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{A}(\mathbf{v},p;\mathbf{w},q) := \int_{\mathcal{F}^{out}} & -\rho v_y \frac{\partial w_y}{\partial t} n_t + \rho(\frac{\partial}{\partial t} + \sigma_y) v_y w_y n_t + \frac{\partial p}{\partial t} w_y n_y - p \frac{\partial w_y}{\partial y} n_t \\
& + \rho v_x w_x n_t + p w_x n_x + \frac{1}{c^2 \rho}(\frac{\partial}{\partial t} + \sigma_y) p q n_t \\
& - \frac{1}{c^2 \rho} p \frac{\partial q}{\partial t} n_t - \mathbf{v} \cdot \nabla q \, n_t + (\frac{\partial \mathbf{v}}{\partial t} \cdot \mathbf{n}) q + \sigma_y v_x q n_x \\
+ \int_{\mathcal{F}^{in}} & (0.5 - \alpha) \left(-\rho v_y \frac{\partial w_y}{\partial t} + \rho(\frac{\partial}{\partial t} + \sigma_y) v_y w_y - p \frac{\partial w_y}{\partial y} + \rho v_x w_x \right) n_t \\
& + (0.5 - \beta) \left(\frac{1}{c^2 \rho}(\frac{\partial}{\partial t} + \sigma_y) p q - \frac{1}{c^2 \rho} p \frac{\partial q}{\partial t} - \mathbf{v} \cdot \nabla q \right) n_t \\
+ \frac{1}{2} \int_{\mathcal{F}^{in}} & \frac{\partial p}{\partial t} w_y n_y + p w_x n_x + (\frac{\partial \mathbf{v}}{\partial t} \cdot \mathbf{n}) q + \sigma_y v_x q n_x \\
+ \int_{\mathcal{F}^{ext}} & \frac{\partial p}{\partial t} w_y n_y + \gamma(\frac{\partial \mathbf{v}}{\partial t} \cdot \mathbf{n}) w_y n_y + p w_x n_x + \gamma(\mathbf{v} \cdot \mathbf{n}) w_x n_x + \sigma_y v_x q n_x \\
+ \int_{\mathcal{F}^{int}} & \left\{ \frac{\partial p}{\partial t} \right\} [\![w_y]\!]_y + \alpha \left[\!\!\left[\frac{\partial \mathbf{v}}{\partial t} \right]\!\!\right]_{\mathbf{x}} [\![w_y]\!]_y + \left\{ \frac{\partial \mathbf{v}}{\partial t} \right\} [\![q]\!]_{\mathbf{x}} + \beta \left[\!\!\left[\frac{\partial p}{\partial t} \right]\!\!\right]_{\mathbf{x}} [\![q]\!]_{\mathbf{x}} \\
& + \{\!\{ p \}\!\} [\![w_x]\!]_x + \alpha [\![\mathbf{v}]\!]_{\mathbf{x}} [\![w_x]\!]_x + \sigma_y \{\!\{ v_x \}\!\} [\![q]\!]_x + \sigma_y \beta [\![p]\!]_x [\![q]\!]_x
\end{aligned}
$$

Drawbacks and Advantages

We can already see some drawbacks for this variational formulation, which encouraged us to look for a different one. Indeed, we need to define a new space $\tilde{\mathbf{T}}$ for the test functions on top of performing a change of variable. This means that we have additional computations to do since we have two sets of functions to determine: one for the basis functions and the other for the test functions, as it is done for instance in Petrov-Galerkin method. Moreover, we can see that the variational formulation has become drastically more complex with many derivatives that we did not have in the formulation without PML, which was is and advantageous characteristic of the Trefftz-DG formulation. In Appendix D, we can see that the expressions of these derivatives are not simple and they increase calculations and burden the comprehension. Finally, using the formulation A, we obtained bad results with a solution which would blow up; they will be presented in the Chapter Implementation. To overcome these drawbacks, we decided to find another variational formulation with auxiliary variables.

8.3 Variational formulation with auxiliary variable

To introduce our auxiliary variables $(p^a, {v_y}^a) \in L^2(\Omega) \times L^2(\Omega)$, let us go back to (8.1). We can see that $\dfrac{\partial_t}{\partial_t + \sigma_y} = 1 - \dfrac{\sigma_y}{\partial_t + \sigma_y}$. So,

$$\begin{cases} \rho \dfrac{\partial v_x}{\partial t} + \dfrac{\partial p}{\partial x} = 0\,, \\ \rho \dfrac{\partial v_y}{\partial t} + \dfrac{\partial p}{\partial y} - \sigma_y (\dfrac{\partial}{\partial t} + \sigma_y)^{-1} \dfrac{\partial p}{\partial y} = 0\,, \\ \dfrac{1}{c^2 \rho} \dfrac{\partial p}{\partial t} + \dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y} - \sigma_y (\dfrac{\partial}{\partial t} + \sigma_y)^{-1} \dfrac{\partial v_y}{\partial y} = 0\,. \end{cases}$$

Hence, setting $p^a = \sigma_y (\dfrac{\partial}{\partial t} + \sigma_y)^{-1} \dfrac{\partial p}{\partial y}$ and ${v_y}^a = \sigma_y (\dfrac{\partial}{\partial t} + \sigma_y)^{-1} \dfrac{\partial v_y}{\partial y}$ we obtain our PML equations with auxiliary variables:

$$\begin{cases} \rho \dfrac{\partial v_x}{\partial t} + \dfrac{\partial p}{\partial x} = 0\,, \\ \rho \dfrac{\partial v_y}{\partial t} + \dfrac{\partial p}{\partial y} - p^a = 0\,, \\ \dfrac{1}{c^2 \rho} \dfrac{\partial p}{\partial t} + \dfrac{\partial v_x}{\partial x} + \dfrac{\partial v_y}{\partial y} - {v_y}^a = 0\,, \\ (\dfrac{\partial}{\partial t} + \sigma_y) p^a = \sigma_y \dfrac{\partial p}{\partial y}\,, \\ (\dfrac{\partial}{\partial t} + \sigma_y) {v_y}^a = \sigma_y \dfrac{\partial v_y}{\partial y}\,. \end{cases} \tag{8.8}$$

We can see that in these new equations, our solution dimension has increased by two and is $(p, \mathbf{v}, p^a, {v_y}^a)$ from now on.

Functional spaces

As before, we need to define our Trefftz spaces. Let us begin with the local variational formulation on a spacetime element K:

$$\begin{aligned} &\frac{1}{c^2 \rho} \int_K \frac{\partial p}{\partial t} q + \int_K \frac{\partial v_x}{\partial x} \boldsymbol{q} + \int_K \frac{\partial v_y}{\partial y} q - \int_K {v_y}^a q = 0\,, \\ &\rho \int_K \frac{\partial v_x}{\partial t} w_x + \int_K \frac{\partial p}{\partial x} w_x = 0\,, \\ &\rho \int_K \frac{\partial v_y}{\partial t} w_y + \int_K \frac{\partial p}{\partial y} w_y - p^a w_y = 0\,, \\ &\int_K \frac{\partial p^a}{\partial t} q^a + \sigma_y \int_K p^a q^a = \sigma_y \int_K \frac{\partial p}{\partial y} q^a\,, \\ &\int_K \frac{\partial {v_y}^a}{\partial t} {w_y}^a + \sigma_y \int_K {v_y}^a {w_y}^a = \sigma_y \int_K \frac{\partial v_y}{\partial y} {w_y}^a\,. \end{aligned} \tag{8.9}$$

Here, $(q, \mathbf{w}, q^a, {w_y}^a)$ are test functions and following the same logic as before, we introduce V and $\mathbf{V}$:

$$\begin{aligned} V =& \Big\{ p \in L^2(\Omega) \mid p \in H^1(K) \Big\} \\ \mathbf{V} =& \Big\{ \mathbf{v} \in (L^2(\Omega))^d \mid \mathbf{v} \in (H^1(K))^d \Big\} \\ H^1(K) =& \Big\{ \Phi \in L^2(K), \partial^\alpha \Phi \in L^2(K), \forall \alpha \in \mathbb{N}^d, |\alpha| \leq 1 \Big\} \end{aligned}$$

As a consequence, (8.9) is well-defined if we assume $p, p^a, {v_y}^a \in V$ and $\mathbf{v} \in \mathbf{V}$.

Now that we have properly defined the functional spaces, it remains to introduce the Trefftz space in which the basis functions will be chosen for the new variational formulation. As already explained before, the idea of the Trefftz method is to take particular solutions as basis functions. According to the system of equations we want to solve, our Trefftz space is defined as:

$$\mathbf{T} = \Big\{ (\mathbf{v}, p, {v_y}^a, p^a) \in \mathbf{V} \times V \times V \times V, \text{ such that } (\mathbf{v}, p, {v_y}^a, p^a) \text{ is solution to (8.8)} \Big\}$$

8.3.1 Variational formulation B

To obtain the variational formulation, let us sum all equations of (8.9) and integrate by parts. This results in:

$$\begin{aligned} &\int_K - \left[\frac{1}{c^2\rho} \frac{\partial q}{\partial t} + \frac{\partial w_x}{\partial x} + \frac{\partial w_y}{\partial y} - \sigma_y \frac{\partial {w_y}^a}{\partial y} \right] p - \left[\frac{\partial q}{\partial y} + \rho \frac{\partial w_y}{\partial t} - \sigma_y \frac{\partial q^a}{\partial y} \right] v_y - \left[\frac{\partial q}{\partial x} + \rho \frac{\partial w_x}{\partial t} \right] v_x \\ &\quad - \left[\frac{\partial q^a}{\partial t} + q - \sigma_y q^a \right] {v_y}^a - \left[\frac{\partial {w_y}^a}{\partial t} + w_y - \sigma_y {w_y}^a \right] p^a \\ &+ \int_{\partial K} \frac{1}{c^2\rho} p q n_t + v_x q n_x + v_y q n_y + \rho v_x w_x n_t + p w_x n_x + \rho v_y w_y n_t + p w_y n_y \\ &\quad + {w_y}^a q^a n_t - \sigma_y v_y q^a n_y + p^a {w_y}^a n_t - \sigma_y p {w_y}^a n_y = 0 \,. \end{aligned}$$

As previously with the first PML formulation, we see that if the test functions are solutions to the problem of interest, the volumic terms do not vanish as it does in classical Trefftz formulations. But this can be achieved if we consider test functions satisfying the following problem:

$$\begin{cases} \dfrac{1}{c^2\rho} \dfrac{\partial q}{\partial t} + \dfrac{\partial w_x}{\partial x} + \dfrac{\partial w_y}{\partial y} - \sigma_y \dfrac{\partial {w_y}^a}{\partial y} = 0 \,, \\ \rho \dfrac{\partial w_x}{\partial t} + \dfrac{\partial q}{\partial x} = 0 \,, \\ \rho \dfrac{\partial w_y}{\partial t} + \dfrac{\partial q}{\partial y} - \sigma_y \dfrac{\partial q^a}{\partial y} = 0 \,, \\ (\dfrac{\partial}{\partial t} - \sigma_y) q^a = -q \,, \\ (\dfrac{\partial}{\partial t} - \sigma_y) {w_y}^a = -w_y \,. \end{cases} \tag{8.10}$$

As before, let us then define the suitable test function space:

$$\tilde{\mathbf{T}} = \Big\{ (\mathbf{w}, q, {w_y}^a, q^a) \in \mathbf{V} \times V \times V \times V, \text{ such that } (\mathbf{w}, q, {w_y}^a, q^a) \text{ is solution to (8.10)} \Big\}$$

Hence, the generalized local Trefftz formulation of the PML problem with auxiliary variables (8.8) reads:

Generalized Trefftz Variational Formulation

Seek $(\mathbf{v}, p, {v_y}^a, p^a) \in \mathbf{T}$*, such that for all* $(\mathbf{w}, q, {w_y}^a, q^a) \in \tilde{\mathbf{T}}$ *and for all* K*, it holds true:*

$$\begin{aligned} &\int_{\partial K} \frac{1}{c^2\rho} pqn_t + v_x q n_x + v_y q n_y \\ &+ \int_{\partial K} \rho v_x w_x n_t + p w_x n_x \\ &+ \int_{\partial K} \rho v_y w_y n_t + p w_y n_y \\ &+ \int_{\partial K} {w_y}^a q^a n_t - \sigma_y v_y q^a n_y + p^a {w_y}^a n_t - \sigma_y p {w_y}^a n_y = 0\,. \end{aligned} \tag{8.11}$$

Notice that, once again, the volumic terms have indeed vanished and we are left with boundary integrals. Moreover, if we compare this formulation with the former one, we can see that there are no more derivatives involved. We end up with a classical expression as with the acoustic wave equation without PML.

Numerical fluxes

Now that we have defined the Trefftz variational formulation, using the same triangulation and approximate wavefields as before, it remains to define the DG numerical fluxes for Tent-Pitching meshes, which we choose as follows:

$$\begin{pmatrix} \mathbf{v}\cdot\mathbf{n} \\ p \\ {v_y}^a \\ p^a \end{pmatrix} = \begin{pmatrix} \{\!\{\mathbf{v}\cdot\mathbf{n}\}\!\} + \beta_1 [\![p]\!]_{\mathbf{x}} \\ \{\!\{p\}\!\} + \alpha_1 [\![\mathbf{v}]\!]_{\mathbf{x}} \end{pmatrix} \quad \text{on } \mathcal{F}^{int}$$

$$\begin{pmatrix} \mathbf{v} \\ p \\ {v_y}^a \\ p^a \end{pmatrix} = \begin{pmatrix} \mathbf{v} \\ p \\ {v_y}^a \\ p^a \end{pmatrix} \quad \text{on } \mathcal{F}^{out}$$

$$\begin{pmatrix} \mathbf{v} \\ p \\ {v_y}^a \\ p^a \end{pmatrix} = \begin{pmatrix} (0.5-\alpha_2)v_y + (0.5+\alpha_2){v_y}^0 \\ (0.5-\beta_2)p + (0.5+\beta_2)p^0 \\ (0.5-\alpha_2){v_y}^a + (0.5+\alpha_2)({v_y}^a)^0 \\ (0.5-\beta_2)p^a + (0.5+\beta_2)(p^a)^0 \end{pmatrix} \quad \text{on } \mathcal{F}^{in}$$

$$\begin{pmatrix} \mathbf{v}\cdot\mathbf{n} \\ p \\ {v_y}^a \\ p^a \end{pmatrix} = \begin{pmatrix} g \\ p+\alpha_1(v\cdot n_x - g) \\ {v_y}^a \\ p^a \end{pmatrix} \quad \text{on } \mathcal{F}^{ext}$$

where α_1 and α_2 are penalty coefficients, taken between 0 and 1. We then inject these fluxes into (8.11) and hence obtain the following **Trefftz-DG PML variational formulation on Tent-Pitched meshes with auxiliary variables**, which we will more simply refer to as "Variational formulation B":

Trefftz-DG Variational Formulation with Tent-Pitching and PML

Seek $(\mathbf{v}, p, {v_y}^a, p^a) \in \mathbf{T}$, *such that for all* $(\mathbf{w}, q, {w_y}^a, q^a) \in \tilde{\mathbf{T}}$, *it holds true:*

$$\begin{aligned}
&\int_{\mathcal{F}^{out}} (\frac{1}{c^2\rho}pq + \rho(\mathbf{v}\cdot\mathbf{w}))n_t + (\mathbf{v}\cdot\mathbf{n})q + p(\mathbf{w}\cdot\mathbf{n}) \\
&\qquad +(p^a q^a + {v_y}^a {w_y}^a)n_t - \sigma_y(pq^a + v_y {w_y}^a)n_y \\
&+\int_{\mathcal{F}^{in}} (0.5-\alpha_2)((\mathbf{v}\cdot\mathbf{w}) + {v_y}^a {w_y}^a)n_t + (0.5-\beta_2)\left(\frac{1}{c^2\rho}pq + p^a q^a\right)n_t \\
&+0.5\int_{\mathcal{F}^{in}} (\mathbf{v}\cdot\mathbf{n})q + p(\mathbf{w}\cdot\mathbf{n}) - \sigma_y(pq^a + v_y {w_y}^a)n_y \\
&+\int_{\mathcal{F}^{ext}} p(\mathbf{w}\cdot\mathbf{n}) + \alpha_1(\mathbf{v}\cdot\mathbf{n})(\mathbf{w}\cdot\mathbf{n}) - \sigma_y(v_y {w_y}^a + pq^a + \alpha_1(\mathbf{v}\cdot\mathbf{n})q^a)n_y \\
&+\int_{\mathcal{F}^{int}} \{\!\{\mathbf{v}\}\!\}\cdot[\![q]\!]_x + \beta_1[\![p]\!]_x\cdot[\![q]\!]_x + \{\!\{p\}\!\}[\![\mathbf{w}]\!]_x + \alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x \\
&-\sigma_y\int_{\mathcal{F}^{int}} \{\!\{v_y\}\!\}[\![{w_y}^a]\!]_y + \gamma_1[\![p]\!]_y[\![{w_y}^a]\!]_y + \{\!\{p\}\!\}[\![q^a]\!]_y + \gamma_2[\![v_y]\!]_y[\![q^a]\!]_y \\
&= -\int_{\mathcal{F}^{in}} (0.5+\alpha_2)((\mathbf{v}^{in}\cdot\mathbf{w}) + ({v_y}^a)^{in}{w_y}^a)n_t + (0.5+\beta_2)\left(\frac{1}{c^2\rho}p^{in}q + (p^a)^{in}q^a\right)n_t \\
&-0.5\int_{\mathcal{F}^{in}} (\mathbf{v}^{in}\cdot\mathbf{n})q + p^{in}(\mathbf{w}\cdot\mathbf{n}) - \sigma_y(p^{in}q^a + {v_y}^{in}{w_y}^a)n_y
\end{aligned}$$

Advantages and Drawbacks

We took an interest in this formulation because all of the information concerning PMLs is concentrated in the auxiliary variables. Hence, it is very easy to switch back to the formulation without PML, allowing us to debug our code more easily by comparing the results of different bricks of code to the acoustic solver without PML. However, we still face the problem of needing an additional functional space for the test functions, hence

computing two sets of functions, once again. After implementing this formulation, we obtained bad results, as the numerical solution does not converge to the exact solution when $\sigma_y = 0$ and when considering $\sigma_y > 0$, the numerical solution blows up over time. These results are explained in more details in Chapter Implementation. We numerically observed that these bad results could have originated from the difference between the basis and the test functions, but we could not find a proof of this claim. However, this idea paved the way for constructing a third formulation avoiding the use of a different space for the test functions.

8.4 Variational formulation C

In Chapter Trefftz-DG Method for Wave equations, we introduced a variant of the variational formulation for the Trefftz-DG method with Tent-Pitching for the acoustic wave equations. We can apply the same reasoning to the PML equations (8.8) and we obtain the following variational formulation:

Trefftz-DG Variational Formulation C with Tent-Pitching and PML

Seek $(\mathbf{v}, p, {v_y}^a, p^a) \in \mathbf{T}$*, such that for all* $(\mathbf{w}, q, {w_y}^a, q^a) \in \mathbf{V} \times V \times V \times V$*, it holds true:*

$$\mathcal{A}(\mathbf{v}, p; \mathbf{w}, q) = l(\mathbf{w}, q)$$

where

$$\begin{aligned}
&\int_{\mathcal{F}^{in}} (0.5+\alpha)(\rho \mathbf{v}\cdot\mathbf{w} + {v_y}^a {w_y}^a) n_t + (0.5+\beta)(\frac{1}{c^2\rho} pq + p^a q^a) \\
+0.5&\int_{\mathcal{F}^{in}} \mathbf{v}q\cdot\mathbf{n} + p\mathbf{w}\cdot\mathbf{n} - \sigma_y(v_y {w_y}^a + pq^a) n_y \\
&+ \int_{\mathcal{F}^{ext}} p(\mathbf{w}\cdot\mathbf{n}) + \alpha_1(\mathbf{v}\cdot\mathbf{n})(\mathbf{w}\cdot\mathbf{n}) - \sigma_y v_y {w_y}^a n_y - \sigma_y pq^a n_y - \sigma_y(\mathbf{v}\cdot\mathbf{n})q^a n_y \\
&+ \int_{\mathcal{F}^{int}} \{\!\{\mathbf{v}\}\!\}\cdot[\![q]\!]_x + \beta_1 [\![p]\!]_x \cdot [\![q]\!]_x + \{\!\{p\}\!\}[\![\mathbf{w}]\!]_x + \alpha_1 [\![\mathbf{v}]\!]_x [\![\mathbf{w}]\!]_x \\
-\sigma_y&\int_{\mathcal{F}^{int}} \{\!\{v_y\}\!\}[\![{w_y}^a]\!]_y + \gamma_1 [\![p]\!]_y [\![{w_y}^a]\!]_y + \{\!\{p\}\!\}[\![q^a]\!]_y + \gamma_2 [\![v_y]\!]_y [\![q^a]\!]_y \\
&= \int_{\mathcal{F}^{in}} (0.5-\alpha_2)((\mathbf{v}^{in}\cdot\mathbf{w}) + ({v_y}^a)^{in} {w_y}^a) n_t + (0.5-\beta_2)\left(\frac{1}{c^2\rho} p^{in} q + (p^a)^{in} q^a\right) n_t \\
+0.5&\int_{\mathcal{F}^{in}} (\mathbf{v}^{in}\cdot\mathbf{n})q + p^{in}(\mathbf{w}\cdot\mathbf{n}) - \sigma_y(p^{in} q^a + {v_y}^{in} {w_y}^a) n_y
\end{aligned}$$

Advantages and Drawbacks

The advantages of this formulation are as expected; we do not need to introduce an additional functional space and thus, we do not have more computations. Additionally, we can even choose any test function in $(\mathbf{V}\times V)$, which is very convenient. However, since

we only have inflow boundaries, we could not compare it with the acoustic solver without PML, which was only implemented with the classical formulation and not the variant formulation C. All the improvements brought by this formulation helped us achieve good results, which will be presented in Chapter Implementation.

8.5 PML in x and y

The aim in this section is to address the problem with two PMLs: one parallel to the x-axis and the other parallel to the y-axis as depicted in Fig. 8.3. The drawbacks of the formulations A and B previously presented and their lack of robustness shown in the numerical experiments with a single horizontal PML in the following chapter encouraged us to restrict ourselves to the formulation C.

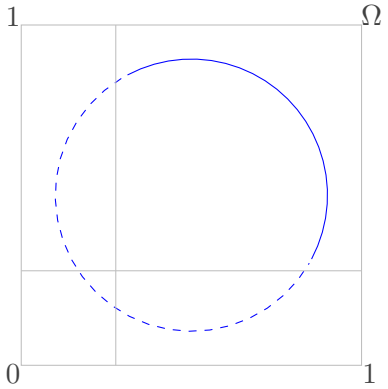


Figure 8.3: Considered domain with two PMLs

We introduce the acoustic wave equations with PML in x- and y-direction, using the variable stretch of Bérenger, as follows:

$$\begin{cases} \frac{1}{c^2\rho}(\frac{\partial}{\partial t}+\sigma_y)(\frac{\partial}{\partial t}+\sigma_x)\frac{\partial p}{\partial t}+(\frac{\partial}{\partial t}+\sigma_y)\frac{\partial}{\partial t}\frac{\partial v_x}{\partial x}+(\frac{\partial}{\partial t}+\sigma_x)\frac{\partial}{\partial t}\frac{\partial v_y}{\partial y}=0\,, \\ \rho(\frac{\partial}{\partial t}+\sigma_x)\frac{\partial v_x}{\partial t}+\frac{\partial}{\partial t}\frac{\partial p}{\partial x}=0\,, \\ \rho(\frac{\partial}{\partial t}+\sigma_y)\frac{\partial v_y}{\partial t}+\frac{\partial}{\partial t}\frac{\partial p}{\partial y}=0\,. \end{cases} \tag{8.12}$$

From which, we derive the equations with auxiliary variables, which can be written as:

$$\begin{cases} \frac{1}{c^2\rho}\frac{\partial p}{\partial t} + \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} - {v_x}^a - {v_y}^a = 0\,, \\ \rho\frac{\partial v_x}{\partial t} + \frac{\partial p}{\partial x} - p_1^a = 0\,, \\ \rho\frac{\partial v_y}{\partial t} + \frac{\partial p}{\partial y} - p_2^a = 0\,, \\ (\partial_t + \sigma_x)p_1^a = \sigma_x\frac{\partial p}{\partial x}\,, \\ (\partial_t + \sigma_y)p_2^a = \sigma_y\frac{\partial p}{\partial y}\,, \\ (\partial_t + \sigma_x){v_x}^a = \sigma_x\frac{\partial v_x}{\partial x}\,, \\ (\partial_t + \sigma_y){v_y}^a = \sigma_y\frac{\partial v_y}{\partial y}\,. \end{cases} \tag{8.13}$$

Functional spaces

As before, let us consider a local variational formulation on a spacetime element K:

$$\begin{aligned} &\int_K (\frac{1}{c^2\rho}\frac{\partial p}{\partial t} + \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} - {v_x}^a - {v_y}^a)q = 0\,, \\ &\int_K (\rho\frac{\partial v_x}{\partial t} + \frac{\partial p}{\partial x} - p_1^a)w_x = 0\,, \\ &\int_K (\rho\frac{\partial v_y}{\partial t} + \frac{\partial p}{\partial y} - p_2^a)w_y = 0\,, \\ &\int_K ((\partial_t + \sigma_x)p_1^a)q_1^a = \int_K \sigma_x\frac{\partial p}{\partial x}q_1^a\,, \\ &\int_K ((\partial_t + \sigma_y)p_2^a)q_2^a = \int_K \sigma_y\frac{\partial p}{\partial y}q_2^a\,, \\ &\int_K ((\partial_t + \sigma_x){v_x}^a){w_x}^a = \int_K \sigma_x\frac{\partial v_x}{\partial x}{w_x}^a\,, \\ &\int_K ((\partial_t + \sigma_y){v_y}^a){w_y}^a = \int_K \sigma_y\frac{\partial v_y}{\partial y}{w_y}^a\,. \end{aligned} \tag{8.14}$$

Here, $(q, \mathbf{w}, {w_x}^a, {w_y}^a, q_1^a, q_2^a)$ are test functions and their functional spaces will be defined in the following. Let us define V and $\mathbf{V}$ as follows:

$$\begin{aligned} V =& \Big\{ p \in L^2(\Omega) \mid p \in H^1(K) \Big\} \\ \mathbf{V} =& \Big\{ \mathbf{v} \in (L^2(\Omega))^d \mid \mathbf{v} \in (H^1(K))^d \Big\} \\ H^1(K) =& \Big\{ \Phi \in L^2(K), \partial^\alpha\Phi \in L^2(K), \forall\alpha \in \mathbb{N}^d, |\alpha| \leq 1 \Big\} \end{aligned}$$

Thus, the system (8.13) is well-defined for $(p, \mathbf{v}, {v_x}^a, {v_y}^a, p_1^a, p_2^a) \in (V \times \mathbf{V} \times V \times V \times V \times V)$. It remains to define the Trefftz space for our problem, which can be written:

$$\begin{aligned} \mathbf{T} = \Big\{ &(p, \mathbf{v}, {v_x}^a, {v_y}^a, p_1^a, p_2^a) \in V \times \mathbf{V} \times V \times V \times V \times V, \text{ such that} \\ &(p, \mathbf{v}, {v_x}^a, {v_y}^a, p_1^a, p_2^a) \text{ is solution to (8.13)} \Big\} \end{aligned}$$

Now that our functional spaces and Trefftz space are properly defined, let us sum all equations of (8.14) and integrate by parts, which results in:

$$
\begin{aligned}
&-\int_K \left[\frac{1}{c^2\rho}\frac{\partial q}{\partial t}+\frac{\partial w_x}{\partial x}+\frac{\partial w_y}{\partial y}-\sigma_x\frac{\partial {w_x}^a}{\partial x}-\sigma_y\frac{\partial {w_y}^a}{\partial y}\right]p\\
&\quad+\left[\frac{\partial q}{\partial x}+\rho\frac{\partial w_x}{\partial t}-\sigma_x\frac{\partial q_1^a}{\partial x}\right]v_x+\left[\frac{\partial q}{\partial y}+\rho\frac{\partial w_y}{\partial t}-\sigma_y\frac{\partial q_2^a}{\partial y}\right]v_y\\
&\quad+\left[\frac{\partial {w_x}^a}{\partial t}+w_x-\sigma_x {w_x}^a\right]p_1^a+\left[\frac{\partial {w_y}^a}{\partial t}+w_y-\sigma_y {w_y}^a\right]p_2^a\\
&\quad+\left[\frac{\partial q_1^a}{\partial t}+q-\sigma_x q_1^a\right]{v_x}^a+\left[\frac{\partial q_2^a}{\partial t}+q-\sigma_y q_2^a\right]{v_y}^a\\
&+\int_{\partial K}\frac{1}{c^2\rho}pqn_t+\rho(\mathbf{v}\cdot\mathbf{w})n_t+p(\mathbf{w}\cdot\mathbf{n})+(\mathbf{v}\cdot\mathbf{n})q\\
&+\int_{\partial K}p_1^a{w_x}^a n_t-\sigma_x p{w_x}^a n_x+p_2^a{w_y}^a n_t-\sigma_y p{w_y}^a n_y\\
&+\int_{\partial K}{v_x}^a q_1^a n_t-\sigma_x v_x q_1^a n_x+{v_y}^a q_2^a n_t-\sigma_y v_y q_2^a n_y=0\,.
\end{aligned}
$$

Since we want to use the same reasoning as for the previous variational formulation C, we will integrate by parts once more, thus obtaining the following formulation:

Trefftz-DG Variational Formulation C with Tent-Pitching and PML in x and y

Seek $(p,\mathbf{v},{v_x}^a,{v_y}^a,p_1^a,p_2^a)\in\mathbf{T}$, *such that for all* $(\mathbf{w},q,{w_y}^a,q_1^a)\in V\times\mathbf{V}\times V\times V\times V\times V$, *it holds true:*

$$\mathcal{A}(\mathbf{v},p;\mathbf{w},q)=l(\mathbf{w},q)$$

where

$$
\begin{aligned}
\mathcal{A}(\mathbf{v},p;\mathbf{w},q):=&\int_{\mathcal{F}^{in}}(0.5+\alpha)(\rho\mathbf{v}\cdot\mathbf{w}+{v_x}^a{w_x}^a+{v_y}^a{w_y}^a)n_t\\
&+(0.5+\beta)(\frac{1}{c^2\rho}pq+p_1^aq_1^a+p_2^aq_2^a)\\
&+0.5\int_{\mathcal{F}^{in}}\mathbf{v}\cdot\mathbf{n}q+p\mathbf{w}\cdot\mathbf{n}\\
&-\sigma_y(v_y{w_y}^a n_y-pq_2^an_y)-\sigma_x(v_x{w_x}^a n_x-pq_1^an_x)\\
&+\int_{\mathcal{F}^{ext}}p(\mathbf{w}\cdot\mathbf{n})+\alpha_1(\mathbf{v}\cdot\mathbf{n})(\mathbf{w}\cdot\mathbf{n})\\
&-\sigma_y(v_y{w_y}^a n_y-pq_2^an_y)-\sigma_x(v_x{w_x}^a n_x-pq_1^an_x)\\
&+\int_{\mathcal{F}^{int}}\{\!\{\mathbf{v}\}\!\}\cdot[\![q]\!]_x+\beta_1[\![p]\!]_x\cdot[\![q]\!]_x+\{\!\{p\}\!\}[\![\mathbf{w}]\!]_x+\alpha_1[\![\mathbf{v}]\!]_x[\![\mathbf{w}]\!]_x\\
&-\sigma_x\int_{\mathcal{F}^{int}}\{\!\{v_x\}\!\}[\![{w_x}^a]\!]_x+\gamma_1[\![p]\!]_x[\![{w_x}^a]\!]_x+\{\!\{p\}\!\}[\![q_1^a]\!]_x+\gamma_2[\![v_x]\!]_x[\![q_1^a]\!]_x\\
&-\sigma_y\int_{\mathcal{F}^{int}}\{\!\{v_y\}\!\}[\![{w_y}^a]\!]_y+\gamma_1[\![p]\!]_y[\![{w_y}^a]\!]_y+\{\!\{p\}\!\}[\![q_2^a]\!]_y+\gamma_2[\![v_y]\!]_y[\![q_2^a]\!]_y
\end{aligned}
$$

$$
\begin{aligned}
l(\mathbf{w},q) := \int_{\mathcal{F}^{in}} & (0.5-\alpha_2)((\mathbf{v}^{in}\cdot\mathbf{w}) + (v_y{}^a)^{in} w_y{}^a) n_t \\
& +(0.5-\beta_2)\left(\frac{1}{c^2\rho} p^{in} q + (p_1^a)^{in} q_1^a\right) n_t \\
+0.5 \int_{\mathcal{F}^{in}} & (\mathbf{v}^{in}\cdot\mathbf{n}) q + p^{in}(\mathbf{w}\cdot\mathbf{n}) \\
& -\sigma_y(p^{in} q_2^a + v_y{}^{in} w_y{}^a) n_y - \sigma_x(v_x{}^{in} w_x{}^a - p^{in} q_1^a) n_x
\end{aligned}
$$

8.6 Computation of basis functions

Now that we have presented the various functional spaces, Trefftz spaces and variational formulation, we need to adress the question of basis functions. Indeed, since we are working in a Trefftz framework, we need to construct local solutions of the studied equations. A natural choice of basis functions would have been polynomial solutions, as the ones we used in the non-PML case. However, to our knowledge, polynomial solutions to the acoustic PML equations have not been derived yet. In this section, we address this question and show that the dependency in time cannot be polynomial.

8.6.1 Polynomial Basis Functions for PML

As explained before, in order to carry out our Trefftz-DG method in a PML framework, we need to construct local solutions, which ideally would be polynomials in our case in order to extend the framework derived in Part I. However, it turns out that we cannot derive spacetime polynomial solutions to the acoustic wave equation with PML. In this section, we will prove this by contradiction.

In order to do so, let us recall the acoustic wave equations with PML in both directions:

$$\frac{1}{c^2\rho}\frac{\partial p}{\partial t} + \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} - {v_x}^a - {v_y}^a = 0\,, \tag{8.15a}$$

$$\rho\frac{\partial v_x}{\partial t} + \frac{\partial p}{\partial x} - p_1^a = 0\,,$$

$$\rho\frac{\partial v_y}{\partial t} + \frac{\partial p}{\partial y} - p_2^a = 0\,, \tag{8.15b}$$

$$(\partial_t + \sigma_x)p_1^a = \sigma_x\frac{\partial p}{\partial x}\,, \tag{8.15c}$$

$$(\partial_t + \sigma_y)p_2^a = \sigma_y\frac{\partial p}{\partial y}\,,$$

$$(\partial_t + \sigma_x){v_x}^a = \sigma_x\frac{\partial v_x}{\partial x}\,,$$

$$(\partial_t + \sigma_y){v_y}^a = \sigma_y\frac{\partial v_y}{\partial y}\,. \tag{8.15d}$$

Let us assume that p, v_x and v_y are polynomials of degree d in $\mathbb{P}^d$, thus they can be written under the following form:

$$p(x,y,t) = \sum_{i+j+l\leq d} p_{i,j,l}x^i y^j t^l$$
$$v_x(x,y,t) = \sum_{i+j+l\leq d} v_{xi,j,l}x^i y^j t^l$$
$$v_y(x,y,t) = \sum_{i+j+l\leq d} v_{y_{i,j,l}}x^i y^j t^l$$

The equations (8.15c)-(8.15d) can be solved and we obtain:

$$p_1^a(x,y,t) = \sigma_x e^{-\sigma_x t}\int_0^t e^{\sigma_x s}\frac{\partial p(x,y,s)}{\partial x}\mathrm{ds}$$
$$p_2^a(x,y,t) = \sigma_y e^{-\sigma_y t}\int_0^t e^{\sigma_y s}\frac{\partial p(x,y,s)}{\partial y}\mathrm{ds}$$
$${v_x}^a(x,y,t) = \sigma_x e^{-\sigma_x t}\int_0^t e^{\sigma_x s}\frac{\partial v_x(x,y,s)}{\partial x}\mathrm{ds}$$
$${v_y}^a(x,y,t) = \sigma_y e^{-\sigma_y t}\int_0^t e^{\sigma_y s}\frac{\partial v_y(x,y,s)}{\partial y}\mathrm{ds}$$

The derivatives of p can be written as follows:

$$\frac{\partial p}{\partial x} = \sum_{i+j+l\leq d} ip_{i,j,l}x^{i-1}y^j t^l$$
$$\frac{\partial p}{\partial y} = \sum_{i+j+l\leq d} jp_{i,j,l}x^i y^{j-1} t^l$$

We can also derive v_x and v_y analogously. We denote $\mathrm{D}^{(d)}[u]$ the d^{th} derivative of u.

According to (8.15a)-(8.15b), we have:

$$\begin{aligned}
&\mathrm{D}^{(d)}[{v_x}^a + {v_y}^a] = \mathrm{D}^{(d)}\left[\frac{1}{c^2\rho}\frac{\partial p}{\partial t} + \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right] = 0\,,\\
&\mathrm{D}^{(d)}[p_1^a] = \mathrm{D}^{(d)}\left[\rho\frac{\partial v_x}{\partial t} + \frac{\partial p}{\partial x}\right] = 0\,,\\
&\mathrm{D}^{(d)}[p_2^a] = \mathrm{D}^{(d)}\left[\rho\frac{\partial v_y}{\partial t} + \frac{\partial p}{\partial y}\right] = 0\,.
\end{aligned} \tag{8.19}$$

We can see that since we assume p, v_x and v_y as polynomials of degree d, taking their d^{th} derivative is equal to zero. Thus, using (8.19) and replacing the derivatives by their polynomial expressions, we obtain:

$$\begin{aligned}
\mathrm{D}^{(d)}[{v_x}^a + {v_y}^a] = \sum_{k=0}^{d}\binom{d}{k}\mathrm{D}^{(d-k)}&\left[\sum_{i+j+l\leq d}(\sigma_x e^{-\sigma_x t} i v_{x\,i,j,l} x^{i-1}y^j + \sigma_y e^{-\sigma_y t} j v_{y\,i,j,l} x^i y^{j-1}\right]\\
&\times \mathrm{D}^{(k)}\left[\int_0^t (e^{\sigma_x s} + e^{\sigma_y s}) s^l \mathrm{ds}\right]\,,\\
\mathrm{D}^{(d)}[p_1^a] = \sigma_x \sum_{k=0}^{d}\binom{d}{k}\mathrm{D}^{(d-k)}&\left[\sum_{i+j+l\leq d} i p_{i,j,l} x^{i-1}y^j\right]\mathrm{D}^{(k)}\left[\int_0^t e^{-\sigma_x(t-s)} s^l \mathrm{ds}\right]\,,\\
\mathrm{D}^{(d)}[p_2^a] = \sigma_y \sum_{k=0}^{d}\binom{d}{k}\mathrm{D}^{(d-k)}&\left[\sum_{i+j+l\leq d} j p_{i,j,l} x^{i}y^{j-1}\right]\mathrm{D}^{(k)}\left[\int_0^t e^{-\sigma_y(t-s)} s^l \mathrm{ds}\right]\,.
\end{aligned}$$

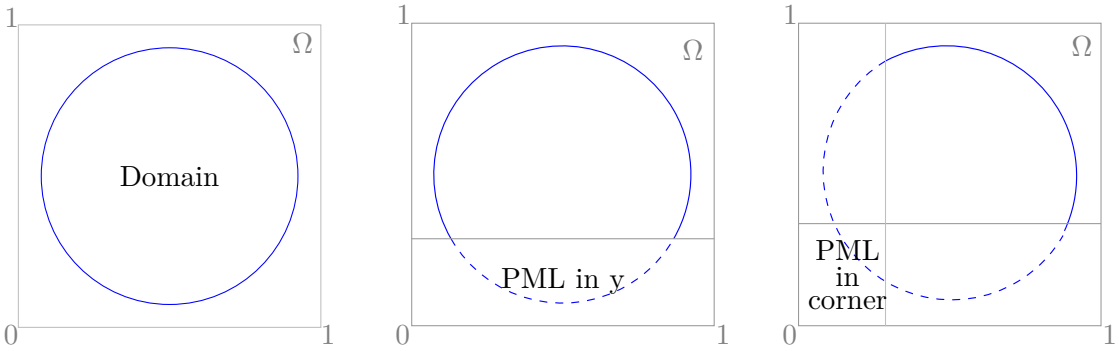
These expressions are never equal to zero, in particular because the k^{th} derivative of the exponential function is never zero. Thus, when assuming that the solutions are polynomials, the conditions (8.19) are never fulfilled and we obtain a contradiction. We can conclude that there are no spacetime polynomial solutions to the acoustic wave equation with PML.

Even though there are no spacetime polynomial solutions, Green's functions are known and have been computed by Diaz in [100] for the second-order wave equation with PML. So, in order to carry out the PML framework we propose to construct a set of local basis functions that are composed of Green's functions. This is the purpose of the following chapter.

Chapter 9

Green's Functions

As explained in the previous chapter, to carry out a Trefftz-DG approximation of the acoustic wave equation with PML we need to construct exact solutions to the acoustic wave equations with Perfectly Matched Layers. For that purpose, we analytically compute the Green's functions for the first-order acoustic wave equation and to do so, we use a tool known as the Cagniard-De Hoop method. The first section will be dedicated to its presentation and explanation. In the second section, we describe the construction of the analytical solutions in the non-PML case, for both the pressure and the velocity. In the third section, we describe the construction of the analytical solutions in a PML parallel to the x-axis. We finish by considering the general case of a PML surrounding the computational domain, which leads us to handle the case of the corner where the absorption is in both directions. We refer the reader to Fig. 9.1 for a description of the different configurations.



(a) Sec. 9.2: Computation of Green's functions without absorption

(b) Sec. 9.3: Computation of Green's functions with absorption in one layer

(c) Sec. 9.4: Computation of Green's functions with absorption in the corner

Figure 9.1: Plan of the chapter

9.1 Cagniard-De Hoop Method

The Cagniard-De Hoop method is a mathematical technique for computing analytical solutions for wave problems in layered media. Originally, this tool was meant for seismic wave problems and has then been extended to other kinds of waves [101, 102, 103]. This method is first attributed to Cagniard ([104], 1939) and then to De Hoop ([105], 1960) who improved it and also proposed an extension to three dimensions. This method exploits the Laplace and Fourier transforms, so let us define them first. The Laplace transform $\mathcal{L}$ of a function $u(t)$ is the following integral:

$$\mathcal{L}[u(t)](s) = \int_0^{+\infty} u(t)e^{-st}\,\mathrm{d}t = \tilde{u}(s)\,,$$

where s is the complex Laplace variable and reciprocally, the inverse Laplace transform $\mathcal{L}^{-1}$ of $\tilde{u}(s)$ is $u(t)$. Let us also introduce the Fourier transform $\mathcal{F}$ of a function $u(x)$ as follows:

$$\mathcal{F}[u(x)](k) = \int_{-\infty}^{+\infty} u(x)e^{ik}\,\mathrm{d}x = U(k)\,,$$

where k is the real dual variable of x and reciprocally, the inverse Fourier transform $\mathcal{F}^{-1}$ of $U(k)$ is defined as follows:

$$\mathcal{F}^{-1}[U(k)](x) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} U(k)e^{-ik}\,\mathrm{d}k = u(x)\,.$$

To apply the Cagniard-De Hoop method, we first perform the above introduced Laplace and Fourier transforms on the considered equations and obtain an explicit expression for $\hat{p}(k,y,s) = \mathcal{F}_x[\tilde{p}(x,y,s)](k,y,s)$, where p is the unknown of our problem, $\tilde{p}(x,y,s)$ is its Laplace transform and $\mathcal{F}_x$ is the partial Fourier transform along x. We then apply an inverse Fourier transform to $\hat{p}(k,y,s)$, which results in the following expression:

$$\tilde{p}(x,y,s) = \mathcal{F}_k^{-1}[\hat{p}(k,y,s)](x,y,s) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \hat{p}(k,y,s)e^{-ikx}dk$$

The next step consists in applying the change of variable $k = {}^{bs}/_{c}$, where b is a real variable. This results in an integral in the following form:

$$\tilde{p}(x,y,s) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} g(x,y,b)e^{-sf(x,y,b)}db.$$

We can see from the previous expression that the change of variable brings out the term $e^{-sf(x,y,b)}$ with the Laplace variable s. This integral can thus be identified with an inverse Laplace transform, if we can find a complex path Γ such that $f(x,y,b) = t$ by considering b as a complex variable. This is the last step and the heart of the Cagniard-De Hoop method. If such a path is found, the previous integral can be rewritten as follows:

$$\tilde{p}(x,y,s) = \frac{1}{2\pi}\int_0^{+\infty} h(x,y,t)e^{-st}dt$$

where $h(x, y, t)$ represents the inverse Laplace transform of $\tilde{p}$, which is also the solution $p(x, y, t)$ we seek by the injectivity of the Laplace transform. Hence:

$$p(x, y, t) = \mathcal{L}^{-1}[\tilde{p}(x, y, s)](t) = h(x, y, t).$$

In the following, we propose to use the Cagniard-De Hoop method for computing the Green's function for the acoustic pressure, as it was formerly done by Diaz in [100]. This will be an opportunity for us to show how the method actually works.

9.2 Green's functions without PML

In this section, we compute the Green's functions of the acoustic wave equation (4.3) using the previously described Cagniard-De Hoop method. We obtain three Green's functions, each associated to the pressure and the velocity fields of the acoustic wave equation.

9.2.1 Pressure

In order to find the Green's function for the pressure field, we apply the Cagniard-De Hoop method to the second order wave equation, which is written as follows:

$$\frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} - \frac{\partial^2 P}{\partial x^2} - \frac{\partial^2 P}{\partial y^2} = \delta(x)\delta(y)f(t). \tag{9.1}$$

where δ represents the Dirac distribution and f the source term. A Green's function is a fundamental solution of a differential equation with point source term. This mathematical object was introduced by George Green in 1828 ([106]) for the Poisson's equation and various methods for deriving Green's functions are presented in Duffy's book ([107]). As long as our problem is concerned, the corresponding Green's function is defined as the solution to (9.2) and the solution to (9.1) is then given by $P = p * f$, where $*$ denotes the convolutional product.

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \frac{\partial^2 p}{\partial x^2} - \frac{\partial^2 p}{\partial y^2} = \delta(x)\delta(y)\delta(t). \tag{9.2}$$

We now introduce the Laplace transform of p denoted by $\tilde{p}$ and, remarking that the Laplace transform of the Dirac distribution is 1, we obtain the following equation for $\tilde{p}$:

$$\frac{s^2}{c^2}\tilde{p} - \frac{\partial^2 \tilde{p}}{\partial x^2} - \frac{\partial^2 \tilde{p}}{\partial y^2} = \delta(x)\delta(y).$$

where s is the Laplace variable. In general, s is a complex number whose real part defines the abscissa of convergence of the Laplace transform. It can also be a positive real which is actually the case here. The next step consists in applying a Fourier transform to $\tilde{p}(x, y, s)$ and since the Fourier transform of the Dirac distribution is 1, the previous equation becomes:

$$\mathcal{F}[\tilde{p}(x, y, s)](k, \omega, s) = \frac{1}{(\omega^2 + k^2 + \frac{s^2}{c^2})} \tag{9.3}$$

where k and ω denote the dual variables of x and y respectively. The system (9.3) can be explicitly solved. Knowing that the Fourier transform of $e^{-a|u|}$ along u is $\mathcal{F}_u[e^{-a|u|}](\omega) = \frac{2a}{a^2+\omega^2}$ with $a > 0$, we deduce that

$$\hat{p}(k, y, s) = \frac{e^{-|y|(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}. \tag{9.4}$$

The next step of the Cagniard-De Hoop method consists in performing a partial inverse Fourier transform along k on $\hat{p}(k, y, s)$, which gives the following:

$$\tilde{p}(x, y, s) = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-|y|\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}-ikx}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

Now, we apply the change of variable $k = bs/c$ where b is a real variable, which results in:

$$\tilde{p}(x, y, s) = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db. \tag{9.5}$$

As explained previously, the motivation behind the change of variable $k = bs/c$ is to obtain an integral in the form $g(x, y, b)e^{-sf(x,y,b)}$, which is the case here and we denote it $\Xi(b)$. The last step of the method consists in considering b as a complex variable and in finding a complex path Γ such that $f(x, y, b) = t$, in order to identify a Laplace transform. Then, we need to introduce the square root of a complex variable as the function $g(z) = z^{1/2}$ where $z \in \mathbb{C}$ is defined as follows:

$$g(z)^2 = z \text{ and } \mathcal{R}e(g(z)) > 0\,.$$

This step of the Cagniard-De Hoop method means that we look for a path Γ defined as follows:

$$\Gamma \subset \tilde{\Gamma} = \{b \in \mathbb{C}, (1+b^2)^{\frac{1}{2}}|y| + ibx = ct \quad \forall t \in \mathbb{R}^+\}. \tag{9.6}$$

Thus, to find our path, we need to solve $(1+b^2)^{\frac{1}{2}}|y| + ibx = ct$. In order to do so, we switch to polar coordinates $x = r\cos\theta\,, y = r\sin\theta$ and (9.6) becomes:

$$\begin{aligned} &(1+b^2)r^2\sin^2\theta = (ct - ibr\cos\theta)^2 \\ \iff &b^2 + 2i\frac{ct}{r}\cos\theta b + \sin^2\theta - \frac{c^2t^2}{r^2} = 0 \end{aligned}$$

So, b is the solution to a second-order equation whose discriminant is $\Delta = \sin^2\theta\Big(\frac{c^2t^2}{r^2} - 1\Big)$. Thus, solving this equation gives us multiple solutions, hence several complex paths

such that $\tilde{\Gamma} = \Gamma^+ \cup \Gamma^- \cup \Upsilon^+ \cup \Upsilon^-$:

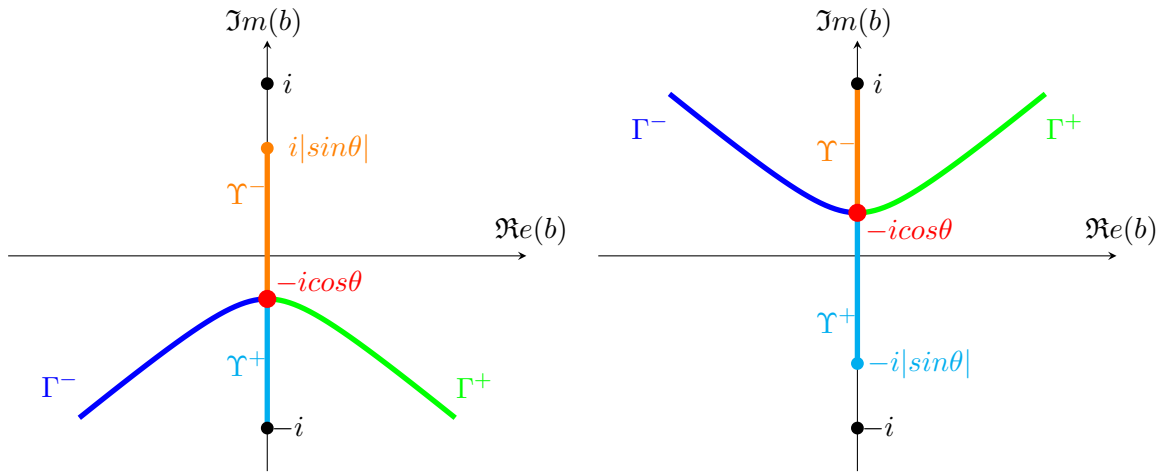
$$\begin{array}{lll} & \Gamma^{\pm} = \{b = \gamma^{\pm}(t)\} & \dfrac{r}{c} \leq t \\ \text{if } \cos\theta > 0 & \Upsilon^{+} = \{b = \nu^{+}(t)\} & \dfrac{r\cos\theta}{c} \leq t \leq \dfrac{r}{c} \\ & \Upsilon^{-} = \{b = \nu^{-}(t)\} & 0 \leq t \leq \dfrac{r}{c} \\ \text{if } \cos\theta < 0 & \Upsilon^{+} = \{b = \nu^{+}(t)\} & 0 \leq t \leq \dfrac{r}{c} \\ & \Upsilon^{-} = \{b = \nu^{-}(t)\} & \dfrac{r|\cos\theta|}{c} \leq t \leq \dfrac{r}{c} \end{array}$$

where

$$\gamma^{\pm} = -i\frac{ct}{r}cos\theta \pm |sin\theta|\sqrt{\frac{c^2t^2}{r^2} - 1}$$

$$\nu^{\pm} = -i\left(\frac{ct}{r}\cos\theta \pm |\sin\theta|\sqrt{1 - \frac{c^2t^2}{r^2}}\right)$$

These different computations are detailed and very well explained in [100]. We illustrate the complex paths we obtain in Fig. 9.2.



(a) Paths for $cos\theta > 0$ (b) Paths for $cos\theta < 0$

Figure 9.2: Representation of complex paths

Since $x > 0$, we have that $\cos\theta > 0$. Thus we consider the path Γ in the lower half-plane. We do not need the paths Υ, but they are necessary for heterogeneous media and other types of waves. In [100] for example, Υ is used in order to derive Green's functions for head waves in the case of a bilayered homogeneous acoustic wave equation.

Now, we need to rewrite the integral (9.5) as an integral on Γ. Let D be the real axis and Ω the region delimited by Γ and D. Let us consider the closed contour $\Gamma_d \cup D_d \cup C_d$, where

$$D_d = \{b \in D, |b| < d\} \qquad \Gamma_d = \{b \in \Gamma, |b| < d\} \qquad C_d = \{b \in \Omega, |b| = d\}$$

C_d contains two arcs of radius d connecting Γ_d and D_d in order to form a closed contour, as depicted in Fig. 9.3. Since $\Xi(b)$ is analytical (*i.e.* infinitely differentiable) and does not have any poles, we obtain that:

$$\int_{D_d} \Xi(b)db + \int_{\Gamma_d} \Xi(b)db + \int_{C_d} \Xi(b)db = 0$$

Since we consider a path which is an arc, the complex numbers in this path can be

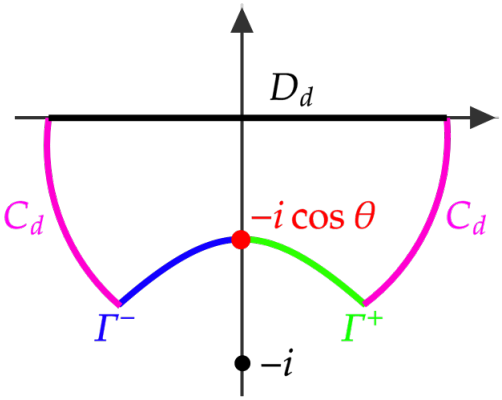


Figure 9.3: Integration contour

decomposed in an exponential form such as $b = Re^{i\Theta}$ where $|b| = R$. We have that $\Re e(|y|(1+b^2)^{1/2} + ibx) \geq 0$. Thus we have the following:

$$\lim_{|b|\to\ +\infty} b\,\Xi(b) = \lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} \frac{e^{i\Theta}}{(\frac{1}{R^2} + e^{2i\Theta})^{\frac{1}{2}}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c} + ie^{i\Theta}\frac{x}{c}\right]} = 0$$

Hence, the following Jordan's lemma applies to $\Xi(b)$:

Jordan's Lemma

Let $\mathcal{S} = \{z = re^{i\theta}, r > 0, 0 \leq \theta_1 \leq \theta \leq \theta_2 < \pi\}$ and $f : \mathcal{S} \to \mathbb{C}$ be a holomorphic function. If

$$\lim_{\substack{|z|\to+\infty \\ z\in\mathcal{S}}} zf(z) = 0$$

then

$$\lim_{r\to+\infty} \int_{\gamma(r,\theta_1,\theta_2)} f(z)e^{iz}dz = 0$$

with $\gamma(r,\theta_1,\theta_2) = \{re^{i\theta}, \theta_1 \leq \theta \leq \theta_2\}$ representing the arc of radius r and of angle θ.

Thus, we can deduce that $\int_{C_d} \Xi(b)db = 0$, which leads us to rewriting (9.5) as the following:

$$\tilde{p}(x,y,s) = -\frac{1}{4\pi}\int_{\Gamma} \frac{e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c} + ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}}db$$

Now, we replace b by its values $\gamma^{\pm}$ and obtain the following quantities:

$$\begin{cases} (1+\gamma^{\pm 2})^{\frac{1}{2}}\dfrac{y}{c} + i\gamma^{\pm}\dfrac{x}{c} = t \\ \dfrac{d\gamma^{\pm}}{(1+\gamma^{\pm 2})^{\frac{1}{2}}} = \pm\dfrac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}} \end{cases}$$

Noticing that Γ^+ varies from $+\infty$ to r/c and that Γ^- varies from r/c to $+\infty$, we obtain the following result:

$$\tilde{p}(x,y,s) = \frac{1}{2\pi}\int_{r/c}^{+\infty} \frac{e^{-st}}{\sqrt{t^2-\frac{r^2}{c^2}}}dt$$

We finally obtain an integral that can be identified to a Laplace transform. Hence, by injectivity of the Laplace transform, we obtain the Green's function for the pressure field of the acoustic wave equation:

$$\begin{aligned} p(x,y,t) &= 0, & t < \frac{r}{c} \\ p(x,y,t) &= \frac{1}{2\pi\sqrt{t^2-\frac{r^2}{c^2}}}, & t > \frac{r}{c} \end{aligned}$$

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

We will now proceed in the same manner to obtain the Green's functions associated to the velocity fields of the acoustic wave equations, which we denote $v_x(x,y,t)$ and $v_y(x,y,t)$.

9.2.2 Velocity in the x-direction

If we perform a Laplace and Fourier transform to the equation linking the pressure and the x-velocity field in (4.3), we obtain the following:

$$\widehat{v_x} = \frac{ik}{\rho s}\hat{p}\,.$$

Thus, injecting the expression of $\hat{p}$ from (9.4) into it, we obtain:

$$\widehat{v_x}(k,y,s) = \frac{ik}{\rho s}\frac{e^{-|y|(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}$$

As done previously, we take the inverse Fourier transform of this quantity, which results in:

$$\tilde{v_x}(x,y,s) = \frac{1}{4\pi}\int_{-\infty}^{+\infty}\frac{ik}{\rho s}\frac{e^{-|y|\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}-ikx}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}}dk$$

We can now perform the change of variable $k = {}^{bs}/_{c}$:

$$\tilde{v_x}(x,y,s) = \frac{1}{4\pi\rho c}\int_{-\infty}^{+\infty} ib \frac{e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db \tag{9.7}$$

Here, let us denote $\Xi(b) = \dfrac{ib}{(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}$ and use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma and evaluate our integral on Γ, we need to have that $\lim\limits_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} \frac{iRe^{2i\Theta}}{(\frac{1}{R^2}+e^{2i\Theta})^{\frac{1}{2}}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]} = 0$$

So, we can rewrite (9.7) as:

$$\widetilde{v_x}(x,y,s) = -\frac{1}{4\pi\rho c}\int_{\Gamma} ib \frac{e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db$$

Since we use the same contour as previously, the following quantities remain unchanged:

$$\begin{cases} b = \gamma^{\pm} = -i\dfrac{ct}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\sqrt{t^2-\dfrac{r^2}{c^2}}, \\ (1+\gamma^{\pm 2})^{\frac{1}{2}}\dfrac{y}{c} + i\gamma^{\pm}\dfrac{x}{c} = t, \\ \dfrac{d\gamma^{\pm}}{(1+\gamma^{\pm 2})^{\frac{1}{2}}} = \pm\dfrac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}}. \end{cases}$$

Remarking once again that Γ^+ varies from $+\infty$ to ${}^{r}/_{c}$ and that Γ^- varies from ${}^{r}/_{c}$ to $+\infty$, we obtain the following:

$$\widetilde{v_x}(x,y,s) = \frac{1}{4\pi\rho c}\int_{r/c}^{+\infty}(\gamma^+ + \gamma^-)ie^{-st}\frac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}} = \frac{1}{2\pi\rho}\int_{r/c}^{+\infty}\frac{t}{r}cos\theta e^{-st}\frac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}}$$

Again, by injectivity of the Laplace transform, we obtain the Green's function for the x-velocity field of the acoustic wave equation:

$$\begin{aligned} v_x(x,y,t) &= 0, & t < \frac{r}{c}, \\ v_x(x,y,t) &= \frac{tx}{2\pi\rho r^2\sqrt{t^2-\frac{r^2}{c^2}}}, & t > \frac{r}{c}, \end{aligned}$$

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

9.2.3 Velocity in the y-direction

If we perform a Laplace and Fourier transform to the equation linking the pressure to the y-velocity field in (4.3), we obtain the following:

$$\widehat{v_y} = -\frac{1}{\rho s}\frac{\partial \hat{p}}{\partial y}\,.$$

Thus, injecting the expression of $\hat{p}$ from (9.4) into it, we obtain:

$$\widehat{v_y}(k,y,s) = \frac{e^{-|y|(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2\rho s}.$$

As done previously, we take the inverse Fourier transform of this quantity, which results in:

$$\widetilde{v_y}(x,y,s) = \frac{1}{4\pi\rho}\int_{-\infty}^{+\infty}\frac{1}{s}e^{-|y|\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}-ikx}dk.$$

We can now perform the change of variable $k = {}^{bs}/_{c}$:

$$\widetilde{v_y}(x,y,s) = \frac{1}{4\pi\rho c}\int_{-\infty}^{+\infty} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}db. \tag{9.8}$$

Here, let us denote $\Xi(b) = e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}$ and use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma and evaluate our integral on Γ, we need to have that $\lim\limits_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} Re^{i\Theta}e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]} = 0.$$

So, we can rewrite (9.8) as:

$$\widetilde{v_y}(x,y,s) = -\frac{1}{4\pi\rho c}\int_{\Gamma} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}db$$

Using the same contour as before, hence the same values of $\gamma^{\pm}$, we can derive the following expressions:

$$\begin{cases} b = \gamma^{\pm} = -i\dfrac{ct}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\sqrt{t^2-\dfrac{r^2}{c^2}},\ \dfrac{y}{c} + i\gamma^{\pm}\dfrac{x}{c} = t, \\ \dfrac{d\gamma^{\pm}}{dt} = -i\dfrac{c}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\dfrac{t}{\sqrt{t^2-\frac{r^2}{c^2}}}. \end{cases}$$

Remarking once again that Γ^+ varies from $+\infty$ to ${}^{r}/_{c}$ and that Γ^- varies from ${}^{r}/_{c}$ to $+\infty$, we obtain the following:

$$\widetilde{v_y}(x,y,s) = -\frac{1}{4\pi\rho c}\int_{r/c}^{+\infty}\left(\frac{d\gamma^-}{dt}-\frac{d\gamma^+}{dt}\right)e^{-st}dt = \frac{1}{2\pi\rho}\int_{r/c}^{+\infty}\frac{t}{r}|sin\theta|e^{-st}\frac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}}.$$

Again, by injectivity of the Laplace transform, we obtain the Green's function for the y-velocity field of the acoustic wave equation:

$$\begin{aligned} v_y(x,y,t) &= 0, & t < \frac{r}{c}, \\ v_y(x,y,t) &= \frac{ty}{2\pi\rho r^2\sqrt{t^2-\frac{r^2}{c^2}}}, & t > \frac{r}{c}, \end{aligned}$$

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

From now on, we denote p, v_x et v_y by G_p, G_{v_x} and G_{v_y} respectively.

9.3 Green's functions in the PML in y-direction

In this section, we compute the Green's functions in a Perfectly Matched Layer parallel to the x-axis. Hence, the equations for which we seek Green's functions are different from (9.1). We will proceed in the same manner as previously and apply the Cagniard-De Hoop method to the second-order wave equation with PML in the y-direction, which can be written as:

$$\frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} - \frac{\partial^2 P}{\partial x^2} - (\frac{\partial}{\partial t}+\sigma_y)^{-2}\frac{\partial^2}{\partial t^2}\frac{\partial^2 P}{\partial y^2} = \delta(x)\delta(y)f(t) \tag{9.9}$$

where σ_y is the absorbing coefficient in the PML, which is constant. We will, as before, compute the Green's functions with absorption in the y-direction for the pressure and the velocity fields.

9.3.1 Pressure

The Green's function associated to the pressure field is defined as the solution to (9.10) and the solution to (9.9) is then given by $P = p * f$, where $*$ denotes the convolutional product. Thus, the problem can be rewritten as follows:

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - \frac{\partial^2 p}{\partial x^2} - (\frac{\partial}{\partial t}+\sigma_y)^{-2}\frac{\partial^2}{\partial t^2}\frac{\partial^2 p}{\partial y^2} = \delta(x)\delta(y)\delta(t). \tag{9.10}$$

We then introduce the Laplace transform of p and, remarking that the Laplace transform of a Dirac distribution is 1, we obtain that:

$$\frac{s^2}{c^2}\tilde{p} - \frac{\partial^2\tilde{p}}{\partial x^2} - \frac{s^2}{(s+\sigma_y)^2}\frac{\partial^2\tilde{p}}{\partial y^2} = \delta(x)\delta(y). \tag{9.11}$$

The next step consists in applying a Fourier transform to $\tilde{p}$ and, also remarking that the Fourier transform of a Dirac distribution is 1, the previous equation becomes:

$$-\frac{s}{s+\sigma_y}\frac{\partial}{\partial y}(\frac{s}{s+\sigma_y}\frac{\partial\hat{p}}{\partial y}) + (k^2+\frac{s^2}{c^2})\hat{p} = \delta(y). \tag{9.12}$$

Here, the procedure differs lightly from the previous section. We perform the following change of variable $Y = y(1 + \frac{\sigma_y}{s})$ in (9.12), which results in the following system:

$$-\frac{\partial^2 \hat{p}}{\partial Y^2} + (k^2 + \frac{s^2}{c^2})\hat{p} = \delta(Y). \tag{9.13}$$

Thus, the system (9.13) can be explicitly solved in the same manner as (9.3), which results in:

$$\hat{p}(k, Y, s) = \frac{e^{-|Y|(k^2+\frac{s^2}{c^2})^{\frac{1}{2}}}}{2(k^2 + \frac{s^2}{c^2})^{\frac{1}{2}}}.$$

Now, by going back to the variable y, we finally obtain the Fourier transform $\hat{p}$ of the pressure field:

$$\hat{p}(k, y, s) = \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}|y(1+\frac{\sigma_y}{s})|}}{2\left(k^2 + \frac{s^2}{c^2}\right)^{\frac{1}{2}}}. \tag{9.14}$$

The next step of the Cagniard-De Hoop method consists in performing an inverse Fourier transform along x on $\hat{p}$, which gives the following:

$$\widetilde{p} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}y(1+\frac{\sigma_y}{s})-ikx}}{\left(k^2 + \frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

Now, we apply the change of variable $k = bs/c$, which results in:

$$\widetilde{p} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1 + b^2)^{\frac{1}{2}}} db. \tag{9.15}$$

As explained previously, the motivation behind this change of variable is to obtain an integral in the form $g(x, y, b)e^{-sf(x,y,b)}$, which is the case here and we denote

$$\Xi(b) = \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1 + b^2)^{\frac{1}{2}}}.$$

The last step of the method consists in finding a complex path Γ such that $f(x, y, b) = t$, in order to identify a Laplace transform. Here, this means that we look for a path Γ which is the same one as in the standard case (9.6). Let us verify that $\Xi(b)$ meets the requirement of Jordan's lemma in order to rewrite (9.15) on Γ. We use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma, we need to have that $\lim_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining that:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} \frac{e^{i\Theta}}{(\frac{1}{R^2} + e^{2i\Theta})^{\frac{1}{2}}} e^{-\frac{y\sigma_y}{c}(1+R^2e^{2i\Theta})^{\frac{1}{2}}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]} = 0.$$

So, we can rewrite (9.15) as:

$$\widetilde{p} = -\frac{1}{4\pi}\int_{\Gamma}\frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}}e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}}db.$$

Since we found the same contour as previously, we also obtain the same values of $\gamma^{\pm}$ and so we can derive the following expressions:

$$\begin{cases} b = \gamma^{\pm} = -i\dfrac{ct}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\sqrt{t^2-\dfrac{r^2}{c^2}}, \\ (1+\gamma^{\pm 2})^{\frac{1}{2}}\dfrac{y}{c} + i\gamma^{\pm}\dfrac{x}{c} = t, \\ \dfrac{d\gamma^{\pm}}{(1+\gamma^{\pm 2})^{\frac{1}{2}}} = \pm\dfrac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}}, \\ \dfrac{(1+\gamma^{\pm 2})^{\frac{1}{2}}}{c} = \dfrac{t}{r}|sin\theta| \mp \dfrac{i}{r}cos\theta\sqrt{t^2-\dfrac{r^2}{c^2}}. \end{cases}$$

Remarking as before that Γ^+ varies from $+\infty$ to r/c and that Γ^- varies from r/c to $+\infty$, we obtain the following:

$$\widetilde{p} = \frac{1}{2\pi}\int_{r/c}^{+\infty} e^{-A(x,y,t)}cos[B(x,y,t)]\frac{e^{-st}}{\sqrt{t^2-\frac{r^2}{c^2}}}dt.$$

Then, by injectivity of the Laplace transform, we obtain the Green's function for the pressure field of the acoustic wave equation with absorption in the y-direction:

$$\begin{aligned} p &= 0, && t < \frac{r}{c}, \\ p &= e^{-A(x,y,t)}cos[B(x,y,t)]G_p\,, && t > \frac{r}{c}, \end{aligned}$$

with

$$\begin{cases} A(x,y,t) = y\sigma_y\dfrac{t}{r}|sin\theta| = \sigma_y\dfrac{ty^2}{r^2}\,, \\ B(x,y,t) = \dfrac{y}{r}cos\theta\sqrt{t^2-\dfrac{r^2}{c^2}} = \sigma_y\dfrac{xy}{r^2}\sqrt{t^2-\dfrac{r^2}{c^2}}\,, \\ r = \sqrt{x^2+y^2}\,. \end{cases}$$

We remark that when setting $\sigma_y = 0$, we retrieve the Green's function for the acoustic pressure without PML.

9.3.2 Velocity in the x-direction

If we perform a Laplace and Fourier transform to the equation linking the pressure and the x-velocity field in (8.2), we obtain the following:

$$\hat{u} = \frac{ik}{\rho s}\hat{p}\,.$$

Thus, injecting the expression of $\hat{p}$ from (9.14) into it, we obtain:

$$\hat{u} = \frac{ik}{\rho s} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}|y|(1+\frac{\sigma_y}{s})}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}}$$

The next step of the method consists in taking the inverse Fourier transform of this quantity, which results in:

$$\widetilde{u} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{ik}{\rho s} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}|y|(1+\frac{\sigma_y}{s})-ikx}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

We can now perform the change of variable $k = {}^{bs}/_{c}$:

$$\widetilde{u} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} ib \frac{e^{-\frac{|y|\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db. \tag{9.16}$$

Here, let us denote $\Xi(b) = \frac{ib}{(1+b^2)^{\frac{1}{2}}} e^{-\frac{|y|\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}$ and use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma and evaluate our integral on Γ, we need to have that $\lim\limits_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} \frac{iRe^{2i\Theta}}{(\frac{1}{R^2}+e^{2i\Theta})^{\frac{1}{2}}} e^{-\frac{y\sigma_y}{c}(1+R^2e^{2i\Theta})^{\frac{1}{2}}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]} = 0.$$

So, we can rewrite (9.16) as:

$$\widetilde{u} = -\frac{1}{4\pi\rho c}\int_{\Gamma} ib \frac{e^{-\frac{|y|\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db.$$

We have the same contour Γ as before, so we can derive the following expressions:

$$\begin{cases} b = \gamma^{\pm} = -i\dfrac{ct}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\sqrt{t^2-\dfrac{r^2}{c^2}}, \\ (1+\gamma^{\pm 2})^{\frac{1}{2}}\dfrac{y}{c} + i\gamma^{\pm}\dfrac{x}{c} = t, \\ \dfrac{d\gamma^{\pm}}{(1+\gamma^{\pm 2})^{\frac{1}{2}}} = \pm\dfrac{dt}{\sqrt{t^2-\frac{r^2}{c^2}}}, \\ \dfrac{(1+\gamma^{\pm 2})^{\frac{1}{2}}}{c} = \dfrac{t}{r}|sin\theta| \mp \dfrac{i}{r}cos\theta\sqrt{t^2-\dfrac{r^2}{c^2}}. \end{cases}$$

Remarking as before that Γ^+ varies from $+\infty$ to r/c and that Γ^- varies from r/c to $+\infty$, we obtain the following:

$$\widetilde{u} = \frac{1}{2\pi\rho}\int_{r/c}^{+\infty}\frac{1}{r}\left(tcos\theta cos[B(x,y,t)] - |sin\theta|\sqrt{t^2-\frac{r^2}{c^2}}sin[B(x,y,t)]\right)\frac{e^{-A(x,y,t)}e^{-st}}{\sqrt{t^2-\frac{r^2}{c^2}}}dt.$$

Then, by injectivity of the Laplace transform, we obtain the Green's function for the x-velocity field of the acoustic wave equation with absorption in the y-direction:

$$\begin{aligned} u &= 0, && t < \frac{r}{c}, \\ u &= \frac{1}{\rho r^2}\left(txcos[B(x,y,t)] - y\sqrt{t^2-\frac{r^2}{c^2}}sin[B(x,y,t)]\right)e^{-A(x,y,t)}G_p\,, && t > \frac{r}{c}, \end{aligned}$$

with

$$\begin{cases} A(x,y,t) = y\sigma_y\dfrac{t}{r}|sin\theta| = \sigma_y\dfrac{t}{r^2}y\,, \\ B(x,y,t) = \dfrac{y\sigma_y}{r}cos\theta\sqrt{t^2-\dfrac{r^2}{c^2}} = \sigma_y\dfrac{xy}{r^2}\sqrt{t^2-\dfrac{r^2}{c^2}}\,, \\ r = \sqrt{x^2+y^2}\,. \end{cases}$$

9.3.3 Velocity in the y-direction

If we perform a Laplace and Fourier transform to the equation linking the pressure and the y-velocity field in (8.2), we obtain the following:

$$\tilde{v} = -\frac{1}{\rho(s+\sigma)}\frac{\partial\tilde{p}}{\partial y}\,.$$

Thus, injecting the expression of $\hat{p}$ from (9.14) into it, we obtain:

$$\hat{v} = \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}y(1+\frac{\sigma_y}{s})}}{2\rho s}.$$

The next step of the method consists in taking the inverse Fourier transform of this quantity, which results in:

$$\widetilde{v} = \frac{1}{4\pi\rho}\int_{-\infty}^{+\infty}\frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}y(1+\frac{\sigma_y}{s})-ikx}}{s}dk.$$

We can now perform the change of variable $k = bs/c$:

$$\widetilde{v} = \frac{1}{4\pi\rho}\int_{-\infty}^{+\infty}e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}}e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}db. \tag{9.17}$$

Here, let us denote $\Xi(b) = e^{-\frac{|y|\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{|y|}{c}+ib\frac{x}{c}\right]}$ and use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma and evaluate our integral on Γ, we need to have that $\lim\limits_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} Re^{i\Theta} e^{-\frac{y\sigma_y}{c}(1+R^2e^{2i\Theta})^{\frac{1}{2}}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]} = 0.$$

So, we can rewrite (9.17) as:

$$\widetilde{v} = -\frac{1}{4\pi\rho c}\int_{\Gamma} e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]} db.$$

We have the same contour Γ as before, so we can derive the following expressions:

$$\begin{cases} b = \gamma^{\pm} = -i\dfrac{ct}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\sqrt{t^2-\dfrac{r^2}{c^2}}, \\ (1+\gamma^{\pm 2})^{\frac{1}{2}}\dfrac{y}{c} + i\gamma^{\pm}\dfrac{x}{c} = t, \\ \dfrac{d\gamma^{\pm}}{dt} = -i\dfrac{c}{r}cos\theta \pm \dfrac{c}{r}|sin\theta|\dfrac{t}{\sqrt{t^2-\frac{r^2}{c^2}}}, \\ \dfrac{(1+\gamma^{\pm 2})^{\frac{1}{2}}}{c} = \dfrac{t}{r}|sin\theta| \mp \dfrac{i}{r}cos\theta\sqrt{t^2-\dfrac{r^2}{c^2}}. \end{cases}$$

Remarking as before that Γ^+ varies from $+\infty$ to r/c and that Γ^- varies from r/c to $+\infty$, we obtain the following:

$$\widetilde{v_y} = -\frac{1}{2\pi\rho}\int_{r/c}^{+\infty}\frac{1}{r}\left(-cos\theta sin[B(x,y,t)] - |sin\theta|\frac{t}{r\sqrt{t^2-\frac{r^2}{c^2}}}cos[B(x,y,t)]\right)e^{-A(x,y,t)}e^{-st}dt.$$

Then, by injectivity of the Laplace transform, we obtain the Green's function for the y-velocity field of the acoustic wave equation with absorption in the y-direction:

$$\begin{aligned} v_y &= 0, && t < \frac{r}{c} \\ v_y &= \frac{1}{\rho r^2}\left(tycos[B(x,y,t)] + x\sqrt{t^2-\frac{r^2}{c^2}}sin[B(x,y,t)]\right)e^{-A(x,y,t)}G_p\,, && t > \frac{r}{c} \end{aligned}$$

with

$$\begin{cases} A(x,y,t) = y\sigma_y\dfrac{t}{r}|sin\theta| = y\sigma_y\dfrac{t}{r^2}|y| \\ B(x,y,t) = \dfrac{y\sigma_y}{r}cos\theta\sqrt{t^2-\dfrac{r^2}{c^2}} = \dfrac{y\sigma_y}{r^2}x\sqrt{t^2-\dfrac{r^2}{c^2}} \end{cases}$$

From now one, we denote p, v_x et v_y by $G_p^{pml_y}$, $G_{v_x}^{pml_y}$ and $G_{v_y}^{pml_y}$ respectively.

9.4 Green's functions in the PML in x- and y-direction

In this section, we compute the Green's functions for the second-order wave equation with PML in the x- and y-direction using the Cagniard-De Hoop method. which can be written as:

$$\frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} - (\frac{\partial}{\partial t} + \sigma_x)^{-2}\frac{\partial^2}{\partial t^2}\frac{\partial^2 P}{\partial x^2} - (\frac{\partial}{\partial t} + \sigma_y)^{-2}\frac{\partial^2}{\partial t^2}\frac{\partial^2 P}{\partial y^2} = \delta(x)\delta(y)f(t). \tag{9.18}$$

where σ_x is the absorbing coefficient in the x-direction and σ_y is the absorbing coefficient in the y-direction, which are both constants. We will, as before, compute the Green's functions with absorption in both directions for the pressure and the velocity fields.

9.4.1 Pressure

The Green's function associated to the pressure field is defined as the solution to (9.19) and the solution to (9.18) is then given by $P = p * f$, where $*$ denotes the convolutional product. Thus, the problem can be rewritten as follows:

$$\frac{1}{c^2}\frac{\partial^2 p}{\partial t^2} - (\frac{\partial}{\partial t} + \sigma_x)^{-2}\frac{\partial^2}{\partial t^2}\frac{\partial^2 p}{\partial x^2} - (\frac{\partial}{\partial t} + \sigma_y)^{-2}\frac{\partial^2}{\partial t^2}\frac{\partial^2 p}{\partial y^2} = \delta(x)\delta(y)\delta(t) \tag{9.19}$$

We then introduce the Laplace transform of p and, remarking that the Laplace transform of a Dirac distribution is 1, we obtain the following:

$$\frac{s^2}{c^2}\tilde{p} - \frac{s^2}{(s+\sigma_x)}\frac{\partial}{\partial x}(\frac{s^2}{(s+\sigma_y)}\frac{\partial \tilde{p}}{\partial x}) - \frac{s^2}{(s+\sigma_y)^2}\frac{\partial^2 \tilde{p}}{\partial y^2} = \delta(x)\delta(y).$$

We use the same idea as before, and perform the following change of variable $X = x(1 + \frac{\sigma_x}{s})$, which results in:

$$\frac{s^2}{c^2}\tilde{p} - \frac{\partial^2 \tilde{p}}{\partial X^2} - \frac{s^2}{(s+\sigma_y)^2}\frac{\partial^2 \tilde{p}}{\partial y^2} = \delta(X)\delta(y).$$

We obtain an equation similar to (9.11), so we carry on the Cagniard-De Hoop method as we did in the previous section. The next step consists in applying a Fourier transform along X to $\tilde{p}$ and, also remarking that the Fourier transform of a Dirac distribution is 1, the previous equation becomes:

$$-\frac{s}{s+\sigma_y}\frac{\partial}{\partial y}(\frac{s}{s+\sigma_y}\frac{\partial \hat{p}}{\partial y}) + (k^2 + \frac{s^2}{c^2})\hat{p} = \delta(y). \tag{9.20}$$

Again, we perform the following change of variable $Y = y(1 + \frac{\sigma_y}{s})$ in (9.20), which leads to the following:

$$-\frac{\partial^2 \hat{p}}{\partial Y^2} + (k^2 + \frac{s^2}{c^2})\hat{p} = \delta(Y).$$

Thus, system (9.20) can be explicitly solved and we obtain:

$$\hat{p}(k, Y, s) = \frac{e^{-|Y|(k^2 + \frac{s^2}{c^2})^{\frac{1}{2}}}}{2(k^2 + \frac{s^2}{c^2})^{\frac{1}{2}}}.$$

Then, by going back to the variable y, we obtain the Fourier transform along x of the pressure field denoted $\hat{p}$:

$$\hat{p}(k,y,s) = \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left|y(1+\frac{\sigma_y}{s})\right|}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}}. \tag{9.21}$$

The next step of the Cagniard-De Hoop method consists in performing an inverse Fourier transform along X on $\hat{p}$, which gives the following:

$$\widetilde{p} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}} y(1+\frac{\sigma_y}{s})-ikX}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

Going back to the variable x, we obtain the following:

$$\widetilde{p} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)-ik(x+\frac{x\sigma_x}{s})}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

Now, we apply the change of variable $k = bs/c$, which results in:

$$\widetilde{G}_p^{\text{pml}} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db. \tag{9.22}$$

As explained previously, the motivation behind this change of variable is to obtain an integral in the form $g(x,y,b)e^{-sf(x,y,b)}$, which is the case here and we denote:

$$\Xi(b) = \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}}.$$

The last step of the method consists in considering b as a complex variable and in finding a complex path Γ such that $f(x,y,b) = t$, in order to identify a Laplace transform. We obtain the same path Γ as in the previous computations. Let us verify that $\Xi(b)$ meets the requirement of Jordan's lemma in order to rewrite (9.22) on Γ. We use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma, we need to have that $\lim_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} \frac{e^{i\Theta}}{(\frac{1}{R^2}+e^{2i\Theta})^{\frac{1}{2}}} e^{-\frac{y\sigma_y}{c}(1+R^2e^{2i\Theta})^{\frac{1}{2}}-iRe^{i\Theta}\frac{x\sigma_x}{c}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]}$$

$$= 0\,.$$

So, we can rewrite (9.22) as:

$$\widetilde{G}_p^{\text{pml}} = -\frac{1}{4\pi}\int_\Gamma \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}} - ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c} + ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db.$$

Using the same contour Γ as before and remarking, once again, that Γ^+ varies from $+\infty$ to r/c and that Γ^- varies from r/c to $+\infty$, we obtain the following:

$$\widetilde{p} = \frac{1}{2\pi}\int_{r/c}^{+\infty} e^{-A(x,y,t)} cos[B(x,y,t)] \frac{e^{-st}}{\sqrt{t^2 - \frac{r^2}{c^2}}} dt.$$

Then, by injectivity of the Laplace transform, we obtain the Green's function for the pressure field of the acoustic wave equation with absorption in the x- and y-direction:

$$\begin{aligned} p &= 0, && t < \frac{r}{c}, \\ p &= e^{-A(x,y,t)} cos[B(x,y,t)] G_p\,, && t > \frac{r}{c}, \end{aligned}$$

with

$$\begin{cases} A(x,y,t) = \dfrac{t}{r}(y\sigma_y|sin\theta| + x\sigma_x cos\theta) = \left(\sigma_y y^2 + \sigma_x x^2\right)\dfrac{t}{r^2}\,, \\ B(x,y,t) = \dfrac{1}{r}\sqrt{t^2 - \dfrac{r^2}{c^2}}(y\sigma_y cos\theta - x\sigma_x|sin\theta|) = (\sigma_y - \sigma_x)\dfrac{xy}{r^2}\sqrt{t^2 - \dfrac{r^2}{c^2}}\,, \\ r = \sqrt{x^2+y^2} \end{cases}$$

We remark here that when setting $\sigma_x = 0$, we retrieve the Green's function for the acoustic pressure with a single PML in the y-direction, and when setting both $\sigma_x = \sigma_y = 0$, we retrieve the Green's function for the acoustic pressure without PML.

9.4.2 Velocity in the x-direction

If we perform a Laplace transform to the equation linking the pressure and the x-velocity field in (8.12), we obtain the following:

$$\rho s\widetilde{v_x} + \frac{s}{s+\sigma_x}\frac{\partial \tilde{p}}{\partial x} = 0\,.$$

As done previously, we apply the change of variable $X = x(1 + \dfrac{\sigma_x}{s})$ to the obtained Laplace transform:

$$\rho s\widetilde{v_x} + \frac{\partial \tilde{p}}{\partial X} = 0\,.$$

Now, we can move on to the next step of the Cagniard-De Hoop method and apply a Fourier transform along X to the previous equation:

$$\widehat{v_x} = \frac{ik}{\rho s}\hat{p}\,.$$

Thus, injecting the expression of $\hat{p}$ from (9.21) into it, we obtain:

$$\widehat{v_x} = \frac{ik}{\rho s} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)}}{2\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}}.$$

The next step of the method consists in taking the inverse Fourier transform of this quantity, which results in:

$$\widetilde{v_x} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{ik}{\rho s} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)-ikX}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

Now, going back to the variable x, we obtain the following:

$$\widetilde{v_x} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} \frac{ik}{\rho s} \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)-ik(x+\frac{x\sigma_x}{s})}}{\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}} dk.$$

We can now perform the change of variable $k = {}^{bs}/_{c}$:

$$\widetilde{v_x} = \frac{1}{4\pi}\int_{-\infty}^{+\infty} ib \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db. \tag{9.23}$$

Here, let us denote

$$\Xi(b) = ib \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}}$$

and use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma and evaluate our integral on Γ, we need to have that $\lim\limits_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} \frac{iRe^{2i\Theta}}{(\frac{1}{R^2}+e^{2i\Theta})^{\frac{1}{2}}} e^{-\frac{y\sigma_y}{c}(1+R^2e^{2i\Theta})^{\frac{1}{2}}-iRe^{i\Theta}\frac{x\sigma_x}{c}} e^{-sR\left[(1+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]}$$

$$= 0\,.$$

So, we can rewrite (9.23) as:

$$\widetilde{v_x} = -\frac{1}{4\pi\rho c}\int_{\Gamma} ib \frac{e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}}{(1+b^2)^{\frac{1}{2}}} db.$$

We have the same contour Γ as before, and noticing that Γ^+ varies from $+\infty$ to ${}^{r}/_{c}$ and that Γ^- varies from ${}^{r}/_{c}$ to $+\infty$, we obtain the following:

$$\widetilde{v_x} = \frac{1}{2\pi\rho}\int_{r/c}^{+\infty}\frac{1}{r}\left(tcos\theta cos[B(x,y,t)] + |sin\theta|\sqrt{t^2-\frac{r^2}{c^2}}sin[B(x,y,t)]\right)\frac{e^{-A(x,y,t)}e^{-st}}{\sqrt{t^2-\frac{r^2}{c^2}}}dt.$$

Then, by injectivity of the Laplace transform, we obtain the Green's function for the x-velocity field of the acoustic wave equation with absorption in the x- and y-direction:

$$\begin{aligned} v_x &= 0, && t < \frac{r}{c}, \\ v_x &= \frac{1}{\rho r^2}\left(txcos[B(x,y,t)] + y\sqrt{t^2-\frac{r^2}{c^2}}sin[B(x,y,t)]\right)e^{-A(x,y,t)}G_p\,, && t > \frac{r}{c}, \end{aligned}$$

with

$$\begin{cases} A(x,y,t) = \dfrac{t}{r}(y\sigma_y|sin\theta| + x\sigma_x cos\theta) = \left(\sigma_y y^2 + \sigma_x x^2\right)\dfrac{t}{r^2}\,, \\ B(x,y,t) = \dfrac{1}{r}\sqrt{t^2-\dfrac{r^2}{c^2}}(x\sigma_x|sin\theta| - y\sigma_y cos\theta) = (\sigma_x-\sigma_y)\dfrac{xy}{r^2}\sqrt{t^2-\dfrac{r^2}{c^2}}\,, \\ r = \sqrt{x^2+y^2}\,. \end{cases}$$

9.4.3 Velocity in the y-direction

If we perform a Laplace and a Fourier transform to the equation linking the pressure and the y-velocity field in (8.12), we obtain the following:

$$\widetilde{v_y} = -\frac{1}{\rho(s+\sigma)}\frac{\partial\tilde{p}}{\partial y}\,.$$

Thus, injecting the expression of $\widehat{p}$ from (9.21) into it, we obtain:

$$\widehat{v_y} = \frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)}}{2\rho s}.$$

The next step of the method consists in taking the inverse Fourier transform along X of this quantity, which results in:

$$\widetilde{v_y} = \frac{1}{4\pi\rho}\int_{-\infty}^{+\infty}\frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)-ikX}}{s}dk.$$

Now, going back to the variable x, we obtain:

$$\widetilde{v_y} = \frac{1}{4\pi\rho}\int_{-\infty}^{+\infty}\frac{e^{-\left(k^2+\frac{s^2}{c^2}\right)^{\frac{1}{2}}\left(y(1+\frac{\sigma_y}{s})\right)-ik(x+\frac{x\sigma_x}{s})}}{s}dk.$$

We can now perform the change of variable $k = {}^{bs}/_{c}$:

$$\widetilde{v_y} = -\frac{1}{4\pi\rho c}\int_{\Gamma} e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]} db.$$

As explained previously, the motivation behind this change of variable is to obtain an integral in the form $g(x,y,b)e^{-sf(x,y,b)}$, in order to identify a Laplace transform, which is the case here and we denote

$$\Xi(b) = e^{-\frac{y\sigma_y}{c}(1+b^2)^{\frac{1}{2}}-ib\frac{x\sigma_x}{c}} e^{-s\left[(1+b^2)^{\frac{1}{2}}\frac{y}{c}+ib\frac{x}{c}\right]}.$$

The last step of the method consists in finding a complex path Γ such that $f(x,y,b) = t$, in order to identify a Laplace transform. We obtain the same path Γ as in the previous computations. Let us verify that $\Xi(b)$ meets the requirement of Jordan's lemma in order to rewrite (9.22) on Γ. We use the same contour as before $D_d \cup \Gamma_d \cup C_d$. In order to apply Jordan's lemma, we need to have that $\lim_{|b|\to+\infty} b\,\Xi(b) = 0$. As previously, we write b in its exponential form, thus obtaining the following:

$$\lim_{R\to\ +\infty} Re^{i\Theta}\,\Xi(Re^{i\Theta}) = \lim_{R\to\ +\infty} Re^{i\Theta} e^{-\frac{y\sigma_y}{c}(1+R^2e^{2i\Theta})^{\frac{1}{2}}-iRe^{i\Theta}\frac{x\sigma_x}{c}} e^{-sR\left[(\frac{1}{R^2}+e^{2i\Theta})^{\frac{1}{2}}\frac{|y|}{c}+ie^{i\Theta}\frac{x}{c}\right]} = 0.$$

We have the same contour Γ as before, and noticing that Γ^+ varies from $+\infty$ to ${}^{r}/_{c}$ and that Γ^- varies from ${}^{r}/_{c}$ to $+\infty$, we obtain the following:

$$\widetilde{v_y} = \frac{1}{2\pi\rho}\int_{r/c}^{+\infty}\frac{1}{r}\left(cos\theta sin[B(x,y,t)] + |sin\theta|\frac{t}{r\sqrt{t^2-\frac{r^2}{c^2}}}cos[B(x,y,t)]\right)e^{-A(x,y,t)}e^{-st}dt.$$

$$\begin{aligned}
v_y &= 0, && t < \frac{r}{c},\\
v_y &= \frac{1}{\rho r^2}\left(tycos[B(x,y,t)] + x\sqrt{t^2-\frac{r^2}{c^2}}sin[B(x,y,t)]\right)e^{-A(x,y,t)}G_p\,, && t > \frac{r}{c},
\end{aligned}$$

with

$$\begin{cases}
A(x,y,t) = \dfrac{t}{r}(y\sigma_y|sin\theta| + x\sigma_x cos\theta) = \left(\sigma_y y^2 + \sigma_x x^2\right)\dfrac{t}{r^2}\,,\\[2ex]
B(x,y,t) = \dfrac{1}{r}\sqrt{t^2-\dfrac{r^2}{c^2}}(y\sigma_y cos\theta - x\sigma_x|sin\theta|) = (\sigma_y-\sigma_x)\dfrac{xy}{r^2}\sqrt{t^2-\dfrac{r^2}{c^2}}\,,\\[2ex]
r = \sqrt{x^2+y^2}\,.
\end{cases}$$

From now one, we denote p, v_x et v_y by $G_p^{pml_{xy}}$, $G_{v_x}^{pml_{xy}}$ and $G_{v_y}^{pml_{xy}}$ respectively.

9.5 Other basis functions

When working with Trefftz methods, we use local solutions as basis functions. Since a linear combination of solutions to a problem results in another solution to the problem, we can combine the Trefftz functions to obtain other kinds of basis functions.

In fact in our framework, if we take

$$p = (\frac{\partial}{\partial t} + \sigma_x)(\frac{\partial}{\partial t} + \sigma_y)G_p^{pml_{xy}} ,$$

we obtain another solution to the acoustic wave equation, where $G_p^{pml_{xy}}$ is the Green's function associated to the pressure of the acoustic wave equation with PML in x and y computed previously. If we inject this into the first and second equations of (8.12), we obtain:

$$\begin{cases} v_x = -\frac{1}{\rho}(\frac{\partial}{\partial t} + \sigma_y)\frac{\partial G_p^{pml_{xy}}}{\partial x} , \\ v_y = -\frac{1}{\rho}(\frac{\partial}{\partial t} + \sigma_x)\frac{\partial G_p^{pml_{xy}}}{\partial y} . \end{cases}$$

Thus, we can express all of our solutions as functions of $G_p^{pml_{xy}}$ and do not need to compute the Green's functions for v_x and v_y.

In the case of auxiliary variables, this technique also comes in handy. Let us also define

$$v_x = (\frac{\partial}{\partial t} + \sigma_x)(\frac{\partial}{\partial t} + \sigma_y)G_{v_x}^{pml_{xy}} ,$$
$$v_y = (\frac{\partial}{\partial t} + \sigma_x)(\frac{\partial}{\partial t} + \sigma_y)G_{v_y}^{pml_{xy}} ,$$

If we inject p, v_x and v_y into the last four equations of (8.13), we obtain the following:

$$p_1^a = \sigma_x(\frac{\partial}{\partial t} + \sigma_y)\frac{\partial G_p^{pml_{xy}}}{\partial x} ,$$
$$p_2^a = \sigma_y(\frac{\partial}{\partial t} + \sigma_x)\frac{\partial G_p^{pml_{xy}}}{\partial y} ,$$
$${v_x}^a = \sigma_x(\frac{\partial}{\partial t} + \sigma_y)\frac{\partial G_{v_x}^{pml_{xy}}}{\partial x} ,$$
$${v_y}^a = \sigma_y(\frac{\partial}{\partial t} + \sigma_x)\frac{\partial G_{v_y}^{pml_{xy}}}{\partial y} .$$

Thus, we can easily compute the auxiliary basis functions in terms of $G_p^{pml_{xy}}$, $G_{v_x}^{pml_{xy}}$ and $G_{v_y}^{pml_{xy}}$.

Chapter 10

Implementation

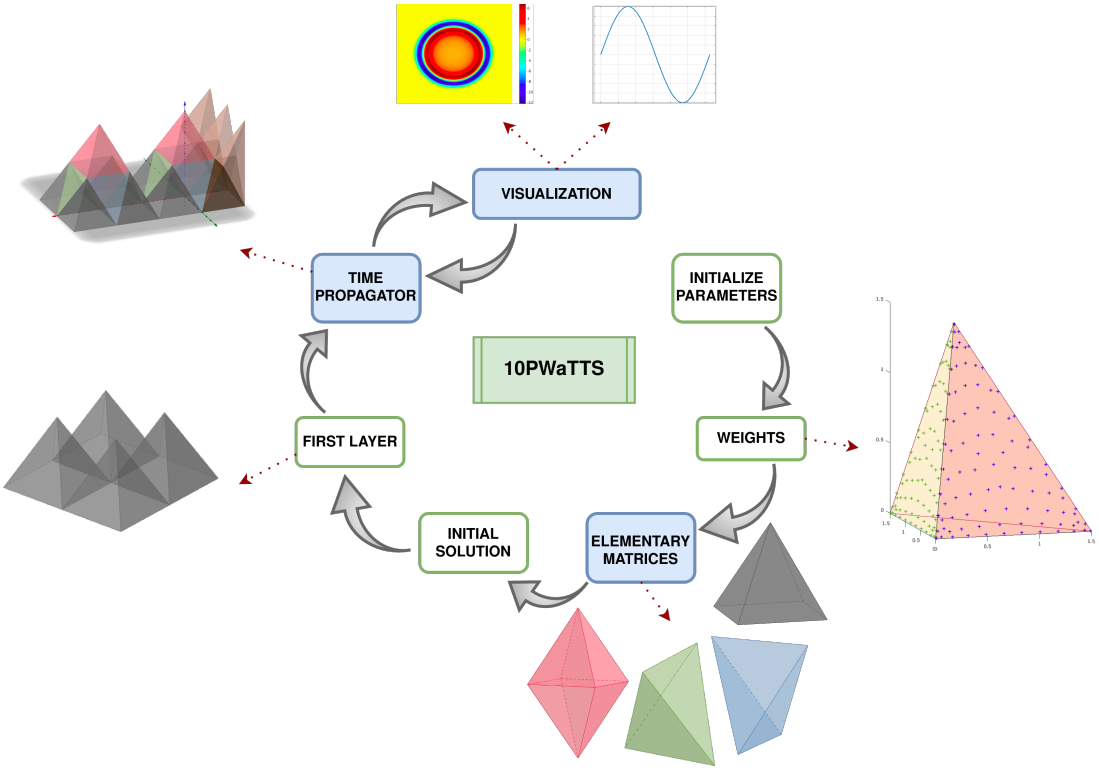


Figure 10.1: Flowchart for the PML case, where the differences from the non-PML case are the bricks depicted in blue

In Part I of the thesis, we presented the implementation process of the Trefftz-DG method with Tent-Pitching, without PML. In this chapter, we aim to detail the implementation of Perfectly Matched Layers into the structured Trefftz-DG Tent-Pitching solver. The overall implementation process stays the same and the changes operate in the details of each brick. The flowchart 10.1 summarizes the steps of the implementation, but let us

recall it in a few words with the changes that need to be taken into account.

1. Initializing the parameters: this function is mostly unchanged, except for the number of degrees of freedom. Since we are not using polynomials but Green's functions, the number of degrees of freedom will be adapted to the Green's functions. The choices of degrees of freedom and their position will be described in section 10.1.

2. Computing the Gaussian weights: this function does not change and is independent from the choice of basis functions. The 2D Gaussian points and weights we use can be found in Appendix B.

3. Computing the elementary matrices: this is where the main changes happen. The elementary matrices are calculated using the basis functions, hence they will be completely different when choosing a different basis. Thus, each time we change the basis (polynomials, Green's functions without absorption, with absorption in one direction or both), we obtain a new set of elementary matrices. We describe each new set of matrices we obtain and how we obtain them in each section.

4. Computing the initial solution: basis functions intervene in the calculation of the initial solution. So, this function is also modified.

5. Constructing the first layer of pyramids: this is unchanged. Once we have the elementary matrices, we do not need basis functions, so the construction of the first layer of pyramids along with the computation of the solution in them is done as previously.

6. Propagating the solution through time: when working in a homogeneous domain, this brick is unchanged. Only elementary matrices are needed in this function, which are pre-computed so nothing changes. However, when considering a bilayered domain, an interface appears and we need to handle it when propagating the solution. So, the time propagation is modified in the second and third phase, when dealing with one or multiple PMLs.

7. Visualization: this function changes here, because we need to reconstruct the global solution in order to visualize it, so, we need the basis functions.

This chapter is articulated based on three different phases of our work, which are:

- Phase I: the first phase consists in substituting the polynomials for Green's functions without PML, which we analytically computed in the previous chapter. We then add absorption in the whole domain by replacing the Green's function with their absorbing analogs,

- Phase II: The second phase addresses the addition of a PML, thus the coupling between the domain of interest with standard Green's functions and the PML with absorbing Green's functions.

- Phase III: In the third phase, we describe the extension to multiple PMLs surrounding the computational domain. Then, we consider the coupling between a domain approximated with polynomials and PMLs with absorbing Green's functions.

10.1 Phase I: Implementation of Green's functions in entire domain

For the first part of phase I, we start by simply substituting the basis and test functions with Green's functions without absorption, thus $\sigma_x = \sigma_y = 0$. Then, we will consider the Green's functions with absorption $\sigma_y > 0$.

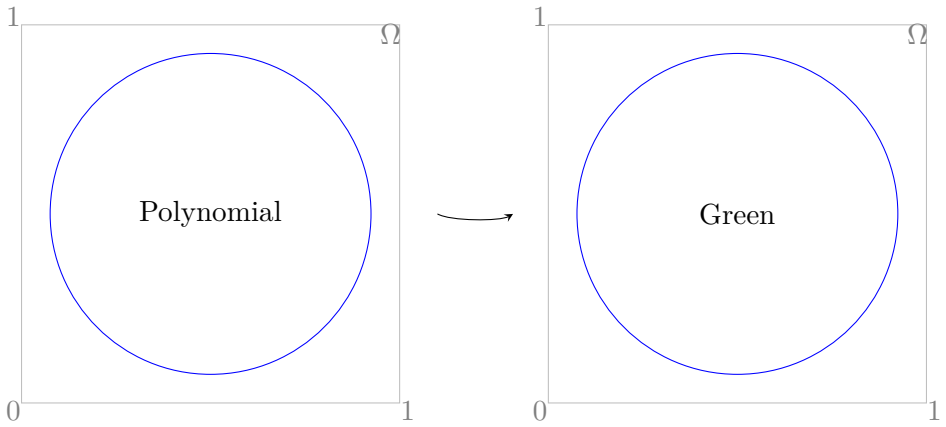


Figure 10.2: In the first phase, we substitute the polynomials for Green's functions

10.1.1 Trefftz-DG method with Green's functions on Tent-Pitching meshes

Let us recall the expression of the elementary matrices and where they come from. Solving the first order acoustic wave equation with the Trefftz-DG method and Tent-Pitching meshes is equivalent to solving the following system:

$$M \begin{pmatrix} p \\ \mathbf{v} \end{pmatrix} = K \begin{pmatrix} p^{in} \\ \mathbf{v}^{in} \end{pmatrix}$$

Or equivalently:

$$MU^n = KU^{n-1}$$

with

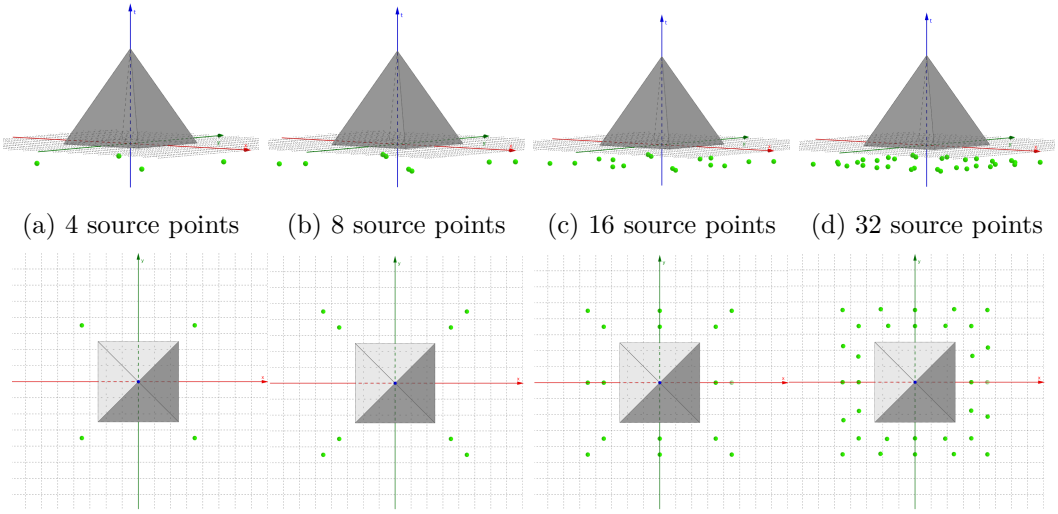
$$\begin{aligned}
M_{ij} =& \int_{\mathcal{F}^{out}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \phi_j^{\mathbf{v}} \cdot \phi_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \phi_i^{\mathbf{w}} + \phi_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \phi_j^{\mathbf{v}} \cdot \phi_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \phi_i^{\mathbf{w}} + \phi_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ \int_{\mathcal{F}^{ext}} \phi_p \phi_{\mathbf{w}} \cdot \mathbf{n} + \gamma (\phi_{\mathbf{v}} \cdot \mathbf{n})(\phi_{\mathbf{w}} \cdot \mathbf{n}) \\
=& M^{out} + M^{in} + M^{ext} \\
K_{ij} =& (1-\gamma) \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \phi_j^{\mathbf{v}} \cdot \phi_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \phi_i^{\mathbf{w}} + \phi_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n}
\end{aligned}$$

$$M = \begin{pmatrix} M_{11} & \dots & M_{1N} \\ \vdots & \ddots & \vdots \\ M_{N1} & \dots & M_{NN} \end{pmatrix} \qquad K = \begin{pmatrix} K_{11} & \dots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \dots & K_{NN} \end{pmatrix}$$

where $N = \text{DoF}$, is the number of degrees of freedom considered in the problem. The vector U^n is the solution vector at time $t = n\Delta t$ and U^{n-1} corresponds to the solution at the previous time step. The matrix M^{out} corresponds to the integral in M posed on $\mathcal{F}^{out}$, which are the outflow boundaries as explained in Chapter Tent-Pitcher Algorithm. Whereas M^{in} corresponds to the one posed on $\mathcal{F}^{in}$, which are the inflow boundaries. What changes in comparison to Part I is the basis $\{\phi\}$, which were polynomials and are Green's functions here. The solution is approximated as a linear combination of Green's function evaluated at different sources $(\mathbf{x}_0, t_0)$. In this case, the number of sources represents the degrees of freedom.

There are no specific rules to follow when choosing the source points for the Green's functions, however we noticed that the solution is extremely sensitive to them. In Fig. 10.3, one of our choices is illustrated. In practice, we will proceed as in Part I and compute the matrices M and K on reference elements (pyramids, tetrahedra and octahedra) and then use a mapping function in order to obtain the matrices for each real element of the mesh.

As explained in Chapter Perfectly Matched Layers with Trefftz-DG methods, we tried out three variational formulations. The matrices for the different formulations are



(a) 4 source points (b) 8 source points (c) 16 source points (d) 32 source points

(e) 4 source points seen from above (f) 8 source points seen from above (g) 16 source points seen from above (h) 32 source points seen from above

Figure 10.3: Placement of source points for Green's functions

written as follows:

$$
\begin{aligned}
M_{ij}^{A} = & \int_{\mathcal{F}^{out}} (-\rho\phi_j^{v_y}\frac{\partial\phi_i^{w_y}}{\partial t} + \rho(\frac{\partial}{\partial t}+\sigma_y)\phi_j^{v_y}\phi_i^{w_y} - \phi_j^{p}\frac{\partial\phi_i^{w_y}}{\partial y} \\
& + \rho\phi_j^{v_x}\phi_i^{w_x}n_t + \frac{1}{c^2\rho}(\frac{\partial}{\partial t}+\sigma_y)\phi_j^{p}\phi_i^{q} - \frac{1}{c^2\rho}\phi_j^{p}\frac{\partial\phi_i^{q}}{\partial t} - \phi_j^{\mathbf{v}}\cdot\nabla\phi_i^{q})n_t \\
& + (\phi_j^{p}\phi_i^{w_x} + \sigma_y\phi_j^{v_x}\phi_i^{q})n_x + \frac{\partial\phi_j^{p}}{\partial t}\phi_i^{w_y}n_y + (\frac{\partial\phi_j^{\mathbf{v}}}{\partial t}\cdot\mathbf{n})\phi_i^{q} \\
+\gamma \int_{\mathcal{F}^{in}} & (-\rho\phi_j^{v_y}\frac{\partial\phi_i^{w_y}}{\partial t} + \rho(\frac{\partial}{\partial t}+\sigma_y)\phi_j^{v_y}\phi_i^{w_y} - \phi_j^{p}\frac{\partial\phi_i^{w_y}}{\partial y} + \rho\phi_j^{v_x}\phi_i^{w_x} \\
& + \frac{1}{c^2\rho}(\frac{\partial}{\partial t}+\sigma_y)\phi_j^{p}\phi_i^{q} - \frac{1}{c^2\rho}\phi_j^{p}\frac{\partial\phi_i^{q}}{\partial t} - \phi_j^{\mathbf{v}}\cdot\nabla\phi_i^{q})n_t \\
& + \frac{\partial\phi_j^{p}}{\partial t}\phi_i^{w_y}n_y + \phi_j^{p}\phi_i^{w_x}n_x + (\frac{\partial\phi_j^{\mathbf{v}}}{\partial t}\cdot\mathbf{n})\phi_i^{q} + \sigma_y\phi_j^{v_x}\phi_i^{q}n_x \\
+ \int_{\mathcal{F}^{ext}} & \frac{\partial\phi_j^{p}}{\partial t}\phi_i^{w_y}n_y + \gamma(\frac{\partial\phi_j^{\mathbf{v}}}{\partial t}\cdot\mathbf{n})\phi_i^{w_y}n_y + \phi_j^{p}\phi_i^{w_x}n_x + \gamma(\phi_j^{\mathbf{v}}\cdot\mathbf{n})\phi_i^{w_x}n_x + \sigma_y\phi_j^{v_x}\phi_i^{q}n_x
\end{aligned}
$$

$$
\begin{aligned}
K_{ij}^{A} = & (1-\gamma)\int_{\mathcal{F}^{in}} (-\rho\phi_j^{v_y}\frac{\partial\phi_i^{w_y}}{\partial t} + \rho(\frac{\partial}{\partial t}+\sigma_y)\phi_j^{v_y}\phi_i^{w_y} - \phi_j^{p}\frac{\partial\phi_i^{w_y}}{\partial y} + \rho\phi_j^{v_x}\phi_i^{w_x} \\
& + \frac{1}{c^2\rho}(\frac{\partial}{\partial t}+\sigma_y)\phi_j^{p}\phi_i^{q} - \frac{1}{c^2\rho}\phi_j^{p}\frac{\partial\phi_i^{q}}{\partial t} - \phi_j^{\mathbf{v}}\cdot\nabla\phi_i^{q})n_t \\
& + \frac{\partial\phi_j^{p}}{\partial t}\phi_i^{w_y}n_y + \phi_j^{p}\phi_i^{w_x}n_x + (\frac{\partial\phi_j^{\mathbf{v}}}{\partial t}\cdot\mathbf{n})\phi_i^{q} + \sigma_y\phi_j^{v_x}\phi_i^{q}n_x
\end{aligned}
$$

$$
\begin{aligned}
M_{ij}^B =& \int_{\mathcal{F}^{out}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \boldsymbol{\phi}_j^{\mathbf{v}} \cdot \boldsymbol{\phi}_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} + \boldsymbol{\phi}_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ (\phi_j^{p^a} \phi_i^{q^a} + \phi_j^{v_y{}^a} \phi_i^{w_y{}^a}) n_t - \sigma_y (\phi_j^p \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y \\
&+ \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \boldsymbol{\phi}_j^{\mathbf{v}} \cdot \boldsymbol{\phi}_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} + \boldsymbol{\phi}_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ (\phi_j^{p^a} \phi_i^{q^a} + \phi_j^{v_y{}^a} \phi_i^{w_y{}^a}) n_t - \sigma_y (\phi_j^p \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y \\
&+ \int_{\mathcal{F}^{ext}} \phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} \cdot \mathbf{n} + \gamma (\boldsymbol{\phi}_j^{\mathbf{v}} \cdot \mathbf{n})(\boldsymbol{\phi}_i^{\mathbf{w}} \cdot \mathbf{n}) - \sigma_y (\phi_j^p \phi_i^{q^a} + \gamma (\boldsymbol{\phi}_j^{\mathbf{v}} \cdot \mathbf{n}) \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y
\end{aligned}
$$

$$
\begin{aligned}
K_{ij}^B =& (1-\gamma) \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \boldsymbol{\phi}_j^{\mathbf{v}} \cdot \boldsymbol{\phi}_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} + \boldsymbol{\phi}_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ (\phi_j^{p^a} \phi_i^{q^a} + \phi_j^{v_y{}^a} \phi_i^{w_y{}^a}) n_t - \sigma_y (\phi_j^p \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y \\
M_{ij}^C =& \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \boldsymbol{\phi}_j^{\mathbf{v}} \cdot \boldsymbol{\phi}_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} + \boldsymbol{\phi}_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ (\phi_j^{p^a} \phi_i^{q^a} + \phi_j^{v_y{}^a} \phi_i^{w_y{}^a}) n_t - \sigma_y (\phi_j^p \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y \\
&+ \int_{\mathcal{F}^{ext}} \phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} \cdot \mathbf{n} + \gamma (\boldsymbol{\phi}_j^{\mathbf{v}} \cdot \mathbf{n})(\boldsymbol{\phi}_i^{\mathbf{w}} \cdot \mathbf{n}) - \sigma_y (\phi_j^p \phi_i^{q^a} + \gamma (\boldsymbol{\phi}_j^{\mathbf{v}} \cdot \mathbf{n}) \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y
\end{aligned}
$$

$$
\begin{aligned}
K_{ij}^C =& \gamma \int_{\mathcal{F}^{in}} \left(\frac{1}{c^2\rho} \phi_j^p \phi_i^q + \rho \boldsymbol{\phi}_j^{\mathbf{v}} \cdot \boldsymbol{\phi}_i^{\mathbf{w}} \right) n_t + \left(\phi_j^p \boldsymbol{\phi}_i^{\mathbf{w}} + \boldsymbol{\phi}_j^{\mathbf{v}} \phi_i^q \right) \cdot \mathbf{n} \\
&+ (\phi_j^{p^a} \phi_i^{q^a} + \phi_j^{v_y{}^a} \phi_i^{w_y{}^a}) n_t - \sigma_y (\phi_j^p \phi_i^{q^a} + \phi_j^{v_y} \phi_i^{w_y{}^a}) n_y
\end{aligned}
$$

We observe that when $\sigma_y = 0$, formulations B and C are equivalent to the Trefftz-DG formulation without PML. Whereas the formulation A is very different even in this case.

Let us now present the results we obtained with each formulation in Table 10.1, where "V.F." stands for "Variational Formulation". We consider a domain of size $L_x = L_y = 1$, discretized with $n_x = n_y = 100$ cells in each direction. We let the code run until it reaches the final time $L_t = 1s$ and visualize the pressure at $t = 0s$, $t = 0.1s$ and $t = 1s$. We use Neumann boundary conditions and a gaussian source function. As explained before, we need to choose the number of source points and how to place them. We take 4 source points and place them as depicted in Fig. 10.3e. We can see in Table 10.1 that the first formulation gives poor results, whereas the second and third ones seems to give better results.

As we explained previously, we are not sure why the first formulation does not work properly. It could be because of an implementational error or a problem with the formulation itself. Since for $\sigma_y = 0$ this formulation is very different from the one without PML (5.1), it was difficult to find a way to compare them in order to debug the code efficiently. This led us to consider the formulation B with auxiliary variables (8.8), because when taking $\sigma_y = 0$, we retrieve the same formulation as (5.1). Thus, we were able to compare both implementations and debug the code efficiently. Note that, unlike the original formulation which does not have any derivatives, formulation B, as

formulation A, contains partial derivatives, which is a clear disadvantage in terms of computational cost and complexity. The third variational formulation shares this trait with the formulation without PMLs, which makes it interesting. In fact, since it is only posed on the inflow boundaries and that we have more flexibility in the choice of the tests functions, we do not need any derivatives.

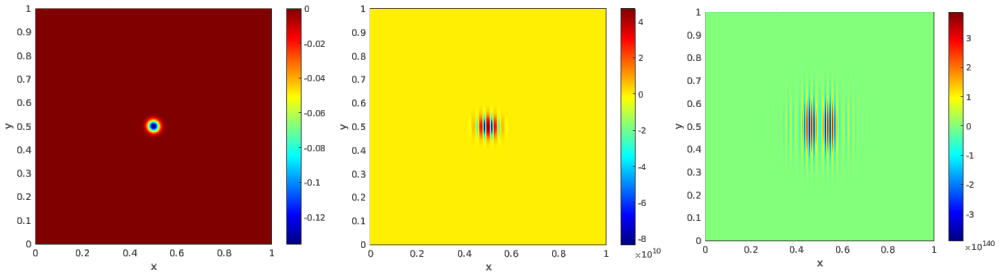
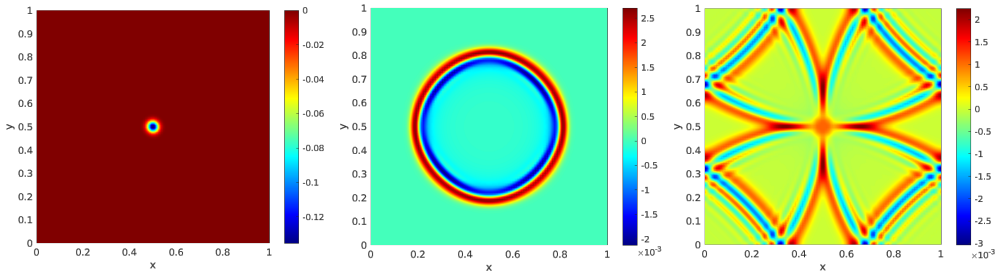
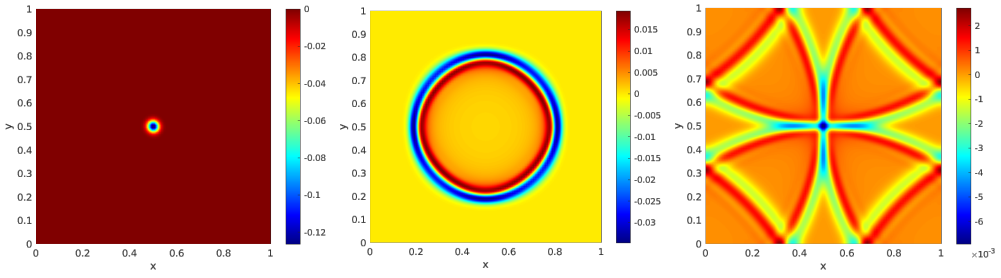
V.F.	$t = 0s$	$t = 0.3s$	$t = 1s$
A			
B			
C			

Table 10.1: Pressure through time for formulations A, B and C

We can see that the results from the second and the third formulation are similar yet quite different. In order to see which solution is the most accurate, we compare the solution from the formulation B and C to the polynomial solution obtained in Part I of the thesis. The relative errors between the different solutions and the polynomial one are presented in Table 10.2 and we can see that the error is approximately the same for both solutions when 4 sources points are taken and is $\approx 10^{-1}(10\%)$, which is quite high. So, we took more source points to see if the Green's functions solutions converge to the polynomial one. We test the formulations with 4, 8, 16 and 32 source points placed as

shown in Fig. 10.3. We compare a cross-section of the solutions at $t = 1s$ and $y = 0.5$. We actually see in Fig. 10.5 that the solution from formulation C converges as the number of sources is increased, in fact the error with 32 sources is $\approx 10^{-3}$ (0.1%), which is correct (see Table 10.2). On the contrary, we see in Fig. 10.4 that the solution from formulation B blows up as soon as we modify the number of sources or their position (in fact, the solution with 32 sources equals NaN). Thus, the third formulation produces more accurate results than the second one.

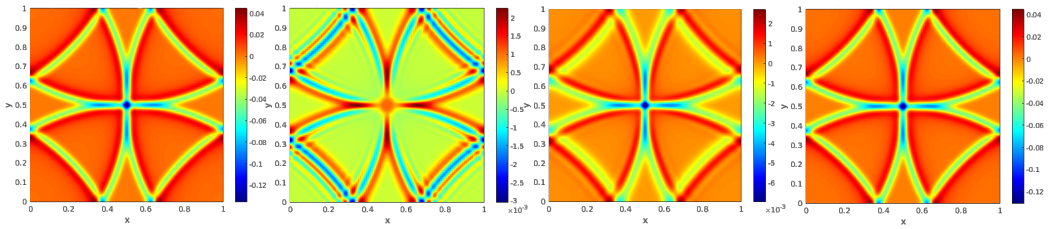
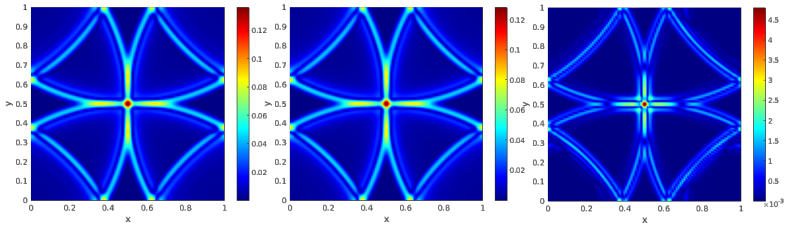
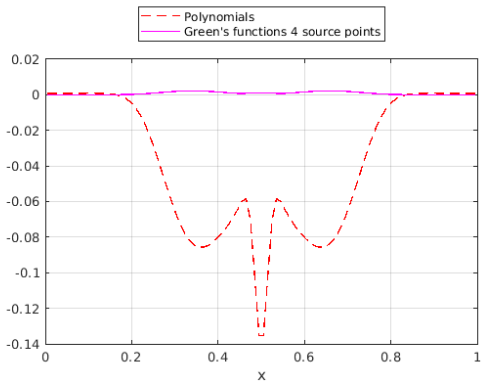
	Polynomials	B	C (4 sources)	C (32 sources)
				
Error				

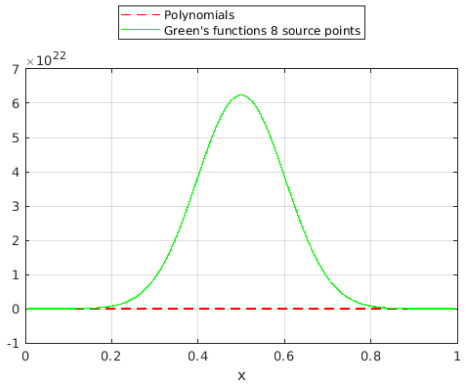
Table 10.2: Error between the polynomial solution and the numerical solutions obtained with formulations B and C, at time $t = 1s$

10.1.2 Absorption in the domain

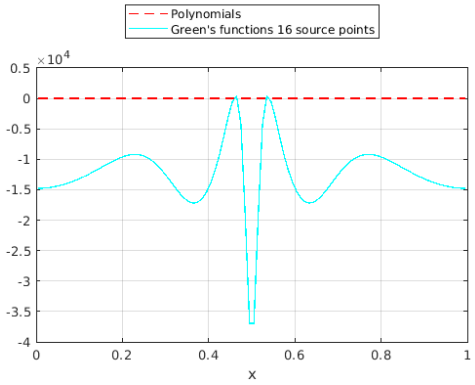
The next part of Phase I consists in adding absorption everywhere in the domain in one direction. The purpose of this test is to verify the validity of the Green's functions with absorption, computed in the previous chapter. Since the first variational formulation A does not give proper results in the non-absorbing case, we expect it to be the same with PML. Hence, we only carry out the tests for the variational formulations B and C. The results we obtain are presented in Table 10.3. We consider a domain of size $L_x = L_y = 1$, discretized with $n_x = n_y = 100$ cells in each direction. We visualize the pressure at time $t = 0s$, $t = 0.1s$ and $t = 1s$. We use Neumann boundary conditions and a Gaussian source function. For the second variational formulation, we use 4 source points since it is the only case for which we obtain the most accurate results and for the third variational formulation, we use 8 source points, because the solution blows up when using more sources with absorption. We actually remarked that depending on the position of the sources, we could take more than 8 sources, but the choice is very sensitive and hard to



(a) Cross-section of the solution obtained with 4 source points



(b) Cross-section of the solution obtained with 8 source points



(c) Cross-section of the solution obtained with 16 source points

Figure 10.4: Comparison between the cross-section of the numerical solution with Green's functions in the formulation B and the polynomial solution, for a varying number of sources

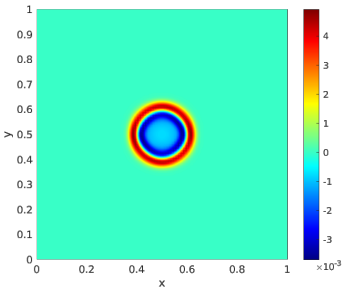
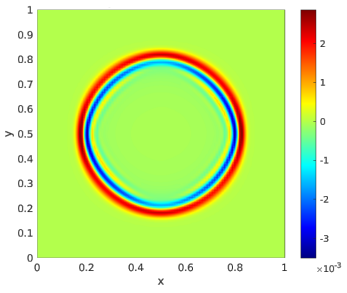
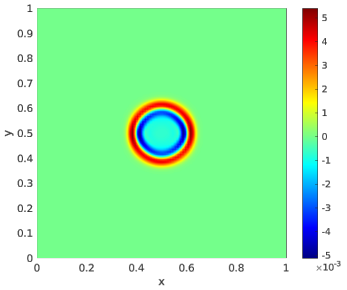
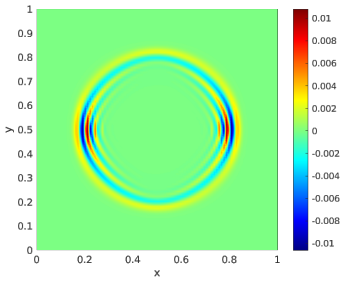
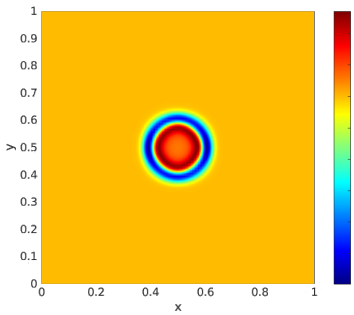
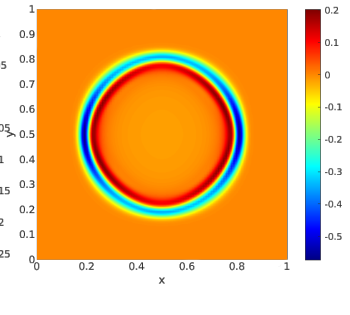
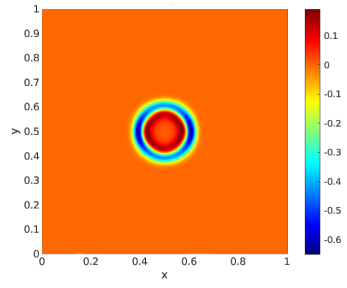
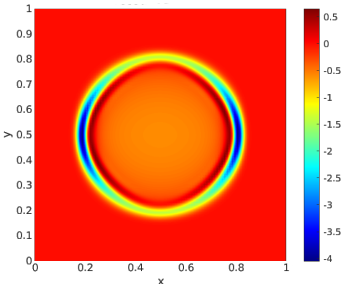
V.F.	σ_y	$t = 0.1s$	$t = 0.3s$
B	0.5		
	1		
C	5		
	10		

Table 10.3: Pressure for varying σ_y with $\sigma_y \neq 0$ in Ω for formulations B and C

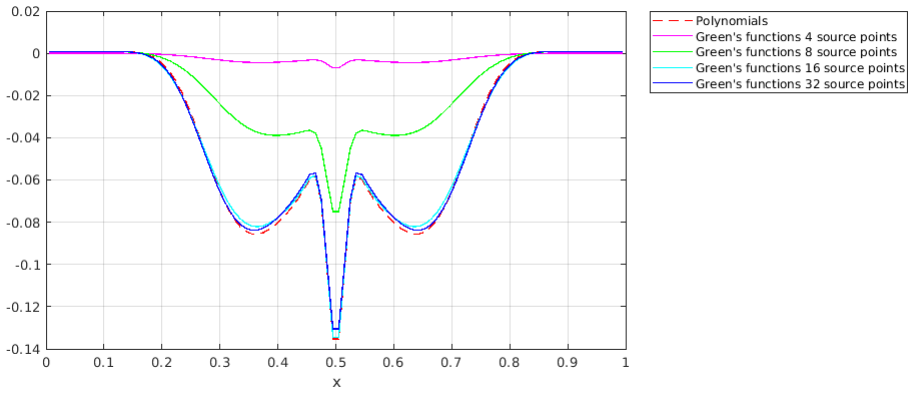


Figure 10.5: Comparison between the cross-section of the numerical solution with Green's functions in the formulation C and the analytical solution, for a varying number of sources

make. We can see that the solution blows up with the second variational formulation, while the third one gives us proper results, and we can see the solution being damped along the y-direction. For all these reasons, we will use the formulation C from now on.

10.2 Phase II

In Phase II, we couple the domain with a single PML parallel to the x-axis. This actually reduces to changing the value of the damping coefficient to zero (in the computational domain) or a non-null value (in the PML), as depicted in Fig. 10.6. As explained

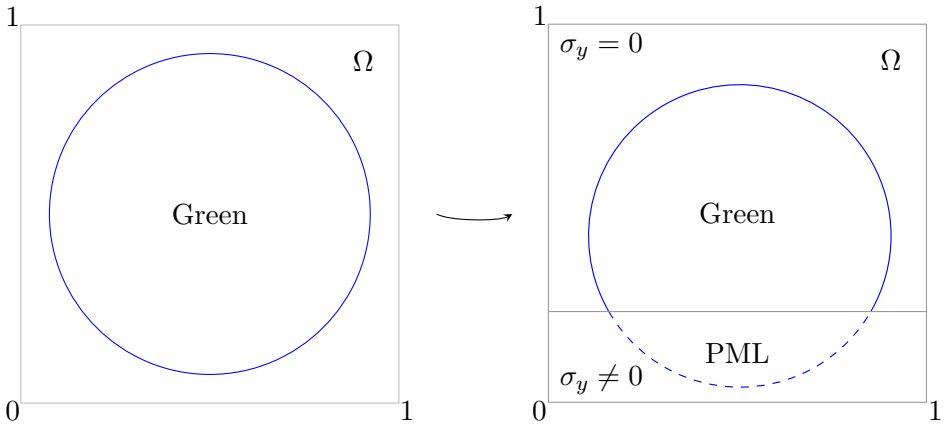


Figure 10.6: In the second phase, we add a PML in the y direction

at the beginning of this chapter, when adding a PML, we need to handle the interface between the two subdomains, because some elements will be shared by the computational domain and the PML as can be seen in Fig. 10.7. To deal with such elements, we will

separate them with a vertical plane as depicted in Fig. 10.9 and construct the associated elementary matrix M, which will contain an additional integral posed on the created internal face, written as follows:

$$CT_{ij} = \int_{\mathcal{F}^{int}} [\![\mathbf{v}]\!]_x \cdot \{\!\{q\}\!\} + \beta_1 [\![p]\!]_x \cdot [\![q]\!]_x + [\![p]\!]_x \{\!\{\mathbf{w}\}\!\} + \alpha_1 [\![\mathbf{v}]\!]_x [\![\mathbf{w}]\!]_x$$
$$-\sigma_y \int_{\mathcal{F}^{int}} [\![v_y]\!]_y \{\!\{{w_y}^a\}\!\} + \gamma_1 [\![p]\!]_y [\![{w_y}^a]\!]_y + [\![p]\!]_y \{\!\{q^a\}\!\} + \gamma_2 [\![v_y]\!]_y [\![q^a]\!]_y$$

where $[\![\cdot]\!]$ and $\{\!\{\cdot\}\!\}$ represent the jump and the mean value respectively.

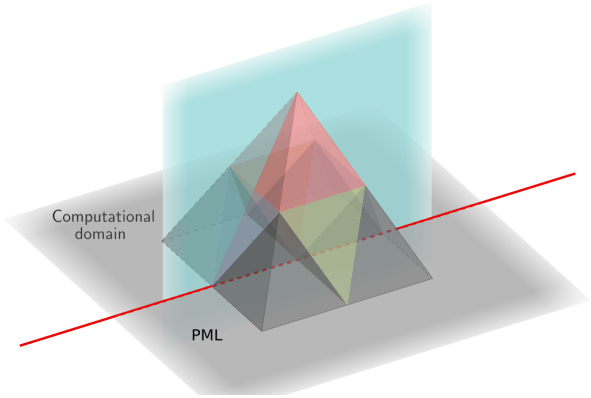
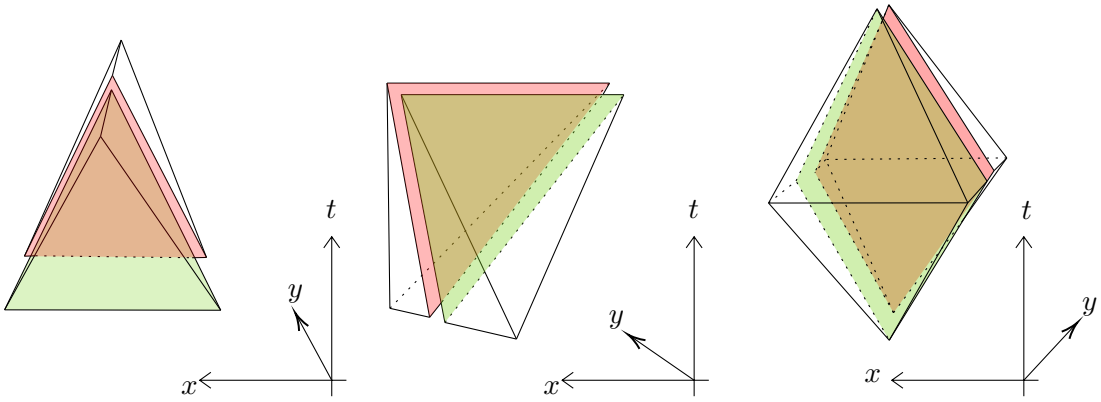


Figure 10.7: Example of elements positioned on the interface between the domain and the PML



(a) Vertical tetrahedra (b) Horizontal tetrahedra (c) Octahedron

Figure 10.8: Example of elements positioned on the interface between the domain and the PML

In fact, the matrix M will actually be of size $2N \times 2N$, because we separate the element in two (see Fig. 10.8) and compute a matrix per sub-element and then fill them into a bigger matrix in its diagonal blocks, along with coupling terms corresponding to the integral on the internal face in the anti-diagonal blocks, as depicted in Fig. 10.9.

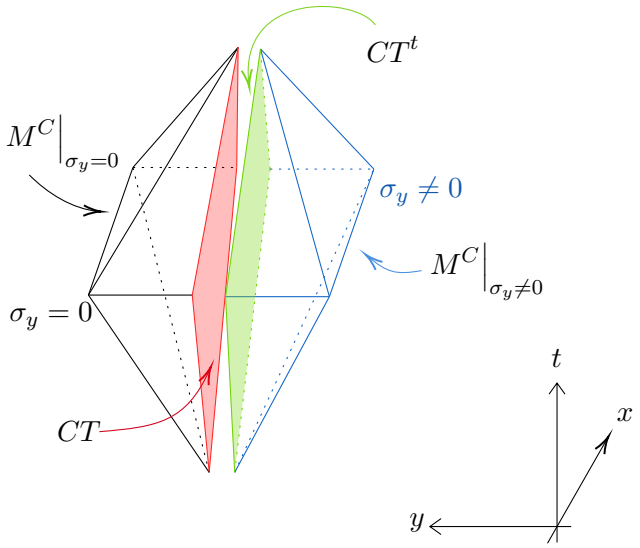


Figure 10.9: Octahedron at the interface of the domain and a PML (rotated by 90° for clarity)

$$M = \begin{pmatrix} M^C_{11}\big|_{\sigma_y=0} & \dots & M^C_{1N}\big|_{\sigma_y=0} & CT_{11} & \dots & CT_{1N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M^C_{N1}\big|_{\sigma_y=0} & \dots & M^C_{NN}\big|_{\sigma_y=0} & CT_{N1} & \dots & CT_{NN} \\ CT_{11} & \dots & CT_{N1} & M^C_{11}\big|_{\sigma_y\neq 0} & \dots & M^C_{1N}\big|_{\sigma_y\neq 0} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ CT_{1N} & \dots & CT_{NN} & M^C_{N1}\big|_{\sigma_y\neq 0} & \dots & M^C_{NN}\big|_{\sigma_y\neq 0} \end{pmatrix}$$

In this case, we need to compute the elementary matrices for each element (pyramids, horizontal and vertical tetrahedra, octahedra) in the computational domain (*i.e.* $M^C\big|_{\sigma_y=0}$ and $K^C\big|_{\sigma_y=0}$), inside the PML (*i.e.* $M^C\big|_{\sigma_y\neq 0}$ and $K^C\big|_{\sigma_y\neq 0}$) and compute the coupling matrices for the elements at the interface. Thus, we have to compute three additional coupling elementary matrices for split tetrahedra (horizontal and vertical) and octahedra. As we can see in Fig. 10.7, the pyramids do not lay on the interface between the domain and the PML, so we do not need to compute coupling matrices for them. Apart from the elementary matrices, there are no major changes. In the time propagator, we only need to handle the interface between the domain and the PML separately by using the coupling matrix computed above. Otherwise, the rest of the implementation remains the same.

Remark. *This need for coupling matrices is not inherent to the PML process, but rather to layered structured domains. In fact, it is because we have two subdomains that we need a coupling matrix. Thus, if we were to study an acoustic-acoustic bi-layered domain, we would have the same needs.*

Let us consider a domain of size $L_x = L_y = 1$, discretized with $n_x = n_y = 100$ cells in each direction. We visualize the pressure at time $t = 0.4s$, $t = 0.5s$ and $t = 0.6s$. We use a first order absorbing boundary condition (defined in Part I) at the external boundary of the PML and a Gaussian source function. We consider a PML of width $w = 0.2$ and several values of the damping coefficient $\sigma_y = 2, 5, 10$. The first results are presented in Table 10.4. In this test, we use different values of σ_y and see how the solution is damped when it enters the PML. As expected, we can see that for a larger σ_y, the absorption is stronger. However, since we are using a constant absorption, if σ_y is too large when the solution enters the PML, reflections can appear and we observe this phenomena with $\sigma_y = 10$. If we want to diminish the reflections and keep a constant absorbing coefficient, we can refine the mesh. This is the second test we performed and the results are presented in Table 10.5. We are placed in the same conditions as the previous test, and we present the pressure for different mesh sizes $h = \Delta x = \Delta y$. We can see that as we refine the mesh, the reflections diminish.

In fact, refining the mesh does not only reduce reflections, it also makes it possible to use a larger σ_y. This is the third test and the obtained results are presented in Table 10.6. In this table, we can see that for σ_y and a mesh of size $h = 10^{-1}$, we have poor results and the solution explodes at the interface. When we refine the mesh, the solution no longer explodes and we obtain proper results with a mesh of size $h = 5.10^{-3}$ and refining even more leads to softening the reflections.

We perform an additional test in order to see the influence of the PML width w on the absorption and the results are presented in Tables 10.7 and 10.8. In the first table, we consider the results in a domain of size $L_x = L_y = 1$, discretized with $n_x = n_y = 100$ cells in each direction. We visualize the pressure at time $t = 0.4s$, $t = 0.5s$ and $t = 0.6s$. We use a first order absorbing boundary conditions and a Gaussian source function. We consider a damping coefficient $\sigma_y = 10$ and several values for the PML width $w = 0.3$, $w = 0.2$, $w = 0.1$ and $w = 0.05$. In the second table, we consider a damping coefficient $\sigma_y = 20$ and two PML width $w = 0.1$ and $w = 0.05$. For $w = 0.1$, we discretize the domain with $n_x = n_y = 200$ cells in each direction and for $w = 0.05$ we discretize it with $n_x = n_y = 400$ cells in each direction, in order to have less reflections. As expected, when the layer is large enough, we can choose smaller σ_y because it will have the time to be fully damped as it goes through the layer. But when the layer is thin, we have to choose a bigger σ_y in order to absorb the solution quickly. In practice, it is better to have a thin absorbing layer because we want to limit computations as much as possible and thus, reduce the computational time. However, as explained before, this would mean using a large σ_y, which involves reflections (which we do not want and are the reasons why we are looking for absorbing conditions) or involves refining the mesh to avoid reflections, so the computational time is also increased. A better option would be the use of a varying σ_y as the one presented in the Introduction, but this implies having to recompute the elementary matrices at each space position, thus also increases the computational time.

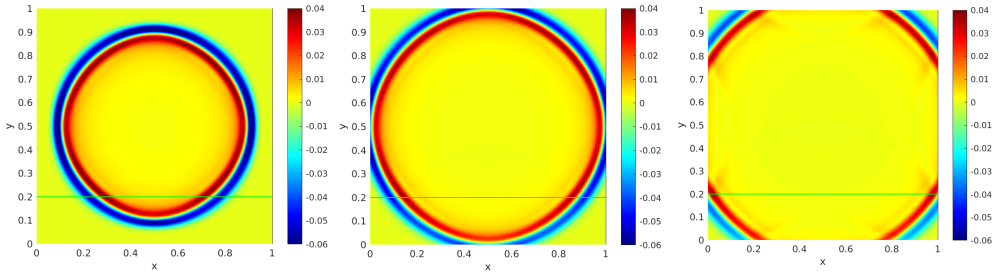
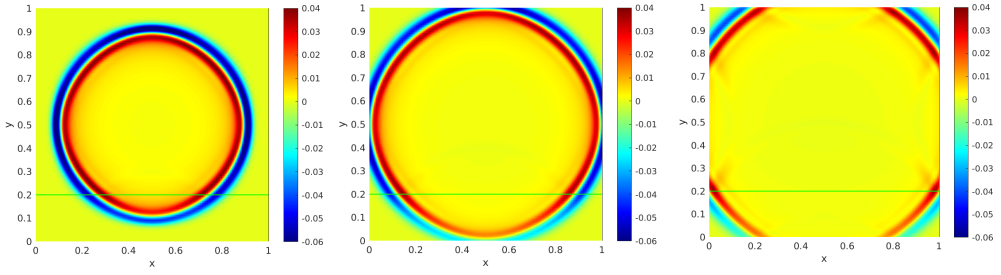
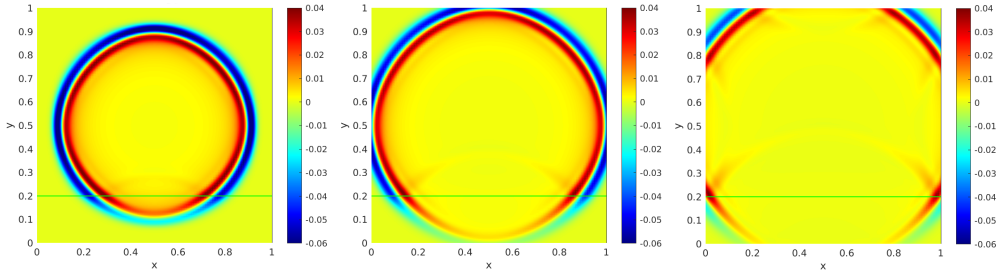
σ_y	$t = 0.4s$	$t = 0.5s$	$t = 0.6s$
2			
5			
10			

Table 10.4: Pressure with a PML of variable damping coefficient σ_y

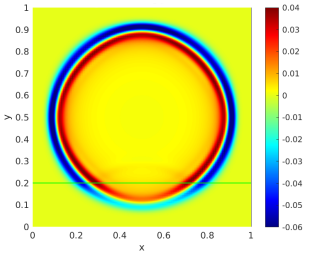
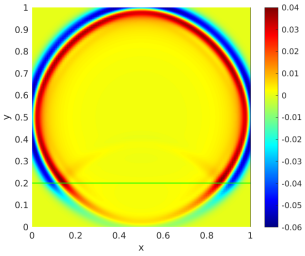
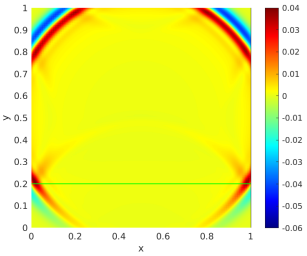
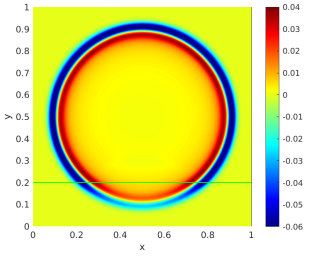
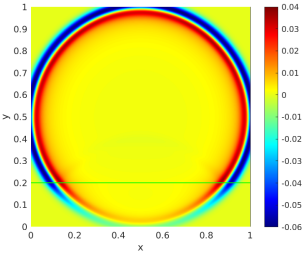
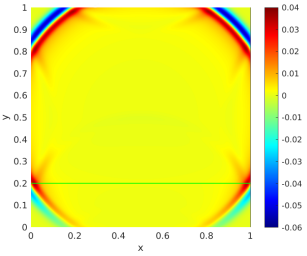
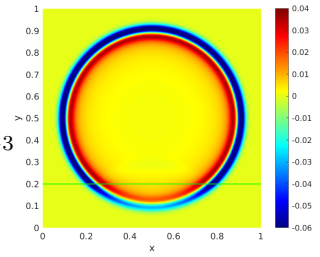
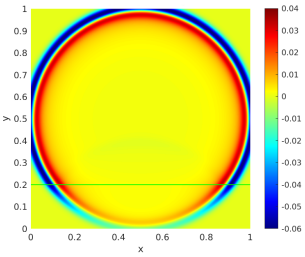
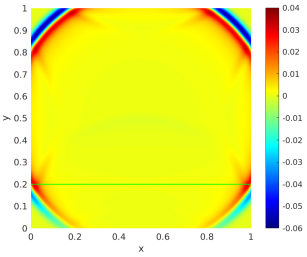
h	$t = 0.4s$	$t = 0.5s$	$t = 0.6s$
10^{-2}			
5.10^{-3}			
$2,5.10^{-3}$			

Table 10.5: Pressure for varying mesh cell size with a PML and $\sigma_y = 10$

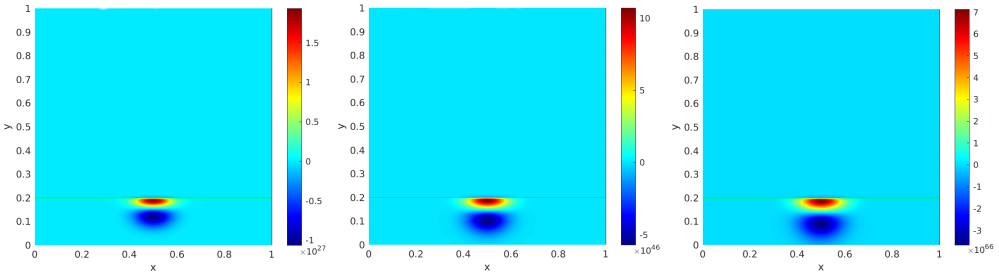
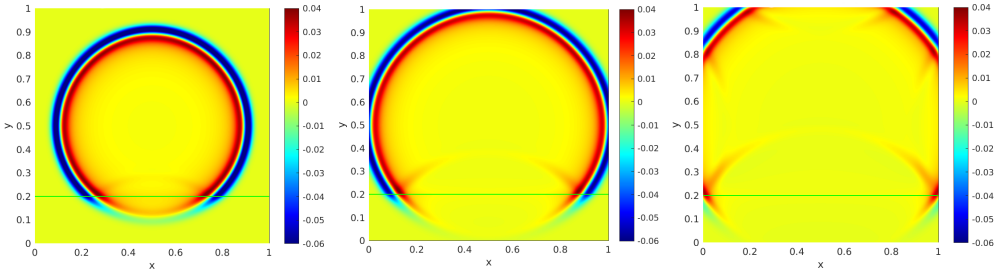
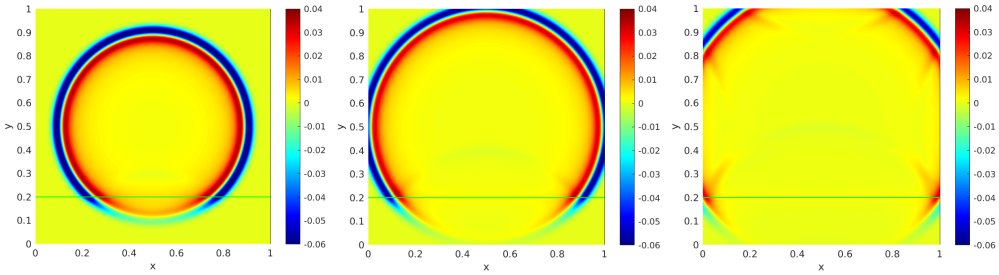
h	$t = 0.4s$	$t = 0.5s$	$t = 0.6s$
10^{-2}			
5.10^{-3}			
$2{,}5.10^{-3}$			

Table 10.6: Pressure for varying mesh cell size with a PML and $\sigma_y = 20$

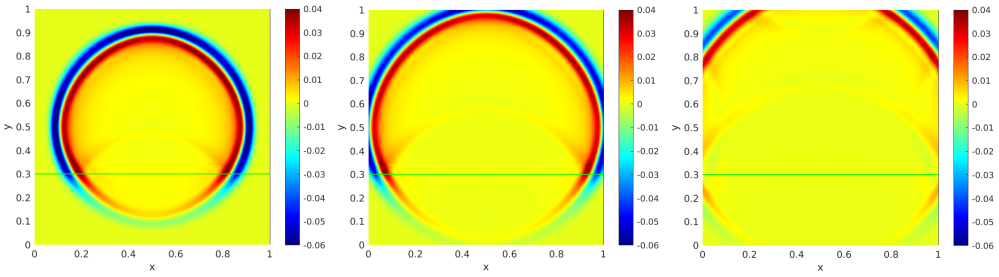
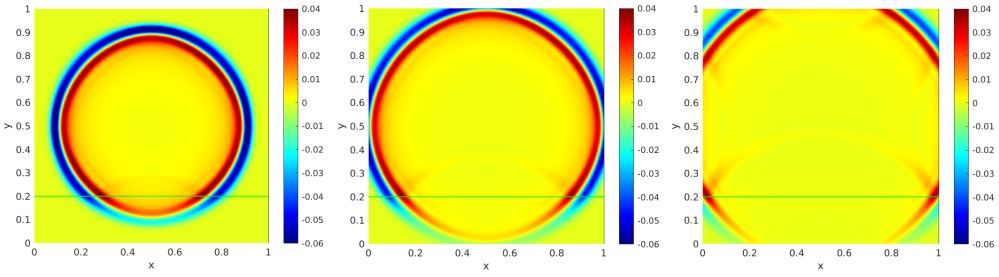
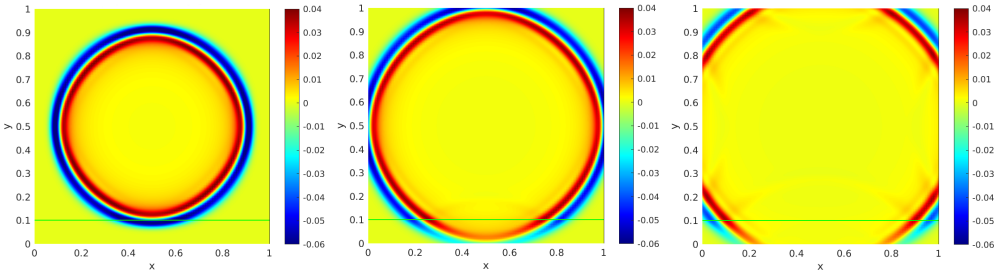
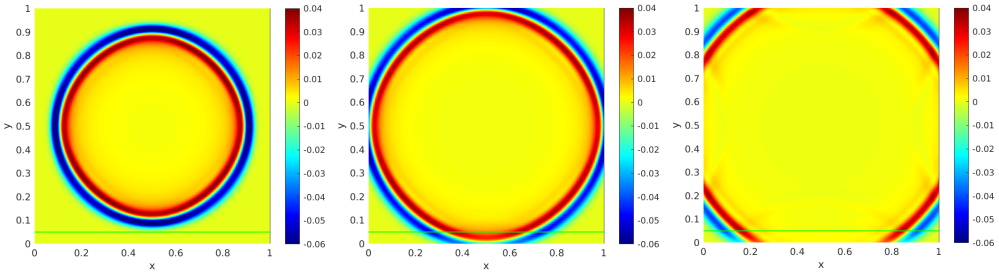
w	$t = 0.4s$	$t = 0.5s$	$t = 0.6s$
0.3			
0.2			
0.1			
0.05			

Table 10.7: Pressure with a PML of variable width and $\sigma_y = 10$

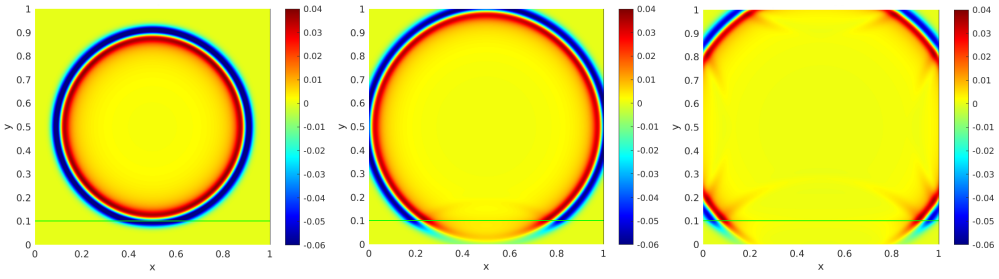
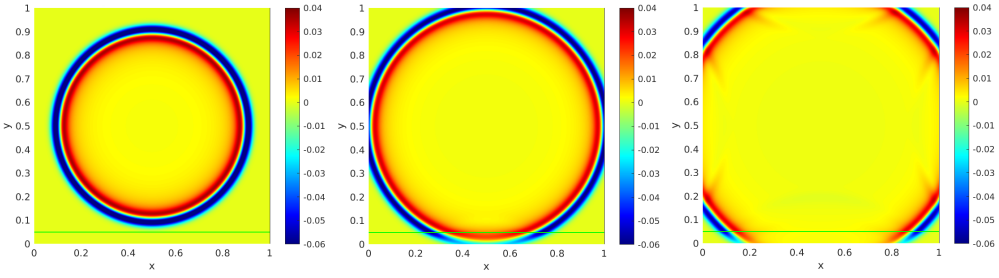
w	$t = 0.4s$	$t = 0.5s$	$t = 0.6s$
0.1			
0.05			

Table 10.8: Pressure with a PML of variable width and $\sigma_y = 20$

10.3 Phase III

In Phase III, we surround the domain with PMLs. This reduces to changing the value of the damping coefficients σ_x and σ_y depending on the sub-domain we consider, as depicted in Fig. 10.10 and of course, use the Green's functions with absorption in the x- and y-direction as basis functions. The first part of this section focuses on the problem with Green's functions in the domain and the PMLs, while the second part will be about the coupling of polynomials in the domain under study with Green's functions in the PMLs.

10.3.1 PML in x and y direction

In the same manner as in the previous section, the main changes operate in the elementary matrices. We now have sixteen sets of elementary matrices to compute, as illustrated in Fig. 10.11:

- the matrices in the computational domain $M^C, K^C\Big|_{\sigma_x=0,\sigma_y=0}$,
- the matrices in the PML in the x-direction $M^C, K^C\Big|_{\sigma_x\neq 0,\sigma_y=0}$ (represented by green layers in Fig. 10.11),
- the matrices in the PML in the y-direction $M^C, K^C\Big|_{\sigma_x=0,\sigma_y\neq 0}$ (represented by yellow layers in Fig. 10.11),

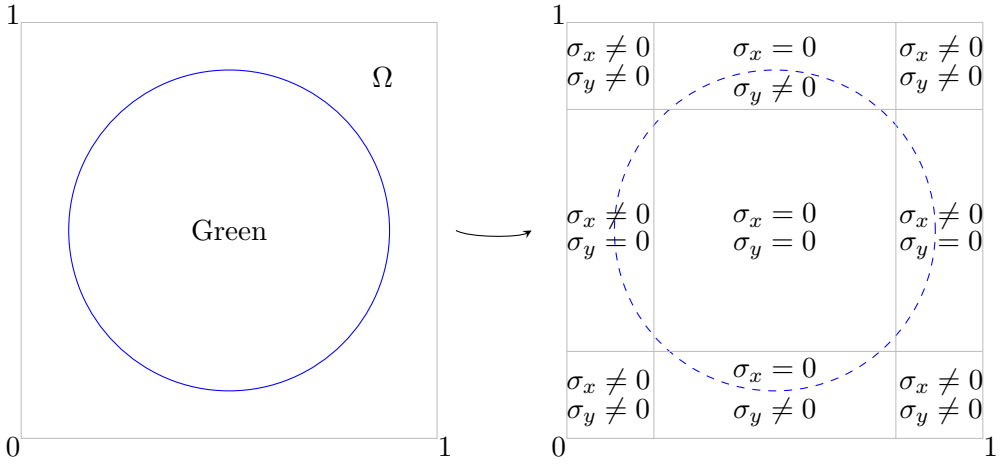


Figure 10.10: In the first part of the third phase, we surround the domain with absorbing layers

- the matrices in the PML in the x- and y-direction $M^C, K^C\big|_{\sigma_x \neq 0, \sigma_y \neq 0}$ (represented by red layers in Fig. 10.11),
- the coupling matrices between the domain and the bottom y-layer (represented by a blue line in Fig. 10.11),
- the coupling matrices between the domain and the top y-layer (represented by a cyan line in Fig. 10.11),
- the coupling matrices between the domain and the left x-layer (represented by a red line in Fig. 10.11),
- the coupling matrices between the domain and the right x-layer (represented by a magenta line in Fig. 10.11),
- the coupling matrices between the bottom y-layer and the bottom-right corner (represented by brown lines in Fig. 10.11),
- the coupling matrices between the bottom y-layer and the bottom-left corner (represented by purple lines in Fig. 10.11),
- the coupling matrices between the left x-layer and the bottom-left corner (represented by green lines in Fig. 10.11),
- the coupling matrices between the left x-layer and the top-left corner (represented by orange lines in Fig. 10.11),
- the coupling matrices at the junctions of all layers, which are all different (represented by shades of gray circles in Fig. 10.11),

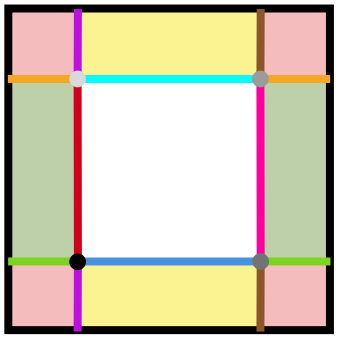


Figure 10.11: Classification of the matrices depending on the considered layer

The different reference elements (pyramids, tetrahedra, octahedra) will be split in the same way as it was in the previous section, and associated elementary matrices will be computed. The process differs in the case of the coupling matrices at the junctions of all layers. Actually, only the octahedron is located at the junction of all layers and since there are four layers, it will be split in four, instead of two as in the previous section. Thus, we obtain a coupling matrix of size $4N \times 4N$, which can be written as the following:

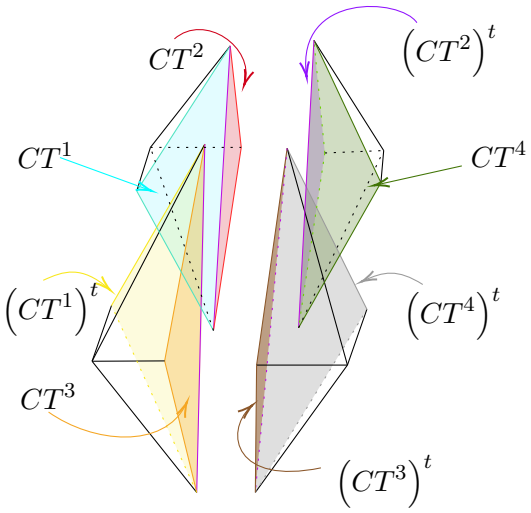


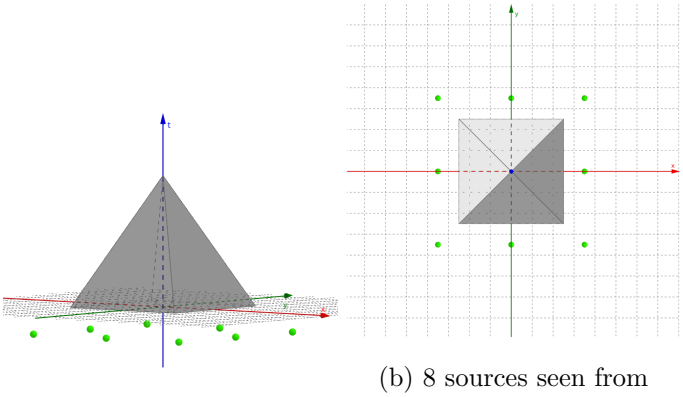
Figure 10.12: Octahedra at the junction of all layers

$$
M = \begin{pmatrix}
M_{11}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y=0}} & \dots & M_{1N}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y=0}} & CT_{11}^1 & \dots & CT_{1N}^1 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
M_{N1}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y=0}} & \dots & M_{NN}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y=0}} & CT_{N1}^1 & \dots & CT_{NN}^1 \\
CT_{11}^1 & \dots & CT_{N1}^1 & M_{11}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y=0}} & \dots & M_{1N}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y=0}} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
CT_{1N}^1 & \dots & CT_{NN}^1 & M_{N1}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y=0}} & \dots & M_{NN}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y=0}} \\
CT_{1N}^2 & \dots & CT_{1N}^2 & & & \\
\vdots & \ddots & \vdots & & 0 & \\
CT_{1N}^2 & \dots & CT_{1N}^2 & & & \\
 & & & CT_{1N}^3 & \dots & CT_{1N}^3 \\
 & 0 & & \vdots & \ddots & \vdots \\
 & & & CT_{1N}^3 & \dots & CT_{1N}^3
\end{pmatrix}
$$

$$
\begin{pmatrix}
CT_{11}^2 & \dots & CT_{1N}^2 & & & \\
\vdots & \ddots & \vdots & & 0 & \\
CT_{N1}^2 & \dots & CT_{NN}^2 & & & \\
 & & & CT_{N1}^3 & \dots & CT_{NN}^3 \\
 & 0 & & & & \\
 & & & CT_{N1}^3 & \dots & CT_{NN}^3 \\
M_{11}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y\neq 0}} & \dots & M_{1N}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y\neq 0}} & CT_{11}^4 & \dots & CT_{1N}^4 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
M_{N1}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y\neq 0}} & \dots & M_{NN}^C\Big|_{\substack{\sigma_x=0\\ \sigma_y\neq 0}} & CT_{N1}^4 & \dots & CT_{NN}^4 \\
CT_{11}^4 & \dots & CT_{N1}^4 & M_{11}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y\neq 0}} & \dots & M_{1N}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y\neq 0}} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
CT_{1N}^4 & \dots & CT_{NN}^4 & M_{N1}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y\neq 0}} & \dots & M_{NN}^C\Big|_{\substack{\sigma_x\neq 0\\ \sigma_y\neq 0}}
\end{pmatrix}
$$

In the above matrices, the matrices CT^i correspond to the integral posed on the interface between each layer, as can be seen in Fig. 10.12.

Let us consider a domain of size $L_x = L_y = 1$. We visualize the pressure at time $t = 0.35s$, $t = 0.4s$, $t = 0.45s$, $t = 0.6s$ and, the final time, $t = 1s$. We use a first order absorbing boundary condition at the extremities of the PMLs and a Gaussian source function. We use 8 source points to evaluate our basis functions. We consider a PML of width $w = 0.2$ and two values of the damping coefficient $\sigma_y = 5, 10$. For $\sigma_y = 5$, , we discretize the domain with $n_x = n_y = 100$ cells in each direction and for $\sigma_y = 10$, we discretize it with $n_x = n_y = 400$ cells in each direction in order to have less reflections as



(a) 8 sources

(b) 8 sources seen from above

Figure 10.13: Another choice of placement for the sources of the Green's functions

in the previous test cases. The first results are presented in Table 10.9. We can see that the solution is absorbed as it enters the PMLs. We can see that the results we obtain are not robust and are very sensitive to the position of the source points. In this test, we work with 8 sources positioned as depicted in Fig. 10.13, because the solution would blow up with the source points used in the previous section and even with more source points. We are on the way to conclude that if Green's functions are very well-suited to define a Trefftz-DG formulation of the time-dependent wave equation, they seem to lead to an unstable numerical method. To understand this problem of lack of robustness, we decided to go back to a case without PML, just to compare the approximation with polynomials to the one with Green's functions.

10.3.2 Comparison between Green's functions and polynomials

In order to compare the robustness of the approximations with polynomials and Green's functions, we use a specific initial function, for which we know the analytical solution. The initial condition is written as follows:

$$f^p(x,y,t) = -\frac{sin(2\pi x)}{c}\,, \qquad f^{v_x}(x,y,t) = sin(2\pi x)\,, \qquad f^{v_y}(x,y,t) = 0$$

and the analytical solution is $p(x,y,t) = -sin(2\pi(x+ct))$. Let us consider a domain of size $L_x = L_y = 1$, discretized with $n_x = n_y = 100$ cells in each direction. We take $\mathbb{P}^3$ polynomials and Green's functions with 32 sources, and compare the results obtained with each of them. In Table 10.10, we present the results we obtain for the two kinds of approximation settings and the relative error between the analytical solution and the numerical solutions at $t = 1.00041s$. To the naked eye, both numerical solutions seem to be quite close to the analytical one. We can see that the relative error with the Green's functions is $\approx 10^{-3} (\approx 0.1\%)$, which is correct. However, the error with the polynomials is $\approx 10^{-9} (\approx 10^{-7}\%)$, which is much better. In order to see this more closely, we display a cross-section of the solution at $y = 0.5$ in Fig. 10.14. We remark that, indeed, the

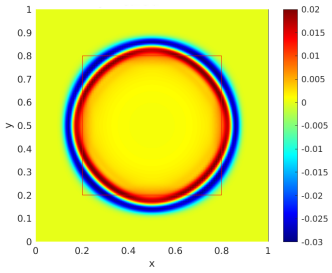
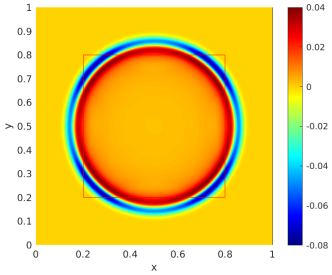
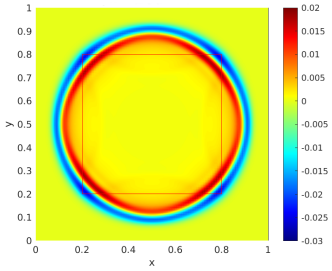
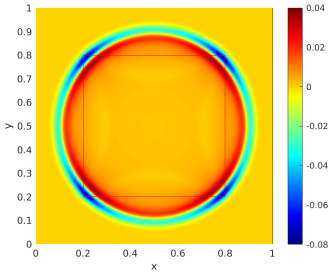
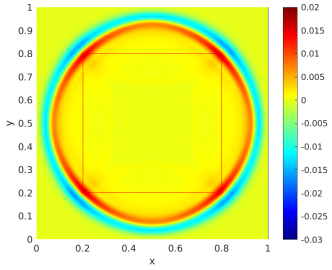
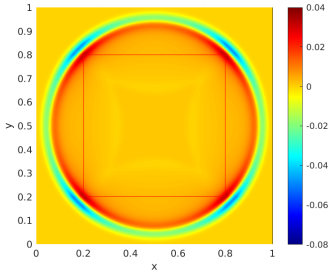
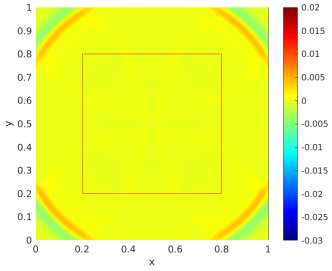
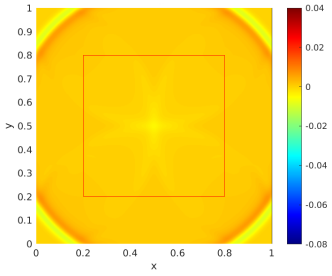
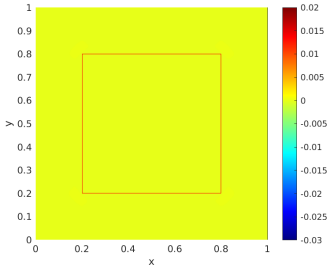
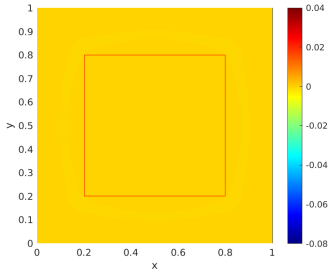
$t(s)$	$\sigma = 5$	$\sigma = 10$
0.35		
0.4		
0.45		
0.6		
1		

Table 10.9: Pressure in a domain surrounded with Perfectly Matched Layers

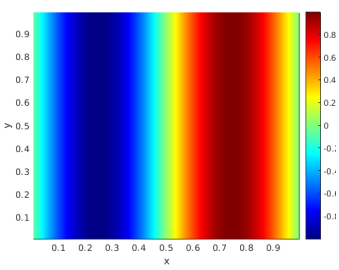
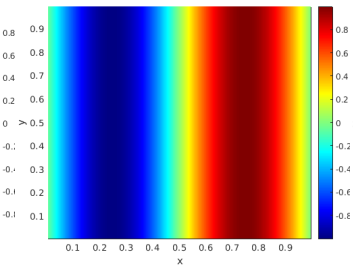
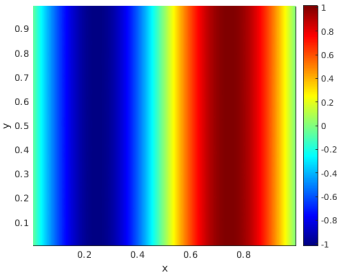
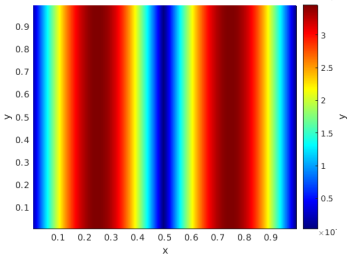
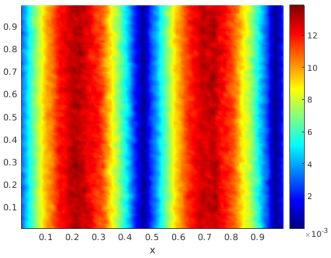
	Analytical solution	Polynomials	Green's functions
			
Error			

Table 10.10: Relative error between the numerical solutions and the analytical solution

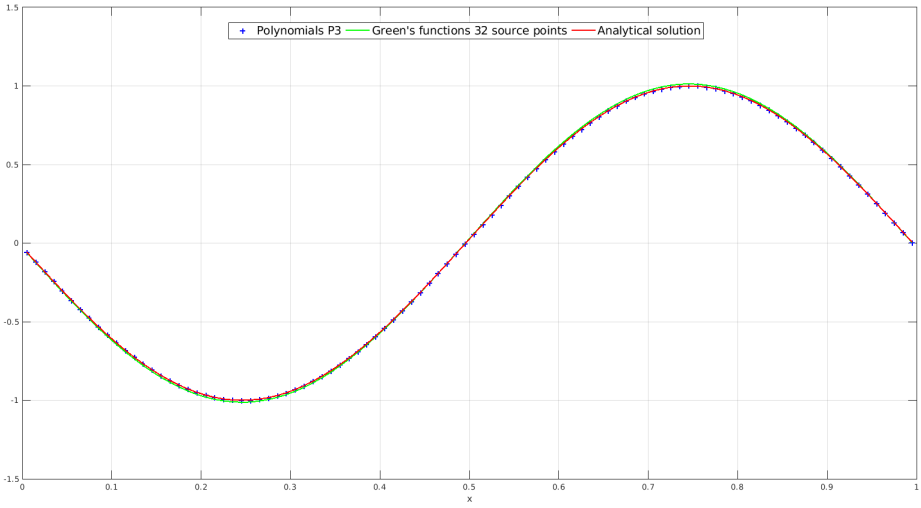


Figure 10.14: Comparison of a cross-section between the numerical solutions and the analytical solution

solution with Green's functions does not have the same amplitude as the analytical solution. Whereas the polynomial solution and the exact solution are indistinguishable. In order to further these tests, we plot the L^2 relative error in function of the cell size in Fig. 10.15. We can see that the approximation with polynomials is very efficient and the solution it produces converges to the analytical solution as the cell size decreases. However, the error of the solution with Green's functions starts decreasing until a plateau is reached, after what the L^2 error increases. We can clearly see that the polynomial solution is more accurate than the solution with Green's functions, and converges to the analytical solution as the mesh is refined. As we explained before, this could be due to our choice of source points. Throughout the various tests we performed, we noticed how sensitive the solution is to them. Yet, we do not know how to choose them in order to obtain better results. This is the reason why we would prefer representing our solution in the computational domain using polynomials and use Green's functions in the PMLs. The solution in the absorbing layers does not matter to us, so it is not a problem if the quality of the solution drops once in the PML (under the assumption that the quality of the solution in the absorbing layers does not affect the quality of the solution in the computational domain, which is not certain and will be discussed in the Conclusion). Hence, the next part consists in coupling the polynomials and the Green's functions.

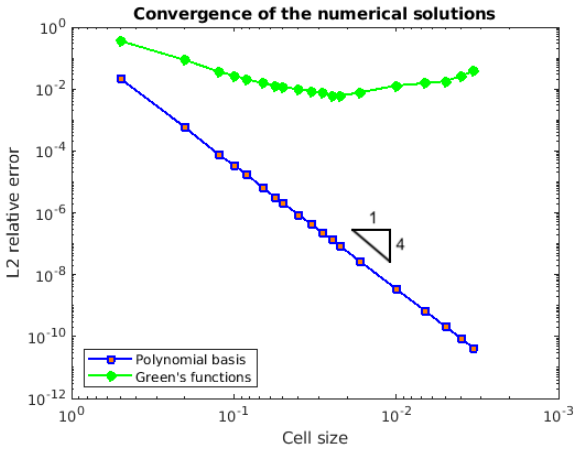


Figure 10.15: Convergence of the numerical solutions

10.3.3 Coupling polynomials and Green's functions

In order to couple the two types of basis functions, we recompute the elementary matrices in the computational domain using the polynomial basis. We also have to recompute the matrices that couple the computation domain with PMLs, and recalculate the contributions from the domain of interest with polynomials.

Let us consider a domain of size $L_x = L_y = 1$, discretized with $n_x = n_y = 100$ cells in each direction for an absorption of $\sigma = 5$ and discretized with $n_x = n_y = 200$ cells in each direction for an absorption of $\sigma = 10$. We visualize the pressure at time

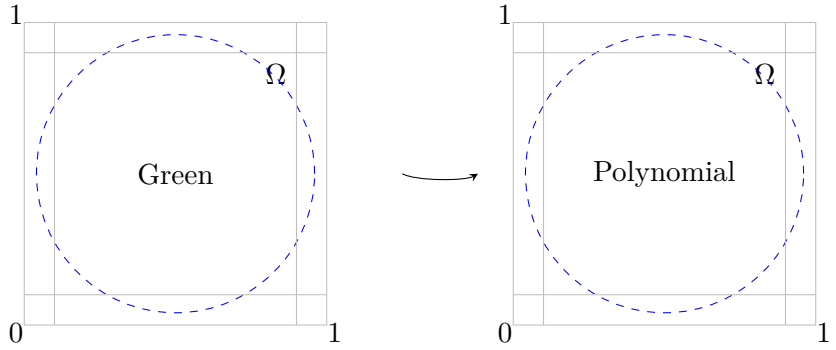


Figure 10.16: In the second part of the third phase, we couple the use of polynomials and Green's functions

$t = 0.3s$, $t = 0.4s$, $t = 0.5s$, $t = 0.6s$ and, the final time, $t = 1s$. We use a first order absorbing boundary condition at the extremities of the PMLs of width $w = 0.2$ and a Gaussian source function. We use 8 source points to evaluate the Green's functions and $\mathbb{P}^0$ polynomials in the domain of interest. The results are presented in Table 10.11. We can see that the solution is absorbed as it enters the PMLs, however, there are very strong reflections even with $\sigma_y = 5$. We present the results for $\mathbb{P}^0$ polynomials because the reflections at the coupling interface are much stronger when increasing the number of polynomials degrees of freedom (DoF^p). We probably need more Green's sources compared to DoF^p, in order to have less reflections, but when increasing the number of source points, the solution explodes depending on the position of these sources.

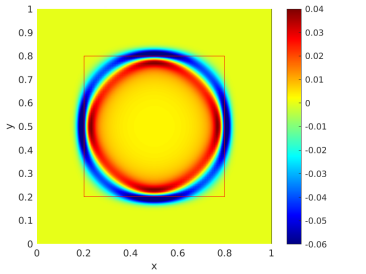
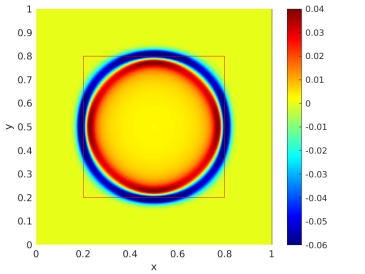
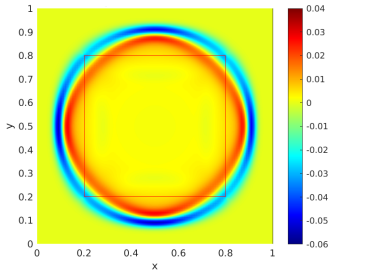
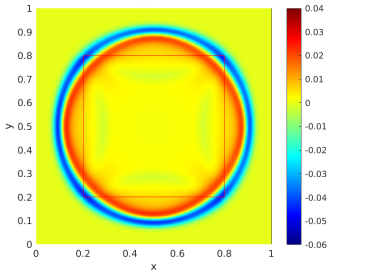
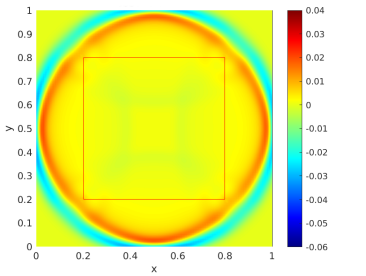
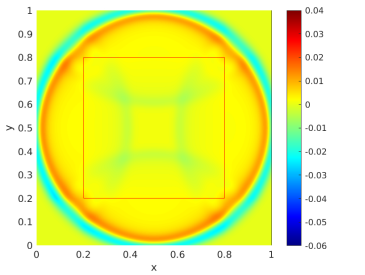
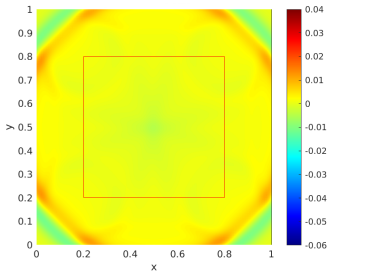
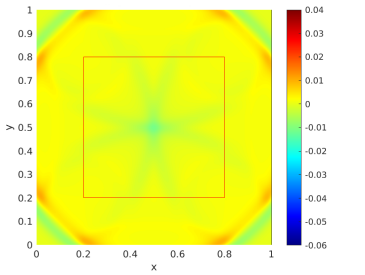
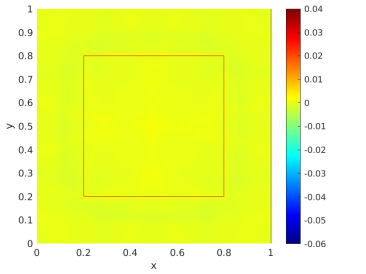
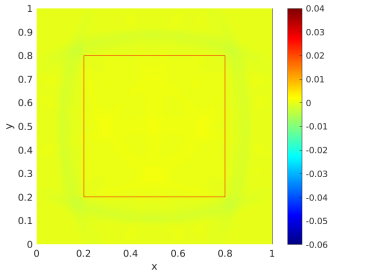
$t(s)$	$\sigma = 5$	$\sigma = 10$
0.3		
0.4		
0.5		
0.6		
1		

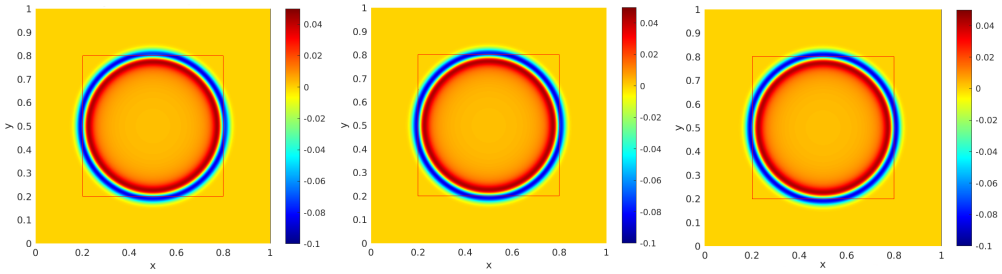
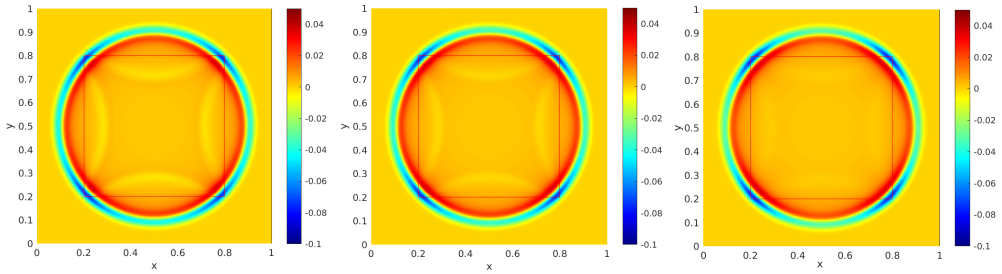
Table 10.11: Pressure represented with coupled polynomials and Green's functions in a domain surrounded with Perfectly Matched Layers

10.3.4 Other basis functions

In Chapter 9 Section 9.5, we introduce alternate basis functions obtained with the Green's function for the acoustic pressure. By using the same principle, we define new functions as follows:

$$\begin{cases} p = (\frac{\partial}{\partial t} + \sigma_x)^2 (\frac{\partial}{\partial t} + \sigma_y)^2 G_p^{pml_{xy}} \,, \\ v_x = -\frac{1}{\rho} (\frac{\partial}{\partial t} + \sigma_x)(\frac{\partial}{\partial t} + \sigma_y)^2 \frac{\partial G_p^{pml_{xy}}}{\partial x} \,, \\ v_y = -\frac{1}{\rho} (\frac{\partial}{\partial t} + \sigma_x)^2 (\frac{\partial}{\partial t} + \sigma_y) \frac{\partial G_p^{pml_{xy}}}{\partial y} \,. \end{cases}$$

whereas the original basis is simply $p = G_p^{pml_{xy}}, v_x = G_{v_x}^{pml_{xy}}, v_y = G_{v_y}^{pml_{xy}}$. We noticed that the results seem more robust with the alternate basis and also have less reflections. In Table 10.12 we present the acoustic pressure simulated with $\mathbb{P}^3$ polynomials in the domain of interest and the original Green's functions with 8 source points in the PMLs compared with the alternate basis with 24 source points in the PMLs. We observe that the original basis produces more reflections even when the mesh is more refined. Since for the alternate basis we use more source points, we also test the original basis with more source points in order to see if this improves the results, but the solution actually blows up, as can be seen in Fig. 10.17.

t(s)	original basis $h = 5e-3$	original basis $h = 2.5e-3$	alternate basis $h = 5e-3$
0.3			
0.4			

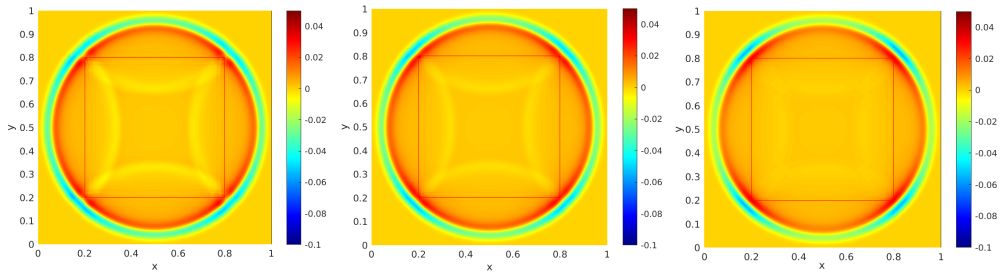
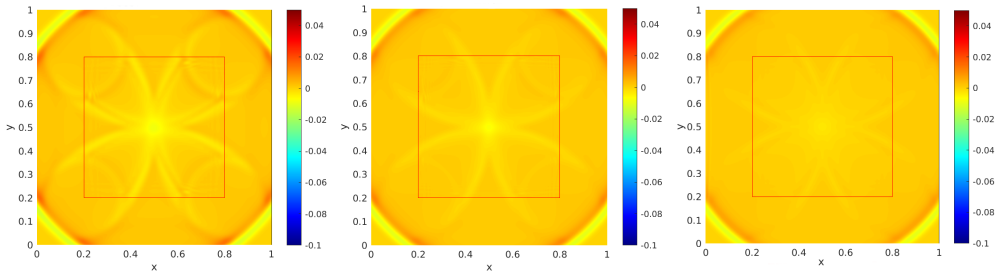
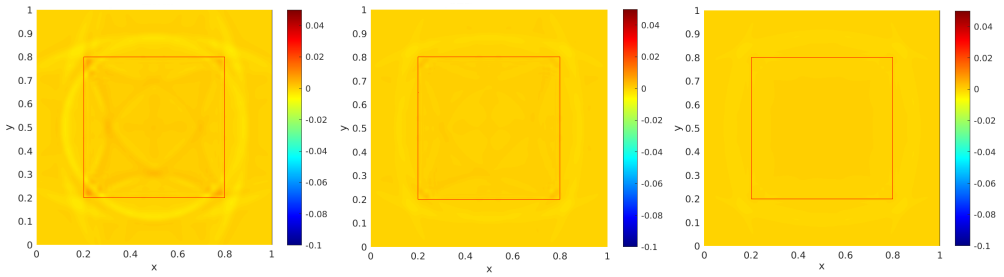
0.45	
0.6	
1	

Table 10.12: Comparison of basis for PML

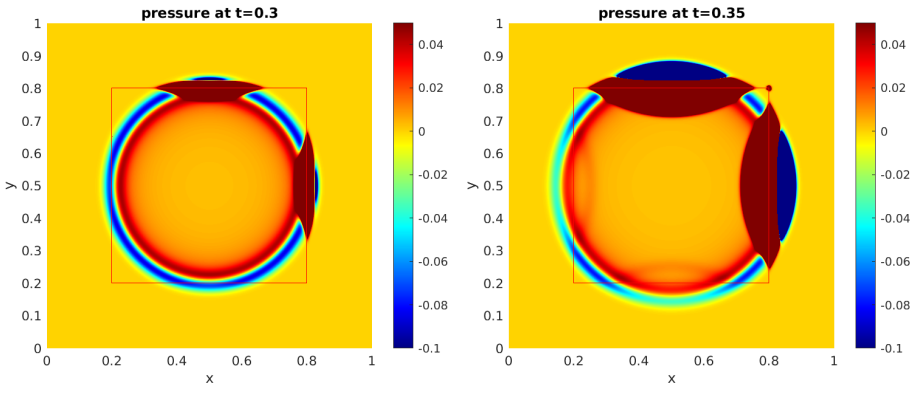


Figure 10.17: Original basis with 16 sources

Conclusion

In this thesis, we presented a Trefftz-DG solver with Tent-Pitching and its implementation in a parallel environment for structured and unstructured meshes. We next address the question of solving PML wave equations with this kind of approach.

The Trefftz-DG solver with Tent-Pitching for space-time acoustic wave equation has already been explored and implemented in [24] for structured meshes, in [26] for unstructured meshes and in [27] in a thread-based parallel environment on unstructured meshes. In this thesis, our purpose was to investigate the potential of this type of solver for geophysical applications. Our starting point was a prototype developed by E. Shishenina [24]. We began with extending her code from structured meshes to unstructured meshes and into a distributed memory parallel environment. Regarding the solver using structured meshes, the results we obtain are very encouraging since the solver has a good scaling when increasing the number of parallel process it is run on and a better scaling when compared to the IPDG solver from the software Hou10ni. Thus, one of the goals, which was to have a faster numerical method, seems to be in our reach with the Trefftz method with Tent-Pitching as candidate. Another aspect we were interested in was to go faster without neglecting the precision of the solution, which is unfortunately often the case. For this point, we remarked that the Trefftz-DG solver also seems more efficient than the IPDG solver. Thus, this allows us to conclude that the Trefftz-DG with Tent-Pitching solver is an encouraging candidate in the search of a faster and more precise numerical method. However, in the case of unstructured meshes we noticed that the performance of our current implementation still requires some optimization. Indeed, since we have to compute a set of matrices and perform an inversion for each constructed tent, the time increases sharply and the IPDG method is much faster in this case. We believe that there is room for improvement, whether in terms of solution methodology or parallelization. In fact, one of the reasons why the unstructured code is slow compared to other methods, is because we need to recompute the matrices at each step. If we had a reference matrix (as is usually done in finite element methods), it would go faster. However the challenging question remains of how to construct a reference element with Trefftz basis functions. Moreover, there is one other important aspect of Tent-Pitching which we do not take advantage of: all tents on a same time level could be computed independently. Thus, implementing the solver in a MPI+Open-MP framework would go faster and the Tent-Pitcher algorithm should facilitate such an implementation. In fact, we have implemented an Open-MP version quite easily and coupling it with MPI is

common. Another thing to observe and that has been explored and tackled in [75] is that it is quite common that a process has to wait for another to finish and in Tent-Pitching, it is is due to the fact that we need to respect causality, so if one process goes too fast it cannot move on if the neighboring process has not constructed enough tents. So, it would be interesting to look into task optimizing in order to overcome this problem and optimize the code even more. Mesh-wise, there has been a lot of work on adaptive Tent-Pitching meshes [46, 47, 48, 49, 50, 51] and it would be an interesting point to explore, since it could be very useful in the context of geophysics.

On the other hand, to our knowledge PMLs have not been implemented in this framework yet and the aim was to test it out and see if it works. We observed that it is complex to obtain solutions in the spacetime PML case and the construction of a working variational formulation is not obvious. In fact, we elaborated several kinds of variational formulations followed by different sets of basis functions. Some of the resulting combinations led to poor results and for the time being, we don't have any clear analysis of what's going on. One originality of our approach has been to consider Green's functions. They provide a natural framework for the Trefftz-DG formulation of our PML wave equation. However, the Green's functions are hard to control, because they are very sensitive to the position and number of sources required for carrying the approximation and considering a continuous damping function seems complex. So, the prospects of this work would be to investigate the question of source points, and find ways to optimize their number and position. To do so, it would be very interesting to see how exactly the choice of these points affects the solution to be able to control it. One of the difficulty is related to the fact that the Green's functions become (at least numerically) linearly dependent when they are too close. This results in the ill-conditioning of the local matrices. The question of reducing the ill-conditioning issue has been addressed for elliptic problems solved with the fundamental solution method (FSM) [108]. FSM approach shares some implementation ideas with ours even if it involves continuous basis functions and focuses on the solution of boundary integral equation. Closer to our work, we can cite the Ultra-Weak Variational Formulation of wave problems which uses plane-waves as basis functions. The same ill-conditioned issue occurs, still because the plane waves tend to become linearly dependent. In [109], a method has been proposed to reduce the ill-conditioning. It consists in using a singular value decomposition of the local matrix and to remove or to increase the singular values that are below a given threshold. We would like to consider the idea of extending the methodology in [109] to spacetime problems. However, we can conclude that Green's functions are not adapted to an industrial context. Since we struggle with the optimal choice of source points, their precision and convergence to the analytical solution is quite poor when compared to polynomials (see 10.3.2). This is the reason why we preferred to couple the use of polynomials in the domain of interest with Green's functions in the PMLs only. In order to conclude if this is the best option, we should measure the precision of the solution with PML and see if the use of Green's functions does not affect the overall convergence to the solution, which is a possibility and unwanted. Another idea can be to change our stance

and explore other kinds of solutions than Green's functions. In section 8.6.1, we saw that it is not possible to obtain full polynomials as solutions for this problem. However, we think that a polynomial multiplied by an exponential function of the damping function could be an alternate solution. In fact, we tried out solutions in this form and the results are encouraging. However, they are not solutions to our PML equation, but rather solutions to other dissipative wave equations. So, this needs to be looked into in more details. In a technical point of view, we only implemented the PML for structured meshes in sequential, so an additional prospect is to extend the work to unstructured meshes and in a parallel environment.

Appendix A

Basis functions

We present below the polynomial functions we used as basis in our Trefftz methods. Trefftz methods use local solutions of the considered problem as basis functions, thus, the polynomials we use are exact solutions to the acoustic wave equation. We give the $\mathbb{P}^0$, $\mathbb{P}^1$, $\mathbb{P}^2$ and $\mathbb{P}^3$ polynomials, which gives us $3, 9, 18$ and 30 degrees of freedom respectively.

p=0, Number of DoF=3		
$\phi_1^p = -c$	$\phi_1^{vx} = 0$	$\phi_1^{vy} = 0$
$\phi_2^p = 0$	$\phi_2^{vx} = 1$	$\phi_2^{vy} = 0$
$\phi_3^p = 0$	$\phi_3^{vx} = 0$	$\phi_3^{vy} = 1$
p=1, Number of DoF=9		
$\phi_4^p = x$	$\phi_4^{vx} = -t$	$\phi_4^{vy} = 0$
$\phi_5^p = -c^2 t$	$\phi_5^{vx} = x$	$\phi_5^{vy} = 0$
$\phi_6^p = 0$	$\phi_6^{vx} = 0$	$\phi_6^{vy} = x$
$\phi_7^p = y$	$\phi_7^{vx} = 0$	$\phi_7^{vy} = -t$
$\phi_8^p = 0$	$\phi_8^{vx} = y$	$\phi_8^{vy} = 0$
$\phi_9^p = -c^2 t$	$\phi_9^{vx} = 0$	$\phi_9^{vy} = y$
p=2, Number of DoF=18		
$\phi_{10}^p = x^2 + c^2 t$	$\phi_{10}^{vx} = -2xt$	$\phi_{10}^{vy} = 0$
$\phi_{11}^p = y^2 + c^2 t$	$\phi_{11}^{vx} = 0$	$\phi_{11}^{vy} = -2yt$
$\phi_{12}^p = xy$	$\phi_{12}^{vx} = -yt$	$\phi_{12}^{vy} = -xt$
$\phi_{13}^p = yt$	$\phi_{13}^{vx} = -xy/c^2$	$\phi_{13}^{vy} = -t^2/2$
$\phi_{14}^p = xt$	$\phi_{14}^{vx} = -t^2/2$	$\phi_{14}^{vy} = -xy/c^2$
$\phi_{15}^p = 0$	$\phi_{15}^{vx} = x^2$	$\phi_{15}^{vy} = -2xy$
$\phi_{16}^p = 0$	$\phi_{16}^{vx} = y^2$	$\phi_{16}^{vy} = 0$
$\phi_{17}^p = 0$	$\phi_{17}^{vx} = 0$	$\phi_{17}^{vy} = x^2$
$\phi_{18}^p = 0$	$\phi_{18}^{vx} = -2xy$	$\phi_{18}^{vy} = y^2$

p=3, Number of DoF=30		
$\phi_{19}^{p} = x^3 + 3c^2xt^2$	$\phi_{19}^{vx} = -c^2t^3 - 3x^2t$	$\phi_{19}^{vy} = 0$
$\phi_{20}^{p} = y^3 + 3c^2yt^2$	$\phi_{20}^{vx} = 0$	$\phi_{20}^{vy} = -c^2t^3 - 3y^2t$
$\phi_{21}^{p} = x^2y + c^2yt^2$	$\phi_{21}^{vx} = -2xyt$	$\phi_{21}^{vy} = -{}^{c^2t^3}/_{3} - x^2t$
$\phi_{22}^{p} = {}^{c^2t^3}/_{3} + x^2t$	$\phi_{22}^{vx} = -xt^2$	$\phi_{22}^{vy} = -{}^{x^2y}/_{c^2}$
$\phi_{23}^{p} = y^2x + c^2xt^2$	$\phi_{23}^{vx} = -{}^{c^2t^3}/_{3} - y^2t$	$\phi_{23}^{vy} = -2xyt$
$\phi_{24}^{p} = {}^{c^2t^3}/_{3} + y^2t$	$\phi_{24}^{vx} = 0$	$\phi_{24}^{vy} = -{}^{y^3}/_{3c^2} - {}^{xt^2}/_{2}$
$\phi_{25}^{p} = xyt$	$\phi_{25}^{vx} = -{}^{y^2t}/_{2}$	$\phi_{25}^{vy} = -{}^{xy^2}/_{2c^2} - {}^{xt^2}/_{2}$
$\phi_{26}^{p} = 0$	$\phi_{26}^{vx} = x^3$	$\phi_{26}^{vy} = -3x^2y$
$\phi_{27}^{p} = 0$	$\phi_{27}^{vx} = y^3$	$\phi_{27}^{vy} = 0$
$\phi_{28}^{p} = 0$	$\phi_{28}^{vx} = x^2y$	$\phi_{28}^{vy} = -xy^2$
$\phi_{29}^{p} = 0$	$\phi_{29}^{vx} = xy^2$	$\phi_{29}^{vy} = -{}^{y^3}/_{3}$
$\phi_{30}^{p} = 0$	$\phi_{30}^{vx} = 0$	$\phi_{30}^{vy} = x^3$

Appendix B

Gaussian quadrature points and weights

The quadrature formula we used is of order 21 and contains the following 91 points of coordinates (x, y) and associated weights.

Coordinates		Weights
x	y	
0.333333333333333	0.333333333333333	0.0137811284764382
0.200935277065085	0.598129445869829	0.0110301077067443
0.200935277065085	0.200935277065085	0.0110301077067443
0.598129445869829	0.200935277065085	0.0110301077067443
0.437659165961927	0.124681668076146	0.0117300079693357
0.437659165961927	0.437659165961927	0.0117300079693357
0.124681668076146	0.437659165961927	0.0117300079693357
0.00343395649059618	0.993132087018808	0.000163444797523595
0.00343395649059618	0.00343395649059618	0.000163444797523595
0.993132087018808	0.00343395649059618	0.000163444797523595
0.0466434847753068	0.906713030449386	0.00163265973146998
0.0466434847753068	0.0466434847753068	0.00163265973146998
0.906713030449386	0.0466434847753068	0.00163265973146998
0.386422251763071	0.227155496473857	0.0058782314577064
0.386422251763071	0.386422251763071	0.0058782314577064
0.227155496473857	0.386422251763071	0.0058782314577064
0.0954354711085309	0.809129057782938	0.00589038420995576
0.0954354711085309	0.0954354711085309	0.00589038420995576
0.809129057782938	0.0954354711085309	0.00589038420995576
0.00876756877638032	0.0357186278731634	0.00113440540940057
0.955513803350456	0.00876756877638032	0.00113440540940057
		Continued on next page

Coordinates		Weights
x	y	
0.0357186278731634	0.955513803350456	0.00113440540940057
0.0357186278731634	0.00876756877638032	0.00113440540940057
0.955513803350456	0.0357186278731634	0.00113440540940057
0.00876756877638032	0.955513803350456	0.00113440540940057
0.0052179616554856	0.108143224915646	0.00129800548221816
0.886638813428868	0.0052179616554856	0.00129800548221816
0.108143224915646	0.886638813428868	0.00129800548221816
0.108143224915646	0.0052179616554856	0.00129800548221816
0.886638813428868	0.108143224915646	0.00129800548221816
0.0052179616554856	0.886638813428868	0.00129800548221816
0.00827270451968939	0.207464449599876	0.00231726489293593
0.784262845880434	0.00827270451968939	0.00231726489293593
0.207464449599876	0.784262845880434	0.00231726489293593
0.207464449599876	0.00827270451968939	0.00231726489293593
0.784262845880434	0.207464449599876	0.00231726489293593
0.00827270451968939	0.784262845880434	0.00231726489293593
0.031391336229483	0.0856847087203169	0.00239716802727443
0.8829239550502	0.031391336229483	0.00239716802727443
0.0856847087203169	0.8829239550502	0.00239716802727443
0.0856847087203169	0.031391336229483	0.00239716802727443
0.8829239550502	0.0856847087203169	0.00239716802727443
0.031391336229483	0.8829239550502	0.00239716802727443
0.00951403254463395	0.321494003014289	0.00285623941836181
0.668991964441077	0.00951403254463395	0.00285623941836181
0.321494003014289	0.668991964441077	0.00285623941836181
0.321494003014289	0.00951403254463395	0.00285623941836181
0.668991964441077	0.321494003014289	0.00285623941836181
0.00951403254463395	0.668991964441077	0.00285623941836181
0.0099856601710977	0.437942218793341	0.00293291380216106
0.552072121035561	0.0099856601710977	0.00293291380216106
0.437942218793341	0.552072121035561	0.00293291380216106
0.437942218793341	0.0099856601710977	0.00293291380216106
0.552072121035561	0.437942218793341	0.00293291380216106
0.0099856601710977	0.552072121035561	0.00293291380216106
0.0404905813398536	0.161916453063578	0.00470688152954579
0.797592965596569	0.0404905813398536	0.00470688152954579
0.161916453063578	0.797592965596569	0.00470688152954579
0.161916453063578	0.0404905813398536	0.00470688152954579
0.797592965596569	0.161916453063578	0.00470688152954579

Continued on next page

Coordinates		Weights
x	y	
0.0404905813398536	0.797592965596569	0.00470688152954579
0.0479805174782336	0.274504767401995	0.00670747189832821
0.677514715119771	0.0479805174782336	0.00670747189832821
0.274504767401995	0.677514715119771	0.00670747189832821
0.274504767401995	0.0479805174782336	0.00670747189832821
0.677514715119771	0.274504767401995	0.00670747189832821
0.0479805174782336	0.677514715119771	0.00670747189832821
0.0516665863359014	0.405335998075007	0.00785845904604162
0.542997415589092	0.0516665863359014	0.00785845904604162
0.405335998075007	0.542997415589092	0.00785845904604162
0.405335998075007	0.0516665863359014	0.00785845904604162
0.542997415589092	0.405335998075007	0.00785845904604162
0.0516665863359014	0.542997415589092	0.00785845904604162
0.106802413773596	0.187737680656435	0.00843184150721845
0.705459905569969	0.106802413773596	0.00843184150721845
0.187737680656435	0.705459905569969	0.00843184150721845
0.187737680656435	0.106802413773596	0.00843184150721845
0.705459905569969	0.187737680656435	0.00843184150721845
0.106802413773596	0.705459905569969	0.00843184150721845
0.119502592167436	0.305696834766055	0.01069501354266
0.574800573066508	0.119502592167436	0.01069501354266
0.305696834766055	0.574800573066508	0.01069501354266
0.305696834766055	0.119502592167436	0.01069501354266
0.574800573066508	0.305696834766055	0.01069501354266
0.119502592167436	0.574800573066508	0.01069501354266
0.216076724624494	0.312144466870891	0.0115383960947463
0.471778808504615	0.216076724624494	0.0115383960947463
0.312144466870891	0.471778808504615	0.0115383960947463
0.312144466870891	0.216076724624494	0.0115383960947463
0.471778808504615	0.312144466870891	0.0115383960947463
0.216076724624494	0.471778808504615	0.0115383960947463

Appendix C

CMake configuration file example

```
cmake_minimum_required(VERSION 2.8)

# set the project name
project (name)
enable_language (Fortran)

# flags for compilation in ../cmake/unstr_flags.cmake
SET(CMAKE_USER_MAKE_RULES_OVERRIDE
"${CMAKE_SOURCE_DIR}/cmake/unstr_flags.cmake")

# compiler
set(CMAKE_Fortran_COMPILER CompilerName)

# MPI
find_package(MPI REQUIRED)

# libs
set(LIB ~/path )

# add all needed directories
add_subdirectory(NameDirectory)

# add dependecies : directory A depends on directory B
add_dependencies(DirectoryA DirectoryB)

# add the executable
set (EXECUTABLES NameExec)
add_executable(NameExec main.f90)
target_link_libraries (main directories -llibs ${LIBS})
```

Appendix D

Green's functions derivatives

$$
\begin{aligned}
I_x(x,y,t) &= \frac{tx}{\rho(x^2+y^2)} \\
\frac{\partial I_x(x,y,t)}{\partial t} &= \frac{x}{\rho(x^2+y^2)} \\
\frac{\partial^2 I_x}{\partial t \partial t} &= 0 \\
\frac{\partial^2 I_x}{\partial x \partial t} &= \frac{(y^2-x^2)}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_x}{\partial y \partial t} &= \frac{-2xy}{\rho(x^2+y^2)^2} \\
\frac{\partial I_x(x,y,t)}{\partial x} &= (y^2-x^2)\frac{t}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_x}{\partial t \partial x} &= \frac{(y^2-x^2)}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_x}{\partial x \partial x} &= (x^2-3y^2)\frac{2tx}{\rho(x^2+y^2)^3} \\
\frac{\partial^2 I_x}{\partial y \partial x} &= (3x^2-y^2)\frac{2ty}{\rho(x^2+y^2)^3} \\
\frac{\partial I_x(x,y,t)}{\partial y} &= -\frac{2txy}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_x}{\partial t \partial y} &= \frac{\partial^2 I}{\partial y \partial t} \\
\frac{\partial^2 I_x}{\partial x \partial y} &= \frac{\partial^2 I}{\partial y \partial x} \\
\frac{\partial^2 I_x}{\partial y \partial y} &= (3y^2-x^2)\frac{2tx}{\rho(x^2+y^2)^3}
\end{aligned}
$$

$$
\begin{aligned}
I_y(x,y,t) &= \frac{ty}{\rho(x^2+y^2)} \\
\frac{\partial I_y(x,y,t)}{\partial t} &= \frac{y}{\rho(x^2+y^2)} \\
\frac{\partial^2 I_y}{\partial t \partial t} &= 0 \\
\frac{\partial^2 I_y}{\partial x \partial t} &= \frac{-2xy}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_y}{\partial y \partial t} &= \frac{(x^2-y^2)}{\rho(x^2+y^2)^2} \\
\frac{\partial I_y(x,y,t)}{\partial x} &= -\frac{2txy}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_y}{\partial t \partial x} &= \frac{\partial^2 I_x}{\partial y \partial t} \\
\frac{\partial^2 I_y}{\partial x \partial x} &= \frac{\partial^2 I}{\partial y \partial x} \\
\frac{\partial^2 I_y}{\partial y \partial x} &= (3y^2-x^2)\frac{2tx}{\rho(x^2+y^2)^3} \\
\frac{\partial I_y(x,y,t)}{\partial y} &= (x^2-y^2)\frac{t}{\rho(x^2+y^2)^2} \\
\frac{\partial^2 I_y}{\partial t \partial y} &= \frac{\partial^2 I_y}{\partial y \partial t} \\
\frac{\partial^2 I_y}{\partial x \partial y} &= \frac{\partial^2 I_y}{\partial y \partial x} \\
\frac{\partial^2 I_y}{\partial y \partial y} &= (y^2-3x^2)\frac{2ty}{\rho(x^2+y^2)^3}
\end{aligned}
$$

$$J_y(x,y,t) = \frac{y}{\rho(x^2+y^2)}\sqrt{t^2-\frac{x^2+y^2}{c^2}}$$

$$\frac{\partial J_y(x,y,t)}{\partial t} = \frac{ty}{\rho(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}$$

$$\frac{\partial^2 J_y}{\partial t \partial t} = \frac{-1}{c^2\rho(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial^2 J_y}{\partial x \partial t} = txy\frac{3(x^2+y^2)-2t^2c^2}{\rho c^2(x^2+y^2)^2(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial^2 J_x}{\partial y \partial t} = \frac{\partial^2 J_y}{\partial x \partial t}$$

$$\frac{\partial J_y(x,y,t)}{\partial x} = -\frac{2xy}{\rho(x^2+y^2)^2}\sqrt{t^2-\frac{x^2+y^2}{c^2}} - \frac{xy}{\rho c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}$$

$$\frac{\partial^2 J_y}{\partial t \partial x} = \frac{\partial^2 J_y}{\partial x \partial t}$$

$$\frac{\partial^2 J_y}{\partial x \partial x} = (3x^2-y^2)\frac{2y}{\rho(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}} + \frac{y(3x^2-y^2)}{\rho c^2(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}} - \frac{x^2y}{\rho c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial^2 J_y}{\partial y \partial x} = (3y^2-x^2)\frac{2x}{\rho(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}} + \frac{x(3y^2-x^2)}{\rho c^2(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}} - \frac{xy^2}{\rho c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial J_y(x,y,t)}{\partial y} = \frac{x^2-y^2}{\rho(x^2+y^2)^2}\sqrt{t^2-\frac{x^2+y^2}{c^2}} - \frac{y^2}{\rho c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}$$

$$\frac{\partial^2 J_y}{\partial t \partial y} = \frac{\partial^2 J_y}{\partial y \partial t}$$

$$\frac{\partial^2 J_y}{\partial x \partial y} = \frac{\partial^2 J_y}{\partial y \partial x}$$

$$\frac{\partial^2 J_y}{\partial y \partial y} = -(3x^2-y^2)\frac{2y}{\rho(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}} - \frac{y(3x^2-y^2)}{\rho c^2(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}} - \frac{y^3}{\rho c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$J_x(x,y,t) = \frac{x}{\rho(x^2+y^2)}\sqrt{t^2-\frac{x^2+y^2}{c^2}}$$

$$\frac{\partial J_x(x,y,t)}{\partial t} = \frac{tx}{\rho(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}$$

$$\frac{\partial^2 J_x}{\partial t \partial t} = \frac{\partial^2 J_y}{\partial t \partial t}$$

$$\frac{\partial^2 J_x}{\partial x \partial t} = \frac{t(x^2-y^2)}{\rho(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}} + \frac{tx^2}{\rho c^2(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial^2 J_x}{\partial y \partial t} = \frac{t(y^2-x^2)}{\rho(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}} + \frac{ty^2}{\rho c^2(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial J_x(x,y,t)}{\partial x} = \frac{y^2-x^2}{\rho(x^2+y^2)^2}\sqrt{t^2-\frac{x^2+y^2}{c^2}} - \frac{x^2}{\rho c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}$$

$$\frac{\partial^2 J_x}{\partial t \partial x} = \frac{\partial^2 J_x}{\partial x \partial t}$$

$$\frac{\partial^2 J_x}{\partial x \partial x} = (x^2-3y^2)\frac{2x}{\rho(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}} + \frac{x(x^2-3y^2)}{\rho c^2(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}} - \frac{x^3}{\rho c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}$$

$$\frac{\partial^2 J_x}{\partial y \partial x} = \frac{\partial^2 J_y}{\partial x \partial x}$$

$$\frac{\partial J_x(x,y,t)}{\partial y} = \frac{\partial J_y(x,y,t)}{\partial x}$$

$$\frac{\partial^2 J_x}{\partial t \partial y} = \frac{\partial^2 J_x}{\partial y \partial t}$$

$$\frac{\partial^2 J_x}{\partial x \partial y} = \frac{\partial^2 J_x}{\partial y \partial x}$$

$$\frac{\partial^2 J_x}{\partial y \partial y} = \frac{\partial^2 J_y}{\partial y \partial x}$$

$$
\begin{aligned}
A(x,y,t) =& (\sigma_y y^2 + \sigma_x x^2)\frac{t}{x^2+y^2} \\
\frac{\partial A(x,y,t)}{\partial t} =& \frac{(\sigma_y y^2 + \sigma_x x^2)}{x^2+y^2} \\
\frac{\partial^2 A}{\partial t \partial t} =& 0 \\
\frac{\partial^2 A}{\partial x \partial t} =& -2(\sigma_y y^2 + \sigma_x x^2)\frac{x}{(x^2+y^2)^2} \\
\frac{\partial^2 A}{\partial y \partial t} =& -2(\sigma_y y^2 + \sigma_x x^2)\frac{y}{(x^2+y^2)^2} \\
\frac{\partial A(x,y,t)}{\partial x} =& -2(\sigma_y y^2 + \sigma_x x^2)\frac{tx}{(x^2+y^2)^2} \\
\frac{\partial^2 A}{\partial t \partial x} =& \frac{\partial^2 A}{\partial x \partial t} \\
\frac{\partial^2 A}{\partial x \partial x} =& -2(\sigma_y y^2 + \sigma_x x^2)\frac{t}{(x^2+y^2)^2} \\
& + (\sigma_y y^2 + \sigma_x x^2)\frac{6tx^2}{(x^2+y^2)^3} \\
\frac{\partial^2 A}{\partial y \partial x} =& (\sigma_y y^2 + \sigma_x x^2)\frac{6txy}{(x^2+y^2)^3} \\
\frac{\partial A(x,y,t)}{\partial y} =& -2(\sigma_y y^2 + \sigma_x x^2)\frac{ty}{(x^2+y^2)^2} \\
\frac{\partial^2 A}{\partial t \partial y} =& \frac{\partial^2 A}{\partial y \partial t} \\
\frac{\partial^2 A}{\partial x \partial y} =& \frac{\partial^2 A}{\partial y \partial x} \\
\frac{\partial^2 A}{\partial y \partial y} =& -2(\sigma_y y^2 + \sigma_x x^2)\frac{t}{(x^2+y^2)^2} \\
& + (\sigma_y y^2 + \sigma_x x^2)\frac{6ty^2}{(x^2+y^2)^3}
\end{aligned}
$$

$$
\begin{aligned}
G_p =& \frac{1}{2\pi\sqrt{t^2 - \frac{x^2+y^2}{c^2}}} \\
\frac{\partial G_p}{\partial t} =& \frac{-t}{(t^2 - \frac{x^2+y^2}{c^2})^{3/2}} \\
\frac{\partial^2 G_p}{\partial t \partial t} =& \frac{2c^2t^2 + x^2 + y^2}{c^4(t^2 - \frac{x^2+y^2}{c^2})^{5/2}} \\
\frac{\partial^2 G_p}{\partial x \partial t} =& \frac{-3tx}{c^2(t^2 - \frac{x^2+y^2}{c^2})^{5/2}} \\
\frac{\partial^2 G_p}{\partial y \partial t} =& \frac{-3ty}{c^2(t^2 - \frac{x^2+y^2}{c^2})^{5/2}} \\
\frac{\partial G_p}{\partial x} =& \frac{x}{c^2(t^2 - \frac{x^2+y^2}{c^2})^{3/2}} \\
\frac{\partial^2 G_p}{\partial t \partial x} =& \frac{\partial^2 G_p}{\partial x \partial t} \\
\frac{\partial^2 G_p}{\partial x \partial x} =& \frac{c^2t^2 + 2x^2 - y^2}{c^4(t^2 - \frac{x^2+y^2}{c^2})^{5/2}} \\
\frac{\partial^2 G_p}{\partial y \partial x} =& \frac{3xy}{c^4(t^2 - \frac{x^2+y^2}{c^2})^{5/2}} \\
\frac{\partial G_p}{\partial y} =& \frac{y}{c^2(t^2 - \frac{x^2+y^2}{c^2})^{3/2}} \\
\frac{\partial^2 G_p}{\partial t \partial y} =& \frac{\partial^2 G_p}{\partial y \partial t} \\
\frac{\partial^2 G_p}{\partial x \partial y} =& \frac{\partial^2 G_p}{\partial y \partial x} \\
\frac{\partial^2 G_p}{\partial y \partial y} =& \frac{c^2t^2 + 2y^2 - x^2}{c^4(t^2 - \frac{x^2+y^2}{c^2})^{5/2}}
\end{aligned}
$$

$$
\begin{aligned}
B(x,y,t) &= (\sigma_y+\sigma_x)\frac{xy}{x^2+y^2}\sqrt{t^2-\frac{x^2+y^2}{c^2}}\\
\frac{\partial B(x,y,t)}{\partial t} &= (\sigma_y+\sigma_x)\frac{txy}{x^2+y^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}}\\
\frac{\partial^2 B}{\partial t\partial t} &= (\sigma_y+\sigma_x)\left(\frac{xy}{x^2+y^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}}-\frac{t^2xy}{x^2+y^2(t^2-\frac{x^2+y^2}{c^2})^{3/2}}\right)\\
\frac{\partial^2 B}{\partial x\partial t} &= (\sigma_y+\sigma_x)\left(\frac{ty(y^2-x^2)}{(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}}+\frac{tx^2y}{c^2(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}\right)\\
\frac{\partial^2 B}{\partial y\partial t} &= (\sigma_y+\sigma_x)\left(\frac{tx(x^2-y^2)}{(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}}+\frac{txy^2}{c^2(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}\right)\\
\frac{\partial B(x,y,t)}{\partial x} &= (\sigma_y+\sigma_x)\left(\frac{y(y^2-x^2)}{(x^2+y^2)^2\sqrt{t^2-\frac{x^2+y^2}{c^2}}}-\frac{x^2y}{c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}\right)\\
\frac{\partial^2 B}{\partial t\partial x} &= \frac{\partial^2 B}{\partial x\partial t}\\
\frac{\partial^2 B}{\partial x\partial x} &= (\sigma_y+\sigma_x)\left(\frac{xy(x^2-3y^2)}{c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}-\frac{x^3y}{c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}\right.\\
&\quad\left.+6\frac{x^3y}{(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}}\right)\\
\frac{\partial^2 B}{\partial y\partial x} &= (\sigma_y+\sigma_x)\left(\frac{(2x^2y^2-y^4-x^4)}{c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}-\frac{x^2y^2}{c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}\right.\\
&\quad\left.+\frac{4x^2y^2-x^4-y^4}{(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}}\right)\\
\frac{\partial B(x,y,t)}{\partial y} &= (\sigma_y+\sigma_x)\left(\frac{x(x^2-y^2)}{(x^2+y^2)^2}\sqrt{t^2-\frac{x^2+y^2}{c^2}}-\frac{xy^2}{c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}\right)\\
\frac{\partial^2 B}{\partial t\partial y} &= \frac{\partial^2 B}{\partial y\partial t}\\
\frac{\partial^2 B}{\partial x\partial y} &= \frac{\partial^2 B}{\partial y\partial x}\\
\frac{\partial^2 B}{\partial y\partial y} &= (\sigma_y+\sigma_x)\left(\frac{xy(x^2-3y^2)}{c^2(x^2+y^2)\sqrt{t^2-\frac{x^2+y^2}{c^2}}}-\frac{xy^3}{c^4(x^2+y^2)(t^2-\frac{x^2+y^2}{c^2})^{3/2}}\right.\\
&\quad\left.+6\frac{xy^3}{(x^2+y^2)^3}\sqrt{t^2-\frac{x^2+y^2}{c^2}}\right)
\end{aligned}
$$

$$
\begin{aligned}
G_p^{pml} =& e^{-A(x,y,t)} cos[B(x,y,t)] G_p \\
\frac{\partial G_p^{pml}}{\partial \bullet} =& \left(\frac{\partial G_p}{\partial \bullet} - G_p \frac{\partial A(x,y,t)}{\partial \bullet} \right) e^{-A(x,y,t)} cos[B(x,y,t)] - G_p B(x,y,t) e^{-A(x,y,t)} sin[B(x,y,t)] \\
\frac{\partial^2 G_p^{pml}}{\partial \circ \partial \bullet} =& \left(\frac{\partial^2 G_p}{\partial \circ \partial \bullet} - \frac{\partial G_p}{\partial \circ} \frac{\partial A(x,y,t)}{\partial \bullet} - G_p \frac{\partial^2 A(x,y,t)}{\partial \circ \partial \bullet} - \frac{\partial A(x,y,t)}{\partial \circ} \left(\frac{\partial G_p}{\partial \bullet} - G_p \frac{\partial A(x,y,t)}{\partial \bullet} \right) \right. \\
& \left. - G_p \frac{\partial B(x,y,t)}{\partial \circ} \frac{\partial B(x,y,t)}{\partial \bullet} \right) \times e^{-A(x,y,t)} cos[B(x,y,t)] \\
& + \left(- \frac{\partial B(x,y,t)}{\partial \circ} \left(\frac{\partial G_p}{\partial \bullet} - G_p \frac{\partial A(x,y,t)}{\partial \bullet} \right) - \frac{\partial G_p}{\partial \circ} \frac{\partial B(x,y,t)}{\partial \bullet} + G_p \frac{\partial A(x,y,t)}{\partial \circ} \frac{\partial B(x,y,t)}{\partial \bullet} \right. \\
& \left. - G_p \frac{\partial^2 B(x,y,t)}{\partial \circ \partial \bullet} \right) \times e^{-A(x,y,t)} sin[B(x,y,t)]
\end{aligned}
$$

$$
\begin{aligned}
K =& e^{-A(x,y,t)} sin[B(x,y,t)] G_p \\
\frac{\partial K}{\partial \bullet} =& \left(\frac{\partial G_p}{\partial \bullet} - G_p \frac{\partial A(x,y,t)}{\partial \bullet} \right) e^{-A(x,y,t)} sin[B(x,y,t)] + G_p \frac{\partial B(x,y,t)}{\partial \bullet} e^{-A(x,y,t)} cos[B(x,y,t)] \\
\frac{\partial^2 K}{\partial \circ \partial \bullet} =& \left(\frac{\partial^2 G_p}{\partial \circ \partial \bullet} - \frac{\partial G_p}{\partial \circ} \frac{\partial A(x,y,t)}{\partial \bullet} - G_p \frac{\partial^2 A(x,y,t)}{\partial \circ \partial \bullet} - \frac{\partial A(x,y,t)}{\partial \circ} \left(\frac{\partial G_p}{\partial \bullet} - G_p \frac{\partial A(x,y,t)}{\partial \bullet} \right) \right. \\
& \left. - G_p \frac{\partial B(x,y,t)}{\partial \circ} \frac{\partial B(x,y,t)}{\partial \bullet} \right) \times e^{-A(x,y,t)} sin[B(x,y,t)] \\
& - \left(- \frac{\partial B(x,y,t)}{\partial \circ} \left(\frac{\partial G_p}{\partial \bullet} - G_p \frac{\partial A(x,y,t)}{\partial \bullet} \right) - \frac{\partial G_p}{\partial \circ} \frac{\partial B(x,y,t)}{\partial \bullet} + G_p \frac{\partial A(x,y,t)}{\partial \circ} \frac{\partial B(x,y,t)}{\partial \bullet} \right. \\
& \left. - G_p \frac{\partial^2 B(x,y,t)}{\partial \circ \partial \bullet} \right) \times e^{-A(x,y,t)} cos[B(x,y,t)]
\end{aligned}
$$

$$
\begin{aligned}
G_{v_x}^{pml} =& G_p^{pml} I_x(x,y,t) - K(x,y,t) J_y(x,y,t) \\
\frac{\partial G_{v_x}^{pml}}{\partial \bullet} =& \frac{\partial G_p^{pml}}{\partial \bullet} I_x(x,y,t) + G_p^{pml} \frac{\partial I_x(x,y,t)}{\partial \bullet} - \frac{\partial K(x,y,t)}{\partial \bullet} J_y(x,y,t) - K(x,y,t) \frac{\partial J_y(x,y,t)}{\partial \bullet} \\
\frac{\partial^2 G_{v_x}^{pml}}{\partial \circ \partial \bullet} =& \frac{\partial^2 G_p^{pml}}{\partial \circ \partial \bullet} I_x + \frac{\partial G_p^{pml}}{\partial \bullet} \frac{\partial I_x}{\partial \circ} + \frac{\partial G_p^{pml}}{\partial \circ} \frac{\partial I_x}{\partial \bullet} + G_p^{pml} \frac{\partial^2 I_x}{\partial \circ \partial \bullet} - \frac{\partial^2 K}{\partial \circ \partial \bullet} J_y - \frac{\partial K}{\partial \bullet} \frac{\partial J_y}{\partial \circ} - \frac{\partial K}{\partial \circ} \frac{\partial J_y}{\partial \bullet} \\
& - K \frac{\partial^2 J_y}{\partial \circ \partial \bullet}
\end{aligned}
$$

$$
\begin{aligned}
G_{v_y}^{pml} =& G_p^{pml} I_y(x,y,t) + K(x,y,t) J_x(x,y,t) \\
\frac{\partial G_{v_y}^{pml}}{\partial \bullet} =& \frac{\partial G_p^{pml}}{\partial \bullet} I_y(x,y,t) + G_p^{pml} \frac{\partial I_y(x,y,t)}{\partial \bullet} + \frac{\partial K(x,y,t)}{\partial \bullet} J_x(x,y,t) + K(x,y,t) \frac{\partial J_x(x,y,t)}{\partial \bullet} \\
\frac{\partial^2 G_{v_y}^{pml}}{\partial \circ \partial \bullet} =& \frac{\partial^2 G_p^{pml}}{\partial \circ \partial \bullet} I_y + \frac{\partial G_p^{pml}}{\partial \bullet} \frac{\partial I_y}{\partial \circ} + \frac{\partial G_p^{pml}}{\partial \circ} \frac{\partial I_y}{\partial \bullet} + G_p^{pml} \frac{\partial^2 I_y}{\partial \circ \partial \bullet} + \frac{\partial^2 K}{\partial \circ \partial \bullet} J_x + \frac{\partial K}{\partial \bullet} \frac{\partial J_x}{\partial \circ} + \frac{\partial K}{\partial \circ} \frac{\partial J_x}{\partial \bullet} \\
& + K \frac{\partial^2 J_x}{\partial \circ \partial \bullet}
\end{aligned}
$$

Bibliography

[1] M. Bonnasse-Gahot, *Simulation de la propagation d'ondes élastiques en domaine fréquentiel par des méthodes Galerkine discontinues.* PhD thesis, Nice, 2015.

[2] N. D. Whitmore, "Iterative depth migration by backward time propagation," in *SEG Technical Program Expanded Abstracts 1983*, pp. 382–385, Society of Exploration Geophysicists, 1983.

[3] E. Baysal, D. D. Kosloff, and J. W. Sherwood, "Reverse time migration," *Geophysics*, vol. 48, no. 11, pp. 1514–1524, 1983.

[4] D. D. Kosloff and E. Baysal, "Forward modeling by a Fourier method," *Geophysics*, vol. 47, no. 10, pp. 1402–1412, 1982.

[5] R. H. Stolt, "Migration by Fourier transform," *Geophysics*, vol. 43, no. 1, pp. 23–48, 1978.

[6] J. Virieux and S. Operto, "An overview of full-waveform inversion in exploration geophysics," *Geophysics*, vol. 74, no. 6, pp. WCC1–WCC26, 2009.

[7] J. Brittan, J. Bai, H. Delome, C. Wang, and D. Yingst, "Full waveform inversion–the state of the art," *first break*, vol. 31, no. 10, 2013.

[8] R. Brossier, S. Operto, and J. Virieux, "Seismic imaging of complex onshore structures by 2D elastic frequency-domain full-waveform inversion," *Geophysics*, vol. 74, no. 6, pp. WCC105–WCC118, 2009.

[9] R. G. Pratt, C. Shin, and G. Hick, "Gauss–newton and full Newton methods in frequency–space seismic waveform inversion," *Geophysical journal international*, vol. 133, no. 2, pp. 341–362, 1998.

[10] M. Bonnasse-Gahot, H. Calandra, J. Diaz, and S. Lanteri, "Hybridizable discontinuous Galerkin method for the 2-D frequency-domain elastic wave equations," *Geophysical Journal International*, vol. 213, no. 1, pp. 637–659, 2018.

[11] B. Cockburn and C.-W. Shu, "TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. II. General framework," *Mathematics of computation*, vol. 52, no. 186, pp. 411–435, 1989.

[12] J. Argyris and D. Scharpf, "Finite elements in time and space," *The Aeronautical Journal*, vol. 73, no. 708, pp. 1041–1044, 1969.

[13] J. T. Oden, "A general theory of finite elements. ii. applications," *International Journal for Numerical Methods in Engineering*, vol. 1, no. 3, pp. 247–259, 1969.

[14] G. M. Hulbert and T. J. Hughes, "Space-time finite element methods for second-order hyperbolic equations," *Computer methods in applied mechanics and engineering*, vol. 84, no. 3, pp. 327–348, 1990.

[15] D. A. French, "A space-time finite element method for the wave equation," *Computer Methods in Applied Mechanics and Engineering*, vol. 107, pp. 145–157, Aug. 1993.

[16] L. Demkowicz, J. Gopalakrishnan, S. Nagaraj, and P. Sepúlveda, "A spacetime DPG method for the schrödinger equation," *SIAM Journal on Numerical Analysis*, vol. 55, no. 4, pp. 1740–1759, 2017.

[17] J. Gopalakrishnan and P. Sepúlveda, "A space-time DPG method for the wave equation in multiple dimensions," *Space-Time Methods*, vol. 25, pp. 117–140, 2019.

[18] O. Steinbach, "Space-time finite element methods for parabolic problems," *Computational methods in applied mathematics*, vol. 15, no. 4, pp. 551–566, 2015.

[19] C. Baldassari, H. Barucq, H. Calandra, B. Denel, and J. Diaz, "Performance analysis of a high-order discontinuous Galerkin method application to the reverse time migration," *Communications in Computational Physics*, vol. 11, no. 2, pp. 660–673, 2012.

[20] F. Ventimiglia, *Schémas numérique d'ordre élevé en temps et en espace pour l'équation des ondes du premier ordre. Application à la Reverse Time Migration.* PhD thesis, Université de Pau et des Pays de l'Adour, 2014.

[21] J. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, vol. 54. 01 2007.

[22] B. Cockburn, J. Gopalakrishnan, and R. Lazarov, "Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems," *SIAM Journal on Numerical Analysis*, vol. 47, no. 2, pp. 1319–1365, 2009.

[23] E. Trefftz, "Ein Gegenstück zum Ritzchen Verfahren," in *Proc. 2nd Int. Cong. Appl. Mech. Zurich*, pp. 131–137, 1926.

[24] E. Shishenina, *Space-Time Discretization of Elasto-Acoustic Wave Equation in Polynomial Trefftz-DG Bases.* PhD thesis, 12 2018.

[25] S. Petersen, C. Farhat, and R. Tezaur, "A space–time discontinuous Galerkin method for the solution of the wave equation in the time domain," *International journal for numerical methods in engineering*, vol. 78, no. 3, pp. 275–295, 2009.

[26] A. Moiola and I. Perugia, "A space-time trefftz discontinuous galerkin method for the first-order acoustic wave equations," *Numerische Mathematik*, vol. 138, 02 2018.

[27] P. Stocker, *Space-time finite element methods*. PhD thesis, 2021.

[28] S. Gómez and A. Moiola, "A space-time Trefftz discontinuous Galerkin method for the linear Schrödinger equation," Nov. 2021. Number: arXiv:2106.04724 arXiv:2106.04724 [cs, math].

[29] L. Banjai, E. H. Georgoulis, and O. Lijoka, "A Trefftz polynomial space-time Discontinuous Galerkin method for the second order wave equation," *SIAM Journal on Numerical Analysis*, vol. 55, no. 1, pp. 63–86, 2017.

[30] H. Egger, F. Kretzschmar, S. M. Schnepp, and T. Weiland, "A space-time Discontinuous Galerkin Trefftz method for time dependent Maxwell's equations," *SIAM Journal on Scientific Computing*, vol. 37, no. 5, pp. B689–B711, 2015.

[31] G. Gabard, "Discontinuous Galerkin methods with plane waves for time-harmonic problems," *Journal of Computational Physics*, vol. 225, no. 2, pp. 1961–1984, 2007.

[32] J. Jirousek and Q. Qin, "Application of hybrid-Trefftz element approach to transient heat conduction analysis," *Computers & Structures*, vol. 58, no. 1, pp. 195–201, 1996.

[33] O. Cessenat, "Application d'une nouvelle formulation variationnelle aux équations d'ondes harmoniques : problèmes de helmholtz 2d et de maxwell 3d," 1996.

[34] O. Cessenat and B. Després, "Application of an ultra weak variational formulation of elliptic pdes to the two-dimensional helmholtz problem," *SIAM journal on numerical analysis*, vol. 35, no. 1, pp. 255–299, 1998.

[35] A. Üngör and A. Sheffer, "Tent-Pitcher: A meshing algorithm for space-time Discontinuous Galerkin methods," in *Proc. 9th int'l meshing roundtable*, pp. 111–122, 2000.

[36] J. Erickson, D. Guoy, J. M. Sullivan, and A. Üngör, "Building Space-Time Meshes over Arbitrary Spatial Domains," Tech. Rep. arXiv:cs/0206002, arXiv, June 2002. arXiv:cs/0206002 type: article.

[37] G. R. Richter, "An explicit finite element method for the wave equation," *Applied Numerical Mathematics*, vol. 16, no. 1-2, pp. 65–80, 1994.

[38] R. Lowrie, P. Roe, and B. van Leer, "Space-time methods for hyperbolic conservation laws," 01 1998.

[39] L. Yin, A. Acharya, N. Sobh, R. B. Haber, and D. A. Tortorelli, "A space-time discontinuous Galerkin method for elastodynamic analysis," in *Discontinuous Galerkin Methods*, pp. 459–464, Springer, 2000.

[40] P. Monk and G. R. Richter, "A discontinuous Galerkin method for linear symmetric hyperbolic systems in inhomogeneous media," *Journal of Scientific Computing*, vol. 22, no. 1, pp. 443–477, 2005.

[41] S. T. Miller and R. B. Haber, "A spacetime discontinuous Galerkin method for hyperbolic heat conduction," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, no. 2, pp. 194–209, 2008.

[42] J. Gopalakrishnan, P. Monk, and P. Sepúlveda, "A tent pitching scheme motivated by Friedrichs theory," *Computers & Mathematics with Applications*, vol. 70, no. 5, pp. 1114–1135, 2015.

[43] C. Howard, A. Mudhakar, J. Erickson, R. Haber, and R. Abedi, "Tent pitching meshes for asynchronous spacetime Discontinuous Galerkin solvers in 3d× time,"

[44] J. Gopalakrishnan, J. Schoeberl, and C. Wintersteiger, "Mapped tent pitching schemes for hyperbolic systems," *SIAM Journal on Scientific Computing*, vol. 39, 04 2016.

[45] J. Gopalakrishnan, J. Schöberl, and C. Wintersteiger, "An explicit mapped tent pitching scheme for hyperbolic systems," in *Proceedings of the 14th International Conference on Mathematical and Numerical Aspects of Wave Propagation*, pp. 272–273, 2019.

[46] R. Abedi, R. B. Haber, S. Thite, and J. Erickson, "An h-adaptive spacetime-discontinuous Galerkin method for linear elastodynamics," *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, vol. 15, no. 6, pp. 619–642, 2006.

[47] R. Abedi, S.-H. Chung, J. Erickson, Y. Fan, M. Garland, D. Guoy, R. Haber, J. M. Sullivan, S. Thite, and Y. Zhou, "Spacetime meshing with adaptive refinement and coarsening," in *Proceedings of the twentieth annual symposium on Computational geometry*, pp. 300–309, 2004.

[48] S. Thite, "Adaptive spacetime meshing for discontinuous Galerkin methods," *Computational Geometry*, vol. 42, no. 1, pp. 20–44, 2009.

[49] R. Abedi, M. A. Hawker, R. B. Haber, and K. Matous, "An adaptive spacetime discontinuous Galerkin method for cohesive models of elastodynamic fracture," *International journal for numerical methods in engineering*, 2010.

[50] O. Omidi, R. Abedi, and S. Enayatpour, "An adaptive meshing approach to capture hydraulic fracturing," in *49th US Rock Mechanics/Geomechanics Symposium*, OnePetro, 2015.

[51] A. D. Mont, "Adaptive unstructured spacetime meshing for four-dimensional spacetime discontinuous Galerkin finite element methods," 2012.

[52] R. Alford, K. Kelly, and D. M. Boore, "Accuracy of finite-difference modeling of the acoustic wave equation," *Geophysics*, vol. 39, no. 6, pp. 834–842, 1974.

[53] W. Ritz, "Über eine neue Methode zur L"osung gewisser Variationsprobleme der mathematischen Physik.," *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1909, pp. 1 – 61.

[54] B. G. Galerkin, "Series solution of some problems of elastic equilibrium of rods and plates," *Vestnik inzhenerov i tekhnikov*, vol. 19, no. 7, pp. 897–908, 1915.

[55] W. H. Reed and T. R. Hill, "Triangular mesh methods for the neutron transport equation," tech. rep., Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[56] B. Cockburn, "Discontinuous Galerkin methods," *ZAMM*, vol. 83, pp. 731–754, Nov. 2003.

[57] C.-W. Shu, "Total-variation-diminishing time discretizations," *SIAM Journal on Scientific and Statistical Computing*, vol. 9, no. 6, pp. 1073–1084, 1988.

[58] F. Bassi and S. Rebay, "A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations," *Journal of computational physics*, vol. 131, no. 2, pp. 267–279, 1997.

[59] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, "Unified analysis of discontinuous Galerkin methods for elliptic problems," *SIAM journal on numerical analysis*, vol. 39, no. 5, pp. 1749–1779, 2002.

[60] B. Rivière, *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[61] I. Herrera, "Trefftz method: a general theory," *Numerical Methods for Partial Differential Equations: An International Journal*, vol. 16, no. 6, pp. 561–580, 2000.

[62] J. Jirousek, "Basis for development of large finite elements locally satisfying all field equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 14, no. 1, pp. 65–92, 1978.

[63] W. Jin, Y. Cheung, and O. Zienkiewicz, "Application of the Trefftz method in plane elasticity problems," *International Journal for Numerical Methods in Engineering*, vol. 30, no. 6, pp. 1147–1161, 1990.

[64] E. Kita and N. Kamiya, "Trefftz method: an overview," *Advances in Engineering Software*, vol. 24, pp. 3–12, Jan. 1995.

[65] R. Hiptmair, A. Moiola, and I. Perugia, "A Survey of Trefftz Methods for the Helmholtz Equation," Sept. 2015. Number: arXiv:1506.04521 arXiv:1506.04521 [math].

[66] L.-M. Imbert-Gérard and B. Després, "A generalized plane-wave numerical method for smooth nonconstant coefficients," *IMA Journal of Numerical Analysis*, vol. 34, no. 3, pp. 1072–1103, 2014.

[67] H. Kreiss and J. Lorenz, *Initial-boundary Value Problems and the Navier-Stokes Equations*. No. vol. 136 in Initial-boundary value problems and the Navier-Stokes equations, Academic Press, 1989.

[68] A. Moiola and I. Perugia, "A space–time Trefftz discontinuous Galerkin method for the acoustic wave equation in first-order formulation," *Numerische Mathematik*, vol. 138, pp. 389–435, jul 2017.

[69] K. Grysa and B. Maciejewska, "Trefftz functions for non-stationary problems," *Journal of Theoretical and Applied Mechanics*, vol. 51, pp. 251–264, 2013.

[70] A. Maciąg, "Solution of the three-dimension wave equation by using wave polynomials," *PAMM*, vol. 4, pp. 706 – 707, 12 2004.

[71] C. Agut, *Schémas numériques d'ordre élevé en espace et en temps pour l'équation des ondes*. PhD thesis, Université de Pau et des Pays de l'Adour, 2011.

[72] J. Hadamard, "Le principe de Huygens," *Bulletin de la Société Mathématique de France*, vol. 52, pp. 610–640, 1924.

[73] A. Üngör, A. Sheffer, R. B. Haber, and S.-H. Teng, "Layer based solutions for constrained space–time meshing," *Applied numerical mathematics*, vol. 46, no. 3-4, pp. 425–443, 2003.

[74] D. French and T. Peterson, "A continuous space-time finite element method for the wave equation," *Mathematics of Computation*, vol. 65, no. 214, pp. 491–506, 1996.

[75] R. Abedi, S.-H. Chung, J. Erickson, Y. Fan, M. Garland, D. Guoy, R. Haber, J. Sullivan, S. Thite, and Y. Zhou, "Spacetime meshing with adaptive refinement and coarsening.," pp. 300–309, 01 2004.

[76] L. Zhang, T. Cui, and H. Liu, "A set of symmetric quadrature rules on triangles and tetrahedra," *Journal of Computational Mathematics*, pp. 89–96, 2009.

[77] H. Barucq, R. Djellouli, and E. Estecahandy, "Efficient dg-like formulation equipped with curved boundary edges for solving elasto-acoustic scattering problems," *International Journal for Numerical Methods in Engineering*, vol. 98, no. 10, pp. 747–780.

[78] J. Diaz, A. Ezziani, and N. Le Goff, "Logiciels Gar6more2D et Gar6more3D," 2010. AP.

[79] J.-P. Bérenger, "A perfectly matched layer for the absorption of electromagnetic waves," *Journal of computational physics*, vol. 114, no. 2, pp. 185–200, 1994.

[80] D. Appelö, T. Hagstrom, and G. Kreiss, "Perfectly matched layers for hyperbolic systems: general formulation, well-posedness, and stability," *SIAM Journal on Applied Mathematics*, vol. 67, no. 1, pp. 1–23, 2006.

[81] H. Barucq, J. Diaz, and M. Tlemcani, "New absorbing layers conditions for short water waves," *Journal of Computational Physics*, vol. 229, no. 1, pp. 58–72, 2010.

[82] F. Q. Hu, "A stable, perfectly matched layer for linearized Euler equations in unsplit physical variables," *Journal of Computational Physics*, vol. 173, no. 2, pp. 455–480, 2001.

[83] J. S. Hesthaven, "On the analysis and construction of perfectly matched layers for the linearized Euler equations," *Journal of computational Physics*, vol. 142, no. 1, pp. 129–147, 1998.

[84] D. Komatitsch and R. Martin, "An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation," *Geophysics*, vol. 72, no. 5, pp. SM155–SM167, 2007.

[85] F. Collino and C. Tsogka, "Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media," *Geophysics*, vol. 66, no. 1, pp. 294–307, 2001.

[86] B. Kaltenbacher, M. Kaltenbacher, and I. Sim, "A modified and stable version of a perfectly matched layer technique for the 3-d second order wave equation in time domain with an application to aeroacoustics," *Journal of computational physics*, vol. 235, pp. 407–422, 2013.

[87] K. Duru and G. Kreiss, "A well-posed and discretely stable perfectly matched layer for elastic wave equations in second order formulation," *Communications in Computational Physics*, vol. 11, no. 5, pp. 1643–1672, 2012.

[88] J. Diaz and P. Joly, "A time domain analysis of PML models in acoustics," *Computer methods in applied mechanics and engineering*, vol. 195, no. 29-32, pp. 3820–3853, 2006.

[89] E. Bécache, S. Fauqueux, and P. Joly, "Stability of perfectly matched layers, group velocities and anisotropic waves," *Journal of Computational Physics*, vol. 188, no. 2, pp. 399–433, 2003.

[90] S. Abarbanel, D. Gottlieb, and J. S. Hesthaven, "Long time behavior of the perfectly matched layer equations in computational electromagnetics," *Journal of scientific Computing*, vol. 17, no. 1, pp. 405–422, 2002.

[91] L. Halpern, S. Petit-Bergez, and J. Rauch, "The analysis of matched layers," *Confluentes Mathematici*, vol. 3, no. 02, pp. 159–236, 2011.

[92] W. C. Chew and W. H. Weedon, "A 3D perfectly matched medium from modified maxwell's equations with stretched coordinates," *Microwave and optical technology letters*, vol. 7, no. 13, pp. 599–604, 1994.

[93] F. Collino and P. B. Monk, "Optimizing the perfectly matched layer," *Computer methods in applied mechanics and engineering*, vol. 164, no. 1-2, pp. 157–171, 1998.

[94] "A mathematical analysis of the pml method," *Journal of Computational Physics*, vol. 134, no. 2, pp. 357–363, 1997.

[95] J. A. Roden and S. D. Gedney, "Convolution PML (CPML): An efficient FDTD implementation of the CFS–PML for arbitrary media," *Microwave and optical technology letters*, vol. 27, no. 5, pp. 334–339, 2000.

[96] J.-P. Bérenger, "Perfectly matched layer (PML) for computational electromagnetics," *Synthesis Lectures on Computational Electromagnetics*, vol. 2, no. 1, pp. 1–117, 2007.

[97] D. Komatitsch and J. Tromp, "A perfectly matched layer absorbing boundary condition for the second-order seismic wave equation," *Geophysical Journal International*, vol. 154, pp. 146–153, 07 2003.

[98] A. Bermúdez, L. Hervella-Nieto, A. Prieto, and R. Rodríguez, "An exact bounded perfectly matched layer for time-harmonic scattering problems," *SIAM Journal on Scientific Computing*, vol. 30, no. 1, pp. 312–338, 2008.

[99] A. Modave, E. Delhez, and C. Geuzaine, "Optimizing perfectly matched layers in discrete contexts," *International Journal for Numerical Methods in Engineering*, vol. 99, no. 6, pp. 410–437, 2014.

[100] J. Diaz, "Approches analytiques et numériques de problèmes de transmission en propagation d'ondes en régime transitoire. Application au couplage fluide-structure et aux méthodes de couches parfaitement adaptées," p. 423.

[101] A. De Hoop and M. Oristaglio, "Application of the modified Cagniard technique to transient electromagnetic diffusion problems," *Geophysical Journal International*, vol. 94, no. 3, pp. 387–397, 1988.

[102] M. L. Shendeleva, "Reflection and refraction of a transient temperature field at a plane interface using Cagniard–de Hoop approach," *Physical Review E*, vol. 64, no. 3, p. 036612, 2001.

[103] J. H. van der Hijden, *Propagation of transient elastic waves in stratified anisotropic media*. Elsevier, 2016.

[104] L. Cagniard, *Réflexion et réfraction des ondes séismiques progressives*. Gauthier-Villars Paris, 1939.

[105] A. De Hoop, "A modification of Cagniard's method for solving seismic pulse problems," *Applied Scientific Research, Section B*, vol. 8, no. 1, pp. 349–356, 1960.

[106] G. Green, "An essay on the application of mathematical analysis to the theories of electricity and magnetism," 1828.

[107] D. G. Duffy, *Green's functions with applications*. Chapman and Hall/CRC, 2015.

[108] P. R. Antunes, "Reducing the ill conditioning in the method of fundamental solutions," *Advances in Computational Mathematics*, vol. 44, no. 1, pp. 351–365, 2018.

[109] H. Barucq, A. Bendali, J. Diaz, and S. Tordeux, "Local strategies for improving the conditioning of the plane-wave ultra-weak variational formulation," *Journal of Computational Physics*, vol. 441, p. 110449, 2021.

Abstract

In this thesis, we explore the optimization of Trefftz-Discontinuous Galerkin methods with the Tent-Pitcher algorithm and the elaboration of boundary conditions adapted to them. We consider these methods in the framework of the first-order acoustic wave equation in time-domain. The idea of Trefftz methods is to use local solutions of the considered Partial Differential Equations as basis functions. Such a formulation has the advantage of being posed on the skeleton of the mesh only and characteristics of the analytical solutions are injected into the discrete solution through the basis functions, which leads us to expect the obtained numerical solution to be more precise. However, the resulting scheme is implicit in time-domain, which leads to an increased computational cost. This is the reason why we investigate Tent-Pitcher algorithms. The Tent-Pitcher algorithm is a space-time meshing algorithm introduced for hyperbolic problems, that consists in constructing a causal mesh which allows to solve the problem element-by-element. This work is divided into two main parts. In the first part, we present an already existing Trefftz-Discontinuous Galerkin solver on structured Tent-Pitching meshes for the acoustic wave equation and extend it to unstructured triangular meshes. The efficiency of the method is validated by comparing the obtained numerical solutions to analytical solutions. Having in mind the optimization of the method in terms of computational time, we implement the different methods in a High Performance Computing environment. We test their performance and present comparisons with the solutions from another numerical method. In the second part, we present an absorbing type of boundary conditions which is the Perfectly Matched Layer. We derive a formulation for the Trefftz-Discontinuous Galerkin methods with Perfectly Matched Layers for the acoustic wave equation, which involves the computation of analytical solutions for this new system of equations because the principle of the Trefftz methods relies on the use of local solutions as basis functions. This is done by computing the Green's functions for the time-domain acoustic wave equation using the Cagniard-De Hoop method. Finally, we implement these boundary conditions into the Trefftz-Discontinuous Galerkin solver with Tent-Pitching on structured meshes and present the obtained results.

Keywords: acoustic waves, spacetime, Tent-Pitcher, Trefftz, perfectly matched layers (PML), MPI

Résumé

Dans cette thèse, nous étudions l'optimisation des méthodes Trefftz-Galerkine Discontinues avec l'algorithme Tent-Pitcher, ainsi que l'élaboration de conditions de bords adaptées. Nous considérons ces méthodes dans le cadre d'une équation des ondes acoustiques de premier-ordre en domaine temporel. L'idée des méthodes de Trefftz est d'utiliser des solutions locales de l'Équation aux Dérivées Partielles considérée comme fonctions de base. Une telle formulation présente l'avantage de n'être posée que sur le squelette du maillage. De plus, certaines caractéristiques des solutions analytiques sont directement injectées dans la solution discrète au travers des fonctions de base, ce qui nous incite à penser que la solution numérique obtenue sera plus précise. Cependant, le schéma qui résulte de cette méthode est implicite en espace-temps, ce qui entraîne un coût de calcul élevé. C'est pourquoi nous nous intéressons à l'algorithme Tent-Pitcher, qui est un algorithme de construction de maillages espace-temps introduit pour les problèmes hyperboliques et consiste en la construction d'un maillage causal permettant de résoudre le problème élément par élément. Le présent travail se découpe en deux parties. Dans la première partie, nous présentons un solveur Trefftz-DG avec Tent-Pitching déjà existant appliqué à l'équation des ondes acoustiques en espace-temps et son extension à des maillages triangulaires non-structurés. L'efficacité de la méthode est validée par comparaison des solutions numériques avec des solution analytiques. Dans l'optique d'optimiser la méthode en termes de temps de calcul, nous implémentons les différentes méthodes dans un environnement de Calcul Haute Performance. La performance des codes est testée et les solutions numériques sont comparées avec des solutions obtenues par une méthode différente. Dans la seconde partie, nous présentons des conditions aux bords de type absorbant que sont les Couches Absorbantes Parfaitement Adaptées pour l'équation des ondes acoustiques, ce qui implique la construction de solutions analytiques pour ce nouveau système d'équations, car le principe des méthodes de Trefftz repose sur l'utilisation de solutions locales comme fonctions de base. Pour ce faire, nous élaborons des fonctions de Green pour les équations d'ondes acoustiques en domaine temporel grâce à la méthode Cagniard-De Hoop. Enfin, nous implémentons ces conditions de bords dans le solveur Trefftz-Galerkine Discontinu avec Tent-Pitching sur des maillages structurés et présentons les résultats obtenus.

Mots-clés : ondes acoustiques, espace-temps, Tent-Pitcher, Trefftz, couches absorbantes parfaitement adaptées, MPI