Time-Domain Full Waveform Inversion using advanced Discontinuous Galerkin method.

Pierre Jacquet






École doctorale des Sciences Exactes et leurs Applications (ED 211)

THÈSE DE DOCTORAT

Spécialité : Mathématiques appliquées

pour l'obtention du grade de
DOCTEUR de l'UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR

INRIA Bordeaux Sud-Ouest,
Équipe-Projet
MAKUTU

Université de Pau et des pays de l'Adour,
Laboratoire de Mathématiques et de leurs
Applications – UMR CNRS 5142

présentée par

Pierre Jacquet

Inversion par forme d'ondes complète en domaine temporel utilisant des méthodes de Galerkine Discontinues avancées

---

Time-Domain Full Waveform Inversion using advanced Discontinuous Galerkin method

---

soutenue le 25 février 2021 devant le Jury composé de:

Hélène Barucq	Directrice de recherche, Inria, Pau	Directrice
Henri Calandra	Ingénieur de recherche, Total, Pau	Examinateur
Gilles Carbou	Professeur, Université de Pau et des Pays de l'Adour	Examinateur
Hervé Chauris	Professeur, MINES ParisTech	Rapporteur
Julien Diaz	Directeur de recherche, Inria, Pau	Co-directeur
Michel Kern	Chargé de recherche, Inria, Paris	Examinateur
Jeanne Pellerin	Ingénieure de recherche, Total, Paris	Examinatrice
Sébastien Pernet	Maître de recherche, Onera, Toulouse	Rapporteur
Florian Faucher	Docteur, Université de Vienne	Membre invité

"Rien dans la vie n'est à craindre, tout doit être compris. C'est maintenant le moment de comprendre davantage, afin de craindre moins."

Marie Curie

Remerciements

Ce mémoire est le résultat d'un travail de prêt de trois ans de recherche. En préambule, je souhaite adresser mes remerciements à toutes les personnes qui ont pu contribuer à la rédaction de ce dernier.

Avant toutes choses, je souhaite adresser ma plus grande gratitude à mes directeurs de thèse Hélène Barucq et Julien Diaz, qui m'ont intégré au sein de l'équipe MAKUTU et qui ont su m'octroyer un cadre de travail prolifique, chaleureux et une écoute permanente. Je vous remercie pour votre encadrement et m'estime chanceux d'avoir pu exercer cette thèse sous votre supervision. Héléne, ton expertise scientifique et ton support moral ont été cruciaux notamment au cours de cette dernière année, riche en péripéties. Merci infiniment pour tes conseils et ta disponibilité.

Je souhaite aussi remercier Henri Calandra, Gilles Carbou, Hervé Chauris, Michel Kern, Jeanne Pellerin, Sébastien Pernet ainsi que Florian Faucher d'avoir participé à mon jury de thèse. Je remercie notamment Hervé Chauris et Sébastien Pernet d'avoir accepté de rapporter mon manuscrit, vos remarques ont beaucoup contribué à l'amélioration de ce dernier. Merci Henri pour ta supervision au cours de cette thèse ainsi que pour ton expertise industrielle apportée lors de ces trois années de développement. Un merci particulier à Florian d'avoir accepté d'être membre invité. Il me semblait inenvisageable que tu ne participes pas à la soutenance au vu de ton expertise et des conseils que tu m'as prodigués tout au long de ma thèse, merci encore.

I would like to add a special thought for Andreas Atle who also advised me during this thesis and who warmly welcomed me twice in Houston. I thank you for your expertise and the time you took for me during long weekly discussions. I wish you a good continuation and the best for the future.

Ces remerciements ne sauraient oublier les magiciens qui ont su participer à l'ambiance chaleureuse et enrichissante dans laquelle j'ai été baigné au sein de l'équipe MAKUTU. Je remercie notamment Chengyi, Elvira, Ha, Izar, Julien B., Justine, Marc, Nathan, Sébastien, Stéfano et Yder. Merci à Aurélien pour le support ingénieur apporté notamment lors de cette dernière année. Merci à Algiane pour ton écoute et pour nos fructueuses discussions sur l'adaptation de maillage ! Un merci particulier à Mamadou, Rose et Vinduja dont l'amitié a su adoucir la crise que nous avons traversé. Courage à toutes deux pour la bonne réalisation de vos travaux de thèse !

Je remercie ma famille pour son soutien et son écoute. Merci à Aurélien, Laura, Karen et Papa d'avoir traversé la France (voire plus) pour être présent le jour de la soutenance. Je conserve des souvenirs indélébiles de ce séjour au Pays Basque passé avec vous. Une pensée particulière à ma Maman qui n'était pas en mesure de faire ce déplacement. Sache que je pense fort à toi.

Je souhaite aussi remercier une autre famille qui me tient à cœur et dont le soutien et l'accueil ont su galvaniser ma motivation et mon moral pendant ces périodes confinées. Merci infiniment à Eve, Frédéric, Guillaume et Maxime d'avoir été là pour moi. Je n'oublie pas non plus Nobel, un camarade de rédaction à quatre pattes dont les promenades m'ont permis de m'oxygéner. Avec un nom comme le sien, je me devais de souligner sa contribution scientifique.

Finalement, le plus grand des mercis revient à Camille, dont l'amour aura été le plus grand des soutiens. Merci pour tes encouragements et pour tous ces moments privilégiés passés ensemble. Je te souhaite plein de courage pour cette dernière année de thèse qui ne saurait être qu'une réussite.

Table of contents

List of Figures

List of Tables

Introduction

When they propagate in the subsurface, seismic waves are very sensitive to the characteristics of the medium through which they pass. Depending on their wavelength, they will react to changes in the environment, which correspond to variations in physical parameters. This property is exploited in geophysical imaging, which aim characterizing natural underground reservoirs likely to contain water, gas, oils, etc. One of the main steps of this characterization is to obtain the location of the reservoir and to draw a map of the subsurface. To achieve this, a set of sensors is deployed to measure the reflected waves generated by seismic sources placed on the surface of the probed area. The seismic data forms an acquisition containing the reflected and diffracted waves and this acquisition is then processed to find the position of the reflectors. This involves extrapolating the recorded data to the reflectors. The recorded data are back-propagated, i.e. they are extrapolated in the opposite direction to the direction of propagation of the source. This technique is called seismic migration and encompasses different approaches with varying degrees of efficiency. Among them, Reverse Time Migration (RTM, [Bednar, 2005]) is known to be the most accurate. It is based on the resolution of the complete wave equation in the time domain and the image of the subsurface is formed by applying an imaging condition that expresses the correlation between the propagated field and the back-propagated field. The reconstruction of the subsurface is very dependent on the imaging condition and this is not always easy to define. RTM is the most common technique used in industrial codes. It is a qualitative method as it only gives reflector locations. It thus indicates the presence of natural reservoirs but does not give any information on their contents. We refer to Zhou et al. [2018] for a recent review on RTM. As far as RTM numerical implementation is concerned, it provides a numerical tool which turns out to be computationally intensive. Indeed, it requires solving as many forward and backward problems as sources and its efficiency requires using advanced numerical methods.

Full Waveform Inversion (FWI) meets the same interest as RTM in having a non-invasive probing tool applied in complex environments penetrated by waves whose reflections can be measured. The strength of this technology lies in the deployment of advanced numerical methods, the efficiency of which is enhanced by modern computing architectures that allow large-scale calculations. In this work, we consider FWI as a numerical technique for reconstructing a medium through which mechanical waves pass, by using as input data the recordings recovered from a set of receivers whose location is known a priori. The reconstruction is achieved by retrieving constitutive parameters such as velocity, density, attenuation, etc. It was introduced in the 1980s by Lailly [1983] and Tarantola [1984], which were preceded by preliminary studies of Bamberger et al. [1977, 1982]. While RTM is a qualitative imaging tool, FWI is a quantitative method since it allows to characterize the physical parameters defining the subsurface. To implement FWI, it is essential to have an efficient tool for calculating synthetic seismograms. Since the success of FWI depends on the quality of the numerical wavefields, geophysics is contributing very significantly to the development and performance analysis of numerical methods for solving wave equations. The first numerical

code developments have been made with finite differences (see Moczo et al. [2014]) for a review of this specific topic) set on regular grids that have been extended to staggered grids (see Virieux [1984, 1986]), arranged to include boundary conditions and material discontinuities [Robertsson, 1996, Ohminato and Chouet, 1997] and anisotropy [Igel et al., 1995]. Finite differences are very efficient for calculations in media that can be meshed with rectilinear grids. Some works have proposed extensions to curved grids using curvilinear coordinate transforms [Fornberg, 1988, Hestholm, 1999, de la Puente et al., 2014, Petersson and Sjögreen, 2015, Shragge, 2016], but each time, the developed technique is computationally expensive and, moreover, is only valid in smooth media. Moreover, taking into account boundary conditions is very difficult. Finally, it is important to note that the stencil of the matrices increases considerably with the order of approximation, which strongly affects the parallel scaling characteristics. To avoid possible weak points of finite differences, finite element and finite volume methods are very good candidates. As regards their use in seismic applications, a consensus was quickly reached in the numerical geophysical community that these methods are not effective if used for low orders of approximation (see for example [Marfurt, 1984]). High-order finite volume methods are indeed very efficient [Dumbser et al., 2007] but they turn out to be computationally extensive. This observation has motivated several studies [de la Puente et al., 2007, Käser et al., 2007, Wilcox et al., 2010, Tago et al., 2012] promoting DG (Discontinuous Galerkin) methods, which are hybrids between standard (continuous) finite element methods and finite volumes. These methods are easily implemented on simplicial unstructured meshes (triangles or tetrahedrons), which is a real asset for solving wave problems in complex geological environments. They are formulated with a block diagonal mass matrix, which makes them more interesting than standard finite element methods whose mass matrix cannot be easily lumped without deteriorating their order of convergence. Moreover, by construction, an element communicates only with its neighbors, which ensures that the stencil of the matrices does not vary with the order of approximation and facilitates the parallelization of the method which comes naturally. But it is important to note that DG methods are more expensive than standard finite element methods because they use a larger number of degrees of freedom and require additional work to manage the fluxes that ensure the communication of neighboring elements. In particular, it has been shown in [Ferroni et al., 2017] that numerical flux management contributes to decrease the Friedrichs Levy Current Constant (CFL) when using explicit time schemes. When the propagation domain can be meshed in hexahedrons, the spectral elements have clearly demonstrated their efficiency and the literature illustrating the interest of their use is very important (without being exhaustive, see for example [Seriani and Priolo, 1994, Faccioli et al., 1996, 1997, Komatitsch and Tromp, 1999, Capdeville et al., 2003, Chaljub et al., 2003, Fichtner and Igel, 2008, Peter et al., 2011, Nissen-Meyer et al., 2014]). The Spectral Element Method (SEM) uses high-order polynomial approximations with degrees of freedom coinciding with Gauss-Lobatto-Legendre points. The mass matrix is then diagonal, which makes the method very efficient in the time domain and well adapted for using SIMD (Single Instruction, Multiple Data) computing architectures. In a recent paper [Afanasiev et al., 2019] focusing on the acoustic wave equation formulated as a second-order equation in time, one can find a very broad description of useful contributions to develop a SEM-based FWI code. These contributions are inspired by several works that gave rise to different packages ([Cupillard et al., 2012] in Specfem, [Logg et al., 2012] in FENicS, [Bangerth et al., 2007] in Deal.II, [Bauman and Stogner, 2016] in GRINS, [Dedner et al., 2010] in DUNE). Based on the resolution of an optimization problem, FWI is a high-definition technique for the reconstruction of the physical parameters of a medium traversed by waves. One of the major obstacles to FWI is the cost of implementation, which, not to mention the preprocessing stages such as the acquisition and processing of the observed data,

is very high because it is based on an iterative method that can take time to converge and that uses a priori as many direct problem-solving as sources. Frequency FWI is the most widely used approach since it leads to the forward problem with multiple right-hand sides [Pocock and Walker, 1998, Operto et al., 2014, Faucher, 2017]. Moreover, the simulations can be done only for a few number of frequencies. In the time domain, FWI requires more computational resources, especially because all wave fields must be stored. In [Singh et al., 2018], a comparison of the two approaches has been carried out using the Marmousi model as a target. It shows that time-domain FWI is more efficient when it comes to solving the direct problem in large propagation domains; the frequency method may indeed be limited by the ability of solvers to solve large-scale problems. The time solver actually requires less memory than the frequency one and in [Sirgue et al., 2010, Brossier et al., 2014], attempts have been done to solve the forward problem in the time-domain and to optimize the cost function in the frequency domain. Unfortunately, this approach is very expensive since it requires using discrete Fourier transforms.

Regarding the industrial problems under consideration, using full time-domain FWI is recommended since the size of the problems often reaches the limit of the solver. FWI, which is written as a strongly non-linear problem, may also have difficulties to converge in the sense that the minimization algorithm tends to converge to a local minimum. This problem may be caused by a wrong initial model, too noisy data, lack of low frequency data and also errors in the approximation of the direct wave problem which is in itself complicated to achieve. Usually to help the algorithm run iteratively, we start with low-frequency calculations, then gradually increase the frequency to refine the reconstruction. The surveyed medium is characterized through the minimization of a cost function that measures the difference between the observed data and the synthetic data. FWI is a strongly non-linear inverse problem that is difficult to solve with a global optimization method due to high computational loads, especially for geophysical applications. In practice, an iterative algorithm is used and it evolves via a local optimization method. This approach depends very strongly on the initial environment and if this is too different from the environment to be reconstructed, one risks remaining stuck on a local minimum. To obtain a more reliable initial environment, different strategies can be applied (see the work of Biondi and Almomin [2012] for tomography traveltime and of Almomin and Biondi [2012] for migration velocity analysis). One can also apply a multiscale method following the ideas of Bunks et al. [1995]. This approach is widely used in the frequency domain (see Faucher [2017] and the bibliography therein). It enables to reconstruct the medium from low to high frequency and thus avoid local minima. However, low-frequency data are generally missing and when they are available, they are most often very noisy. In the time domain, envelope inversion [Wu et al., 2014, Luo and Wu, 2013] can also be applied to extract low-frequency content in order to image structures at the long wavelength scale. Another idea is to use attenuated data and this approach seems to be very effective in limiting the skipping cycle effect [Chen et al., 2015]. Some attemps have also been done by enlarging the search space [Van Leeuwen and Herrmann, 2013]. Over the last few decades, FWI has been the result of numerous methodological developments that have been aided by access to quality data in sufficient numbers to hopefully yield information that can be used by the application world. For example, FWI has been applied in global seismology [Lekić and Romanowicz, 2011, French and Romanowicz, 2014], and local seismology [Fichtner et al., 2009, Zhu et al., 2015, Simutė et al., 2016], geophysical exploration [Igel et al., 1995, Virieux and Operto, Sirgue et al., 2011, Warner et al., 2013]. This is a high-resolution numerical technique for estimating propagation medium parameters with the following main features:

- FWI is the result of minimizing a cost function;

- the gradient of the function is prevented by the adjoint state method;
- minimizing is done iteratively;
- the vast size of the geophysical environment requires using numerous sources.

If we look at the FWI only at the level of the optimization method used to estimate the medium parameters, attention naturally turns to the minimized cost function and with it to the way its gradient is determined, i.e., how the adjoint state method is implemented. The idea of using different costs functions to mitigate the local minima issue has been investigated quite recently [Bozdağ et al., 2011, Chi et al., 2014].

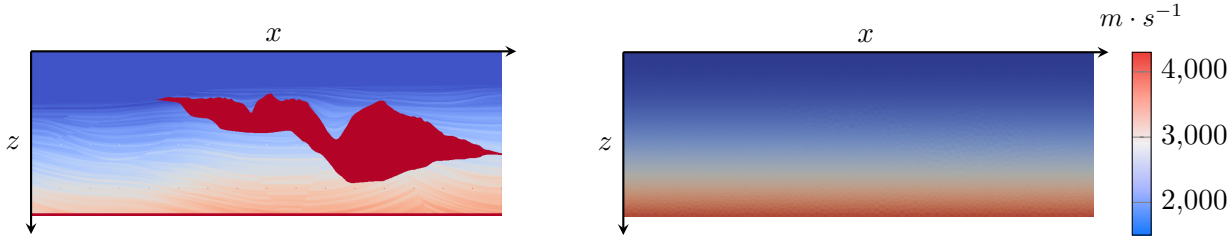
The objective of this thesis is to develop an inversion code based on FWI for the acoustic wave equation in time-domain. The numerical method chosen is a discontinuous Galerkin method (DG). It is important to note that the code was developed in a computing platform maintained and developed by the company Total. This implies certain constraints that will be exposed, when necessary, in the document. We will follow the next plan. In the first chapter, we will introduce the inverse problem and describe the optimization method used. The second chapter will be dedicated to the direct problem. We will introduce the discretization method in space and the time schemes used. We will discuss the Bernstein-Bézier polynomial base application, which could be an interesting option to reduce the computational costs when using high order polynomials. Indeed, it has been shown by Chan and Warburton [2016] that in the case of a GPU implementation, this basis allows speeding up computations. In the context of this thesis, we will use CPUs, which is why we will carry out a study on this polynomial basis in order to know if the conclusions of Chan and Warburton [2016] are still valid. Then, we will present a variant of the DG method, the so-called Weight Adjusted Discontinuous Galerkin (WADG) method [Chan et al., 2017], which allows taking into account the variations of physical parameters within elements. This approach seemed to us to be inseparable from the idea of using high-order elements in meshes whose cells can be very large. The third chapter is devoted to the characterization of the adjoint state. This can be determined in two different ways: either one constructs the adjoint problem of the continuous direct problem and one calculates the adjoint state as the numerical solution of the continuous adjoint problem; or one constructs a discrete adjoint problem defined as the adjoint of the system resulting from the space-time discretization of the direct problem. It will be emphasized that the second method applied to a discrete problem built on DG approximations requires additional tedious developments that are not always easy to implement in a large industrial code and in a massively parallel context. Chapter 4 will describe the development environment in which the code was developed. This chapter will also be the opportunity to present first inversion results in 2D and 3D. However, these results have been obtained using a single mesh during the whole inversion, which can be expensive. In Chapter 5, we propose to use tools dedicated to mesh adaptation in order to generate a mesh adapted to the model parameters but also to the reconstruction frequency. In this chapter we will define the mesh adaptation process and finally determine criteria for optimizing the meshes for wave propagation simulations using the DG method. This chapter will be the opportunity to benefit from all the specificities of DG method (hp-adaptivity, choice of the polynomial basis, WADG) by proposing a workflow, compatible with the industrial code, which will be based upon an adapted and evolutive discretization during the FWI course.

Chapter 1

The Inverse Problem

1.1 General setting

To reconstruct the propagation environment, one of the most popular techniques is the Full Waveform Inversion (FWI), which is a numerical procedure that is based upon an iterative update of the constituent parameters until the synthetic data are very close to the observed data (ideally equal). In this thesis, we focus on the reconstruction of a wavespeed model, i.e., the propagation medium (see Figure 1.1a for an example of a wavespeed model). The initialization of the FWI requires on the one hand collecting data (the observables) and on the other hand calculating numerical data by solving a direct problem posed in a blindly constructed medium from the characteristic magnitudes collected during the acquisition campaign (see Figure 1.1b).



(a) Target Sigsbee wavespeed model ($m.s^{-1}$) (b) Blind wavespeed model ($m.s^{-1}$)

Figure 1.1: Comparison between blind model and target wavespeed model.

FWI relies on the solution of an inverse problem that can be defined from the three following fundamental spaces:

- the space of parameters, denoted by M;
- the space of data or observations, denoted by D;
- the space of wavefields, denoted by U.

Then, the parameters are explicitly related to the wavefield through the state equation F (here, it will be the acoustic wave equation) as follows:

$$F(m,u) = s, \qquad m \in M, u \in U\,, \tag{1.1}$$

where s denotes the source field. One can use the relationship

$$u = P_s(m) = u_{m,s}\,, \tag{1.2}$$

by using the argument that the direct problem (operator P) is well-posed and therefore, even if we restrict the space of the parameters, we can always associate to a parameter (or wavespeed model) and a source, a single wavefield u_m. It is also interesting to write the problem in terms of observations. This is done with an observation equation that aims at getting information from the wavefield in terms of measurements. It reads:

$$d = Q(u), \qquad u \in U\,. \tag{1.3}$$

It is worth noting that, if we inject (1.2) inside (1.3), we obtain the relation

$$d = Q(u) = Q(P_s(m))\,. \tag{1.4}$$

We then end up with the formulation of the inverse problem: let d_{obs} be the observation data; find the parameters m such that

$$d_{obs} = \Phi(m)\,, \tag{1.5}$$

where Φ is the operator defined in (1.5). Here, we have dropped the s index to simplify the notations. It is important to note that although the state equation, and possibly the observation equation, is linear, the operator Φ is non-linear. This suggests that the inverse problem will be more difficult to solve the problem straightforwardly than the direct problem. But the difficulties do not stop at the non-linearity of the inverse problem. Indeed, it is also ill-posed, and to explain it, let us begin by recalling what is a well-posed problem in the sense of Hadamard [1923]. A well-posed problem is a problem for which we have:

- existence of a solution;
- uniqueness of the solution;
- the solution depends continuously on the parameters.

The inverse problem we are interested in is not well-posed because it is based on noisy experimental data that therefore differ from the real data and are not for sure the data that correspond to the model that we are seeking. Moreover, the uniqueness of the solution cannot be guaranteed, even if the data are not noisy. It is therefore difficult to solve and in practice, it is replaced by a minimization problem, most often written in the sense of least squares. Equation (1.5) is therefore replaced by minimizing:

$$\mathcal{J}(m) = \frac{1}{2} \parallel \Phi(m) - d_{obs} \parallel^2, \qquad m \in M\,. \tag{1.6}$$

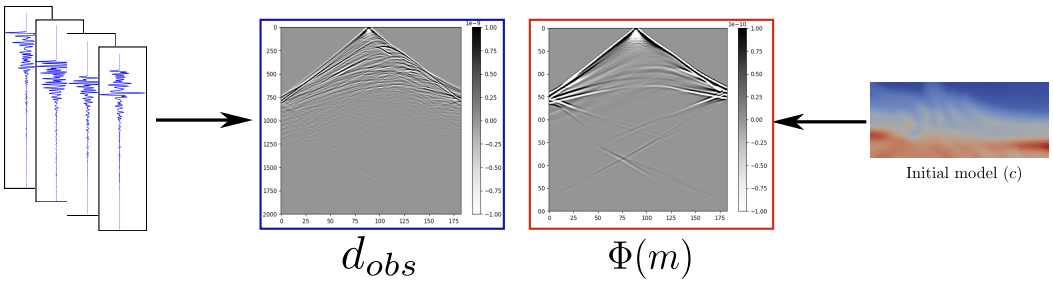


Figure 1.2: Comparisons of seismograms from observations and simulations to illustrate the role of the cost function $\mathcal{J}$.

The function $\mathcal{J}$ is the cost function (or misfit function or error function) that is used to measure the difference between numerical and observed parameters (see Figure 1.2). Obviously, this formulation does not prevent from the fact that the inverse problem is ill-posed but it allows having the existence of a solution because the cost function is positive. However, the cost function is in general non-convex and thus admits local minima. The underlying iterative process for minimizing is thus likely to converge on a local minimum which is not the right solution. Another problem is related to a possible lack of data (typically low-frequency data) which explains that different parameters may produce the same observations. Noisy data may also be difficult if not impossible to invert. Last but not least is the very high computational burden: the cost function computation requires solving the state equation. This point should be of particular interest to us in the application context we are interested in. Indeed, the geophysical environments we are trying to reconstruct are large, which means that we have to use numerous sources to collect information on the entire surface of the surveyed domain (see Figure 1.3). In practice, we have one source every several hundred meters and thus, if the surface on which they are deployed is of the order of five square kilometers and if we chose to place one source each 250m, we must use about four hundred sources. This means that we have to solve four hundred wave equations for each iteration. These equations are posed in heterogeneous media, which also contributes to increasing the computational burden, whether it is related to the solution of the direct problem or to the inversion itself.

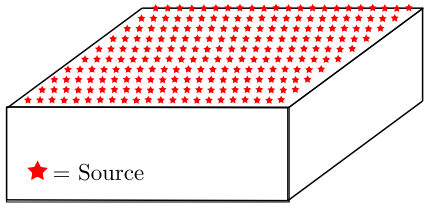


Figure 1.3: Illustration of source deployment on 3D regular domain.

Indeed, in order to properly integrate the heterogeneous characteristics of the medium, the numerical method used for the direct problem must be of very high precision. Moreover, a significant increase in the number of iterations is to be expected in order to converge towards a set of discontinuous parameters, as it is the case to describe a heterogeneous medium. It is therefore important to address the challenge of avoiding local minima while keeping calculation costs acceptable for the majority of end-users. For solving the minimization problem (1.6), one can use a local or a global approach. A local approach is based upon the following condition: in order to be a global minimum of the cost function, m must be a zero of its gradient. This condition is sufficient if the cost function is convex, which is not the case here. Hence, by adopting this approach, there is a chance to converge to a local minimum. It is possible to control this drawback by using the Hessian [Métivier et al., 2013, Fichtner and Trampert, 2011] but taking the second-order derivatives will increase the computational costs.

Regarding global algorithms, they search for a global minimum by exploring a larger parameter space. These methods avoid the computation of the gradient of the cost function and require an important amount of cost function evaluations. Such methods are more suitable for retrieving a low number of parameters in problems where the cost function evaluation can be performed in a short time. To mention few global optimization algorithms, we can cite the simulated annealing [van Laarhoven and Aarts, 1987] and genetic algorithm [Davis, 1991]. The first approach is inspired by the annealing process used in metallurgy, where shaping the metal is easier at high temperatures. This algorithm introduces a numerical temperature which, if elevated, allows the cost function to be worse (in the case of a minimization) than

in the previous state. This kind of behavior encourages the exploration of the parameter space and prevents getting stuck in a local minimum. Concerning the genetic algorithm, the parameter space coverage is ensured by genetic mixing and the introduction of random mutations in a population of solutions that are studied during several generations.

However, such methods cannot be applied to the geophysical inverse problems we aim to solve. On the one hand, the number of parameters to recover is very high in geophysics, on the ohther and, the evaluation of the cost function is computationally challenging. For these reasons, we will only use local optimization algorithms. Such an optimization relies on the computation of the gradient of the cost function $\mathcal{J}$ with respect to the geophysical parameters m (see Figure 1.4). Two methods are commonly used to determine the gradient. The first one, which is the most intuitive, is the sensitivity functions method that computes the Jacobian of the cost function with respect to the physical parameters. The gradient evaluation is then proportional to the number of parameters we aim to recover. The second option is the adjoint state method that gives access to the gradient by computing the state equation and its adjoint, which have essentially the same computational cost. With this method, it is then possible to compute the gradient of the cost function with respect to the parameters for a cost which is equivalent to evaluate the state equation twice regardless of the number of parameters. The adjoint state method is then recommended for recovering geophysical models.

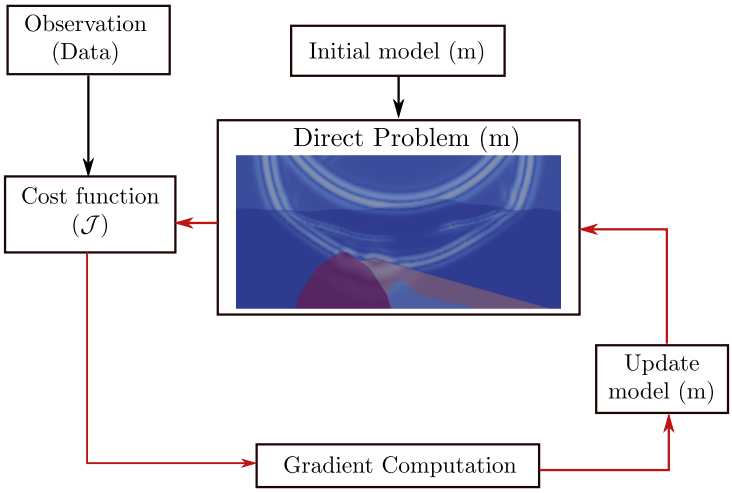


Figure 1.4: Classical workflow of minimization problem.

In order to recover the physical model, this inversion method requires the simulation of the overall behavior of the wave in the domain of interest, which gives the name Full Waveform Inversion. The term "Full" comes from the fact that this method is going way beyond refraction and reflection tomography techniques, which only use the travel time kinematics of the seismic data.

The FWI, which is a minimization problem that aims to recover physical properties by adjoint state method, has been introduced in the early 80s by Patrick Lailly [1983] and Albert Tarantola [1984].

In this section, we placed the general setting of the FWI. We did not go into details in the definition of the objective function neither on the parameterization. The objective of this chapter is to define all the required notions that will be addressed in the thesis.

In what follows, we will describe more precisely the cost function we aim to minimize. We will also come back on the gradient computation, and we will go further in details to compare the sensitivity functions and the adjoint state method. Then, we will make an overview of the

optimization algorithm that can be used, and we will make some benchmarks on well-known synthetic optimization problems to define an optimization strategy in the remainder of the thesis.

1.2 Geophysical context

In this section, we aim to go further into details concerning the geophysical context and all the mechanics and notions inherited from the problem we are considering. We will present the context and familiarize the reader with notions and vocabulary specific to geophysics.

We will more precisely describe the classical setup used in seismic campaigns in order to obtain data from the subsurface. Then, we will describe what the observed data look like and how they are recorded. In particular, we will discuss the different types of receivers that exist and the physical quantities of interest.

Finally, we will present the cost function that will be studied throughout the thesis where we will list alternatives to the usual least square norm commonly used.

1.2.1 Seismic data acquisition

The objective of the inverse problem is to minimize a cost function that aims to quantify the differences between the data from numerical simulations, for a given model m, with the observed data d_{obs}. The objective of the seismic acquisition campaign is to collect these data d_{obs}. This consists in using a set of sources and receivers placed close enough to the surface, so that the problem is neither too intrusive nor too expensive. The acquisition devices are therefore divided into two categories, the sources and the receivers. In seismic exploration the sources are artificial. We then have a complete control of their location and a fairly good understanding of their formulation. In the same way, we know where the receivers are located. The sources generate a disturbance in the environment of interest Ω that propagates and is recorded by the receivers.

So, for each source, each receiver records a disturbance during the time of the experiment. The receivers can register different physical quantities, but we will come back to this a little later. Let us first consider that receivers register a pressure perturbation (in $\mathrm{kg.m^{-1}.s^{-2}}$ or Pa) over time. This disturbance, an example of which we display in Figure 1.5, is what we call a seismic trace. This kind of trace is the result of the propagation of the perturbation generated by the source and owns information from the different geophysical reflectors that the wave has passed through.

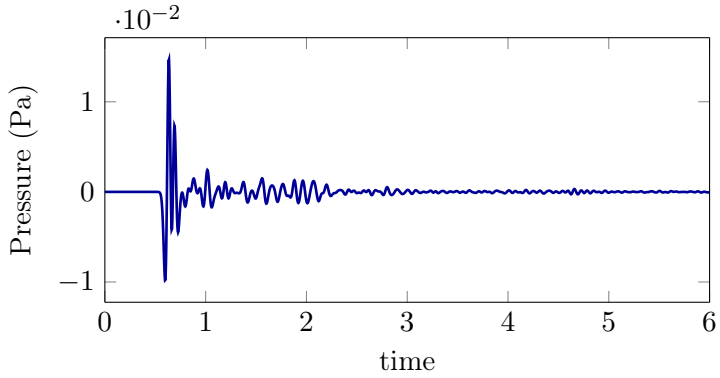


Figure 1.5: Example of seismic trace.

For one source, we then record as many seismic traces as there are receivers. We denote by n_{rcv} the number of receivers used during the acquisition. We represent in Figure 1.6 a scheme of a 2D synthetic seismic acquisition campaign for one central source.

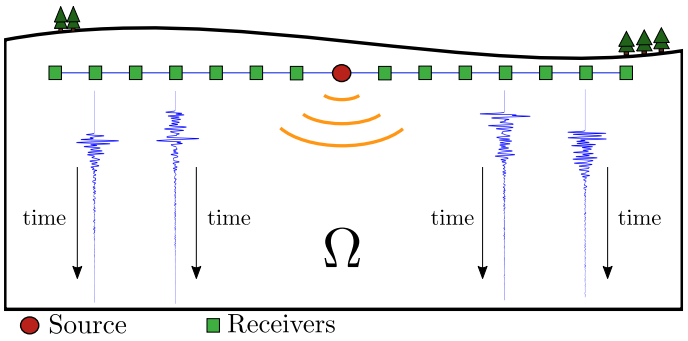


Figure 1.6: Illustration of seismic data acquisition for one source.

By collecting all the seismic traces thus obtained, we can form what is called a seismogram.A seismogram represents on the X-axis the index of the receiver, on the Y-axis the time and the color map represents the amplitude of the measurement. Figure 1.7 represents the synthetic seismogram of a 2D problem where the source is located in the center of the domain of interest. The picture highlights the cone of propagation, which is typical in this type of representation. The cone is explained by the time of the first arrival, which is shorter if the receiver is located close to the source. We have also tried to be faithful to this phenomenon in the illustration in Figure 1.6.

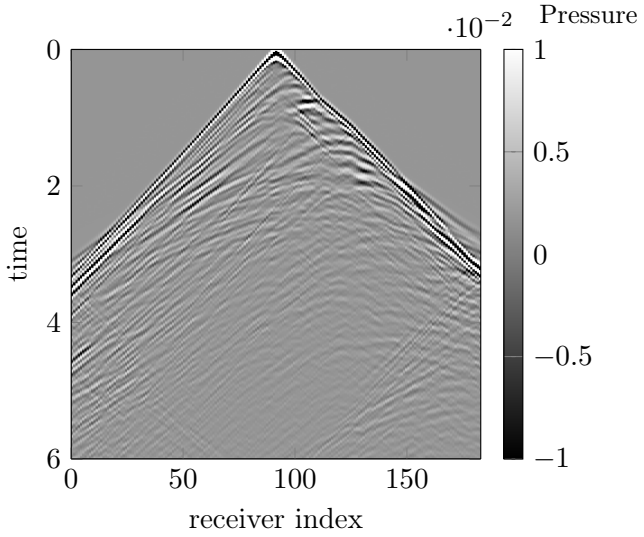


Figure 1.7: Example of seismogram.

Note that, for better coverage of the domain of interest Ω, several sources are used. For each source, the number of receivers n_{rcv} and their locations may vary. If we denote the number of sources by n_{src}, the set of observed data d_{obs} is then defined by $n_{src} \times n_{rcv}$ seismic traces in the case where n_{rcv} is constant for each acquisition.

Now that we have described the experimental configuration of data acquisition, we will focus the following on how the data are defined, and we will also make a brief overview of the different geophysical devices used to record them.

1.2.2 Observations and measurement tools

Seismic data are naturally defined in the time domain. In the case where the observed quantity is the field u, we can define the spatial restriction function $\mathcal{R}_{xr}$, which consists in keeping only the information located at the point of coordinate $\boldsymbol{x}_r$ where the receiver r is located, i.e.,

$$\mathcal{R}_{xr}(u) = u(\boldsymbol{x}_r)\,.$$

In a more general way, we can note $\mathcal{R}_x$ the global spatial restriction on all receivers. We then have:

$$\mathcal{R}_x(u) = \{u(\boldsymbol{x}_r)\}_{1\leq r\leq n_{rcv}}\,.$$

The observed data are not continuous and are defined on a sample at specific times. We will note n_t the number of sampling points in time of the observations and dt the time step between two samples. To give estimates of dt, we usually record the signal every 2 or 4 microseconds.

If we define the observed data associated with the source s by $d_s = \{d_{s_i}\}$, where $i \in \{0, ..., n_t\}$, it is then possible to define the application Q from (1.3) by the successive application of a temporal and spatial restriction on the field u:

$$d_s = Q(u) = \mathcal{R}_t\mathcal{R}_x(u)\,.$$

We can get back to expression (1.4) by using the P_s operator introduced in (1.2). Then, we have:

$$d_s = \Phi_s(m) = Q(P_s(m)) = \mathcal{R}_t\mathcal{R}_x(P_s(m))\,. \tag{1.7}$$

We have introduced above the measured field u. It is the field of interest and it may vary, depending on the acquisition. In marine seismic acquisition the receivers, or hydrophones, measure the pressure disturbance. For land acquisitions, it is geophones that are used. They are exclusively located on the surface of the domain of interest. They measure the normal displacement at the surface. For more technical details on the configuration of sources and receivers in real land seismic acquisition, we refer to [Baeten, 1989].

Some receivers measure two physical components, they are called dual-sensor devices. For instance, they record the pressure, as well as the normal velocity of the disturbance [Carlson et al., 2007]. The access to the traces on two physical variables offers more information on the subsurface and gives new perspectives, especially on the description of the cost function.

Being aware of the configuration in which the data acquisition took place is crucial to numerically reproduce the same acquisition. The cost function $\mathcal{J}$ is a function that quantifies, for a given model m, how far away the simulated data are from the observed data. In what follows, we will define the cost function used during the thesis. We will also review the different cost functions used in geophysical exploration.

1.2.3 The cost function

We have previously shown in (1.7) that the operator Φ includes the evaluation of the state equation P_s that gives the field u according to the parameters m. The operator P_s represents the wave operator that governs the propagation of the perturbation from the source s in an acoustic, elastic, elasto-acoustic, or other media. In this thesis, we will consider exclusively acoustic media. The field u then represents the pressure p and the velocity $\boldsymbol{v}$ of the wavefield.

To obtain the simulated observations, it is necessary to evaluate the wavefield u and its restriction to the receivers (1.7).

The cost function can then be set up in the following way:

$$\mathcal{J}(m) = \frac{1}{2} \sum_{s=1}^{n_{src}} \| \phi_s(m) - {d_{obs}}^{(s)} \|_D^2 .$$

For simplification, we will put aside the s index, which will not impact the following. We can then establish the generic cost function formulation:

$$\mathcal{J}(m) = \frac{1}{2} \| \phi(m) - d_{obs} \|_D^2 .$$

The choice of the norm $\| . \|_D$ remains to be defined. The most common choice is the L^2 standard norm and we have then:

$$\begin{aligned} \mathcal{J}(m) &= \frac{1}{2} \| \phi(m) - d_{obs} \|_2^2, \\ &= \frac{1}{2} \sum_{i=0}^{n_t} \sum_{r=1}^{n_{rcv}} ([\Phi(m)]_{i,r} - [d_{obs}]_{i,r})^2 , \end{aligned}$$

where $\Phi(m)$ and d_{obs} can be expressed as matrices of size $n_t \times n_{rcv}$, which is perfectly represented by the seismogram we pictured previously in Figure 1.7.

Within the framework of this thesis, we will exclusively use pressure measurements, which is commonly the case in marine acquisitions. In this case, the L^2 standard norm is expressed as follows:

$$\mathcal{J}(m) = \frac{1}{2} \sum_{i=0}^{n_t} \sum_{r=1}^{n_{rcv}} ([Q(p(m))]_{i,r} - [d_{obs}]_{i,r})^2 , \tag{1.8}$$

which we have slightly adapted in order to handle normalized quantity:

$$\mathcal{J}(m) = \frac{1}{2} \sum_{i=0}^{n_t} \sum_{r=1}^{n_{rcv}} \frac{([Q(p(m))]_{i,r} - [d_{obs}]_{i,r})^2}{\sigma_r^2}, \tag{1.9}$$

where $\sigma_r = \sqrt{\frac{1}{n_t+1} \sum_{i=0}^{n_t} [d_{obs}]_{i,r}^2}$ represents the standard deviation of the data observed on the seismic trace associated with the receiver r. In this way, there are no preponderant receivers in the evaluation of the cost function. Since in the L^2 classical cost function defined in (1.8) the amplitude of the wavefield is naturally more important for the receivers close to the source, the normalization by the variance σ_r allows to balance the participation of the receivers according to their distance to the source.

The cost function defined in (1.9) is the one that is used in the framework of the thesis. However, there exist several cost functions and norms that can be used to quantify the misfit between simulations and observations. Brossier et al. [2010] focuses on the L^1 norm applied to frequency FWI, which has been demonstrated to be weakly sensitive to noise. Some hybrid L^1/L^2 strategies have also been studied by Bube and Langan [1997] and also shown improved robustness to noise compared with conventional L^2 norm.

Still in the frequency domain, we can cite the works [Shin et al., 2007, Bednar et al., 2007, Pyun et al., 2007], which compare the results obtained for different cost functions based on the phase and/or amplitude of the signal. These works focus on the logarithmic function first

mentioned in [Tarantola, 1987]. The logarithm misfit function has also been employed by Shin and Min [2006], Faucher [2017], which reveals to be central for complex frequency inversion.

More recently, a new class of misfit functions based on optimal transport theory has been studied in [Métivier et al., 2016a, Yang et al., 2017, Chen et al., 2018]. Mainly based on Wasserstein [Engquist and Froese, 2013] or Kantorovich-Rubenstein [Métivier et al., 2016b] norms, optimal transport misfit is not a 'point by point' objective function, which means that we do not compute the difference in amplitude at each sampling time. The optimal transport methods compute the minimal cost to rearrange one signal into another. In that way, these methods bring new criteria to quantify the difference between observed and simulated data. Furthermore, such methods are less sensitive to noise and offer better convexity properties that reduce cycle skipping effects [Yunan, 2018].

In this subsection, we have defined, in a geophysical context, the cost function that we wish to minimize. As explained previously, this minimization problem will be solved by local optimization methods, which we will detail later in this chapter. These methods depend on the calculation of the gradient of the cost function by the set of physical parameters we are trying to retrieve m. The calculation of the gradient $\nabla\mathcal{J}$ is then a key step of the FWI that, as explained above, is calculated by the adjoint state method. In order to motivate this choice, we will present in the next section the options available to calculate the gradient $\nabla\mathcal{J}$ and discuss the possibilities in light of the geophysical context.

1.3 Gradient computation

We introduced in Section 1.1 the three spaces defining the inverse problem that we recall here:

- M, the space of parameters;
- D, the space of observations;
- U, the wavefield or state space.

These spaces, which are generally defined as Hilbert spaces, require, for the sake of numerical optimization, to be reduced into a finite-dimensional space. Note that this is often the case for the D space because we generally have a finite number of observations. Without describing the discretization of U, which will be discussed in more details in Chapters 2 and 3 of this document, we will approach M by $\mathbb{R}^{n_p}$, which is intrinsically defined by the discretization of the U space. The set of parameters will be represented by the vector $m = \{m_i\}_{1\leq i\leq n_p}$ and n_p denotes the total number of parameters.

Definition 1.3.1. The function $\mathcal{J} : M \to \mathbb{R}$, is Fréchet-differentiable or F-differentiable at point m_0, if and only if there is a bounded linear operator $\mathcal{J}'(m_0) : W \to \mathbb{R}$, with $W \subset M$ such that:

$$\lim_{\|\delta_m\|_M\to 0} \frac{|\mathcal{J}(m_0+\delta_m) - \mathcal{J}(m_0) - \mathcal{J}'(m_0)\delta_m|}{\| \delta_m \|_M} = 0, \qquad \forall\, \delta_m \in M\,.$$

Then, $\mathcal{J}'(m_0)$ is the differential of $\mathcal{J}$ at the point m_0. In the remaining of the chapter we will also denote the differential operator by a capital D. The Fréchet derivative of the cost function $\mathcal{J}$ with respect to the parameter at point m_0 will be then noted that way:

$$D_m\mathcal{J}(m_0)\delta_m = \mathcal{J}'(m_0)\delta_m\,.$$

Remark. It is then possible to obtain the first order expansion, we have:

$$\mathcal{J}(m_0 + \delta_m) = \mathcal{J}(m_0) + \mathcal{J}'(m_0)\delta_m + o(\| \delta_m \|_M).$$

Let us recall the Riesz representation theorem:

Theorem 1.3.1. Let H being a Hilbert space, with inner product $\langle \cdot, \cdot \rangle$ and $f \in H^*$ a continuous linear form on H,

$$\exists! y \in H, \qquad \forall x \in H, \qquad f(x) = \langle y, x \rangle.$$

By applying the Riesz representation theorem, one will denote by $\nabla\mathcal{J}(m_0)$ the gradient of $\mathcal{J}$ at the point m_0 that is the unique vector of M defined as:

$$< \nabla\mathcal{J}(m_0), \delta_m >_M = \mathcal{J}'(m_0)\delta_m \qquad \delta_m \in M\,.$$

Thus, the gradient of the cost function at m_0 with respect to the model parameters is represented by the vector:

$$\nabla\mathcal{J}(m_0) = \{\frac{\partial \mathcal{J}}{\partial m_i}(m_0)\}_{1 \leq i \leq n_p}\,.$$

Intuitively, we could calculate $\nabla\mathcal{J}(m_0)$ by using finite difference approximation:

$$\{\nabla\mathcal{J}(m_0)\}_i \approx \frac{\mathcal{J}(m_0 + h\boldsymbol{e}_i) - \mathcal{J}(m_0)}{h} \qquad \forall i \in \{1, ..., n_p\},$$

where $\boldsymbol{e}_i$ represents the canonical vector of size n_p that is equal to 0 everywhere but at the ith component where it is equal to 1. This calculation requires $n_p + 1$ times the evaluation of the objective functional. The cost of this method is prohibitive if the number of parameters n_p is large and if the calculation burden to evaluate $\mathcal{J}$ is important. This method, to be avoided, can however be used as a means of verifying the calculation of the gradient by the methods we will see below.

1.3.1 Sensitivity functions method

The sensitivity function method is a natural way to determine the gradient of the cost function $\mathcal{J}$. We refer to [Kern, 2002, Chavent, 2010] for more details concerning this method. It consists in determining explicitly the derivative of $\mathcal{J}$ with respect to the parameters m.

We remind that we will consider the following standard formulation of the cost function:

$$\mathcal{J}(m) = \frac{1}{2} \| \phi(m) - d_{obs} \|^2_D \ .$$

Using the derivative chain rule, its gradient is given by:

$$\nabla\mathcal{J}(m) = \phi'(m)^*(\phi(m) - d_{obs})\,, \tag{1.10}$$

where ϕ' represents the Jacobian of the observation functional ϕ with respect to the model m. The symbol "*" represents the adjoint operator.

Proof:

$$\begin{aligned} D_m\mathcal{J}(m)\delta m &= D_m \frac{1}{2} < \phi(m) - d_{obs}, \phi(m) - d_{obs} >_D \delta m, \\ &=< D_m(\phi(m) - d_{obs})\delta m, \phi(m) - d_{obs} >_D, \\ &=< \phi'(m)\delta m, \phi(m) - d_{obs} >_D, \\ &=< \delta m, \phi'(m)^*\phi(m) - d_{obs} >_M\,, \end{aligned}$$

where we denote by $D_m\mathcal{J}(m)$ the Fréchet-derivative of the cost function $\mathcal{J}$.

Then we retrieve the results given in (1.10) by identification.

This formula shows that we have to determine Φ' and for that purpose, we apply the implicit function theorem. We recall this theorem in the following.

Theorem 1.3.2. Let X, Y and Z be three Banach spaces, and the mapping $f : X \times Y \to Z$ be continuously Fréchet-differentiable. If $(x_0, y_0) \in X \times Y$, $f(x_0, y_0) = 0$, and $y \to D_y f(x_0, y_0)(0, y)$ is a Banach space isomorphism from Y to Z, then, there exist a neighborhood U of x_0 and V of y_0 and a Fréchet-differentiable function $g : U \to V$ such that :

$$\forall (x, y) \in U \times V \, , f(x, y) = 0 \iff \forall x \in U, y = g(x) \, .$$

Then, we go back to the state equation given for a model m_0 and a source field s:

$$F(m_0, u) = s.$$

By passing s to the left-hand side, we can rewrite the state equation as:

$$F_s(m_0, u) = 0.$$

Then the mapping $F_s : M \times U \to Z$, is Fréchet-differentiable with respect to both variables.

By applying the implicit function theorem, there exists a Fréchet-differentiable function $g : W \subset M \to U$, where $\forall m \in W$ neighborhood of m_0 we have:

$$F_s(m, u) = F_s(m, g(m)) = 0 \, .$$

The derivative of the state equation according to the model parameters writes:

$$D_m F_s(m, g(m)) + D_u F_s(m, g(m)) g'(m) = 0 \, .$$

This expression shows that the Jacobian of g is obtained as a solution of a linear system. Now we go back to the derivative of Φ. According to the above calculations, since we have $\Phi(m) = Qg(m)$, we get:

$$g'(m) = -D_u F_s(m, g(m))^{-1} D_m F_s(m, g(m)) \, .$$

Finally, in the neighborhood W of m_0 we have:

$$\begin{aligned} \phi'(m) &= Qg'(m), \\ &= -QD_u F_s(m, g(m))^{-1} D_m F_s(m, g(m)). \end{aligned}$$

The model is represented with a finite number n_p of parameters. Hence, the computational cost to evaluate $\phi'(m)$ is equivalent to resolve n_p linear systems. The computational cost of the sensitive functions method is then proportional to the number of parameters n_p characterizing the model parameter space M. Hence, if n_p is very large as it is for describing a geophysical medium, this method could be too much expensive.

However, this method gives access to the Jacobian of the observation operator ϕ, which brings new possibilities concerning the choice of the optimization method used to solve the minimization problem under study. Besides, the gradient obtained with such a method is exact.

In order to have a gradient whose calculation cost does not depend on the number of parameters to be reconstructed, a very favored alternative is to use the adjoint state method. We will see in the next section that it is possible to define the gradient $\nabla\mathcal{J}$ for an additional cost close to the one of the evaluation of the state equation.

1.3.2 Adjoint State method

We have seen in the previous section, that we can calculate the gradient of the function in the following way:

$$\begin{aligned}\nabla\mathcal{J}(m) &= -(QD_uF_s(m,g(m))^{-1}D_mF_s(m,u))^*(Qu(m)-d_{obs}), \\ &= -(D_mF_s(m,u)^*(D_uF_s(m,g(m))^*)^{-1}Q^*)(Qu(m)-d_{obs}).\end{aligned} \tag{1.11}$$

Kern [2002] shows that, with a trivial manipulation of the parentheses, we can fundamentally modify the reading of expression (1.11):

$$\nabla\mathcal{J}(m) = -(D_mF_s(m,u))^*\left(D_uF_s(m,g(m)^*)^{-1})Q^*(Qu(m)-d_{obs})\right).$$

Let us denote by λ the solution of:

$$D_uF_s(m,g(m))^*)\lambda = -Q^*(Qu(m)-d_{obs})\,..$$

Then, the gradient of the cost function is written in a simple way as follows:

$$\nabla\mathcal{J}(m) = (D_mF_s(m,u))^*\lambda\,.$$

As a matter of fact, obtaining the gradient involves non-trivial adjoint operators.

This result can also be obtained as a solution to a constrained minimization problem. This formulation is based on the calculation trick that consists in considering that u and m are independent variables and that u is constrained to satisfy the state equation (1.1). A justification of this formalism can be found in [Chavent, 2010, Allaire, 2019]. We introduce the Lagrangian $\mathcal{L}$ as follows:

$$\mathcal{L}(u,m,\lambda) = \hat{\mathcal{J}}(u) + \langle F_s(m,u),\lambda\rangle_Z\,,$$

where we have the mapping function $F_s : M\times U\to Z$ and

$$\hat{\mathcal{J}}(u) = \frac{1}{2}\parallel Q(u)-d_{obs}\parallel^2_D\,,$$

which only depends on the field u. The Lagrangian is a common way to define an optimization problem under constraint and λ is then called the Lagrange variable.

This Lagrangian has four fundamental properties that should be listed to automate the calculations leading to the $\nabla\mathcal{J}(m)$ gradient.

- If u satisfies the state equation (1.1), then:

  $$\forall\lambda\in Z\,,\qquad \mathcal{L}(u(m),m,\lambda) = \hat{\mathcal{J}}(u(m)) = \mathcal{J}(m)\,,$$

  the Lagrangian $\mathcal{L}$ equals the cost function $\mathcal{J}$ we aim to minimize.

- By computing the derivative $D_\lambda\mathcal{L}$ with respect to the Lagrange variable:

  $$\begin{aligned}D_\lambda\mathcal{L}(u,m,\lambda)\delta\lambda &= D_\lambda\hat{\mathcal{J}}(u)\delta\lambda + D_\lambda\langle F_s(m,u),\lambda\rangle_Z\delta\lambda \\ &=< F_s(m,u),\delta\lambda >_Z\,,\end{aligned}$$

  and requiring $D_\lambda\mathcal{L}\delta\lambda = 0$ for all $\delta\lambda\in Z$, gives us the state equation:

  $$F_s(m,u) = 0\,.$$

- By computing the derivative $D_u\mathcal{L}$, we have:

$$\begin{aligned} D_u\mathcal{L}(u,m,\lambda)\delta u &= D_u\hat{\mathcal{J}}(u)\delta u + D_u\langle F_s(m,u),\lambda\rangle_Z\delta u, \\ &= \frac{1}{2}D_u\langle Qu-d_{obs},Qu-d_{obs}\rangle_D\delta u + \langle D_uF_s(m,u)\delta u,\lambda\rangle_Z, \\ &= \langle\delta u, Q^*(Qu-d_{obs})\rangle_U + \langle\delta u, D_uF_s(m,u)^*\lambda\rangle_U. \end{aligned}$$

  Requiring $D_u\mathcal{L}\delta u = 0$ for all $\delta u \in U$ gives us the adjoint state equation:

$$D_uF_s(m,u)^*\lambda + Q^*(Qu-d_{obs}) = 0\,. \tag{1.12}$$

- Since we know that if u is solution of the state equation, then $\mathcal{L}(u,m,\lambda) = \mathcal{J}(m)$, we can express the derivative of the cost function with respect to m as follows:

$$\mathcal{J}'(m)\delta m = D_m\mathcal{L}(u,m,\lambda)\delta m + D_u\mathcal{L}(u,m,\lambda)D_mu\delta u.$$

By choosing $\lambda \in Z$ satisfying the adjoint state equation (1.12) then the term $D_u\mathcal{L}(u,m,\lambda)$ vanishes. Then, the differential of $\mathcal{J}$ simplifies to:

$$\mathcal{J}'(m)\delta_m = \langle D_mF_s(m,u(m))\delta m,\lambda\rangle_Z$$

It is then possible to identify the gradient $\nabla\mathcal{J}(m)$:

$$\nabla\mathcal{J}(m) = D_mF_s(m,u(m))^*\lambda\,, \tag{1.13}$$

which is the expression we determined before in (1.11).

Contrary to sensitivity functions methods, the adjoint state methods determine directly the gradient of the cost function without computing its Jacobian. This makes this approach particularly interesting because the optimization process may only deal with the gradient. The gradient can then be obtained by solving first the state equation and then the adjoint state equation. More importantly, the computational cost of the gradient does not depend on the number of parameters used for representing the wavespeed model. Indeed, geophysical models usually involve several thousands to millions of parameters.

The introduction of a Lagrangian offers a chain rule to be followed to determine the gradient of the considered problem. The resulting gradient is then easily expressed as long as the adjoint state λ is well determined. Once λ is calculated, the gradient can be determined by the formula (1.13) with a negligible extra cost compared with the one of the evaluation of u and λ.

1.3.3 Concluding remarks

We saw in the previous section that we can obtain the first order derivative of the cost function with respect to the model parameters m applying several methods:

- finite difference approximation;
- sensitivity functions method;
- adjoint state method.

The first method is really expensive and required the evaluation of $n_p + 1$ cost functions to get an approximation of the gradient. However, this method is easy to implement and can easily be adjusted to the choice of the parameterization and the choice of the objective functional. We have also presented the sensitivity functions and adjoint state methods that enable the computation of the gradient. Each method has its own advantages and disadvantages, and following Kern [2016], we displayed them in Table 1.1.

	Sensitivity functions	Adjoint state
Jacobian	✓	✗
Gradient	✓	✓
Computation cost	proportional to n_p	1 state equation + 1 adjoint state equation
Adaptative parameterization	✗	✓
Adaptative cost function	✗	✓
Optimization Algorithm	Gauss-Newton	Quasi-Newton

Table 1.1: Comparison of sensitivity functions and adjoint state method to evaluate gradient of the objective function with respect to the model parameters.

It is clear that, with a high number of parameters, the adjoint state is the best candidate method to compute the gradient. Besides, this method offers a lot of flexibility not only concerning the choice of parameterization, but also the choice of the cost function. Indeed, as explained previously in this section, there exist plenty of objective functions that can quantify the differences between observed and numerical solutions. With the adjoint state method, it may be simpler to conserve the overall workflow for different cost functions or parameterization.

The computation of the gradient then requires to compute the state variable u also called "Forward wavefield", by computing the forward simulation, and the adjoint state λ called "Backward wavefield" by computing the backward simulation. We presented in figure Figure 1.4 on page 26 the flowchart of the optimization process that can be updated including the adjoint state method. Figure 1.8 represents the optimization loop used in the FWI.

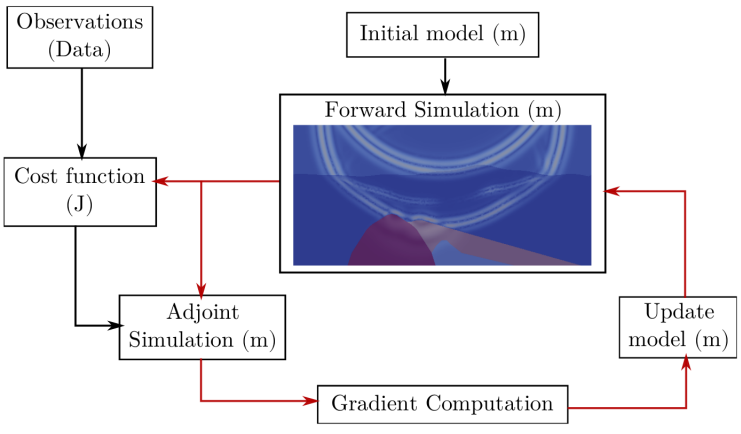


Figure 1.8: FWI workflow using the adjoint state method.

It remains for us to study the possible optimization methods that can be used for minimizing the cost function.

1.4 Optimization Method

We have previously justified the choice of a local optimization algorithm and to carry it out, the access to the gradient of the cost function which is a prerequisite.

As shown in Figure 1.8, FWI relies on successive updates of the physical parameters to match as well as possible the observations with the simulation. Newton method proves its efficiency for resolving such nonlinear iterative problems. Based on Taylor expansion of the gradient of the cost function, we may write for a small perturbation δ_m:

$$\nabla\mathcal{J}(m+\delta_m) = \nabla\mathcal{J}(m) + H(m)\delta_m + o(\| \delta_m^2 \|)\,,$$

where H represents the second derivative of the cost function called the Hessian.

The perturbation δ_m is a sequence depending on m which is defined for making $\nabla\mathcal{J}(m+\delta_m)$ converge to zero ideally. By writing that the gradient of the updated model is zero, we get an approximate expression of the perturbation up to order 2:

$$\delta_m = -H(m)^{-1}\nabla\mathcal{J}(m)\,,$$

and it leads to the Newton optimization update formula at the i^{th} iteration of the process:

$$m^{i+1} = m^i - H(m^i)^{-1}\nabla\mathcal{J}(m^i)\,.$$

Remark. Unfortunately, the condition $\nabla\mathcal{J} = 0$ does not guarantee to retrieve the global minimum and may converge to a local minimum. Regularization, and multiscale reconstruction, may be used to ensure the convergence toward the global minimum. However, regularization may be difficult to set since it requires adding prior information of the targeted model. Furthermore, a strong regularization can fade the reconstruction and may remove small structures. We refer to Lopez [2014] for a detailed description of regularization methods applied to FWI. In this thesis, we will only use multiscale reconstruction [Bunks et al., 1995] to force the convergence toward the global minimum. This issue will be discussed later in Chapter 4.

However, Newton optimization requires the Hessian computation, which can be very expensive with an important memory burden especially when the number of parameters is high. Such limitations make the access to the second order information unavailable and justify the use of Quasi-Newton algorithm. Furthermore, in the context of seismic reconstruction, it has been mentioned by Tarantola [1987] that the second order information does not provide enough interesting improvements to privilege Newton-type methods over classical descent algorithm. It might even be preferable to introduce additional iterations of descent direction, than trying to calculate the Hessian matrix at each iteration. Nevertheless, the opposite view is defended in [Pratt et al., 1998] where an interpretation of the information contained in the Hessian is provided in a context of seismic reconstruction. Faucher [2017] makes the comparisons by using Hessian conjugate gradient approximation and gradient method seismic model reconstruction in frequency domain. At high frequencies, using Hessian seems to improve convergence but the computational cost seems really unreasonable in view of the final reconstructed model which is very close for both approaches.

We choose, for this thesis, to consider only gradient descent methods that we will describe in this section. At the i^{th} iteration, these methods can be written in such a way:

$$m^{i+1} = m^i + \alpha^i d^i\,,$$

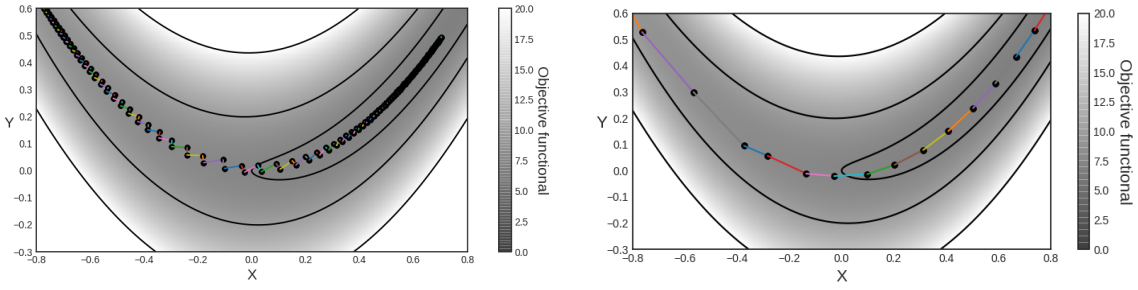
where,

- the scalar α: represents the step length real coefficient ($\alpha \geq 0$);
- the vector $d = \{d_j\}_{1 \leq j \leq n_p}$: represents the vector of model updates also known as search direction.

For more information on the algorithms that will be used, we refer to the book of Nocedal and Wright [2006], which proposes a vast documentation of existing optimization techniques. The study of these methods can then be split in two. We will first review the different search direction method implemented and then the methods giving an adapted line search step length. Finally, we will propose a benchmark of these methods on synthetic cases of optimizations in order to identify a strategy to be applied for FWI.

1.4.1 Steepest descent method

A simple choice for the search direction is to choose the opposite of the gradient direction, $d^i = -\nabla \mathcal{J}(m^i)$. This method, called the steepest descent, is really easy to implement once the gradient is computed. However, this search direction is a bad choice to decrease $\mathcal{J}$. If $\mathcal{J}$ is complex and the initial guessed parameters m^0 is far from the solution, the steepest descent requires small step length α increasing the computational burden. Such methods are also sensitive to "zigzags". We illustrate this effect by plotting the resolution path of a minimization problem based on the Banana Rosenbrock function using the steepest descent method and Newton algorithm in Figure 1.9.



(a) Optimization with steepest descent method. (b) Optimization with Newton method.

Figure 1.9: Path realized by the couple (x^i, y^i) we aim to retrieve to minimize the Banana Rosenbrock functional for steepest descent method (a) and Newton method (b).

Unfortunately, according to the book of Bonnans [2006], this method may never converge and for this reason, it must be abandoned in favor of other methods that we will present in the following and that have similar calculation costs. However, it is common for the steepest descent method to be used to initialize the optimization loop which is then implemented with another more efficient method. In the following subsections, we will develop other methods that are compatible with the memory and computational restriction of the FWI.

1.4.2 Non-linear Conjugate Gradient

Non-linear Conjugate Gradient (NLCG) algorithm is a generalization of the classical Conjugate Gradient (CG) method for nonlinear optimization problems [Nocedal and Wright, 2006]. NLCG method offers an alternative approach to the steepest descent and requires the knowledge of the gradient only.

Basically, the search direction in NLCG methods is computed following this formula:

$$d^i = -\nabla \mathcal{J}(m^i) + \beta^i d^{i-1}\,,$$

where the first iteration is simply the steepest descent step. There exist several formulations to define the coefficient β^i. In this thesis we have implemented the four main formulations:

- Fletcher and Reeves [1964] formula (FR), $\beta^i = \frac{\nabla \mathcal{J}(m^i)^\top \nabla \mathcal{J}(m^i)}{\nabla \mathcal{J}(m^{i-1})^\top \nabla \mathcal{J}(m^{i-1})}$;
- Polak and Ribiere [1969] formula (PR), $\beta^i = \frac{\nabla \mathcal{J}(m^i)^\top (\nabla \mathcal{J}(m^i) - \nabla \mathcal{J}(m^{i-1}))}{\nabla \mathcal{J}(m^{i-1})^\top \nabla \mathcal{J}(m^{i-1})}$;
- Hestenes et al. [1952] formula (HS), $\beta^i = -\frac{\nabla \mathcal{J}(m^i)^\top (\nabla \mathcal{J}(m^i) - \nabla \mathcal{J}(m^{i-1}))}{d^{i-1\top} (\nabla \mathcal{J}(m^i) - \nabla \mathcal{J}(m^{i-1}))}$;
- Dai and Yuan [1999] formula (DY), $\beta^i = -\frac{\nabla \mathcal{J}(m^i)^\top \nabla \mathcal{J}(m^i)}{d^{i-1\top} (\nabla \mathcal{J}(m^i) - \nabla \mathcal{J}(m^{i-1}))}$.

NLCG method does not have an excessive computational cost since it only requires to compute the scalar β^i, which is obtained by evaluating two inner products. However, this method requires saving in memory the gradient and the search direction of the previous iteration. In practice, we save this information on disk since the computational time required by the gradient computation is much more important than the one of optimization routines. It is then more valuable to spend some time reading and writing on the disk than trying to keep everything in memory.

For an in-depth study of NLCG methods and an interesting analogy with quasi-Newton methods we refer to [Bonnans, 2006].

1.4.3 BFGS and L-BFGS

We saw at the beginning of this section that Newton methods cannot be used for the problem we are considering for computational and memory issues. We recall that the search direction computed by Newton method is given by:

$$d^{i+1} = -H(m^i)^{-1} \nabla \mathcal{J}(m^i)\,.$$

The class of method called Quasi-Newton methods are meant to approximate the inverse of the Hessian by using only the gradient information. A quasi-Newton method is based upon the approximation of $H(m^i)^{-1}$ by a Symmetric Definite Positive (SPD) matrix W^i, which evolves recursively as follows:

$$W^{i+1} = W^i + B^i\,,$$

where B^i represents a correction. Once the operator B^k is defined the Quasi-Newton search direction becomes:

$$d^{i+1} = -W^{i^{-1}} \nabla \mathcal{J}(m^i)\,.$$

It remains to define the initial matrix W^0 and the correction B^i.

The most popular Quasi-Newton method is the BFGS method elaborated by Broyden, Fltecher, Goldfarb and Shanno. This algorithm gives:

$$W^{i+1} = (I - p^i s^i y^{i\top}) W^i (I - p^i s^i y^{i\top}) + p^i s^i s^{i\top},$$

where:

- $s^i = m^{i+1} - m^i = \alpha^i d^i$;
- $y^i = \nabla \mathcal{J}(m^{i+1}) - \nabla \mathcal{J}(m^i)$;

- $p^i = \frac{1}{{y^i}^\top s^i}$.

Concerning the choice of W^0, an effective choice is to use an approximation of the Hessian. But since this is what we avoid calculating using a quasi-newton method, we mainly choose the identity operator I.

BFGS has been proved to be robust and to have a superlinear convergence [Nocedal and Wright, 2006]. Despite being less efficient than the Newton method, Quasi-Newton methods are more affordable since they do not require to compute the Hessian and avoid resolution of a linear system.

However, since the method is recursive, it requires to store the approximate matrix W^i, which may be unfeasible because it is of size $n_p \times n_p$. To reduce the memory load, Nocedal [1980] proposes an algorithm that approximates the BFGS product $W^i \nabla\mathcal{J}(m^i)$, using the n previous stages of the optimization that is to say the n previous m^i and $\nabla\mathcal{J}(m^i)$ vectors. Initially called Sampled Quasi-Newton algorithm, this method is now known as Limited-memory BFGS or L-BFGS. We display in Algorithm 1 the computation of the search direction using L-BFGS method.

Input: m^i, $\nabla\mathcal{J}(m^i)$
Stored: m^{i-1},..., m^{i-n-1}, $\nabla\mathcal{J}(m^{i-1})$,..., $\nabla\mathcal{J}(m^{i-n-1})$
Delete from disc m^{i-n-1}, $\nabla\mathcal{J}(m^{i-n-1})$
Save to disc m^i, $\nabla\mathcal{J}(m^i)$
$q \leftarrow \nabla\mathcal{J}(m^i)$
for $k = (i-1), (i-2), ..., (i-n)$ **do**
  $g^k \leftarrow \nabla\mathcal{J}(m^k)$
  $s^k \leftarrow m^{k+1} - m^k$
  $y^k \leftarrow g^{i+k} - g^k$
  $p^k \leftarrow \frac{1}{{y^k}^\top s^k}$
  $\alpha^k \leftarrow p^k {s^k}^\top y^k$
  $q \leftarrow q - \alpha^k y^k$
end
$\gamma^i \leftarrow \frac{{y^{i-1}}^\top s^{i-1}}{{y^{i-1}}^\top y^{i-1}}$
$d^i \leftarrow \gamma^i q$
for $k = (i-1), (i-2), ..., (i-n)$ **do**
  $\beta^k \leftarrow p^k {y^k}^\top d^i$
  $d^i \leftarrow d^i + (\alpha^k - \beta^k) s^k$
end
$d^i \leftarrow -d^i$
Output: d^i

Algorithm 1: Compute search direction d^i using L-BFGS algorithm.

This algorithm requires only the storage of the n previous states of m and $\nabla\mathcal{J}(m)$, which is manageable by saving on disc as suggested in the NLCG subsection. Métivier et al. [2013] compares the reconstructions for several choices of n (5, 20 and 40), but no particular differences emerge from this test. For the thesis we have chosen arbitrarily $n = 8$, allowing a compromise between efficiency, computational cost and memory storage.

In this section, we have established the different methods of search direction developed during the thesis in order to solve the problem of minimization raised by the FWI. In Chapter 4 at Subsection 4.2.1, we perform a convergence study, of these search direction methods, applied

to the reconstruction of a geophysical medium. Now that we have reviewed the different ways to calculate the search direction, we will discuss the strategies to compute an efficient line search.

1.4.4 Line search method

The minimization is carried out by activating an iterative process. At each iteration of the optimization, we define a search direction d^i to update the parameters as follows:

$$m^{i+1} = m^i + \alpha^i d^i \,.$$

The optimal step length at the i^{th} iteration consists in finding α^i such that:

$$\alpha^i = \underset{\alpha}{inf} \mathcal{J}(m^i + \alpha d^i) \,. \tag{1.14}$$

However, the optimal step length is solution of another minimization problem (1.14) which is not affordable. The choice of the line search coefficient is then a trade-off between the computational time and the decrease of $\mathcal{J}$.

A step length α^i is considered as valid if it satisfies Armijo conditions [Armijo, 1966]. This conditions guarantees that the step α^i is not too large and is defined by the following relation:

$$\mathcal{J}(m^i + \alpha^i d^i) \leq \mathcal{J}(m^i) + c_1 \alpha^i {d^i}^\top \nabla \mathcal{J}(m^i) \,, \tag{1.15}$$

where c_1 is a real parameter such as $c_1 \leq 1$, and in practice it is chosen small ($c_1 = 10^{-4}$) [Nocedal and Wright, 2006].

It is important to note that such a condition is not sufficient by itself to have a well-behaved optimization since the condition (1.15) is always satisfied for small steps. To avoid undesirable small updates, we may use the Wolfe condition defined as follows:

$$ {d^i}^\top \nabla \mathcal{J}(m + \alpha^i d^i) \geq c_2 {d^i}^\top \nabla \mathcal{J}(m^i) \,, \tag{1.16}$$

where c_2 is a real factor such that : $c_1 < c_2 < 1$. However, condition (1.16) requires the computation of the gradient at the new update parameter $m + \alpha^i d^i$, which may be expensive when using adjoint state method. Bonnans [2006] suggests that, for situations that require as few computations of the gradient as possible, it may be more efficient to use the Goldstein criterion:

$$\mathcal{J}(m^i + \alpha^i d^i) \geq \mathcal{J}(m^i) + c_2 {\alpha^i}^\top \nabla \mathcal{J}(m^i) \,.$$

We described criteria that define if the step length α^i is 'too large' (Armijo condition) or 'too short' (Wolfe or Goldstein condition). We then propose to use a backtracking line search algorithm to find an appropriate step length. We display in Algorithm 2 the backtracking procedure. This algorithm requires starting with a large enough initial step. If it is too small, it may take several iterations to be efficient.

```
Input: m, d
define α                                              # Initial step length
define p_1                                            # growth factor p_1 > 1
define p_2                                            # reduction coefficient p_2 < 1
while do
    compute J(m + αd)
    if (Armijo condition) then
        if (Wolfe or Goldstein condition) then
            exit                                      # Keep step length
        end
        else
            α ← p_1 * α                               # Increase the step length
            cycle
        end
    end
    else
        α ← p_2 * α                                   # Decrease the step length
        cycle
    end
end
Output: α
```

Algorithm 2: Compute step length α^i using Backtrack method:

The backtracking algorithm requires introducing the definition of two parameters, p_1 and p_2, that represent respectively the growth and reduction factor. These parameters enable to adjust the step length if Armijo condition or Wolfe/Goldstein condition fails. We usually choose $p_1 = 1.5$ and $p_2 = 0.5$. According to the choice of c_1 c_2, p_1, p_2 there is no preset recipe to define the step length.

The backtracking algorithm is interesting because it requires only the computation of one cost function at each iteration of the minimization process of the algorithm (one extra gradient also if Wolfe condition is used). Other line search strategies exist, we can mention the cubic fitting developed in [Bonnans, 2006] which aims to fit the cost function in the direction of the search direction by a third order polynomial or the Maximum Projected Curvature (MPC) from [Khoury and Chavent, 2006] which requires the Jacobian of the cost function. These alternatives are technically better but require more computation.

We have defined all the strategies considered in the thesis, both in terms of search direction and in terms of line search. We propose in the next section to test these algorithms on different known cases assuming that the calculation of the gradient is obtained by adjoint state method in order to see if a strategy emerges knowing that the cost of the calculation of the gradient with this method is significant.

1.4.5 Numerical analysis

In this subsection, we want to test the minimization strategies previously introduced. The interest of these tests is of several orders. First, they allow us to validate the implementation on known synthetic cases, but also, in a second step, to see if there are more suitable methods for an optimization using adjoint state method. To do so, we will propose a set of problems to minimize. Since, for FWI, the gradient is computed by adjoint state methods, we will consider the cost of the gradient to be twice the cost of the evaluation of the cost function. In this way, by accounting for all the evaluations of $\mathcal{J}$ and $\nabla\mathcal{J}$, we will be able to quantify

the total cost of the method for each given problem.

We display in Table 1.2 the set of optimization problem we implement in order to test the optimization routine.

Problem	n_p	**Formula**	**Global minimum**	**Initial location**
"Sphere" [Tang et al.]	100	$\mathcal{J}(X) = \sum_i^{n_p} i\, X_i^2$	(0,. . . ,0)	(10,. . . ,10)
Rosenbrock [Rosenbrock, 1960]	2	$\mathcal{J}(x,y) = (1-x)^2 + 100(y-x^2)^2$	(1.0,1.0)	(-1.2,1.0)
Beale [Jamil and Yang, 2013]	2	$\mathcal{J}(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	(3,0.5)	(0.0,0.0)
Three-hump [Branin, 1972]	2	$\mathcal{J}(x,y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$	(0,0,0,0)	(3.0,3.0)
Matyas [Hedar, 2007]	2	$\mathcal{J}(x,y) = 0.26(x^2+y^2) - 0.48xy$	(0.0,0.0)	(-9.0,9.0)
Booth [Jamil and Yang, 2013]	2	$\mathcal{J}(x,y) = (x+2y-7)^2 + (2x+y-5)^2$	(1.0,3.0)	(7.5,5.5)

Table 1.2: Collection of benchmark functions for optimization problem.

Industrial context

There are already developed optimization libraries that are the result of several years of development. I will quote in particular the optimized procedure from scipy and the toolbox of optimization from Matlab. Unfortunately, in the context of this thesis, these tools can not be used for compatibility reasons. It is required to have an inverse problem optimizer compatible with the massively parallel context already existing. There are also consistency reasons that require the code to deal with class and types developed in Total environment. So there is no other choice than to develop the optimization algorithm. These benchmarks are thus the opportunity to test and validate the optimization routines that have been incorporated into the industrial code.

First of all, we aim to determine the most efficient search direction strategy between the steepest descent, NLCG (FR,PR,DY,HS), and L-BFGS. For these tests, we propose to fix the line search strategy by employing backtracking using Armijo and Wolfe line search conditions. As said before, the efficiency of the method will be assessed by comparing the overall cost of the method. We will then quantify the number of iterations N_{it}, the number of cost function evaluations $N_{\mathcal{J}}$, and the number of gradient computations $N_{\nabla\mathcal{J}}$. We will consider that the

overall cost of the optimization N_{all} satisfies the following relation: $N_{all} = N_{\mathcal{J}} + 2N_{\nabla\mathcal{J}}$, since we supposed that the adjoint state method enables to compute the gradient with twice the cost of the cost function evaluation. For all these problems, we consider that the algorithm has converged if and only if we have: $|\mathcal{J}| < 10^{-9}$ or $\| \nabla\mathcal{J} \|_2 < 10^{-3}$.

Table 1.3 highlights the superiority of L-BFGS over the other search direction methods. This result was expected since we aim by this method gives access to Hessian information. It also shows that the steepest descent is most of the time much more expensive than other search directions. This is more particularly visible in the three first problems displayed in Table 1.3. For the Three-Hump problem, some figures are not available since the algorithm does not converge to the global minima. We then do not take into account these configurations in such a case.

Let us now consider the choice of the line search. We propose to evaluate the global computational cost the same way we did for the search direction, by fixing the search direction method to be L-BFGS and by changing the line search strategy using single Armijo condition, Armijo and Wolfe, or Armijo and Goldstein conditions. The objective is to know if it is worth computing one extra gradient during the line search for Wolfe criterion, knowing that one gradient evaluation is twice as expensive as computing the cost function. We display in Table 1.4 the results of the experiment. It is clearly shown that, despite the fact that the line search using Armijo + Wolfe condition can give the lowest number of iterations, the global cost of the method is way more expensive than the one using single Armijo or Armijo and Goldstein conditions. For the different experiments, adding Goldstein condition to avoid 'too short' step length seems to improve the efficiency of the method when using single Armijo condition, but both configurations seem to have a similar cost.

In this section concerning optimization, we made an inventory of methods that complied with FWI requirements. Indeed, when the number of parameters is very important and the evaluation of the function is time-consuming, not all optimization methods are reasonable, hence we focus on low memory Quasi-Newton search directions such as Non-linear conjugate gradient or the Limited-Memory BFGS. We also defined an adapted line search strategy that consists in using a backtracking algorithm to find a step length satisfying several conditions we have defined (Armijo, Wolfe, and Goldstein conditions). We then proceeded to a benchmark of these methods on various well-known optimization problems (see Table 1.2). The so-obtained results allowed us to validate our implementation, but also to define an optimization strategy adapted to problems whose gradient is obtained by adjoint state method.

As a consequence, to reconstruct physical parameters with FWI, we propose to use L-BFGS search direction and a backtracking line search based on single Armijo or Armijo + Goldstein conditions.

1.5 Conclusion

In this chapter, we defined the inverse problem in a geophysical context, and we described the modalities of a seismic acquisition campaign. These campaigns provoke, at the levels of seismic sources a disturbance that spreads in the subsoil. The wavefield that propagates is then recorded at the receivers levels. The inverse problem consists then in finding the physical parameters of the subsoil such as the simulations coincide 'at best' with the observed data. We can quantify this difference between observations and numerical results by defining a cost function. In this thesis, we chose to consider a least square cost function.

We have then introduced optimization algorithms. Global algorithms seem impossible to use in view of the large number of parameters to be reconstructed but also in view of the cost function that is revealed to be computationally challenging. For local optimization algorithms,

Problem	**Search direction**	**Line search**	N_{it}	$N_{\mathcal{J}}$	$N_{\nabla\mathcal{J}}$	N_{all}
"Sphere"	Steepest	Armijo + Wolfe	264	620	885	2390
	NLCG_FR	Armijo + Wolfe	120	279	418	1115
	NLCG_PR	Armijo + Wolfe	232	467	699	1865
	NLCG_HS	Armijo + Wolfe	82	189	271	731
	NLCG_DY	Armijo + Wolfe	89	179	268	715
	L-BFGS	Armijo + Wolfe	85	173	258	689
Rosenbrock	Steepest	Armijo + Wolfe	744	2113	2857	7827
	NLCG_FR	Armijo + Wolfe	122	421	543	1507
	NLCG_PR	Armijo + Wolfe	75	441	516	1473
	NLCG_HS	Armijo + Wolfe	48	211	259	729
	NLCG_DY	Armijo + Wolfe	70	163	233	629
	L-BFGS	Armijo + Wolfe	38	88	126	340
Beale	Steepest	Armijo + Wolfe	156	416	572	1560
	NLCG_FR	Armijo + Wolfe	59	118	177	472
	NLCG_PR	Armijo + Wolfe	64	221	286	793
	NLCG_HS	Armijo + Wolfe	36	118	154	426
	NLCG_DY	Armijo + Wolfe	25	56	81	218
	L-BFGS	Armijo + Wolfe	26	53	79	211
Three-hump	Steepest	Armijo + Wolfe	#	#	#	#
	NLCG_FR	Armijo + Wolfe	#	#	#	#
	NLCG_PR	Armijo + Wolfe	#	#	#	#
	NLCG_HS	Armijo + Wolfe	30	62	92	246
	NLCG_DY	Armijo + Wolfe	63	127	190	507
	L-BFGS	Armijo + Wolfe	18	39	57	153
Matyas	Steepest	Armijo + Wolfe	35	89	124	337
	NLCG_FR	Armijo + Wolfe	46	107	153	413
	NLCG_PR	Armijo + Wolfe	30	97	127	351
	NLCG_HS	Armijo + Wolfe	19	47	66	179
	NLCG_DY	Armijo + Wolfe	23	47	70	187
	L-BFGS	Armijo + Wolfe	5	14	19	52
Booth	Steepest	Armijo + Wolfe	29	63	92	247
	NLCG_FR	Armijo + Wolfe	26	53	79	211
	NLCG_PR	Armijo + Wolfe	28	62	89	240
	NLCG_HS	Armijo + Wolfe	28	61	89	239
	NLCG_DY	Armijo + Wolfe	19	40	59	158
	L-BFGS	Armijo + Wolfe	12	25	37	99

Table 1.3: Benchmark results for several search direction strategies using Armijo and Wolfe condition line search.

Problem	Search direction	Line search	N_{it}	$N_{\mathcal{J}}$	$N_{\nabla\mathcal{J}}$	N_{all}
Sphere	L-BFGS	Armijo	64	81	64	209
	L-BFGS	Armijo + Wolfe	85	173	258	689
	L-BFGS	Armijo + Goldstein	54	87	54	195
Rosenbrock	L-BFGS	Armijo	59	80	59	198
	L-BFGS	Armijo + Wolfe	38	88	126	340
	L-BFGS	Armijo + Goldstein	43	84	43	170
Beale	L-BFGS	Armijo	17	21	17	55
	L-BFGS	Armijo + Wolfe	26	53	79	211
	L-BFGS	Armijo + Goldstein	13	19	13	45
Three-hump	L-BFGS	Armijo	26	30	26	82
	L-BFGS	Armijo + Wolfe	18	39	57	153
	L-BFGS	Armijo + Goldstein	20	37	20	77
Matyas	L-BFGS	Armijo	15	19	15	49
	L-BFGS	Armijo + Wolfe	5	14	19	52
	L-BFGS	Armijo + Goldstein	5	12	5	22
Booth	L-BFGS	Armijo	21	25	21	67
	L-BFGS	Armijo + Wolfe	12	25	37	99
	L-BFGS	Armijo + Goldstein	18	31	18	67

Table 1.4: Benchmarks results for several line search strategies using L-BFGS search direction.

it is necessary to obtain the gradient of the cost function according to the parameters that we are trying to reconstruct. This gradient can be obtained by different ways, but the adjoint state method [Chavent, 2010] is recommended because it allows great flexibility (change of parameterization and cost function) while having a fixed calculation cost whatever the number of parameters to be reconstructed. This process is called Full Waveform Inversion (FWI) [Lailly, 1983, Tarantola, 1984].

In the last part of this chapter, we have discussed the optimization to be followed for efficient resolution. We defined different search direction and line search strategies. Following tests on well-known synthetic cases, we were able to benchmarks the cost of the different methods by taking into account that the gradient is computed using the adjoint state method. We then concluded that the FWI should be implemented by using Limited-memory BFGS [Nocedal, 1980] and a backtracking line search defined by the conditions of Armijo and Goldstein [Bonnans, 2006].

Now that we have defined the inverse problem and a guideline to solve it, we will devote the next chapter to define in a precise manner the forward problem from its continuous formulation to its discretization by Discontinuous Galerkin methods (DGm). DGm have several properties that make this space discretization very suitable in geophysical context and more particularly in terms of High Performance Computation.

Chapter 2

The Forward Problem

The objective of this chapter is to introduce the direct problem dealt with in the thesis. It is important to define the physical and mathematical context in which we will evolve in the continuation of the manuscript, in particular because it is driven by industrial constraints of development. All the work carried out during this thesis has been done in an acoustic environment. It is for this reason that this chapter aims to introduce acoustic wave equations and their discretizations. The core of this work is the resolution of the inverse problem in a time domain geophysical context using Discontinuous Galerkin method in space. A particular care will thus be taken, during this chapter, to define the discretized model in space and time.

In order to promote DG methods, which are mainly implemented using Lagrange nodal polynomial basis, we will investigate on the asset of Bernstein-Bézier polynomial basis. The latter brings interesting properties and allows reducing computational costs when using high order polynomials, which is exactly the framework in which DG methods have proved their interest in front of continuous methods (e.g. [Dumbser, 2005, Baldassari, 2009]).

In the majority of cases, and for obvious reasons of practicality, the DG method is implemented on the assumption that the physical parameters of the medium are constant per cell. This leads to an undersampling of the physical parameters and to a misrepresentation of the geological model. In this chapter, we will also investigate Weight Adjusted Discontinuous Galerkin method, which provides more information on the physical parameters within each cell. This technology changes the parameterization of the physical model, which in particular influences the inverse problem. It is therefore important to develop also this aspect of the direct problem in this chapter that introduces all tools used in the perspective of solving the inverse problem.

2.1 The Continuous Acoustic Problem

In this section, we will define the physical equation describing the propagation of mechanical waves. We will more precisely describe the continuous acoustic wave equation, which is the physical phenomenon we consider in this thesis.

2.1.1 The Acoustic wave equations

In fluid medium, an acoustic wave is a vibration that propagates information, from a point to another, like the circular fronts on the surface of water if you throw a rock. An acoustic wave is generated when the molecules that compose the propagating medium start to move, pushing each other. A wave is characterized by a fundamental frequency that, if it is too

high (ultrasound) or too low (infrasound), will correspond to a wave that cannot be heard by the human ear. We then speak of pressure or compressional waves or P waves to refer to the mode of propagation of acoustic waves (see Figure 2.1).

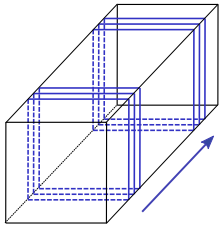


Figure 2.1: Illustration of a compressional wave.

An acoustic wave can be represented by a scalar field p which corresponds to the pressure field. We can add to it the velocity field $\boldsymbol{v}$, which is a vector with two or three components, depending on the dimension (2 or 3) of the propagation medium. The wavefield $(p, \boldsymbol{v})$ is the solution of the following system of equations that describes the wave propagation knowing a certain initial state $(p_0, \boldsymbol{v}_0)$ and a space and time perturbation defined by f:

$$\begin{cases} \dfrac{\partial p}{\partial t}(t,\boldsymbol{x}) + \kappa(\boldsymbol{x})\nabla\cdot\boldsymbol{v}(t,\boldsymbol{x}) = f(t,\boldsymbol{x})\,, & (2.1)\\ \rho(\boldsymbol{x})\dfrac{\partial \boldsymbol{v}}{\partial t}(t,\boldsymbol{x}) + \nabla p(t,\boldsymbol{x}) = 0\,, & (2.2)\\ p(t_0,\boldsymbol{x}) = p_0\,,\\ \boldsymbol{v}(t_0,\boldsymbol{x}) = \boldsymbol{v}_0\,. \end{cases}$$

These equations are set in the time-dependent domain described by the time variable t and the space variable $\boldsymbol{x}$. In this system, the propagation medium is depicted by the bulk modulus κ and the density ρ but can also be represented by the wavespeed c.

The wavefield and the physical model parameters are defined by the following units:

p = pressure ($\mathrm{kg.m^{-1}.s^{-2}}$);
$\boldsymbol{v}$ = velocity ($\mathrm{m.s^{-1}}$);
κ = bulk modulus ($\kappa = \rho c^2$) ($\mathrm{kg.m^{-1}.s^{-2}}$);
ρ = density ($\mathrm{kg.m^{-3}}$);
c = wavespeed of wave propagation ($\mathrm{m.s^{-1}}$).

This system of equations is obtained by combining the conservation of momentum and conservation of mass equations derived from the linearized Euler equations (see for instance [Dahlen and Tromp, 1998]). It is possible to get the pressure field p alone as the solution to the second-order wave equation:

$$\begin{cases} \dfrac{1}{\kappa}\dfrac{\partial^2 p}{\partial t^2} - \nabla\cdot\left(\dfrac{1}{\rho}\nabla p\right) = g\,, & (2.3)\\ p(t_0,\boldsymbol{x}) = p_0\,, \end{cases}$$

which is obtained after eliminating the velocity field of the previous system by time derivation of the equations.

The second order acoustic equation (2.3) is equivalent to the first order ones (2.1), (2.2) but as far as the computational costs are concerned, it is fundamentally different because the only

unknown managed here (p) is a scalar. In this thesis, we will principally work with the first order formulation. The propagation code under development in Total uses this formulation. This choice can be motivated by the fact that we do not need extra computation to get access to the velocity field which is stored in memory. For instance, this simplifies implementation of absorbing boundary condition, which require the both fields p and $\boldsymbol{v}$. It is also important to point out that the velocity field, $\boldsymbol{v}$, is also a key variable for inversion. It is therefore interesting to access it at the same time as the pressure field, p, without any post-processing.

2.1.2 The Acoustic model

Acoustic modeling is one of the pillars of the inversion technique that we are going to develop. Here, we will use the first order formulation (see (2.1), (2.2)), which gives us access to the pressure field and the velocity field in the same time. A distinction is made between the direct problem, which consists in simulating the wave field in a given medium, and the inverse problem, which associates a given wave field with a set of parameters to be retrieved in order to reconstruct the propagation medium.

The real physical problem is set in a half-space Ω_{inf} that is infinite in the depth direction. The problem lasts between the time $t = 0$ and the final time $t = T_f$. The surface Γ_1 corresponds to the boundary of the half-space Ω_{inf}, it is basically the surface of a portion of the Earth.

The propagation medium (see Figure 2.2) is then defined by physical parameters like the wavespeed, c, the bulk modulus κ, the density ρ, and a source field (in red) generates a perturbation inside the propagation medium represented by the pair (p, $\boldsymbol{v}$). In Figure 2.2, we have represented in blue receivers that can be used to record signals. These signals are the result of the reflections of the waves generated by the source, which indicate discontinuities in the medium.

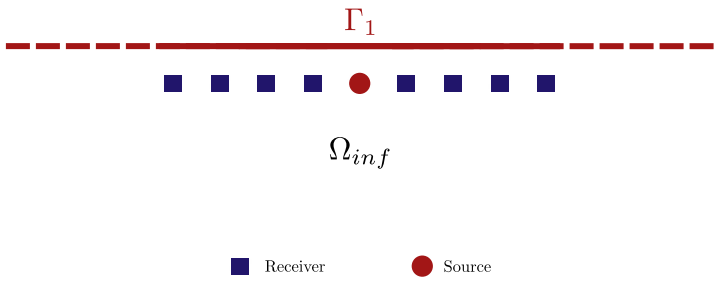


Figure 2.2: Semi-infinite propagation domain.

As far as numerical simulations are concerned, it is convenient to limit the computation to a truncated domain. Artificial boundaries are then introduced to make these boundaries as invisible as possible to the computed waves within the truncated domain. Thus, the solution computed within the bounded computational domain provides an accurate simulation of the real wavefield that would only be evaluated regionally within the bounded domain (see Figure 2.3). In the following of this manuscript, we denote by Ω the truncated domain and Γ_2 the artificial boundary, $\Gamma_2 = \partial\Omega \backslash \Gamma_1$.

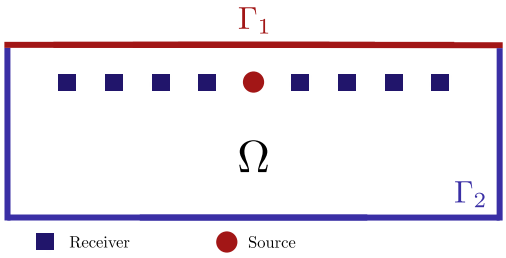


Figure 2.3: Truncated infinite domain.

Unsurprisingly, a difficulty of the approach is contained in the definition of the Boundary Condition that is imposed on artificial boundaries. The literature is quite rich on the subject and either in the time domain or in the frequency domain, these conditions can be divided into two large families that are the absorbing boundary conditions (ABC) and the perfectly matched layers (PML).

Absorbing Boundary Condition (ABC)

The idea of using ABCs has been promoted by Engquist and Majda [1977]). The ABC construction can be the result of different approaches. For example, in [Engquist and Majda, 1977], ABCs are constructed as an approximation of a transparent condition that conveys the perfect transmission of a wave through an artificial surface. In [Bayliss and Turkel, 1980], these conditions are derived from the expression of an analytical solution that can be written in the harmonic regime for particular geometries. Here, we use the simplest ABC for the time-dependent acoustic wave equation, following for instance [Engquist and Majda, 1977]. It reads:

$$\frac{\partial p}{\partial t}(t,\boldsymbol{x}) + c(\boldsymbol{x})\nabla p(t,\boldsymbol{x}).\boldsymbol{n} = 0\,, \quad x \in \Gamma_2\,, \tag{2.4}$$

where $\boldsymbol{n}$ stands for the unitary normal vector outwardly directed to Γ_2.

Knowing that at time $t = t_0$ the pressure and velocity fields are null, the physical acoustic wave in Ω is then solution of the following system:

$$\begin{cases} \dfrac{\partial p}{\partial t}(t,\boldsymbol{x}) + \kappa(\boldsymbol{x})\nabla\cdot\boldsymbol{v}(t,\boldsymbol{x}) = f\,, & \text{on } [t_0,T_f]\times\Omega\,,\\ \rho(\boldsymbol{x})\dfrac{\partial \boldsymbol{v}}{\partial t}(t,\boldsymbol{x}) + \nabla p(t,\boldsymbol{x}) = 0\,, & \text{on } [t_0,T_f]\times\Omega\,,\\ p(t,\boldsymbol{x}) = 0\,, & \text{on } [t_0,T_f]\times\Gamma_1,\\ p(t,\boldsymbol{x}) - c(\boldsymbol{x})\rho(\boldsymbol{x})\boldsymbol{v}(t,\boldsymbol{x}).\boldsymbol{n} = 0\,, & \text{on } [t_0,T_f]\times\Gamma_2,\\ p(t_0,\boldsymbol{x}) = 0\,, \quad \boldsymbol{v}(t_0,\boldsymbol{x}) = 0 & \text{on } \Omega. \end{cases}$$

The term f represents the pressure perturbation induced by the source field. The boundary Γ_1 is characterized by a free surface condition. This condition allows the ground surface to move freely into the upper part (which is air in geophysical exploration). In acoustics, the free surface is represented by a Dirichlet boundary condition on Γ_1 :

$$p|_{\Gamma_1} = 0 \qquad \text{acoustic free surface condition.}$$

ABCs are generally developed by assuming that possible reflected waves are generated by waves impinging the artificial surface with an incidence following a direction which is close to the one of normal vector (see for instance [Keys, 1985]). Obviously, their efficiency is thus

optimal for waves traveling in close proximity of the normal incidence. It has been shown that it is possible to widen the angle of incidence for which the ABC is efficient by increasing the order of approximation of the transparent condition. The ABC involves then higher order differential operators (see for instance [Givoli et al., 2006, Engquist and Majda, 1977, Higdon, 1986, Diaz, 2005]). Here, we chose ABCs because it turns out to be quite efficient for geophysical applications.

Perfectly Matched Layer (PML)

As previously mentioned, ABCs can become less efficient, especially in heterogeneous environments that allow waves to propagate in all directions and thus reduce the percentage of waves hitting the artificial boundary to an incidence close to the direction of the normal vector. A solution was first proposed to simulate electromagnetic waves in the time domain by Bérenger [1994] with the idea of using a system of perturbed equations written as the initial system of equations to be solved modified only in an area surrounding the initial field of study (see Figure 2.4). The perturbation of the system is made so that the waves propagating in the layer are sufficiently attenuated and thus, the possible waves reflected by the outer boundary of the layer cannot perturb the computed field in the inner domain. The difficulty of this approach lies in the construction of the perturbation which must be done in such a way that the waves propagating from the inner domain to the layer must be perfectly transmitted, as suggested by the generic terms of perfectly matched layers (PML). Like the work of Engquist and Majda [1977], Bérenger's work has been followed by numerous publications, extending PMLs to other systems, either in the time domain or in the frequency domain [Turkel and Yefet, 1998].

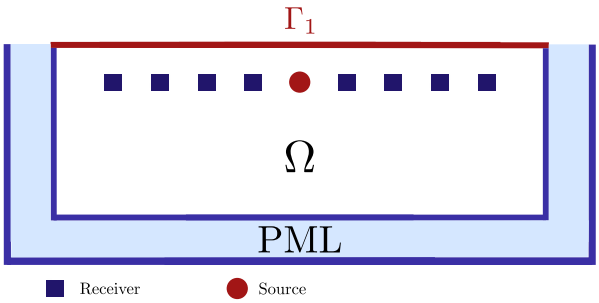


Figure 2.4: Illustration of PML surrounding the computational media.

For the work carried out in this manuscript, we will only consider ABC. This choice is motivated by the observation made by many users that PMLs could make DG formulations unstable as it was clearly demonstrated in [Citrain, 2019] which uses exactly the same DG solver as we do. Using an ABC seems to us the most appropriate choice to implement FWI in the time domain.

2.2 Discretized Acoustic Problem

In this section, we will focus on the discretization of these equations using time domain Discontinuous Galerkin methods (DGm). We will first describe what is DGm, then we will introduce the space discretization related to such methods, and we will finally describe the time discretization used in this thesis.

2.2.1 Introduction to Discontinuous Galerkin method

Discontinuous Galerkin Methods (DGm) appeared around 1970 for solving the neutron transport equations [Reed and Hill, 1973, Lesaint and Raviart, 1974]. It was then applied to hyperbolic equations by [Johnson et al., 1984] but it did not spread immediately into the geophysical community and more generally in numerical simulation of waves. It was in the 2000s that DGm was really launched thanks to an important promotion delivered by the works of Shu and Cockburn Cockburn et al. [2000], Cockburn [2001] followed by [Hesthaven and Warburton, 2007]. Namely, the review article [Cockburn, 2003], shows very well how DGm has been extensively used by many authors between 2000 and 2002.

DGm is a finite element method using discontinuous polynomial basis functions whose variational formulation is constructed element by element, each local formulation being glued with the others thanks to numerical fluxes defined on the edges or faces of the elements. Numerical fluxes makes it possible to see DGm as a method combining finite elements and finite volumes. With a DG approximation, the continuity of the solution at the interface between the elements is thus relaxed and the orders of the polynomial basis functions can be different from one element to another. We can see here that DGm allows to easily implement p-adaptivity (see Figure 2.5) which contributes a lot to obtain a numerical method with limited costs.

Discontinuous polynomial basis generates a bloc diagonal mass matrix, which is convenient to invert and encourage explicit time schemes. Furthermore, the fluxes make the element communicating only with its neighbors. Those two properties make it natural to implement the calculation in parallel. Each element works individually except for the fluxes which are only with the neighbors. Such fluxes prevent heavy communications which are required for high order Finite Difference and Finite Element Method and also give a block structure to any resulting matrix.

We have already talked about the p-adaptivity which is easily implemented with a DGm. The p-adaptivity could also easily be coupled with the h-adaptivity (see Figure 2.6) to further increase the accuracy of the method for reduced calculation costs. Indeed, as the formulation is written element by element, it is quite possible to mix elements of very different sizes to use even unstructured meshes with hanging nodes. It is therefore easy to carry out local anisotropic refinements in order to adapt the size of the elements to the physical parameters of the environment.

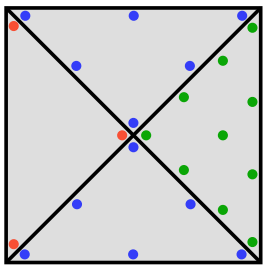


Figure 2.5: Illustration of p-adaptivity.

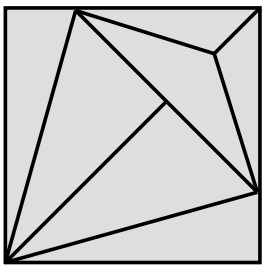
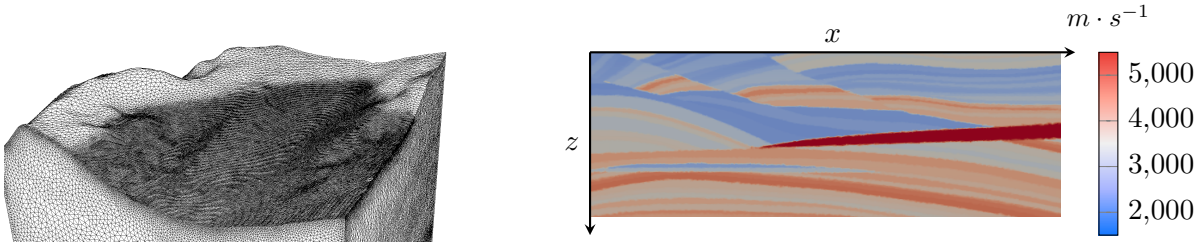


Figure 2.6: Illustration of h-adaptivity.

Why is it interesting to use a DGm to reconstruct a geological reservoir? To begin with, the surveyed domain is very heterogeneous, represented by a large number of parameters. For example, we know that a wavefield can only be correctly represented if the mesh is adapted to the characteristic wavelengths of the medium. Thus, the h-adaptivity is an asset of DGm giving it all the flexibility required to adapt the mesh size without difficulty. One can then adjust

the approximation orders according to the mesh size by also implementing p-adaptivity. This provides an ideal compromise between accuracy and computation costs. Next, a geological reservoir is a complex object whose discrete representation with regular grids (like those used with Finite Difference methods) can be difficult, especially in the presence of sharp zones or topography (Figure 2.7). The ability to work with unstructured grids is therefore another asset of DGm. Finally, the parallelization of DGm calculations is intrinsic, which is particularly important because a geological reservoir is generally large and its reconstruction requires a large number of simulations. We are therefore faced with a large-scale problem that has no chance of being solved sequentially. In particular, it has been shown that high-order DG approximations lead to a better scalability of the solver in a HPC environment. This is a major feature as compared to other space discretization methods (see [Shragge, 2014] for a comparison with Finite Difference method). This is mainly due to the fact that the matrices associated with most of the methods have a stencil which increases significantly with the order of approximation, requiring the interaction between distant elements. Then, the number of communications are strongly increased, which hampers the performance of the parallelization.



(a) 3D unconstrained meshes following topography.

(b) Sharp wavespeed model ($m.s^{-1}$).

Figure 2.7: Complex areas motivating unstructured mesh utilization.

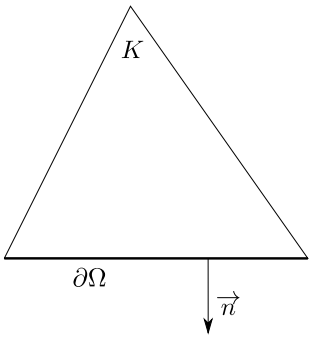
2.2.2 DGm Formulation for Acoustic wave equations

In this section, we will describe the discretization of the first order acoustic wave equations (2.1)-(2.2) based upon DG formulation. But before initiating any calculation, let us introduce some useful notations for the formulation used here.

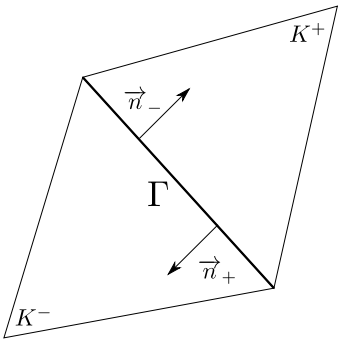
NOTATIONS

- Ω: the computational domain;
- $\partial\Omega$: the boundary of the computational domain Ω;
- dim: the space dimension ($dim = 2$ in 2D, $dim = 3$ in 3D);
- N: the polynomial order of approximation chosen;
- DoF: number of degrees of freedom per element, since DoF depends on the polynomial order approximation of the current element, we will also denote, when required, the number of degrees of freedom of an element of order N by $DoF(N)$;
- $\mathcal{T}_h$: triangulation of Ω;
- K: an element of $\mathcal{T}_h$ (triangles in 2D, tetrahedrons in 3D);
- N_e: the total number of elements in $\mathcal{T}_h$;

- $\boldsymbol{x}$: position vector in Ω of size dim ($\boldsymbol{x} = (x_1, x_2)^\top$ in 2D, $\boldsymbol{x} = (x_1, x_2, x_3)^\top$ in 3D);
- Γ: intersection between two elements or with the boundary of the domain;
- $\boldsymbol{n}$: the unitary normal vector outwardly directed to the domain it is associated to. It may be the normal associated to the boundary of Ω (Figure 2.8a) or the normal associated to an element K (Figure 2.8b).
- E_K: set of edges (in 2D) or faces (in 3D) of the element K.
- $\mathcal{B}_{int}$: interior set of edges (in 2D) or faces (in 3D) of $\mathcal{T}_h$ that belong to $\Omega \backslash \partial\Omega$.
- $\mathcal{B}_{ext}$: exterior set of edges (in 2D) or faces (in 3D) of $\mathcal{T}_h$ that belong to $\partial\Omega$.



(a) Normal vector in element/boundary configuration.



(b) Normal vector in element/element configuration.

Figure 2.8: 2D illustration of the outgoing normal whether the interface is external (a) or internal (b).

The DG formulation involves numerical fluxes that ensures the communication between internal elements. The main feature of DG approximation is to use piecewise continuous basis functions elementwise defined. That leads to introducing jump $[\![.]\!]$ and average $\{.\}$ of a quantity w (a scalar or a vector) defined at the interface Γ between two elements K^+ and K^-. The index $\pm$ denotes the value of w in the element $K^\pm$ at the interface Γ in $\mathcal{B}_{int}$. If w is a scalar, the jump is the vector defined by:

$$[\![w]\!] = w_+ \boldsymbol{n}^+ + w_- \boldsymbol{n}^- .$$

If $\boldsymbol{w}$ is a vector, the jumps is the scalar defined by:

$$[\![\boldsymbol{w}]\!] = \boldsymbol{w}_+ \cdot \boldsymbol{n}^+ + \boldsymbol{w}_- \cdot \boldsymbol{n}^- .$$

The average of w is defined by:

$$\{w\} = \frac{w_+ + w_-}{2},$$

whether w is a scalar or a vector.

Acoustic wave equation discretization

In order to understand DG formulation, let us consider the following acoustic wave equation system where the boundary condition has been simplified to be pressure free-surface boundary condition all over $\partial\Omega$:

$$\begin{cases} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{v} = \frac{1}{\kappa}f\,, & \text{in } \Omega\times[t_0,T_f], \\ \rho\frac{\partial \boldsymbol{v}}{\partial t} + \nabla p = 0\,, & \text{in } \Omega\times[t_0,T_f], \\ p = 0, & \text{on } \partial\Omega\times[t_0,T_f]\,, \\ p(t_0,\boldsymbol{x}) = 0\,, \quad u(t_0,\boldsymbol{x}) = 0, & \text{in } \Omega\,. \end{cases} \tag{2.5}$$

Theorem 2.2.1 (Well-posedness)**.** Let κ and ρ be positive piecewise constant functions on Ω. Let f be given in $L^2([t_0,T_f];L^2(\Omega))$. Then (2.5) admits a unique solution $(p,\boldsymbol{v})$ in $L^2([t_0,T_f];H_0^1(\Omega))\times L^2([t_0,T_f];H(div,\Omega))$.

This results can be established with Hille-Yosida theory (see for instance Dautray and Lions [2012]). We denote by $H_0^1(\Omega)$ the standard Sobolev space and $H(div,\Omega) = \{\boldsymbol{w}\in L^2(\Omega)^{dim}, \nabla\cdot\boldsymbol{w}\in L^2(\Omega)\}$.

Let $\mathcal{T}_h$ be a triangulation of Ω. Let K be one element of $\mathcal{T}_h$. DG formulations start with local variational problems that are defined by:
find $(p,\boldsymbol{v})$ such that for any $(q,\boldsymbol{w})\in L^2(\Omega)\times L^2(\Omega)^{dim}$ satisfying $q_{|K}\in H^1(K)$, $\boldsymbol{w}_{|K}\in (H^1(K))^{dim}$, we have:

$$\begin{cases} \displaystyle\int_K \left(\frac{1}{\kappa}\frac{\partial p}{\partial t}q + \nabla\cdot\boldsymbol{v}q\right)dK = \int_K \frac{1}{\kappa}fq dK, \\ \displaystyle\int_K \left(\rho\frac{\partial \boldsymbol{v}}{\partial t}.\boldsymbol{w} + \nabla p.\boldsymbol{w}\right)dK = 0\,. \end{cases} \tag{2.6}$$

To obtain (2.6), we multiply by q and $\boldsymbol{w}$ the two first equations of (2.5) and we integrate on the element K.

In order to construct a global DG formulation, communication must be established between the elements that comprise the triangulation covering the computational domain. A natural way to introduce these communications is to carry out an integration by part of the local problem that brings out boundary terms shared in the neighboring elements (see Figure 2.8b). We then have, at element level:

$$\begin{cases} \displaystyle\int_K \frac{1}{\kappa}\frac{\partial p}{\partial t}q dK - \int_K \boldsymbol{v}.\nabla q dK + \int_{\partial K}\boldsymbol{v}.\boldsymbol{n}q ds = \int_K \frac{1}{\kappa}fq dK, \\ \displaystyle\int_K \rho\frac{\partial \boldsymbol{v}}{\partial t}.\boldsymbol{w} dK - \int_K p\nabla\cdot\boldsymbol{w} dK + \int_{\partial K} p\boldsymbol{w}.\boldsymbol{n} ds = 0\,. \end{cases}$$

Remark. Here both $\int_{\partial K}$ are in fact duality products $(H^{-\frac{1}{2}}(\partial K), H^{\frac{1}{2}}(\partial K))$.

Then, the global formulation is obtained by summing over all the elements K with a specific treatment of boundary terms. To do this, we have to identify the internal edges (faces) which correspond to the interfaces between two different elements and the external edges (faces) which bear the boundary conditions. Let $\mathcal{B}_{int}$ be the set of internal edges (faces) and $\mathcal{B}_{ext}$ be the set of external edges (faces). We then have:

$$\begin{cases} \int_\Omega \frac{1}{\kappa}\frac{\partial p}{\partial t} q dK - \int_\Omega \boldsymbol{v}.\nabla q dK + \sum_K \int_{\partial K} \boldsymbol{v}\cdot\boldsymbol{n} q ds = \int_\Omega \frac{1}{\kappa} f q dK, \\ \int_\Omega \rho \frac{\partial \boldsymbol{v}}{\partial t}.\boldsymbol{w} dK - \int_\Omega p\nabla\cdot\boldsymbol{w} ds + \sum_K \int_{\partial K} p\boldsymbol{w}\cdot\boldsymbol{n} ds = 0\,, \end{cases}$$

and

$$\sum_K \int_{\partial K} \boldsymbol{v}\cdot\boldsymbol{n} q ds = \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma \left(\boldsymbol{v}^+\cdot\boldsymbol{n}^+ q^+ + \boldsymbol{v}^-\cdot\boldsymbol{n}^- q^-\right) ds + \sum_{\Gamma\in\mathcal{B}_{ext}} \int_\Gamma \boldsymbol{v}\cdot\boldsymbol{n} q ds\,,$$

along with

$$\sum_K \int_{\partial K} p\boldsymbol{w}.\boldsymbol{n} ds = \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma \left(p^+\boldsymbol{w}^+\cdot\boldsymbol{n}^+ + p^+\boldsymbol{w}^+\cdot\boldsymbol{n}^+\right) ds + \sum_{\Gamma\in\mathcal{B}_{ext}} \int_\Gamma p\boldsymbol{w}\cdot\boldsymbol{n} ds\,,$$

where $q \in \mathcal{Q}$ and $\boldsymbol{w} \in \boldsymbol{\mathcal{W}}$, the functional spaces being defined such that:

$$\begin{aligned} \mathcal{Q} &= \{q \in L^2(\Omega),\, q_{|_K} \in H^1(K)\}\,, \\ \boldsymbol{\mathcal{W}} &= \{\boldsymbol{w} \in (L^2(\Omega))^2,\, \boldsymbol{w}_{|_K} \in (H^1(K))^2\}\,. \end{aligned}$$

Since we are using free-surface condition on the pressure field on $\partial\Omega$, we have:

$$\sum_{\partial K\in\mathcal{B}_{ext}} \int_{\partial K} p\boldsymbol{w}\cdot\boldsymbol{n} ds = 0.$$

Regarding the terms $\Gamma \in \mathcal{B}_{int}$, we can rewrite both equations in terms of jumps:

$$\sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma (\boldsymbol{v}^+\cdot\boldsymbol{n}^+ q^+ + \boldsymbol{v}^-\cdot\boldsymbol{n}^- q^-) ds = \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma [\![q\boldsymbol{v}]\!] ds\,,$$

and

$$\sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma (p^+\boldsymbol{w}^+\cdot\boldsymbol{n}^+ + p^-\boldsymbol{w}^-\cdot\boldsymbol{n}^-) ds \;= \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma [\![p\boldsymbol{w}]\!] ds.$$

We thus end up with the following global DG variational formulation:

$$\begin{cases} \int_\Omega \frac{1}{\kappa}\frac{\partial p}{\partial t} q dK - \int_\Omega \boldsymbol{v}.\nabla q dK + \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma [\![q\boldsymbol{v}]\!] ds + \sum_{\Gamma\in\mathcal{B}_{ext}} \int_\Gamma q\boldsymbol{v}\cdot\boldsymbol{n} ds = \int_\Omega \frac{1}{\kappa} f q dK, \\ \int_\Omega \rho \frac{\partial \boldsymbol{v}}{\partial t}.\boldsymbol{w} dK - \int_\Omega p\nabla\cdot\boldsymbol{w} ds + \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma [\![p\boldsymbol{w}]\!] ds = 0\,. \end{cases} \tag{2.7}$$

Then, it is convenient to rewrite jump terms in (2.7) in order to exhibit interesting properties. For that purpose, we state:

Property 1. Let α be a scalar and $\boldsymbol{w}$ be a vector. Then on $\Gamma \in \mathcal{B}_{int}$:

$$[\![\alpha\boldsymbol{w}]\!] = \{\alpha\}[\![\boldsymbol{w}]\!] + \{\boldsymbol{w}\}\cdot[\![\alpha]\!]$$

Proof: For $\Gamma \in \mathcal{B}_{int}$:

$$\{\alpha\}[\![\boldsymbol{w}]\!] + \{\boldsymbol{w}\} \cdot [\![\alpha]\!] = \frac{1}{2}\left[(\alpha^+ + \alpha^-)(\boldsymbol{w}^+ \cdot \boldsymbol{n}^+ + \boldsymbol{w}^- \cdot \boldsymbol{n}^+) + (\boldsymbol{w}^+ + \boldsymbol{w}^-) \cdot (\alpha^+ \boldsymbol{n}^+ + \alpha^- \boldsymbol{n}^+)\right]$$

$$= \frac{1}{2}\left[(\alpha^+ \boldsymbol{w}^+ \cdot 2\boldsymbol{n}^+ + \alpha^+ \boldsymbol{w}^- \cdot (\boldsymbol{n}^+ + \boldsymbol{n}^+) + \alpha^- \boldsymbol{w}^+ \cdot (\boldsymbol{n}^+ + \boldsymbol{n}^+) + \alpha^- \boldsymbol{w}^- \cdot 2\boldsymbol{n}^+)\right]$$

$$= \alpha^+ \boldsymbol{w}^+ \cdot \boldsymbol{n}^+ + \alpha^- \boldsymbol{w}^- \cdot \boldsymbol{n}^+$$

because $\boldsymbol{n}^+ = -\boldsymbol{n}^+$ by definition, hence the two last terms cancel each other out. We then obtain:

$$\{\alpha\}[\![\boldsymbol{w}]\!] + \{\boldsymbol{w}\}.[\![\alpha]\!] = [\![\alpha \boldsymbol{w}]\!]$$

We have then completed the proof of Property 1 for $\Gamma \subset \mathcal{B}_{int}$. Then by injecting the relationship of Property 1 in (2.7) we get:

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p}{\partial t} q dK - \int_\Omega \boldsymbol{v}.\nabla q dK + \sum_{\Gamma \in \mathcal{B}_{int}} \int_\Gamma \left(\{q\}[\![\boldsymbol{v}]\!] + \{\boldsymbol{v}\}.[\![q]\!]\right) ds + \sum_{\Gamma \in \mathcal{B}_{ext}} \int_\Gamma q\boldsymbol{v} \cdot \boldsymbol{n} ds = \int_\Omega \frac{1}{\kappa} f q dK, \\ \displaystyle\int_\Omega \rho \frac{\partial \boldsymbol{v}}{\partial t}.\boldsymbol{w} dK - \int_\Omega p \nabla \cdot \boldsymbol{w} dK + \sum_{\Gamma \in \mathcal{B}_{int}} \int_\Gamma \left(\{p\}[\![\boldsymbol{w}]\!] + \{\boldsymbol{w}\}.[\![p]\!]\right) ds = 0\,, \end{cases} \tag{2.8}$$

which displays a sort of symmetry in the interface terms and more importantly a way to separate the solution trace from the test function one. By this way, it is possible to inject one regularity of the solution, that is:

$$\begin{aligned} &[\![p]\!] = 0 \text{ on } \mathcal{B}_{int} \cup \mathcal{B}_{ext}, && \text{since } \ p(t,.) \in H_0^1(\Omega) \subset H^1(\Omega)\,, \\ &[\![\boldsymbol{v}]\!] = 0 \text{ on } \mathcal{B}_{int}, && \text{since } \ \boldsymbol{v}(t,.) \in H(div, \Omega)\,. \end{aligned}$$

Hence, (2.8) changes to:

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p}{\partial t} q dK - \int_\Omega \boldsymbol{v}.\nabla q dK + \sum_{\Gamma \subset \mathcal{B}_{int}} \int_\Gamma \{\boldsymbol{v}\}.[\![q]\!] ds + \sum_{\Gamma \subset \mathcal{B}_{ext}} \int_\Gamma q\boldsymbol{v} \cdot \boldsymbol{n} ds = \int_\Omega \frac{1}{\kappa} f q ds, \\ \displaystyle\int_\Omega \rho \frac{\partial \boldsymbol{v}}{\partial t}.\boldsymbol{w} dK - \int_\Omega p \nabla \cdot \boldsymbol{w} dK + \sum_{\Gamma \subset \mathcal{B}_{int}} \int_\Gamma \{p\}[\![\boldsymbol{w}]\!] ds = 0\,. \end{cases} \tag{2.9}$$

Then, let us introduce the approximation spaces $\mathcal{Q}_h^N$ and $\boldsymbol{\mathcal{W}_h^N}$ that are defined by:

$$\mathcal{Q}_h^N = \{q \in L^2(\Omega),\, q|_K \in P^N(K),\, \forall K \in \mathcal{T}_h\,\},$$

$$\boldsymbol{\mathcal{W}_h^N} = \{\boldsymbol{w} \in L^2(\Omega)^{dim},\, \boldsymbol{w}|_K \in (P^N(K))^{dim},\, \forall K \in \mathcal{T}_h\},$$

where $P^N(K)$ denotes the set of polynomials of order N defined on an element K. Then, we can define a polynomial approximation of p and $\boldsymbol{v}$ that we seek in the form:

$$p_h = \sum_{K \in \mathcal{T}_h} {p_h}^K, \qquad \boldsymbol{v_h} = \sum_{K \in \mathcal{T}_h} {\boldsymbol{v_h}}^K$$

where ${p_h}^K$ and ${\boldsymbol{v_h}}^K$ are defined as:

$$p_h{}^K = \sum_{j=1}^{DoF} P_{h_j}{}^K \varphi_j^K\,, \qquad (\boldsymbol{v_h}{}^K)_d = \sum_{j=1}^{DoF} \boldsymbol{V_h}_{dj}{}^K \varphi_j^K\,, \quad \text{for } d = 1 \text{ to } dim\,,$$

and $\varphi_j^K \in P^N(K)$.

The number of Degrees of Freedom per elements (DoF) is determined by the polynomial order (N) and the space dimension (dim) as reminded in Table 2.1.

$DoF(N, dim)$	$dim = 1$	$dim = 2$	$dim = 3$
DoF	$N+1$	$\frac{(N+2)(N+1)}{2}$	$\frac{(N+3)(N+2)(N+1)}{6}$

Table 2.1: DoF per element as a function of dim and N .

Here, it is worth noting that once the continuous solutions in (2.9) are substitute with the approximate solutions, there is no reason to have p_h and $\boldsymbol{v_h}$ both continuous at the interface between each element. Hence, to guarantee the approximate solutions keep this property, we introduce penalization terms led by penalization parameters $\boldsymbol{\alpha}$, β, $\boldsymbol{\gamma}$ and δ.

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p_h}{\partial t} q dK - \int_\Omega \boldsymbol{v_h}.\nabla q dK + \sum_{\Gamma\in\mathcal{B}_{int}} \int_\Gamma (\{\boldsymbol{v_h}\}.[\![q]\!] + \boldsymbol{\alpha}[\![\boldsymbol{v_h}]\!].[\![q]\!] + \beta[\![p_h]\!].[\![q]\!])\, ds \\ \qquad\qquad + \displaystyle\sum_{\Gamma\in\mathcal{B}_{ext}} \int_\Gamma q\boldsymbol{v_h}\cdot\boldsymbol{n} ds = \int_\Omega \frac{1}{\kappa} f q dK, \\ \displaystyle\int_\Omega \rho\frac{\partial \boldsymbol{v_h}}{\partial t}.\boldsymbol{w} dK - \int_\Omega p_h \nabla\cdot\boldsymbol{w} dK + \sum_{\Gamma\in\mathcal{B}_{int}} \int_{\partial K} (\{p_h\}[\![\boldsymbol{w}]\!] + \boldsymbol{\gamma}.[\![p_h]\!][\![\boldsymbol{w}]\!] + \delta[\![\boldsymbol{v_h}]\!][\![\boldsymbol{w}]\!])\, ds = 0\,. \end{cases} \tag{2.10}$$

Hesthaven and Warburton [2007], proposed, once the penalization is introduced, to go back to the strong variational formulation of the volume equations. It requires to apply an integration by part which allows to only consider the jump of the approximate solution. This way of formulating the DG solution is then given by:

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p_h}{\partial t} q dK + \int_\Omega \nabla\cdot\boldsymbol{v_h} q dK - \sum_{\Gamma\subset\mathcal{B}_{int}} \int_\Gamma ([\![\boldsymbol{v_h}]\!]\{q\} + \boldsymbol{\alpha}[\![\boldsymbol{v_h}]\!].[\![q]\!] + \beta[\![p_h]\!].[\![q]\!])\, ds = \int_\Omega \frac{1}{\kappa} f q dK, \\ \displaystyle\int_\Omega \rho\frac{\partial \boldsymbol{v_h}}{\partial t}.\boldsymbol{w} dK + \int_\Omega \nabla p_h.\boldsymbol{w} dK - \sum_{\Gamma\subset\mathcal{B}_{int}} \int_\Gamma ([\![p_h]\!].\{\boldsymbol{w}\} + \boldsymbol{\gamma}.[\![p_h]\!][\![\boldsymbol{w}]\!] + \delta[\![\boldsymbol{v_h}]\!][\![\boldsymbol{w}]\!])\, ds \\ \qquad\qquad - \displaystyle\sum_{\Gamma\subset\mathcal{B}_{ext}} \int_\Gamma p_h\boldsymbol{w}\cdot\boldsymbol{n} ds = 0\,. \end{cases} \tag{2.11}$$

Remark. Note that this formulation (2.11) is mathematically equivalent to (2.10) but is computationally different. Indeed, it requires to only compute jumps of the approximate wavefield which simplifies the implementation, contrary to (2.10) where the mean value is also necessary.

With free-surface condition, we can adapt the jump and the mean formula on $\Gamma \in \mathcal{B}_{ext}$ to be:

$$[\![p_h]\!] = p_h\boldsymbol{n}\,, \qquad [\![\boldsymbol{v_h}]\!] = 0 \qquad \{p_h\} = p_h \qquad \{\boldsymbol{v_h}\} = \boldsymbol{v_h}\,.$$

With this notation, we can have the general expression of (2.11):

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p_h}{\partial t} q dK + \int_\Omega \nabla\cdot \boldsymbol{v_h} q dK - \sum_\Gamma \int_\Gamma \left([\![\boldsymbol{v_h}]\!]\{q\} + \boldsymbol{\alpha}[\![\boldsymbol{v_h}]\!].[\![q]\!] + \beta[\![p_h]\!].[\![q]\!]\right) ds = \int_\Omega \frac{1}{\kappa} f q dK, \\ \displaystyle\int_\Omega \rho \frac{\partial \boldsymbol{v_h}}{\partial t}.\boldsymbol{w} dK + \int_\Omega \nabla p_h.\boldsymbol{w} dK - \sum_\Gamma \int_\Gamma \left([\![p_h]\!].\{\boldsymbol{w}\} + \boldsymbol{\gamma}.[\![p_h]\!][\![\boldsymbol{w}]\!] + \delta[\![\boldsymbol{v_h}]\!][\![\boldsymbol{w}]\!]\right) ds = 0\,. \end{cases} \tag{2.12}$$

Note that in this expression, the penalization parameters $\boldsymbol{\alpha}$ and δ are zero if $\Gamma \in Bext$.

By choosing $q = \varphi_i^K$ and $\boldsymbol{w} = \varphi_i^K \boldsymbol{e}_d$, where $\boldsymbol{e}_d$ is the canonical vector of size dim that is null except 1 at the d^{th} component (for $d = 1$ to dim), we explore the whole space of approximation ($\mathcal{Q}_h^N$ and $\boldsymbol{\mathcal{W}_h^N}$), and (2.12) changes to a system of $(dim+1) \times DoF \times N_e$ equations.

We then have the following expression for jumps and averages of the test functions:

$$\begin{aligned} \{q\} = \{\varphi_i^K\} &= \frac{1}{2}{\varphi_i^K}_{|\partial K}\,, \\ [\![q]\!] = [\![\varphi_i^K]\!] &= {\varphi_i^K}_{|\partial K}\boldsymbol{n}\,, \\ \{\boldsymbol{w}\} = \{\varphi_i^K\} &= \frac{1}{2}{\varphi_i^K}_{|\partial K}\boldsymbol{e}_d && \text{for } d = 1 \text{ to } dim\,, \\ [\![\boldsymbol{w}]\!] = [\![\varphi_i^K]\!] &= {\varphi_i^K}_{|\partial K}\boldsymbol{e}_d.\boldsymbol{n} && \text{for } d = 1 \text{ to } dim. \end{aligned}$$

System (2.11) can thus be written as a system composed of $(dim+1)$ equations:

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p_h}{\partial t}\varphi_i^K dK + \sum_{d=1}^{dim}\int_\Omega \frac{\partial v_{hd}}{\partial x_d}\varphi_i^K dK \\ \displaystyle -\sum_\Gamma \int_\Gamma \left(\frac{1}{2}[\![\boldsymbol{v_h}]\!] + \boldsymbol{\alpha}[\![\boldsymbol{v_h}]\!].\boldsymbol{n} + \beta[\![p_h]\!].\boldsymbol{n}\right)\varphi_i^K ds = \int_\Omega \frac{1}{\kappa} f\varphi_i^K dK, \\ \displaystyle\int_\Omega \rho\frac{\partial v_{hd}}{\partial t}\varphi_i^K dK + \int_\Omega \frac{\partial p_h}{\partial x_d}\varphi_i^K dK \\ \displaystyle -\sum_\Gamma \int_\Gamma \left(\frac{1}{2}[\![p_h]\!] + \boldsymbol{\gamma}.[\![p_h]\!]\boldsymbol{n} + \delta[\![\boldsymbol{v_h}]\!]\boldsymbol{n}\right).\varphi_i^K \boldsymbol{e}_d ds = 0\,, \quad \text{for } d = 1 \text{ to } dim. \end{cases} \tag{2.13}$$

It is then convenient to introduce the numerical fluxes:

$$\begin{aligned} \mathcal{F}_p{}^\Gamma(p_h, \boldsymbol{v_h}) &= \frac{1}{2}[\![\boldsymbol{v_h}]\!] + \boldsymbol{\alpha}[\![\boldsymbol{v_h}]\!].\boldsymbol{n} + \beta[\![p_h]\!].\boldsymbol{n} \\ \mathcal{F}_{vd}^\Gamma(p_h, \boldsymbol{v_h}) &= \left(\frac{1}{2}[\![p_h]\!] + \boldsymbol{\gamma}.[\![p_h]\!]\boldsymbol{n} + \delta[\![\boldsymbol{v_h}]\!]\boldsymbol{n}\right).\boldsymbol{e}_d \end{aligned}$$

so that system (2.13) takes the form:

$$\begin{cases} \displaystyle\int_\Omega \frac{1}{\kappa}\frac{\partial p_h}{\partial t}\varphi_i^K dK + \sum_{d=1}^{dim}\int_\Omega \frac{\partial v_{hd}}{\partial x_d}\varphi_i^K dK - \sum_\Gamma \int_\Gamma \mathcal{F}_p{}^\Gamma(p_h, \boldsymbol{v_h})\varphi_i^K ds = \int_\Omega \frac{1}{\kappa} f\varphi_i^K dK, \\ \displaystyle\int_\Omega \rho\frac{\partial v_{hd}}{\partial t}\varphi_i^K dK + \int_\Omega \frac{\partial p_h}{\partial x_d}\varphi_i^K dK - \sum_\Gamma \int_\Gamma \mathcal{F}_{vd}^\Gamma(p_h, \boldsymbol{v_h})\varphi_i^K ds = 0\,, \quad \text{for } d = 1 \text{ to } dim. \end{cases}$$

Obviously, this system is solved elementwise, this is the essence of DGm. Hence, we consider elemental systems of the form:

$$\begin{cases} \displaystyle\int_K (\frac{1}{\kappa}\frac{\partial}{\partial t}\sum_{j=1}^{DoF} \boldsymbol{P_h}_j^K \varphi_j^K \varphi_i^K + \sum_{d=1}^{dim}\frac{\partial}{\partial x_d}\sum_{j=1}^{DoF} \boldsymbol{V_h}_{dj}^K \varphi_j^K \varphi_i^K) dK \\ \displaystyle + \sum_\Gamma \int_\Gamma \mathcal{F}_p{}^\Gamma(p_h, \boldsymbol{v_h})\varphi_i^K ds = \int_K \frac{1}{\kappa}\sum_{j=1}^{DoF} \boldsymbol{F_h}_j^K \varphi_j^K \varphi_i^K dK, \\ \displaystyle\int_K (\rho\frac{\partial}{\partial t}\sum_{j=1}^{DoF} \boldsymbol{V_h}_{dj}^K \varphi_j^K \varphi_i^K + \frac{\partial}{\partial x_d}\sum_{j=1}^{DoF} \boldsymbol{P_h}_j^K \varphi_j^K \varphi_i^K) dK \\ \displaystyle + \sum_\Gamma \int_\Gamma \mathcal{F}_{vd}^\Gamma(p_h{}^K, \boldsymbol{v_h}{}^K)\varphi_i^K ds = 0 \qquad (\text{for } d = 1 \text{ to } dim)\,. \end{cases} \tag{2.14}$$

To simplify (2.14), we introduce the following matricial operators:

$$\begin{aligned} [M]_{i,j}^K &= \int_K \varphi_i^K \varphi_j^K dK, && \text{Local Mass matrix on } K, \\ [S_{x_d}]_{i,j}^K &= \int_K \varphi_i^K \frac{\partial \varphi_j^K}{\partial x_d} dK, && \text{Local Stiffness matrix on } K, \\ [M_\Gamma]_{i,j}^K &= \int_\Gamma \varphi_i^K \varphi_j^K ds, && \text{Local surface Mass matrix on } \Gamma, \\ \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}{}^\Gamma \quad &\text{and} \quad \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}d}^\Gamma, && \text{The flux vectors of size } DoF, \end{aligned}$$

where we consider:

$$\mathcal{F}_p{}^\Gamma(p_h, \boldsymbol{v_h}) = \sum_{j=1}^{DoF} \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}_j}^\Gamma \varphi_j^K\,,$$

$$\mathcal{F}_{vd}^\Gamma(p_h, \boldsymbol{v_h}) = \sum_{j=1}^{DoF} \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}d_j}^\Gamma \varphi_j^K.$$

We also define the source term such that:

$$f|_K = \sum_{j=1}^{DoF} \boldsymbol{F_h}_j^K \varphi_j^K\,.$$

Remark. The flux computation depends on the polynomial approximation order of the two elements apart from the edge Γ. If those elements do not have the same polynomial order approximation, some projections to the local space are required. We do not describe the calculations of $\bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}$ and $\bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}d}$, but we keep in mind that the flux terms allow the communications between elements whether they have the same order or not. This is the idea of p-adaptivity.

These operators lead to rewrite (2.14) into the simplified matricial system that follows:

$$\begin{cases} \displaystyle\frac{1}{\kappa}M^K \frac{\partial \boldsymbol{P_h}{}^K}{\partial t} + \sum_{d=1}^{dim} S_{x_d}^K \boldsymbol{V_h}_d{}^K + \sum_\Gamma M_\Gamma^K \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}{}^\Gamma = \frac{1}{\kappa}M^K \boldsymbol{F_h}{}^K, \\ \displaystyle\rho M^K \frac{\partial \boldsymbol{V_h}_d^K}{\partial t} + S_{x_d}^K \boldsymbol{P_h}{}^K + \sum_\Gamma M_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}d}^\Gamma = 0 \qquad (\text{for } d = 1 \text{ to } dim)\,. \end{cases} \tag{2.15}$$

Concerning the fluxes, in the industrial DG solver used in this thesis, the choice has been done to implement the acoustic upwind fluxes as described by Hesthaven and Warburton [2007]. They are:

$$\mathcal{F}_p^{\Gamma}(p_h, \boldsymbol{v_h}) = \frac{1}{2}(\llbracket \boldsymbol{v_{hi}} \rrbracket + \tau_p \llbracket p_{hi} \rrbracket . \boldsymbol{n}), \qquad \text{Upwind fluxes,} \tag{2.16}$$

$$\mathcal{F}_{vd}^{\Gamma}(p_h, \boldsymbol{v_h}) = \frac{1}{2}(\llbracket p_{hi} \rrbracket + \tau_u \llbracket \boldsymbol{v_{hi}} \rrbracket \boldsymbol{n}) . \boldsymbol{e}_d, \qquad \text{Upwind fluxes,} \tag{2.17}$$

where $\tau_p = \frac{1}{\{\rho c\}}$ and $\tau_u = \{\rho c\}$. Those fluxes are obtained by setting the penalization parameters as follows:

$$\boldsymbol{\alpha} = 0, \qquad \beta = \frac{1}{2}\tau_p,$$
$$\boldsymbol{\gamma} = 0, \qquad \delta = \frac{1}{2}\tau_v,$$

The choice of the penalization parameters is important for the modeling process. Without penalization, the simulation is polluted by spurious numerical modes. Adding penalization terms affects the conditioning of the stiffness matrix that may reduce the computational time step while solving the equation in time domain using explicit time schemes [Ventimiglia]. Here we chose the upwind fluxes defined in [Hesthaven and Warburton, 2007]. It has been shown that such fluxes are dissipative giving smaller jumps in between two elements. However, a smoother solution is not necessarily more accurate but this smoothness damp the spurious numerical mode. For an interesting study concerning the choice of the penalization parameters we refer to [Ainsworth et al., 2006].

We end up with a symmetrical system (2.15) where the unknowns are the coefficients p_h and $\boldsymbol{v_h}_d$ for $d = 1$ to dim. It is then possible to write a synthetic that summed up all those equations:

$$\begin{cases} M\dfrac{\partial}{\partial t}\boldsymbol{U} + S\boldsymbol{U} = M\boldsymbol{F}\,, \\ \boldsymbol{U}_K = (\boldsymbol{P_h}^K, \boldsymbol{V_h}_1^K, \ldots, \boldsymbol{V_h}_{dim}^K)^\top\,, \\ \boldsymbol{U} = (\boldsymbol{U}_{K_1}, \boldsymbol{U}_{K_2}, \ldots, \boldsymbol{U}_{K_{N_e}})^\top\,, \\ \boldsymbol{F}_K = (\boldsymbol{F_h}^K, 0, \ldots, 0)^\top \qquad \text{by supposing that: } f|_K \approx f_h^K = \displaystyle\sum_{i=1}^{DoF} \boldsymbol{F_h}_i^K \varphi_i^K\,, \\ \boldsymbol{F} = (\boldsymbol{F}_1, \boldsymbol{F}_2, \ldots, \boldsymbol{F}_{N_e})^\top\,, \end{cases}$$

Matrix M represents the global mass matrix on the entire domain Ω. S is the global stiffness matrix that contains the volume and surface terms (fluxes).

Remark. Since M is block-diagonal, we can consider without any additional difficulty the formulation:

$$\frac{\partial}{\partial t}\boldsymbol{U} + M^{-1}S\boldsymbol{U} = \boldsymbol{F}\,, \tag{2.18}$$

which is well-suited for explicit time integration.

With DGm, the Mass operator (M) is then easier to invert than the one obtained using classical Finite Element Methods:

$$M = \begin{pmatrix} \bar{M}^{K_1} & & & & \\ & \ddots & & & \\ & & \bar{M}^{K_i} & & \\ & & & \ddots & \\ & & & & \bar{M}^{K_{Ne}} \end{pmatrix}, \quad M^{-1} = \begin{pmatrix} (\bar{M}^{K_1})^{-1} & & & & \\ & \ddots & & & \\ & & (\bar{M}^{K_i})^{-1} & & \\ & & & \ddots & \\ & & & & (\bar{M}^{K_{Ne}})^{-1} \end{pmatrix},$$

where each matrix $\bar{M}^K$ is a $(dim + 1) \times DoF$ square matrix:

$$\bar{M}^K = \begin{pmatrix} \frac{1}{\kappa} M^K & & & \\ & \rho M^K & & \\ & & \ddots & \\ & & & \rho M^K \end{pmatrix} \quad \text{and} \quad (\bar{M}^K)^{-1} = \begin{pmatrix} \kappa (M^K)^{-1} & & & \\ & \frac{1}{\rho}(M^K)^{-1} & & \\ & & \ddots & \\ & & & \frac{1}{\rho}(M^K)^{-1} \end{pmatrix}.$$

Note that we are considering here, for sake of simplicity, that the model parameters ρ and κ are piecewise constant. Hence, those parameters are outside the integrals. We will see farther that it is possible to have parameters varying inside each element using Weight Adjusted Discontinuous Galerkin (WADG) method.

Now that we have defined how to compute each local operator on each element K of $\mathcal{T}_h$, we are going to introduce the notion of reference element that will make the computation of the mass and stiffness matrices easier.

Generalization using a reference element

With this way of defining local operators, it would be necessary to store as many of these operators as there are elements. In order to have a more generic expression but also to reduce the memory size required by the solver, we introduce the calculations on a reference element $\hat{K}$. This element will be represented in 1D by the segment [0,1] in 2D by the triangle [(0,0);(1,0);(0,1)] and in 3D by the tetrahedron [(0,0,0);(1,0,0);(0,1,0);(0,0,1)]. We will then define the transformation of this reference element to an element K by the linear and bijective application T_K as illustrated in Figure 2.9. We will also denote T_Γ the bijective transformation that changes $\hat{\Gamma} \subset E_{\hat{K}}$ into $\Gamma \subset E_K$.

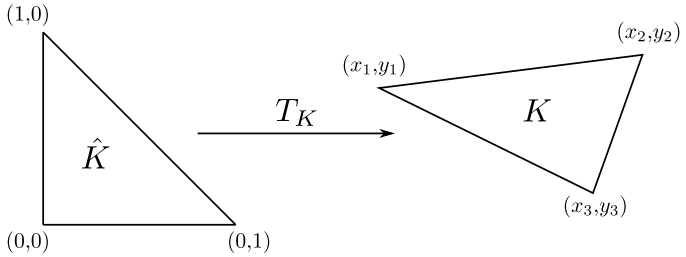


Figure 2.9: Transformation of $\hat{K}$ into K in 2D.

Thanks to this transformation, all the calculations are expressed according to the coefficients on the reference element. It is a real advantage to be able to proceed in this way because in addition to drastically simplifying the calculations, we can avoid storing the mass and stiffness matrices. These operators can therefore be calculated thanks to their equivalent form established on the reference element.

The local operators can, indeed, be computed thanks to reference operators as follows:

$$\begin{aligned}
[M^K]_{i,j} &= \int_K \varphi_i^K \varphi_j^K dK = |det(J_{T_K})| \int_{\hat{K}} \varphi_i^{\hat{K}} \varphi_j^{\hat{K}} d\hat{K} = |det(J_{T_K})| [\hat{M}]_{i,j} \,, \\
[S_{x_d}^K]_{i,j} &= \int_K \varphi_i^K \frac{\partial \varphi_j^K}{\partial x_d} dK \\
&= |det(J_{T_K})| \int_{\hat{K}} \varphi_i^{\hat{K}} (J_{T_K}^{-\top} \nabla \varphi_j^{\hat{K}}).\boldsymbol{e_d} d\hat{K} \\
&= |det(J_{T_K})| \sum_k^{dim} [J_{T_K}^{-\top}]_{d,k} [\hat{S}_{\hat{\boldsymbol{x}}_k}]_{i,j} \,, \\
[M_\Gamma]_{i,j}^K &= \int_\Gamma \varphi_i^K \varphi_j^K ds = |det(J_{T_\Gamma})| \int_{\hat{\Gamma}} \varphi_i^{\hat{K}} \varphi_j^{\hat{K}} d\hat{s} = |det(J_{T_\Gamma})| [\hat{M}_\Gamma]_{i,j} \,.
\end{aligned} \tag{2.19}$$

We are now able to compute all local operators thanks to the reference element. The reference matrices are defined by:

$$\begin{aligned}
&[\hat{M}]_{i,j} = \int_{\hat{K}} \hat{\varphi}_i \hat{\varphi}_j d\hat{K}, && \text{The reference mass matrix of size } DoF \times DoF. \\
&[\hat{S}_{\hat{x}_d}]_{i,j} = \int_{\hat{K}} \hat{\varphi}_i \frac{\partial \hat{\varphi}_j}{\partial x_d} d\hat{K}, && \text{The reference stifness matrices of size } DoF \times DoF. \\
&[\hat{M}_\Gamma]_{i,j} = \int_\Gamma \hat{\varphi}_i \hat{\varphi}_j d\hat{s} \;\text{ with }\; \Gamma \subset E_{\hat{K}}, && \text{The reference surface mass matrices of size } DoF \times DoF. \\
&J_{T_K}, && \text{The Jacobian matrix of } T_K \text{ of size } dim \times dim \,.
\end{aligned} \tag{2.20}$$

In (2.19) we also requires some determinants derived from J_{T_K}. We then introduce the notations:

$$\begin{aligned}
|T_K| &= |det(J_{T_K})| \,, \\
|T_\Gamma| &= |det(J_{T_\Gamma})| \,,
\end{aligned}$$

where *det* stands for the determinant.

It is then possible to generalize system (2.15) defined on each element by using reference operators:

$$\begin{cases}
\dfrac{1}{\kappa} |T_K| \hat{M} \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} + |T_K| \hat{M} \sum_{k=1}^{dim} (\sum_{d=1}^{dim} [J_{T_K}]_{k,d} \hat{S}_{\hat{x}_d} \boldsymbol{V_h}_d^K) + \sum_\Gamma |T_\Gamma| \hat{M}_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}^{\,\Gamma} = \dfrac{1}{\kappa} |T_K| \hat{M} \boldsymbol{F_h}^K, \\
\rho |T_K| \hat{M} \dfrac{\partial \boldsymbol{V_h}_d^K}{\partial t} + |T_K| \sum_{k=1}^{dim} [J_{T_K}]_{k,d} \hat{S}_{\hat{x}_d} \boldsymbol{P_h}^K + \sum_\Gamma |T_\Gamma| \hat{M}_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}_d}^{\,\Gamma} = 0 \qquad (\text{for } d = 1 \text{ to } dim) \,.
\end{cases}$$

We then apply the reference matrix $\hat{M}^{-1}$ to get a formulation in which the time derivatives are isolated:

$$\begin{cases}
\dfrac{\partial \boldsymbol{P_h}^K}{\partial t} = -\kappa \sum_{k=1}^{dim} (\sum_{d=1}^{dim} [J_{T_K}^{-\top}]_{k,d} \hat{M}^{-1} \hat{S}_{\hat{x}_d} \boldsymbol{V_h}_d^K) - \kappa \sum_\Gamma \dfrac{|T_\Gamma|}{|T_K|} \hat{M}^{-1} \hat{M}_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}^{\,\Gamma} + \boldsymbol{F_h}^K, \\
\dfrac{\partial \boldsymbol{V_h}_d^K}{\partial t} = -\dfrac{1}{\rho} \sum_{k=1}^{dim} [J_{T_K}^{-\top}]_{k,d} \hat{M}^{-1} \hat{S}_{\hat{x}_d} \boldsymbol{P_h}^K - \dfrac{1}{\rho} \sum_\Gamma \dfrac{|T_\Gamma|}{|T_K|} \hat{M}^{-1} \hat{M}_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}_d}^{\,\Gamma} \qquad (\text{for } d = 1 \text{ to } dim) \,.
\end{cases} \tag{2.21}$$

By using the reference mass matrix we define above (2.20), only a single inversion of a small matrix has to be computed. This is one real advantage of using DGm instead of classical Finite Element method. This enables to deal with p-adaptivity naturally. Furthermore, DGm have interesting HPC properties thanks to employing fluxes. Indeed, fluxes allow communications at the immediate neighbors which is not the case while using Finite Difference or Finite Element methods where the space scheme stencil requires more communications than the one coming from the direct neighborhood.

As all DG operators can be determined thanks the reference element $\hat{K}$, a unique basis $(\hat{\varphi})$ of $P^N(\hat{K})$ has to be implemented. In the next part, we will illustrate the choice of the Lagrange polynomial basis function.

Lagrange polynomial basis functions

Lagrange polynomial approximation is commonly used in DGm. The Lagrange polynomial basis is defined as follows:

$$\hat{\varphi}_i \in P^N(\hat{K}),\ \hat{\varphi}_i(\hat{\boldsymbol{x}}_j) = \delta_i^j = \begin{cases} 1 & \text{if } \ i = j,\ 1 \leq i, j \leq DoF, \\ 0 & \text{otherwise}\,. \end{cases}$$

For instance, we illustrate the Lagrange polynomial basis in 1D, expressed in (2.22), on the reference element $[0, 1]$ in Figure 2.10 at order five.

$$\begin{cases} \forall x \in [0,1],\ \ell_i^N(x) = \displaystyle\prod_{1 \leq j \leq N+1, j \neq i} \frac{x - x_j}{x_i - x_j}, \\ \text{with:} \quad x_i = \dfrac{(i-1)}{N}. \end{cases} \tag{2.22}$$

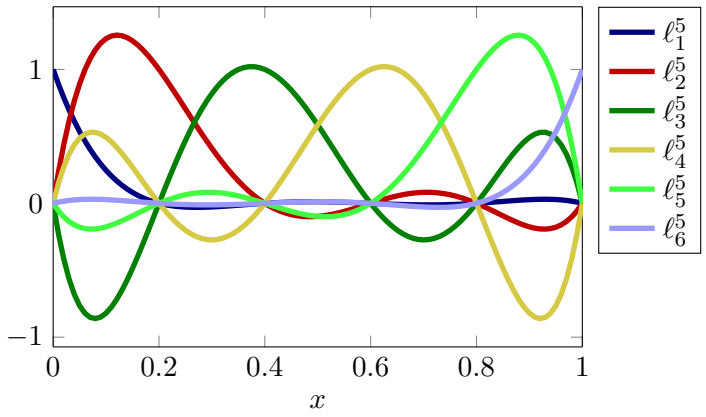


Figure 2.10: Illustration of $P^5([0, 1])$ Lagrange basis.

The Lagrange basis is a nodal basis since we have $\ell_i^N(\hat{\boldsymbol{x}}_j) = \delta_i^j$. Therefore, we get:

$$p_h{}^K(\boldsymbol{x}_j) = \boldsymbol{P}_{\boldsymbol{h}_j}{}^K\,, \boldsymbol{v}_{\boldsymbol{h}_d}{}^K(\boldsymbol{x}_j) = \boldsymbol{V}_{\boldsymbol{h}_d}{}^K{}_j\,,$$

where $\boldsymbol{x}_j$ is defined by $\boldsymbol{x}_j = T_K(\hat{\boldsymbol{x}}_j)$, where $\hat{\boldsymbol{x}}_j$ represent the j^{th} interpolation point on the reference element $\hat{K}$.

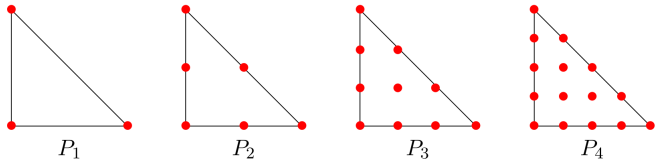


Figure 2.11: Location of the degrees of freedom in 2D on the reference element.

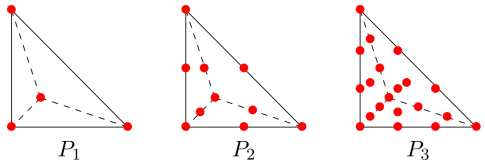


Figure 2.12: Location of the degrees of freedom in 3D on the reference element.

We illustrate the location of the interpolation points in the reference element for the first polynomial orders in 2D and 3D respectively in Figure 2.11 and Figure 2.12.

In the above pictures, the interpolation points are all equidistant. It is noteworthy that at very high order, this nodal basis employed on equidistant nodes suffers the well-known Runge phenomenon [Davis, 1975]. The Runge phenomenon can by illustrate, in 1D, by interpolating a function regular enough (for instance : $f(\hat{x_1}) = \frac{1}{(10\hat{x_1}-5)^2+1}$) in the Lagrange polynomial basis. In Figure 2.13 we clearly see what we are calling the Runge effect, located in the vicinity of 0 and 1. We can observe high undesirable values between two interpolation points.

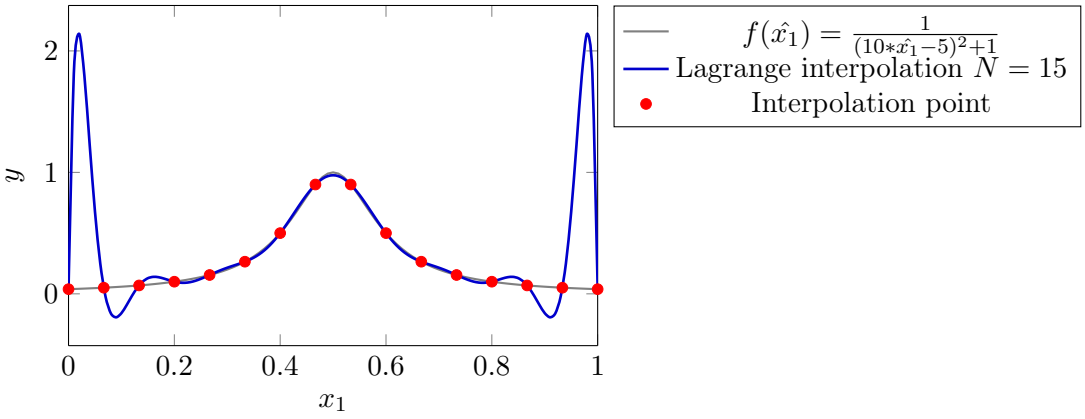


Figure 2.13: Illustration of the Runge effect coming from a Lagrange interpolation on 16 equidistant points.

As a solution to avoid this phenomenon, Hesthaven and Warburton [2007] propose to use Legendre-Gauss-Lobatto (LGL) quadrature points as interpolation points. Avoiding high parasitic values also improves the conditioning number of the DG mass matrix $\hat{M}$ because it guarantees that its components are not included in a too wide range of values. As far as this thesis is concerned, the (LGL) nodes are not implemented in Total propagator. For now, only equidistant points, pictured in Figure 2.11 and Figure 2.12, are used.

Lagrange polynomial bases are convenient and intuitive ways to represent a polynomial representation of a solution. Evaluating the approximate solution at the interpolation point is trivial. Since it is a nodal basis, the approximate solution is given at each interpolation point by the value of the coefficient associated. Everywhere else, the evaluation is more expensive

and requires the computation of the whole polynomial at the point of interest. We will see farther that we can employ other polynomial basis functions in a DG framework that seems more efficient than nodal polynomials regarding the computational effort.

Now that we have specified the discretization in space of the wave equation, it remains for us to describe its integration in time. This is precisely the purpose of the following section.

2.2.3 Time schemes

We have shown in the previous section that DGm yields a global semi-discrete system (2.18) that can be written using this synthetic expression:

$$\frac{\partial}{\partial t}\boldsymbol{U} + M^{-1}S\boldsymbol{U} = \boldsymbol{F}\,,$$

where $\boldsymbol{U} = (\boldsymbol{P_h}, \boldsymbol{V_h}1, \ldots, \boldsymbol{V_h}_{dim})^\top$ is the global vector of all the coefficients of the polynomial approximation p_h of p and $\boldsymbol{v}_h$ of $\boldsymbol{v}$ and $\boldsymbol{F}$ represents the source terms. DGm offers a block diagonal mass matrix that is shown to be easy to invert. Let us consider $A = M^{-1}S$.

$$\begin{cases} \dfrac{\partial}{\partial t}\boldsymbol{U}(t) = A\boldsymbol{U}(t) + \boldsymbol{F}(t)\,, & \text{for } t \in [t_0, T_f]\,, \\ \boldsymbol{U}(t_0) = 0\,. \end{cases} \tag{2.23}$$

The synthetic equation (2.23) can also be written by using a unique operator $\mathcal{L}(U(t), t)$ describing the right-hand side $A\boldsymbol{U}(t) + \boldsymbol{F}(t)$. The following expression is the one commonly used to describe time scheme procedures:

$$\begin{cases} \dfrac{\partial}{\partial t}\boldsymbol{U}(t) = \mathcal{L}(\boldsymbol{U}(t), t)\,, & \text{for } t \in [t_0, T_f], \\ \boldsymbol{U}(t_0) = 0. \end{cases} \tag{2.24}$$

The time is discretized along the regular time grid (t_n) for $n \in [0, N]$ that represents $[t_0, T_f]$ as a collection of points $t_{n+1} = t_n + \Delta t$ and $t_N = T_f$. Integrating (2.24) between two time steps gives :

$$\boldsymbol{U}(t_{n+1}) - \boldsymbol{U}(t_n) = \int_{t_n}^{t_{n+1}} \mathcal{L}(\boldsymbol{U}(t), t)dt\,. \tag{2.25}$$

The right hand side replaced by the general function $\mathcal{L}$ is symptomatic of the way the program is structured. Once this function is implemented, it is enough to follow the procedures that we will explain in what follows.

We make the choice of using explicit time schemes whose stability is ensured providing the time step is chosen small enough. This choice is dictated by the need of reducing as much as possible intermediate calculations such as matrix inversions which would be necessary for the implementation of an implicit scheme. The value of the time step is calculated as a function of the size of the smallest cell of the mesh, the mean values of the physical parameters describing the propagation medium and the polynomial order of approximation. A global time step might induce very high computational costs that could be reduced significantly by using local time stepping [Diaz and Grote, 2015, Gödel et al., 2010, Baldassari, 2009], but it has been observed that local time stepping tends to hamper the parallelization performances due to balancing issues arising from the domain decomposition [Rietmann et al., 2015]. Implicit time schemes have better stability properties but since they require solving one linear system at each time step, we prefer to favor explicit formulations, in particular in the context of FWI

which is computationally intensive since requiring solving many forward problems. However, it has been discussed in [N'Diaye, 2017] that implicit schemes promote using Hybridizable Discontinuous Galerkin methods because of their structure which allow main computation on the skeleton of the mesh. It is indeed convenient to implement hybrid schemes as in [Barucq et al.]. Herein, we limit out study to DGm with explicit time schemes.

We now describe the time schemes that have been developed in the industrial code of Total.

Explicit Runge-Kutta 2 time scheme

The Runge-Kutta time scheme of order 2 (RK2) is obtained by applying the midpoint quadrature formula to the integral from (2.25). We then have:

$$\begin{aligned}\boldsymbol{U}(t_{n+1}) - \boldsymbol{U}(t_n) &= \int_{t_n}^{t_{n+1}} \mathcal{L}(\boldsymbol{U}(t), t)dt, \\ &\approx \Delta t \underbrace{(\mathcal{L}(\boldsymbol{U}(t_n + \frac{1}{2}\Delta t), t_n + \frac{1}{2}\Delta t)}_{\boldsymbol{k_2}}.\end{aligned}$$

The explicit RK2 is then given by:

$$\begin{aligned}&\boldsymbol{U}^{n+1} \approx \boldsymbol{U}^n + \Delta t \boldsymbol{k_2} \quad \text{with:} \\ &\boldsymbol{k_1} = \mathcal{L}(\boldsymbol{U}^n, t_n)\,, \\ &\boldsymbol{k_2} = \mathcal{L}(\boldsymbol{U}^n + \frac{1}{2}\Delta t \boldsymbol{k_1})\,.\end{aligned}$$

This can be represented by the Butcher table (Table 2.2) [J.C. Butcher, 2016] as follows.

RK2 :		
	0	
	0.5	0.5
	0	1

Table 2.2: Butcher table for RK2 time scheme.

The Butcher table sums up the quadrature coefficients used to define a Runge-Kutta time scheme. There exist several Runge Kutta time schemes of order two, we display the corresponding table, in Table 2.2, for discerning reader to determine at first glance which time scheme we are dealing with.

The explicit RK2, described here, requires two computations of the right-hand-side function and one extra stage to be saved.

Explicit Runge-Kutta 4 time scheme

The standard fourth-order explicit Runge-Kutta time scheme (RK4) is given by the following four stage procedure:

$$\begin{aligned}&\boldsymbol{U}^{n+1} \approx \boldsymbol{U}^n + \frac{1}{6}\Delta t(\boldsymbol{k_1} + 2\boldsymbol{k_2} + 2\boldsymbol{k_3} + \boldsymbol{k_4}) \quad \text{with :} \\ &\boldsymbol{k_1} = \mathcal{L}(\boldsymbol{U}^n, t_n)\,, \\ &\boldsymbol{k_2} = \mathcal{L}(\boldsymbol{U}^n + \frac{1}{2}\Delta t \boldsymbol{k_1}, t_n + \frac{1}{2}\Delta t)\,, \\ &\boldsymbol{k_3} = \mathcal{L}(\boldsymbol{U}^n + \frac{1}{2}\Delta t \boldsymbol{k_2}, t_n + \frac{1}{2}\Delta t)\,, \\ &\boldsymbol{k_4} = \mathcal{L}(\boldsymbol{U}^n + \Delta t \boldsymbol{k_3}, t_n + \Delta t)\,.\end{aligned}$$

and the associated Butcher table is given in Table 2.3.

	0				
	0.5	0.5			
RK4 :	0.5	0	0.5		
	1	0	0	1	
		1/6	1/3	1/3	1/6

Table 2.3: Butcher table for RK4 time scheme.

RK4 requires computing four right-hand-sides. It is more expensive than RK2 but this scheme, in addition to being more accurate, is more stable. The time step Δt that satisfies the stability of the scheme is bigger using RK4 instead of using RK2. For more details concerning the stability of such schemes in the context of wave equations, we refer to the thesis of N'Diaye [2017].

The drawback of this scheme is that it needs four additional stages to be stored, which can lead to a undesirable memory burden. Alternatives exist, notably with the Low Storage Explicit Runge Kutta 4 (LSERK4) [Carpenter and Kennedy, 1994], which only needs one extra state to be stored in return for an extra calculation of the right-hand side function.

Adams-Bashforth 3 time scheme

Adams-Bashforth time scheme uses another strategy to approximate the time derivative. It aims to reduce the number of right-hand side computations. Contrary to Runge-Kutta schemes that are using intermediate stages, the computation of $\boldsymbol{U}^{n+1}$ uses several previous states $\boldsymbol{U}^n$,$\boldsymbol{U}^{n-1}$,...,$\boldsymbol{U}^{n-p}$. We will denote by p the number of previous stages we are keeping.

By storing this information, it is possible to compute the polynomial Lagrange interpolation of $\mathcal{L}$. We note $\mathcal{L}^{(n-i)} = \mathcal{L}(\boldsymbol{U}^i, t_i)$. We have then the following polynomial interpolation:

$$\mathcal{L}(\boldsymbol{U}(t), t) \approx \sum_{i=1}^{p} \mathcal{L}^{(n-i)} l_{n,i,p}(t),$$

with:

$$l_{n,i,p}(t) = \prod_{0 \leq j \leq p,\, j \neq i} \frac{t - t_{n-j}}{t_{n-i} - t_{n-j}}.$$

It gives rise to the following approximation of $\boldsymbol{U}^{n+1}$:

$$\begin{aligned}
\boldsymbol{U}^{n+1} &\approx \boldsymbol{U}^n + \sum_{i=0}^{p} \mathcal{L}^{(n-i)} \int_{t_n}^{t_{n+1}} l_{n,i,p}(t)\,, \\
&\approx \boldsymbol{U}^n + \sum_{i=0}^{p} \mathcal{L}^{(n-i)} \Delta t \int_0^1 l_{n,i,p}(t_n + \Delta t\, t')dt'\,, \\
&\approx \boldsymbol{U}^n + \Delta t \sum_{i=0}^{p} \mathcal{L}^{(n-i)} \beta_{i,p}\,.
\end{aligned}$$

Adams-Bashforth time schemes are proved to be of order p+1, the case $p = 0$ corresponding to the classical explicit Euler scheme. Those schemes require only one evaluation of the right-hand side per time step but with the price of storing p stages. In the industrial code from

Total, the Adams-Bashforth 3 ($p = 2$) scheme has been implemented. This scheme is defined by the following ($\beta_{i,2}$) coefficients:

$$\beta_{0,2} = \frac{23}{12}\,, \qquad \beta_{1,2} = -\frac{16}{12}\,, \qquad \beta_{2,2} = -\frac{5}{12}\,,$$

which yields the following scheme:

$$\boldsymbol{U}^{n+1} \approx \boldsymbol{U}^n + \Delta t(\beta_{0,2}\mathcal{L}^n + \beta_{1,2}\mathcal{L}^{(n-1)} + \beta_{2,2}\mathcal{L}^{(n-2)})\,.$$

Note that the two previous estimations of the right-hand are needed to perform the current step. It is then impossible for AB3 to evaluate $\mathcal{L}^1$ and $\mathcal{L}^2$ without using another time scheme (RK2 for instance) to start the process. The choice that has been done in the industrial code of Total is to trick the coefficients $\beta_{i,2}$ to be:

$$\beta_{0,2} = 1\,, \qquad \beta_{1,2} = 0\,, \qquad \beta_{2,2} = 0\,,$$

at the first iteration. Here we are performing an AB1 time step which is nothing but an explicit Euler time step. For the second iteration, the set of coefficients $\beta_{i,2}$ is chosen as:

$$\beta_{0,2} = \frac{3}{2}\,, \qquad \beta_{1,2} = -\frac{1}{2}\,, \qquad \beta_{2,2} = 0\,,$$

defining the AB2 time scheme. Schemes AB1 then AB2 are thus involved to perform the first and second iterations. The remaining time steps are then carried out by following the process AB3.

The main asset of Adams-Bashforth time scheme is to require only one computation of the right-hand side. However, and unfortunately, those methods suffer from a bad conditioning that can be controlled by reducing the time step size Δt and thus, they may have a high computational cost.

The low computational cost per steps of the method makes this scheme a good candidate for accurate multi-rate local time stepping. In the case where the size of the mesh and the physical model vary from small to large scales in the computational domain, multi-rate Adams-Bashforth scheme has already proved its efficiency [Gödel et al., 2010].

Remark. Those time schemes are used because they are stable and compatible with the upwind fluxes. It is worth noting that such fluxes prevent using Leap-Frog time scheme which would not be explicit anymore. The choice of the fluxes influences the conditioning of the system. For example, using centered fluxes with RK2 time scheme is unconditionally unstable [Deriaz, 2012], which is not the case while using upwind fluxes.

The purpose of what follows is to determine an empirical stability condition on the time step Δt by introducing the Courant–Friedrichs–Lewy (CFL) condition.

Courant–Friedrichs–Lewy (CFL) condition

As explained before, we focus on explicit time schemes. Contrary to implicit time schemes, they are constrained by a stability condition.

Definition 2.2.1. A method is stable if it exists a constant $\alpha \in \mathbb{R}^+$ such that for two sequences (y_n) and $(\tilde{y}_n)$ defined as follows:

$$y_{n+1} = y_n + \Delta t \Psi(t_n, y_n, \Delta t),$$
$$\tilde{y}_{n+1} = \tilde{y}_n + \Delta t \Psi(t_n, \tilde{y}_n, \Delta t) + \varepsilon_n,$$

we have:

$$\underset{0\leq n\leq N}{max} |\tilde{y}_n - y_n| \leq \alpha \left(|\tilde{y}_0 - y_0| + \sum_{0\leq n\leq N} |\varepsilon_n| \right) .$$

That is to say that the error at each iteration can be controlled by the initial error $|\tilde{y}_0 - y_0|$ and the rounding error ε_n done at each step. It is easy to see that both RK and AB schemes can be rewritten as sequences (y_n) and $(\tilde{y}_n)$ of Definition 2.2.1.

In our case, the stability condition depends on the spatial operator $\mathcal{L}$ and defines the proper time step Δt for the explicit time scheme. To obtain the optimal time step, an eigenvalue study can be carried out on $\mathcal{L}$. This is generally done by an iterated power method for compatible time schemes such as Leap-Frog [Citrain, 2019]. Note that explicit Leap-Frog time scheme is only available with centered fluxes. Here we are using upwind fluxes that we defined in (2.16) and (2.17). For the multistep time schemes presented here, the eigenvalue analysis is difficult to implement. In practice, we do approximate the stability condition by using a heuristic CFL condition. This condition gives an estimate of the ideal time step by defining an empirical relation between time step Δt, space discretization size Δx and physical parameters (here the wavespeed c).

The relation defining the CFL condition is most of the time inspired by advection problems. The geometrical relation derived from the CFL condition expresses the distance travelled by the wave during one time step.

Inspired by the CFL condition defined in [Hesthaven and Warburton, 2007], we define the geometrical condition:

$$\Delta t^K \leq C \frac{r^K}{c^K p^K} , \tag{2.26}$$

with:

- C: the CFL condition coefficient to be configured for the different time schemes;
- r^K: the inner radius of the cell K;
- c^K: the wavespeed parameter associated to K;
- p^K: the polynomial order of approximation in element K.

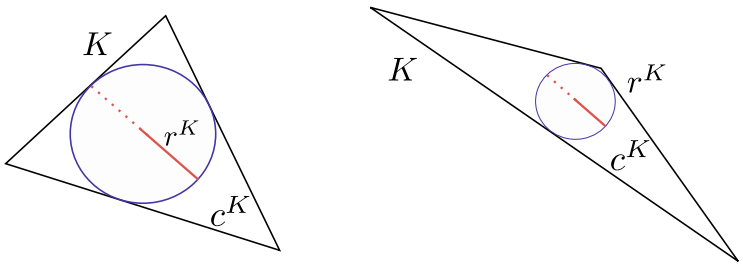


Figure 2.14: Illustration of inner radius for two cell configurations.

We have chosen to configure the CFL condition using the inner radius instead of the minimal edge as done in [Hesthaven and Warburton, 2007]. If the element is really acute as illustrated in Figure 2.14, the size of the edges is, indeed, not representative of the distance between each node in the element. This scenario is likely to occur in geophysical applications where the meshes can be strongly anisotropic.

With this CFL condition (2.26), we are able to quantify an approximation of the time step Δt^K for each cell. The selected global time step will be the more restrictive one that is to say $\min_K(\Delta t^K)$.

The central point is therefore the computation of the coefficient C for the different time schemes. Those coefficients are displayed in Table 2.4 and have been determined by dichotomy from a collection of numerical experiments. These values are by no way absolute references. They can probably be the subject of adaptation in the future. As far as the work done in this thesis is concerned, these are the coefficients that are used. In any case, these values represent and give an order of idea of the ranges of the stability conditions for the different schemes, but they have advantage of giving stable simulations.

	RK2	RK4	AB3
C	0.66	0.84	0.18

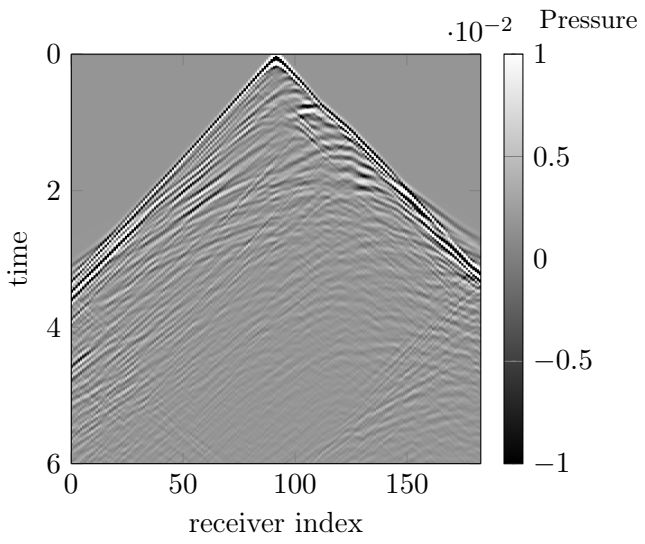
Table 2.4: CFL coefficients for different time schemes.

2.2.4 Numerical Experiments

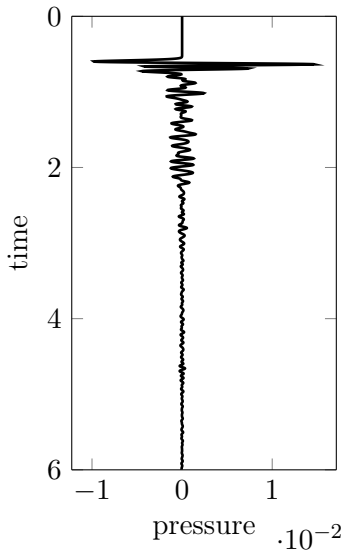
In the foregoing sections, we defined the space and time discretization. The theoretical foundations are in place to perform direct problem simulations. In this part, we aim to validate the time domain acoustic DG solver implemented in Total environment.

We will compare the output of this solver with the results obtained with two other pieces of software developed by Inria Bordeaux Sud-Ouest, in the Project-Team Magique 3D. The first software is [Gar6more2D]. This code displays acoustic, elastic and poroelastic simulations on 2D homogeneous or bilayered media based on Cagniard-de-Hoop method. We will use it in order to validate our numerical results thanks to a comparison with analytic solutions on bilayered models. The second software is [Hou10ni] which uses Interior Penalty Discontinuous Galerkin (IPDG) to solve acoustic, elastic and elasto-acoustic wave equation. We will use this software in order to validate our results in a more complex heterogeneous media.

In what follows, the results we obtain will be on the form of seismograms. A seismogram represents on the X-axis all the receivers successively and on the Y-axis the simulated time. Each column represents a trace, that is to say the perturbation recorded by the associate receiver. This signal recorded can be either a pressure, a displacement or a velocity perturbation. In the tests developed in this section we will only consider the recorded pressure at receivers. We illustrate in Figure 2.15 an example of seismograms and the corresponding trace associated to the receiver n^{o}110.



(a) Example of seismograms.

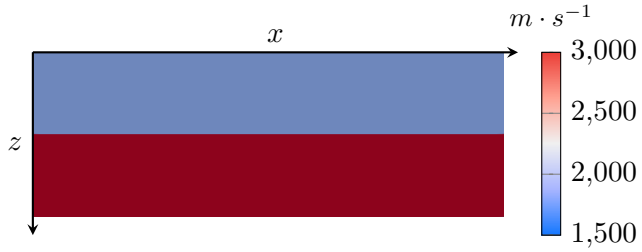


(b) Recorded signal at receiver n^o 110.

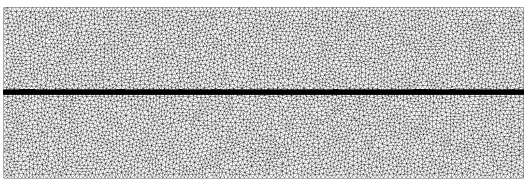
Figure 2.15: Illustration of a seismograms (a) and the trace corresponding to the receiver n^o 110 (b).

Bilayered Acoustic problem

For this experiment, we are considering a 2D bilayered acoustic medium. The domain of interest is a rectangle of size 9200m × 3000m. The first upper part is composed of a homogeneous wavespeed model of 1500m.s^{-1}. The second part is defined by a wavespeed model of 3000m.s^{-1} (see Figure 2.16a). The interface is located at 1500m depth. We will consider the density to be constant, on the entire domain, ρ=1000kg.m^{-3}.



(a) bilayered wavespeed model.



(b) Mesh used for wave propagation on bilayered model (interface forced for sake of visibility).

Figure 2.16: Illustration of the bilayered wavespeed model and the associated mesh for the current experiment.

We consider 183 receivers located at 50m depth equally distributed on the X-axis from 50m to 9150m. One source located at (4600m,60m) perturbs the medium: it is a first order Ricker function pressure disturbance with $f_{peak} = 10$Hz and $t_{peak} = 0.2$s.

It is given by (see Figure 2.17):

$$f(t) = (t - t_{peak}) \exp^{-(\pi f_{peak}(t-t_{peak}))^2} .$$

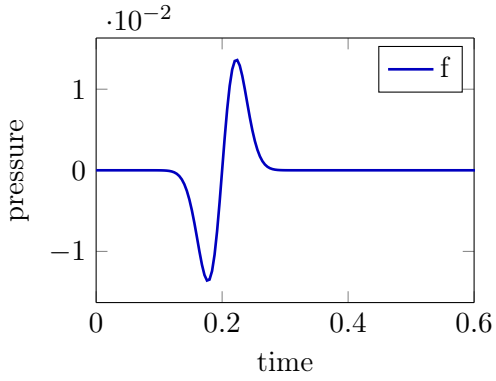


Figure 2.17: First order Ricker function for $f_{peak} = 10$Hz and $t_{peak} = 0.2$s.

Concerning the boundary condition, we choose a free surface condition at the top boundary ($p_{|\Gamma_1}$) and absorbing boundary condition on the others. We recall the absorbing boundary condition introduced in (2.4) in page 50:

$$\frac{\partial p}{\partial t}(t, \boldsymbol{x}) + c(\boldsymbol{x}) \nabla p(t, \boldsymbol{x}).n = 0\,, \quad x \in \Gamma_2\,, t \in [t_0, T_f].$$

We aim to validate the industrial solver by comparing the traces obtained after 2.6s time of simulation. We used a mesh constituted of 12556 P4 elements and a RK2 time scheme. The interface has been taken into account in the mesh generation. The interface is then perfectly respected by the mesh as shown in Figure 2.16b. Even if we are using piecewise constant parameters per element here, with this kind of mesh, we do not misrepresent the wavespeed model since the mesh is exactly following the interface.

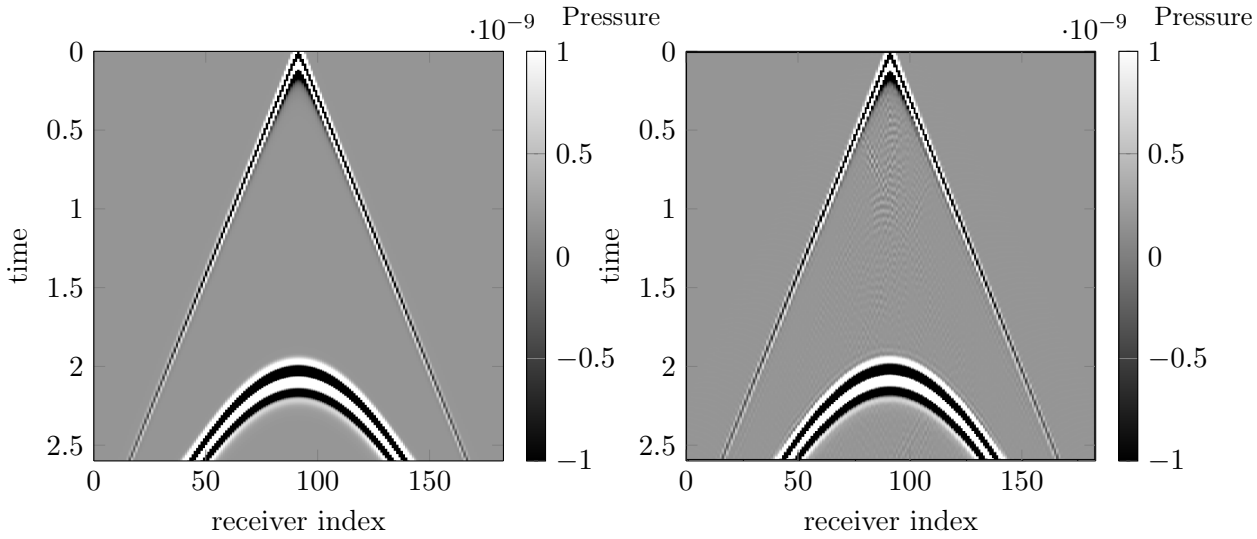
We display in Figure 2.18 the two seismograms recorded by the receivers and obtained with the analytical solution generator Gar6more2D and the DG solver. We can see that both are very similar, in particular regarding the kinematic.

In order to get significant information, we can compute the global L^2 error on the seismograms that is defined as:

$$Error = \sqrt{\frac{\sum_{rcv} \sum_t^{T_f} (p_{ref}(rcv, t) - p_{num}(rcv, t))^2}{\sum_{rcv} \sum_t^{T_f} (p_{ref}(rcv, t))^2}}\,.$$

With such formula, we obtain a relative error of 1%, which validates the numerical experiment.

For further visual assessment, we also display both traces obtained on the receiver n^{o}110 in Figure 2.19. We can see that both perturbations perfectly match.



(a) Seismogram obtained using Gar6more2D. (b) Seismogram obtained using Total's code.

Figure 2.18: Comparison of bilayered seismograms between: Reference solution with Gar6more2D (a) and Numerical solution using Total DG solver (b).

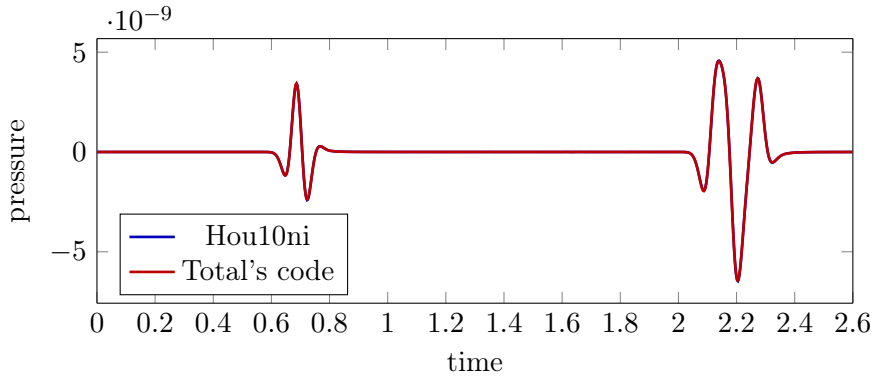


Figure 2.19: Trace comparison between references and numerical solutions recorded by the 110^{th} receiver.

Moving forward on the validation of the DG solver, we propose to deal with a more complex wavespeed model.

Marmousi Acoustic problem

For this study, we will use the Marmousi wavespeed model (see Figure 2.20) with a constant density parameter on the entire domain ($\rho = 1000\text{kg.m}^{-3}$). The domain is a rectangle of dimension 9200m $\times$ 3000m. We set 183 receivers at the same location that in the previous study, that is to say, at 50m depth and equally separated on the X-axis from 50m to 9150m. One single source is located at (4550m,100m) which perturbs the pressure wavefield as a first order Ricker function with $f_{peak} = 10\text{Hz}$ and $t_{peak} = 0.2\text{s}$. For the space discretization, we use a mesh constituted of 59266 P4 elements for both Total DG solver and Hou10ni simulations. Indeed, we cannot employ the analytical solution generator in this case. We are using the same mesh in order to have a comparison that is not affected by the difference of the model approximation. Concerning the time schemes, we are using a RK2 time scheme

for the simulation performed by Total solver and Leap Frog time scheme for the one from Hou10ni. The top surface is defined by a free surface boundary condition and absorbing boundary condition are set for the three last boundaries.

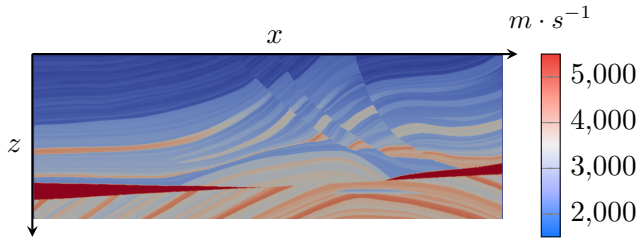
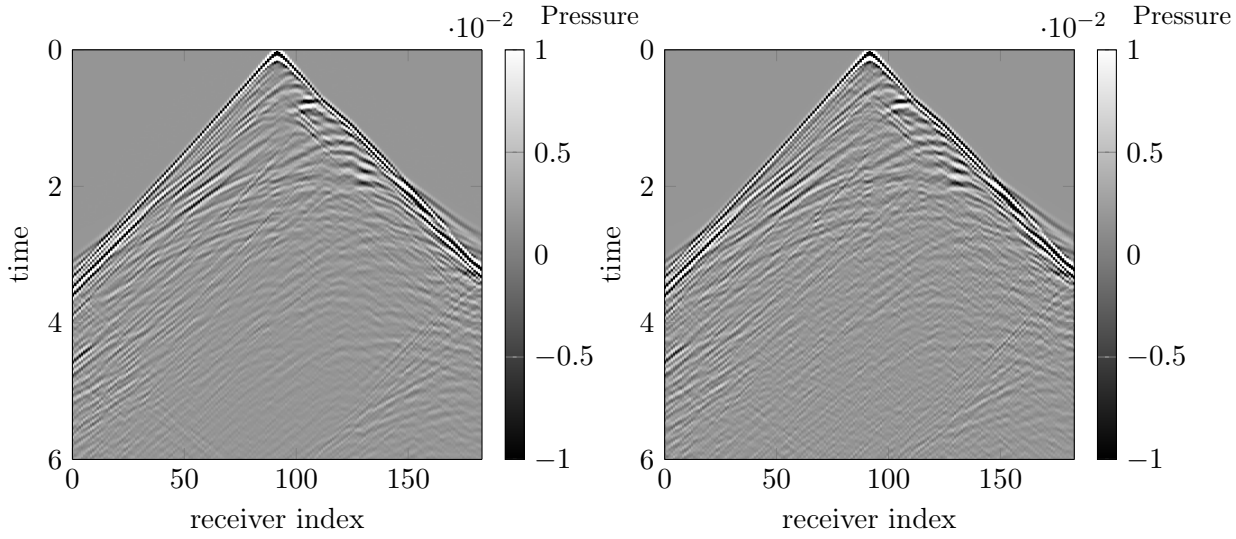


Figure 2.20: Illustration of the Marmousi wavespeed model.

In Figure 2.21 we display the resulting seismograms. Both of them look really similar. As we did before, we can compute a relative L2 error to quantify the error between the two seismograms. Here we obtain an error of 0.16% which validates the capability of Total's solver to simulate acoustic wave propagation in complex media using Hou10ni solver as reference.



(a) Seismogram obtained using Hou10ni.

(b) Seismogram obtained using Total's code.

Figure 2.21: Comparison of Marmousi seismograms between: Reference solution (a), Numerical solution Total's code (b).

In Figure 2.22, we display the solution recorded by the 110^{th} receiver, and we can see that both traces match perfectly.

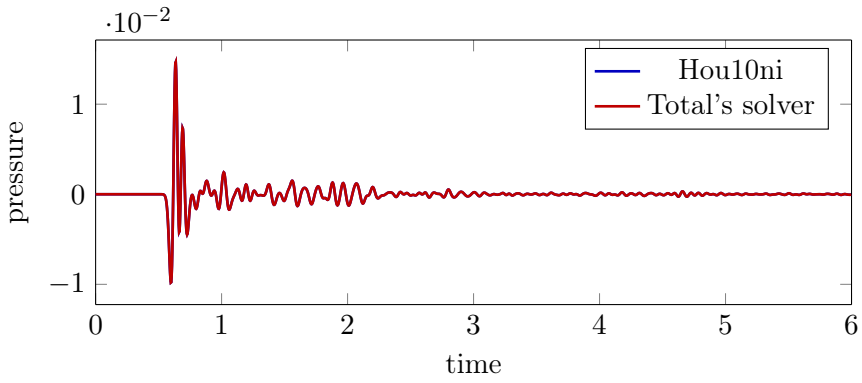


Figure 2.22: Trace comparison between references and numerical solutions recorded by the 110^{th} receiver.

The comparisons that have well been performed here in a bilayered media and on Marmousi acoustic domain show that we recover the reference signal. On the first model, the comparison has been performed with an analytical solution giving good accuracy. Concerning the second medium, which is more complex, the comparison has been done with another DG solver using another time scheme. The seismograms we obtained by Total and Hou10ni solver are similar. The small relative L2 error, of 0.16% between those two results proves the capability of Total DG acoustic solver to simulate propagation in complex media.

Those experiments have been performed using nodal representation of Legendre polynomial basis set on equidistant points which is then equivalent to the Lagrange polynomial basis we introduced before. For further information concerning the nodal representation of the Legendre basis, we refer to [Hesthaven and Warburton, 2007]. We aim, in the next section, to explore other polynomial bases and more precisely the advantage brought by the modal Bernstein-Bézier polynomial basis.

2.3 Bernstein-Bézier Polynomial Basis

Sergei Natanovich Bernstein introduced a basis of polynomials at the beginning of the 20th century in the purpose of implementing an elegant proof of the Weierstrass approximation theorem [Bernštein, 1912]. Then it is necessary to wait until the 1960s to see this polynomial basis, currently call Bernstein polynomials, regaining an interest no longer in probability but in geometry. Two rival French automobile engineers, Pierre Etienne Bézier and Paul de Faget de Castlejau, were trying to express the Bézier curves analytically in order to reduce the memory cost of geometrical vehicle parameterization at the dawn of computer science technology. Bézier curves are actually very useful for representing automobile parts digitally, and they can be expressed in terms of Bernstein polynomial basis. It is for this reason that the polynomials derived from this basis are commonly called Bernstein-Bézier polynomials. For further information we refer to the work of Farouki [2012], which historically but also mathematically retraces a century of retrospection on this polynomial basis.

Recently, Bernstein polynomials have had an increasing interest in the context of implementing high order finite elements which generate extra costs. Some of their properties indeed make it possible to reduce the computational cost caused by treatment of the high-order matrices of mass and stiffness [Ainsworth et al., 2011, Kirby and Thinh, 2012]. In the acoustic DG solver developed by Total, the Bernstein polynomials implementation is inspired by the architecture proposed by Jesse Chan and Tim Warburton in [Chan and Warburton, 2016].

In this section dedicated to Bernstein polynomial basis, we will first state the main properties of this polynomial basis. By exploiting these properties, we will show how such basis can enhance the DG operators defined in the previous section. Finally, we will study and assess how the use of Bernstein basis can improve numerical performance of DGm, thanks to numerical experiments.

2.3.1 Bernstein polynomial basis description and properties

Let us introduce the Bernstein polynomial basis in one dimension as we did for Lagrange polynomial basis before (2.2.2). For $g \in P(\hat{K})^N$, where $\hat{K}$ corresponds to the $[0,1]$ segment, we have:

$$g(\hat{x_1}) = \sum_{i=0}^{N} G_i B_i^N(\hat{x_1}) \,,$$

with $B_i^N(\hat{x_1}) = C_i^N(1-\hat{x_1})^{N-i}\hat{x_1}^i$, and $C_i^N = \binom{N}{i} = \frac{N!}{(N-i)!i!}$.

Each B_i^N represents a binomial distribution, that is to say, the probability of i successes on N draws with replacement where the probability of one success is $\hat{x_1}$. For an illustration of the basis at order $N = 5$, see Figure 2.23.

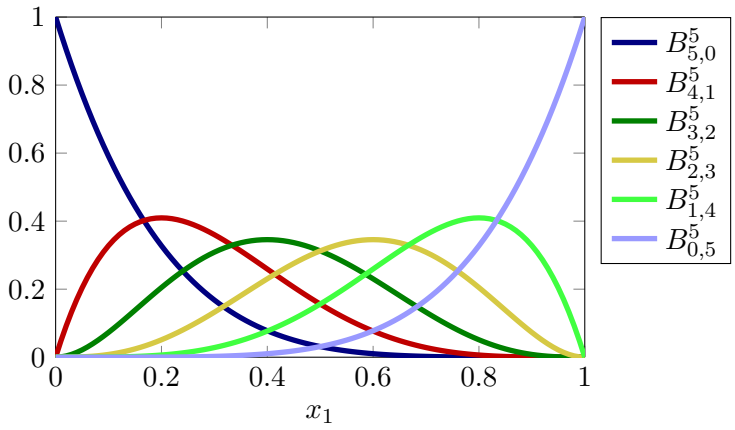


Figure 2.23: Representation of the Bernstein polynomial basis of order 5.

To study the Bernstein polynomial in higher dimension, the position of a point $\hat{\boldsymbol{x}}$ in the reference element $\hat{K}$ has to be expressed in the barycentric coordinates ($\hat{\boldsymbol{\lambda}}$). We note $\hat{\boldsymbol{\lambda}}$ the vector of size $dim + 1$ where $\hat{\lambda}_i$ is the i^{th} component of the barycentric coordinates. By considering the reference elements (in 1D, 2D and 3D) previously defined , such coordinates can be expressed in this way:

$$\begin{array}{llllll}
\text{In 1D:} & \hat{\lambda_0} = (1-\hat{x_1}), & \hat{\lambda_1} = \hat{x_1}. & & & \\
\text{In 2D:} & \hat{\lambda_0} = (1-\hat{x_1}-\hat{x_2}), & \hat{\lambda_1} = \hat{x_1}, & \hat{\lambda_2} = \hat{x_2}. & & \\
\text{In 3D:} & \hat{\lambda_0} = (1-\hat{x_1}-\hat{x_2}-\hat{x_3}), & \hat{\lambda_1} = \hat{x_1}, & \hat{\lambda_2} = \hat{x_2}, & \hat{\lambda_3} = \hat{x_3}.
\end{array} \tag{2.27}$$

We introduce the multi-indices notation of the polynomials to generalize Bernstein polynomial basis properties at different orders and dimensions. For instance, in 3D ($dim = 3$), a basis function $B_{ijkl}^N(\hat{\lambda_0}, \hat{\lambda_1}, \hat{\lambda_2}, \hat{\lambda_3})$ can be noted as $B_{\boldsymbol{\alpha}}^N(\hat{\boldsymbol{\lambda}})$; where $\boldsymbol{\alpha}$ represents the tuple (i,j,k,l) and $(\hat{\lambda_0}, \hat{\lambda_1}, \hat{\lambda_2}, \hat{\lambda_3})$ are the 3D barycentric coordinates in the reference element. In

an arbitrary dimension, $\boldsymbol{\alpha} \in \mathbb{N}^{dim+1}$ with $\sum_{i=0}^{N} \alpha_i = N$ and $\hat{\boldsymbol{\lambda}} \in [0,1]^{dim+1}$ with $\sum_{i=0}^{N} \hat{\lambda}_i = 1$. For example, we have in 3D the following notation:

$$B_{\boldsymbol{\alpha}}^{N}(\hat{\boldsymbol{\lambda}}) = B_{ijkl}^{N}(\hat{\lambda_0}, \hat{\lambda_1}, \hat{\lambda_2}, \hat{\lambda_3}) = C_{ijkl}^{N} \hat{\lambda_0}^{i} \hat{\lambda_1}^{j} \hat{\lambda_2}^{k} \hat{\lambda_3}^{l} = \frac{N!}{i!j!k!l!} \hat{\lambda_0}^{i} \hat{\lambda_1}^{j} \hat{\lambda_2}^{k} \hat{\lambda_3}^{l}$$

Now that the notation is set, we will develop the properties of interest offered by the Bernstein polynomial basis. These properties are inherited from the binomial distribution which is a key idea of the construction of these polynomials.

Since each basis is a probability law, we have the following statements:

- **Positivity:** $\forall \hat{\boldsymbol{\lambda}} \in \hat{K},\ 0 \leq B_{\boldsymbol{\alpha}}^{N}(\hat{\boldsymbol{\lambda}})$.
- **Boundedness:** $\forall \hat{\boldsymbol{\lambda}} \in \hat{K},\ B_{\boldsymbol{\alpha}}^{N}(\hat{\boldsymbol{\lambda}}) \leq 1$.
- **Partition of unity:** $\forall \hat{\boldsymbol{\lambda}} \in \hat{K},\ \sum_{\boldsymbol{\alpha}} B_{\boldsymbol{\alpha}}^{N}(\hat{\boldsymbol{\lambda}}) = 1$.

These statements enable us to define more interesting properties that can be exploited in the context of high order finite element method discretization.

Property 2. Lower and Upper bounds.
For $g \in P(\hat{K})^N$ such as $g(\hat{\boldsymbol{\lambda}}) = \sum_{\boldsymbol{\alpha}} G_{\boldsymbol{\alpha}} B_{\boldsymbol{\alpha}}^{N}(\hat{\boldsymbol{\lambda}})$ with $G_{\boldsymbol{\alpha}}$ the polynomial coefficients associated to the $(dim+1)$-tuple $\boldsymbol{\alpha}$, we have $\min_{\boldsymbol{\alpha}}(G_{\boldsymbol{\alpha}}) \leq g(\hat{\boldsymbol{\lambda}}) \leq \max_{\boldsymbol{\alpha}}(G_{\boldsymbol{\alpha}})$. The value of the polynomial g is thus bounded by the minimal and maximal values of its coefficient in the Bernstein polynomial basis.

This inequality prevents from Runge effects. To illustrate this property we can reuse the interpolation example introduced before in Figure 2.24.

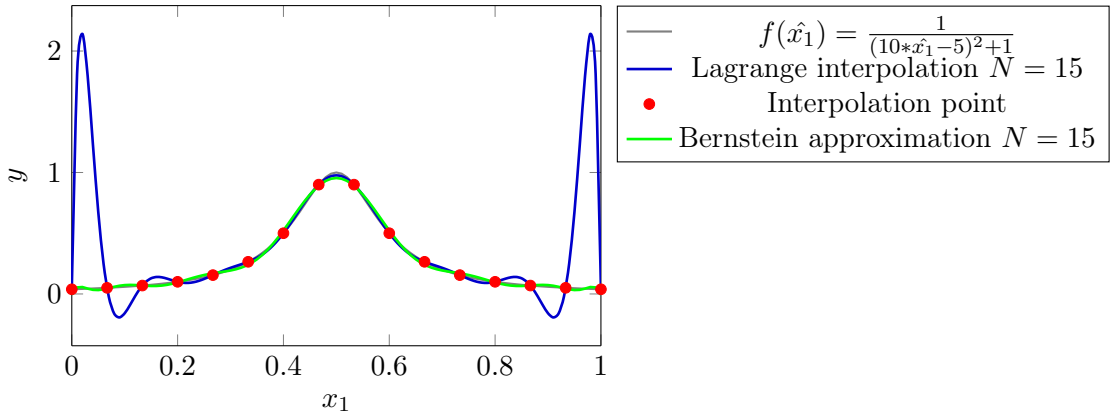


Figure 2.24: Comparison the Lagrange polynomial interpolation and the Bernstein approximation regarding the Runge effect.

We clearly see in Figure 2.24 that the Bernstein approximation in not hampered by Runge effects. It is worth noting that when considering Bernstein polynomials we do not talk about interpolation but approximation. The notion of interpolation is associated to the fact that the reconstructed solution is achieved on a finite set of interpolation points.

Avoiding Runge effects leads to reduce the range of the value distribution in the DG operator, which enhances the conditioning of finite element operators.

Property 3. Efficient evaluation.
For $g \in P(\hat{K})^N$ such as $g(\hat{\boldsymbol{\lambda}}) = \sum_{\boldsymbol{\alpha}} G_{\boldsymbol{\alpha}} B^N_{\boldsymbol{\alpha}}(\hat{\boldsymbol{\lambda}})$, knowing $P_{\boldsymbol{\alpha}}$, $p(\hat{\boldsymbol{\lambda}})$ can be efficiently computed using the pyramidal de Casteljau algorithm [Ainsworth, 2014].

This algorithm is suboptimal in terms of complexity but its recursive structure leads to have a stable algorithm easy to implement for a minimal memory cost.

This algorithm is termed with pyramidal because it is a recursive algorithm that constructs a vector of size $DoF - i$ at the i_{th} iteration. Then, at the first iteration, we are constructing a vector of size $DoF - 1$ from the initial vector of size DoF representing the coefficient of a polynomial $g \in P(\hat{K})^N$. After $DoF - 1$ iterations, the single value that remains is the evaluation of the polynomial g at the point of interest $\hat{x_1}$.

Here is a brief description that generalizes the behavior of this pyramidal algorithm. To illustrate the de Casteljau algorithm we are considering this algorithm in 1D. We will define G as a vector of size DoF in which the polynomial coefficients in the Bernstein basis are stored. To compute $g(\hat{x_1})$ we can run an algorithm similar to this simplistic program:

```
for j in range(DoF):
   for i in range(DoF-(j+1)):
      G[i] = (1-x)*G[i] + x*G[i+1] # weighted mean

return G[0] # contains the evaluation of g at point x
```

The term pyramidal becomes clear in 1D when the algorithm is depicted graphically. The Figure 2.25 illustrates this algorithm for a polynomial of order $N = 3$. At each step, we are computing, recursively, a weighted mean between two points.

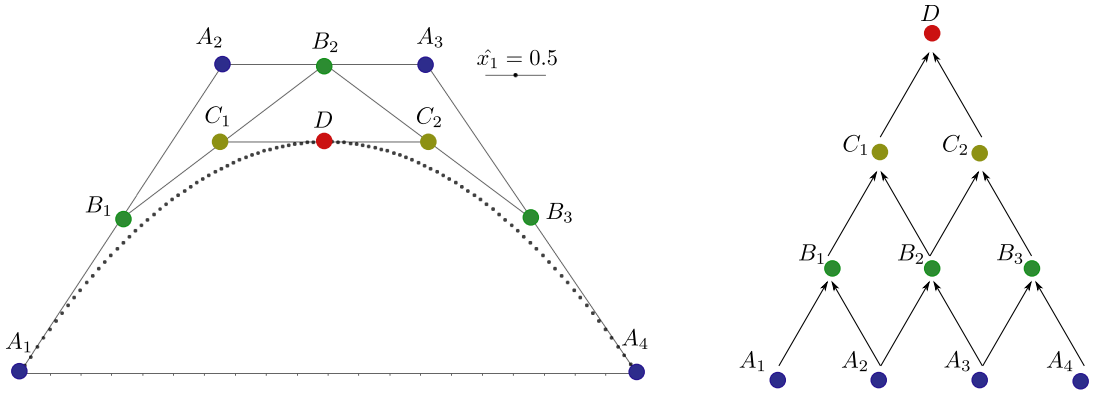


Figure 2.25: Illustration of pyramidal de Casteljau algorithm in 1D for a P3 function.

Concerning Lagrange polynomial basis, the evaluation is trivial. The evaluation on the interpolation point consists in taking the proper coefficient associated to the basis function that is equal to one at this point. If the polynomial has to be evaluated in between two interpolation points then, there is no underlying efficient algorithm. Such evaluation can be used, for instance, when considering high order visualization where it is required to evaluate the polynomial solution in between interpolation points for an enhanced rendering.

Property 4. Sparse elevation order
A Bernstein polynomial of order $N-1$ can be represented with a linear combination of $dim+1$ Bernstein polynomials of order N as follows:

$$B^{N-1}_{\boldsymbol{\alpha}}(\hat{\boldsymbol{\lambda}}) = \sum_{i=0}^{dim} \frac{\alpha_i + 1}{N} B^N_{\boldsymbol{\alpha}+\boldsymbol{e}_i}(\hat{\boldsymbol{\lambda}})\,. \tag{2.28}$$

Proof:

$$\begin{aligned} B^{N-1}_{\boldsymbol{\alpha}}(\hat{\boldsymbol{\lambda}}) &= C^{N-1}_{\boldsymbol{\alpha}} \prod_{j=0}^{dim} \lambda_d^{\alpha_j}\,, = C^{N-1}_{\boldsymbol{\alpha}} \prod_{j=0}^{dim} \lambda_d^{\alpha_j} \sum_{i=0}^{dim} \hat{\lambda}_i\,, \\ &= \sum_{i=0}^{dim} C^{N-1}_{\boldsymbol{\alpha}} \prod_{j=0}^{dim} \lambda_d^{\alpha_j+\delta^i_j} = \sum_{i=0}^{dim} \frac{C^{N-1}_{\boldsymbol{\alpha}}}{C^N_{\boldsymbol{\alpha}+\boldsymbol{e}_i}} B^N_{\boldsymbol{\alpha}+\boldsymbol{e}_i}(\hat{\boldsymbol{\lambda}})\,, \\ &= \sum_{i=0}^{dim} \frac{\alpha_i + 1}{N} B^N_{\boldsymbol{\alpha}+\boldsymbol{e}_i}(\hat{\boldsymbol{\lambda}})\,, \end{aligned}$$

with $\boldsymbol{e}_i$ the canonical tuple, whose components are all zero except the i^{th} equaling 1.

We call this an elevation order because it enables to express the Bernstein polynomial coefficients from the basis generating $P^{N-1}(\hat{K})$ into the one generating $P^N(\hat{K})$. This operator is considered as sparse because it only comprises a linear combination of $dim + 1$ terms when most of the time we have: $dim + 1 << DoF$.

Property 5. Sparse derivative operator
The derivative of a Bernstein polynomial $B^N_{\boldsymbol{\alpha}}$ with respect to $\hat{\lambda}_i$ is given by:

$$\frac{\partial B^N_{\boldsymbol{\alpha}}}{\partial \hat{\lambda}_i}(\hat{\boldsymbol{\lambda}}) = N B^{N-1}_{\boldsymbol{\alpha}-\boldsymbol{e}_i}(\hat{\boldsymbol{\lambda}})\,.$$

Proof:

$$\frac{\partial B^N_{\boldsymbol{\alpha}}}{\partial \hat{\lambda}_i}(\hat{\boldsymbol{\lambda}}) = \frac{\partial}{\partial \hat{\lambda}_i} C^N_{\boldsymbol{\alpha}} \prod_{j=0}^{dim} \lambda_j^{\alpha_j} = C^N_{\boldsymbol{\alpha}} \alpha_i \prod_{j=0}^{dim} \lambda_j^{\alpha_j - \delta^i_j} = N B^{N-1}_{\boldsymbol{\alpha}-\boldsymbol{e}_i}(\hat{\boldsymbol{\lambda}}).$$

Coupling this result with Property 4, we get a sparse derivative operator by computing $dim + 1$ combination of Bernstein polynomials:

$$\frac{\partial B^N_{\boldsymbol{\alpha}}}{\partial \hat{\lambda}_i}(\hat{\boldsymbol{\lambda}}) = N B^{N-1}_{\boldsymbol{\alpha}-\boldsymbol{e}_i}(\hat{\boldsymbol{\lambda}}) = \sum_{j=0}^{dim} (\alpha_j + 1) B^N_{\boldsymbol{\alpha}-\boldsymbol{e}_i+\boldsymbol{e}_j}(\hat{\boldsymbol{\lambda}})\,. \tag{2.29}$$

This operator is cleary sparse since it allows to express the derivative of one element of the polynomial basis by one of the barycentric coordinate $\hat{\lambda}_i$ with a single linear combination of $dim+1$ terms of the other elements of the basis (2.29). This is sparse since $dim+1 << DoF$.

A discussion and experimental results will be addressed in the next section to quantify the sparsity of these operators.

Those properties enable us to compute differently the DG operators defined in Subsection 2.2.2. In the following part, we will use those properties, more precisely the two last properties, to define sparse operators in the semi-discrete DGm formulation.

2.3.2 Sparse operators using Bernstein polynomial basis

We developed in the previous sections a local DG semi-discrete formulation (2.21), which is recalled here:

$$\begin{cases} \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} = -\kappa \displaystyle\sum_{k=1}^{dim}(\sum_{d=1}^{dim}[J_{T_K}^{-\top}]_{k,d}\hat{M}^{-1}\hat{S}_{\hat{x_d}}\boldsymbol{V_h}_d) - \kappa \sum_{d=1}^{dim}\sum_{\Gamma\in K}\frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma {\mathcal{F}_p}^\Gamma + {\boldsymbol{F_h}}^K, \\ \dfrac{\partial \boldsymbol{V_h}_d{}^K}{\partial t} = -\dfrac{1}{\rho}\displaystyle\sum_{k=1}^{dim}[J_{T_K}^{-\top}]_{k,d}\hat{M}^{-1}\hat{S}_{\hat{x_d}}{\boldsymbol{P_h}}^K - \frac{1}{\rho}\sum_{\Gamma}\frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma \mathcal{F}_{v_d}^\Gamma \qquad (\text{for } d=1 \text{ to } dim)\,. \end{cases} \tag{2.30}$$

We introduce the reference derivative matrix operator $\hat{D}_d$ for $d = 1,$ to dim such that:

$$\frac{\partial {p_h}^K}{\partial \hat{x_d}} = \sum_{i=1}^{DoF}(\hat{D}_d {\boldsymbol{P_h}}^K)_i \hat{\varphi}_i,$$

with $(\hat{\varphi})$ any polynomial basis generating $P^N(\hat{K})$. It is then possible to replace this derivative operator in (2.30),

$$\begin{cases} \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} = -\kappa \displaystyle\sum_{k=1}^{dim}(\sum_{d=1}^{dim}[J_{T_K}^{-\top}]_{k,d}\hat{D}_d\boldsymbol{V_h}_d) - \kappa \sum_{d=1}^{dim}\sum_{\Gamma\in K}\frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma {\mathcal{F}_p}^\Gamma + {\boldsymbol{F_h}}^K, \\ \dfrac{\partial \boldsymbol{V_h}_d{}^K}{\partial t} = -\dfrac{1}{\rho}\displaystyle\sum_{k=1}^{dim}[J_{T_K}^{-\top}]_{k,d}\hat{D}_d{\boldsymbol{P_h}}^K - \frac{1}{\rho}\sum_{\Gamma}\frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma \mathcal{F}_{v_d}^\Gamma \qquad (\text{for } d=1 \text{ to } dim)\,. \end{cases}$$

By choosing the Bernstein polynomial basis we proved previously that this basis offers a sparse spatial derivative operators with respect to the barycentric coordinates. We have:

$$\frac{\partial B_{\boldsymbol{\alpha}}^N}{\partial \hat{\lambda}_i} = N B_{\boldsymbol{\alpha}-\boldsymbol{e}_i}^{N-1} = \sum_{j=0}^{dim}(\alpha_j+1)B_{\boldsymbol{\alpha}-\boldsymbol{e}_i+\boldsymbol{e}_j}^N\,.$$

Then we can write the derivative with respect to the barycentric coordinates of any polynomial on K such as:

$$\begin{aligned} \frac{\partial {p_h}^K}{\partial \hat{\lambda}_i} &= \frac{\partial}{\partial \hat{\lambda}_i}\sum_{\boldsymbol{\alpha}} {\boldsymbol{P_h}}_{\boldsymbol{\alpha}}^K B_{\boldsymbol{\alpha}}^N\,, \\ &= \sum_{\boldsymbol{\alpha}} {\boldsymbol{P_h}}_{\boldsymbol{\alpha}}^K \sum_{j=0}^{dim}(\alpha_j+1)B_{\boldsymbol{\alpha}-\boldsymbol{e}_i+\boldsymbol{e}_j}^N\,, \\ &= \sum_{\boldsymbol{\alpha}}(\hat{D}_{\hat{\lambda}_i}{\boldsymbol{P_h}}^K)_{\boldsymbol{\alpha}} B_{\boldsymbol{\alpha}}^N\,. \end{aligned}$$

The derivatives of the quantity ${p_h}^K$ with respect to $\hat{\lambda}_i$ are matrices containing $dim+1$ terms per row. For any degree of approximation N, the rows of the derivative operator keeps a constant amount of terms. This property ensures that the matrices are sparser and sparser for high degrees of approximation.

Knowing the geometrical transformation from cartesian coordinates $\hat{\boldsymbol{x}}$ to barycentric $\hat{\boldsymbol{\lambda}}$ expressed in 1D, 2D and 3D in (2.27), we have, by using derivative chain rule, these expressions of the regular derivative operators:

$$\frac{\partial}{\partial \hat{x_1}}p_h = (\frac{\partial}{\partial \hat{\lambda_1}} - \frac{\partial}{\partial \hat{\lambda_0}})p_h, \qquad \frac{\partial}{\partial \hat{x_2}}p_h = (\frac{\partial}{\partial \hat{\lambda_2}} - \frac{\partial}{\partial \hat{\lambda_0}})p_h, \qquad \frac{\partial}{\partial \hat{x_3}}p_h = (\frac{\partial}{\partial \hat{\lambda_3}} - \frac{\partial}{\partial \hat{\lambda_0}})p_h\,.$$

Then :

$$\hat{D}_{\hat{x_1}} = \hat{D}_{\hat{\lambda_1}} - \hat{D}_{\hat{\lambda_0}}, \qquad \hat{D}_{\hat{x_2}} = \hat{D}_{\hat{\lambda_2}} - \hat{D}_{\hat{\lambda_0}}, \qquad \hat{D}_{\hat{x_3}} = \hat{D}_{\hat{\lambda_3}} - \hat{D}_{\hat{\lambda_0}} .$$

For Lagrange polynomial basis, for instance, there is no underlying formula describing the derivative operator. This operator is then computed using the reference inverse mass matrix and the reference stiffness matrices described above. In this case we have then:

$$\hat{D}_{\hat{x_1}} = \hat{M}^{-1}\hat{S}_{\hat{x_1}}, \qquad \hat{D}_{\hat{x_2}} = \hat{M}^{-1}\hat{S}_{\hat{x_2}}, \qquad \hat{D}_{\hat{x_3}} = \hat{M}^{-1}\hat{S}_{\hat{x_3}} .$$

With such construction, knowing that $\hat{M}$ and $\hat{S}_{\hat{x_d}}$ are full matrices, there is no guarantee that the derivative operator from nodal polynomial basis has the same sparsity properties as Bernstein operators.

Concerning the surface terms, sparse operators can be constructed, still thanks to from the Bernstein polynomial basis properties. [Chan and Warburton, 2016] introduced the lift operator L^{Γ}:

$$L^{\Gamma} = \hat{M}^{-1}M_{\Gamma} = E_L^{\Gamma}L_0 ,$$

where E_L^{Γ} is the face reduction matrix and L_0 is a sparse matrix of size $DoF(\Gamma) \times DoF(\Gamma)$ where $DoF(\Gamma)$ represents the number of degrees of freedom on the face Γ.

To get the explicit expression of E_L^{Γ} and L_0 we refer to [Chan and Warburton, 2016]. It has to be noted that the sparse combination used to create those two operators is derived from the elevation order property of Bernstein polynomials.

Let us consider ${p_h}^K$ of order $N-1$ we have then :

$${p_h}^K = \sum_{\boldsymbol{\beta}} P_{h\boldsymbol{\beta}}^K B_{\boldsymbol{\beta}}^{N-1} ,$$

where $\boldsymbol{\beta}$ is one of the $dim+1$-tuple such that $\sum_{i=0}^{dim}\beta_i = N-1$. Then, by using the sparse elevation order (2.28), we have:

$$\begin{aligned} {p_h}^K &= \sum_{\boldsymbol{\beta}} P_{h\boldsymbol{\beta}}^K \sum_{i=0}^{N-1} \frac{\alpha_i+1}{N} B_{\boldsymbol{\beta}+\boldsymbol{e}_i}^N , \\ {p_h}^K &= \sum_{\boldsymbol{\beta}} P_{h\boldsymbol{\beta}} B_{\boldsymbol{\beta}}^{N-1} = \sum_{\boldsymbol{\alpha}} {P_h}'^K_{\boldsymbol{\alpha}} B_{\boldsymbol{\alpha}}^N . \end{aligned}$$

Here, $\boldsymbol{P_h}' = E_{N-1}^N \boldsymbol{P_h}$. E_{N-1}^N is a $DoF(N)\times DoF(N-1)$ matrix where $DoF(N)$ and $DoF(N-1)$ correspond respectively to the number of degrees of freedom per element of order N and $N-1$. We recall that the formulation of DoF as a function of N and dim is presented in Table 2.1 (see page 58). The elevation order operator is a sparse operator because, by construction, there is no more than $dim+1$ elements per row.

It is then possible to benefit from this advantage to construct recursively an elevation operator between $N-i$ and N with $i \leq N$ in such a way that:

$$E_{N-i}^N = E_{N-1}^N \, E_{N-2}^{N-1} \, ... \, E_{N-i+1}^{N-i+2} \, E_{N-i}^{N-i+1} .$$

The reduction order operator is obtained by taking the transpose of the elevation operator. We then have:

$$(E_{N-i}^N)^{\top} = (E_{N-i}^{N-i+1})^{\top} \, (E_{N-i+1}^{N-i+2})^{\top} \, ... \, (E_{N-2}^{N-1})^{\top} \, (E_{N-1}^N)^{\top} .$$

Such decomposition is a keystone to compute efficiently the surface operator E_L^Γ. We refer to [Chan and Warburton, 2016], to get the technical fully detailed algorithm to compute the lift operator L^Γ. This algorithm computes the lift matrix in $O(N^{dim})$ operations, and it is important to note that this operator is computed with $O(N^{2dim-1})$ operations using nodal basis.

It has been shown that strategies exist, thanks to Bernstein polynomial basis properties, to combine sparse operators in order to compute combination of the reference matrices developed in Subsection 2.2.2. The resulting sparse operator, $\hat{D}$ for the volume terms and L^Γ for the surface terms are both computed in $O(N^{dim})$ operations. The complexity is then optimal because the number of degrees of freedom also grows like $O(N^{dim})$.

In the next part we will validate the industrial DG solver and we will quantify the saving in computational time by comparing experiments in 1D, 2D and 3D.

2.3.3 1D Numerical Experiments

First of all, in order to better understand the DG technology and Bernstein polynomial basis we have developed a 1D acoustic propagator in Fortran90 from scratch. This 1D program solves the following equation on the 1D domain Ω pictured in Figure 2.26:

$$\begin{cases} \dfrac{\partial}{\partial t}p + \kappa \dfrac{\partial}{\partial x_1}v = f\,, & \text{in } \Omega \times [0, T_f]\,, \\ \rho \dfrac{\partial}{\partial t}v + \dfrac{\partial}{\partial x_1}p = 0\,, & \text{in } \Omega \times [0, T_f]\,, \\ p(t,0) = p(L,t)\,, & \text{in } \partial\Omega \times [0, T_f]\,, \\ v_{x_1}(t,0) = v(L,t)\,, & \text{in } \partial\Omega \times [0, T_f]\,, \\ p(0,x_1) = p_0(x_1)\,, & \text{in } \Omega\,, \\ v_{x_1}(0,x_1) = v_0(x_1)\,, & \text{in } \Omega\,. \end{cases} \tag{2.31}$$

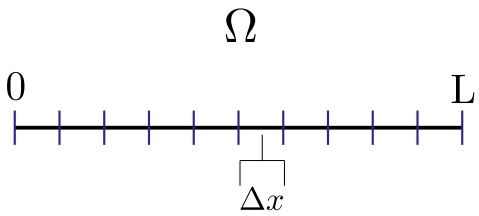


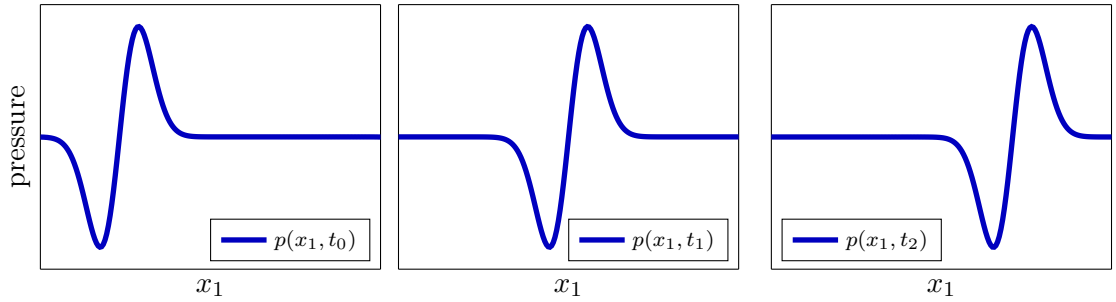
Figure 2.26: 1D domain illustration.

To validate the 1D propagator developed in this thesis, we choose periodic boundary conditions between $x_1 = 0$ (the left boundary) and $x_1 = L$ (the right boundary). We do not add source terms, but we choose to propagate an initial state $(p_0, v_{x_1 0})$. We choose an initial pressure and velocity state representing a first order Ricker function centered in x_0 of frequency peak of f_0:

$$p_0(x_1) = v_0(x_1) = (x_1 - x_0)e^{-(\pi f_0(x_1 - x_0))^2}\,.$$

In this specific case, when we have $p(x_1, 0) = v(x_1, 0)$ and with $\rho = 1$ and $\kappa = 1$ in Ω we are solving the acoustic equation that behaves as an advection problem (see Figure 2.27). Both equations on Ω in (2.31) are the same. We can then easily determine the analytical solution p_{ref}:

$$p_{ref}(t, x_1) = p_0((x_1 - t)[L])\,.$$



(a) Solution at time $t = t_0$. (b) Solution at time $t = t_1$. (c) Solution at time $t = t_2$.

Figure 2.27: Pressure behaving as an advection problem ($t_0 < t_1 < t_2$).

Since we have an analytical solution, we are able to validate the scheme developed by achieving a convergence study. We choose to use the RK4 time scheme to perform the convergence tests. For this purpose, we assess the relative L2 error between the analytical and the numerical solution while decreasing the size of the space discretization Δx. For each test, we determine the corresponding time step Δt respecting the CFL condition. The L2 error, ε_{L2}, defined in equation (2.32) quantifies the differences between the analytical and the numerical solutions at time t:

$$\varepsilon_{L2} = \frac{\sqrt{(\sum_i (p_{ref}(x_{1i}, t) - p_{num}(x_{1i}, t))^2)}}{\sqrt{(\sum_i (p_{ref}(x_{1i}, t)^2))}}\,. \tag{2.32}$$

The convergence study enables us to draw the convergence graphs in Figure 2.28. On the one hand, we observe that we have the exact same results whatever the polynomial basis is, either Bernstein or Lagrange polynomials. On the other hand, the slopes obtained by studying the $log(\varepsilon_{L2})$ as a function of $log(\Delta x_1)$ highlight that the developed method is well implemented since we retrieve the DGm convergence order of $N + 1$ [Arnold et al., 2002]. Surprisingly, we have a super-convergence behaviour for $N = 4$, where we expect to have a global maximal order of 4 determined by the time scheme. A smaller time step Δt than necessary can explain the phenomenon. In any case, those orders prove the validity of DGm implementation in 1D.

Now that we have proved the 1D program correctness, we illustrate the sparsity advantage offered by the Bernstein polynomial basis. We will more precisely show results concerning the sparse derivative operator, that it to say, the volume terms.

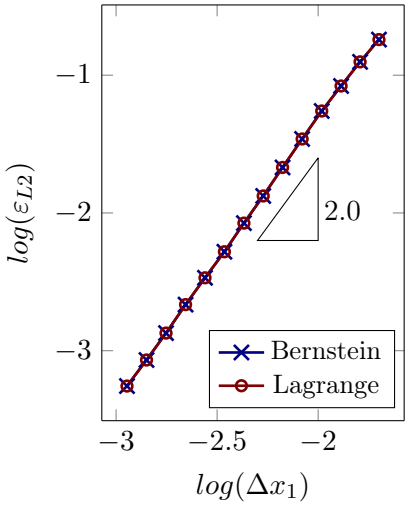
The quantity of interest here is the local derivative operator $\hat{D}_{\hat{x_1}}$. As explained above, there is no underlying decomposition to compute this operator using nodal polynomial basis. We are then computing $\hat{D}_{\hat{x_1}}$ such as:

$$\hat{D}_{\hat{x_1}} = \hat{M}^{-1}\hat{S}_{\hat{x_1}}.$$

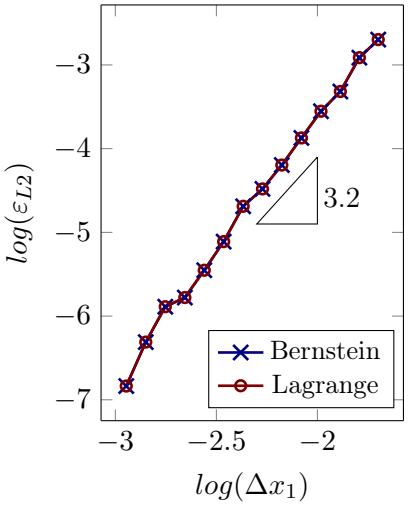
$\hat{M}^{-1}$ and $\hat{S}_{\hat{x_1}}$ are full matrices. There is then no guarantee that $\hat{D}_{\hat{x_1}}$ has a sparse pattern. Concerning Bernstein polynomials we have shown that:

$$\hat{D}_{\hat{x_1}} = \hat{D}_{\hat{\lambda_1}} - \hat{D}_{\hat{\lambda_0}}\,,$$

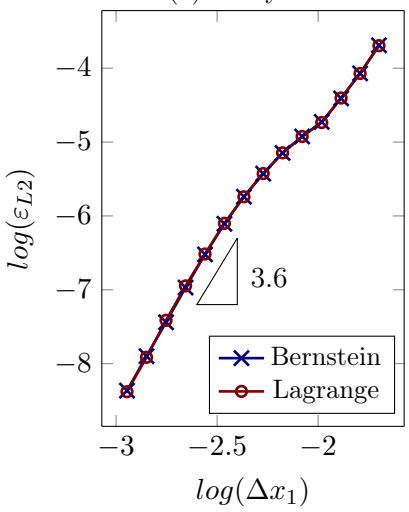
where $\hat{D}_{\hat{\lambda_1}}$ and $\hat{D}_{\hat{\lambda_0}}$ are sparse operators with only $dim + 1$ non-zero values per row. In Figure 2.29, we compare the sparseness of the local reference matrix $\hat{D}_{\hat{x_1}}$ using Lagrange and Bernstein polynomial basis for $N = 4$ and $N = 10$.



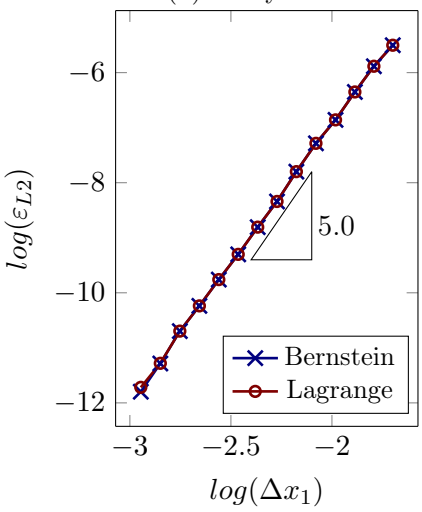
(a) Study for $N = 1$.



(b) Study for $N = 2$.

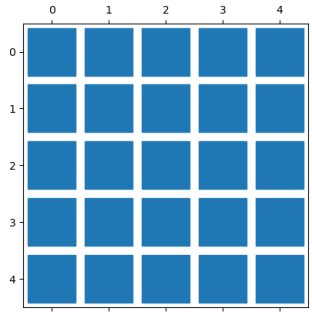


(c) Study for $N = 3$.

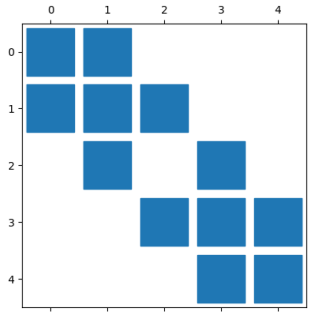


(d) Study for $N = 4$.

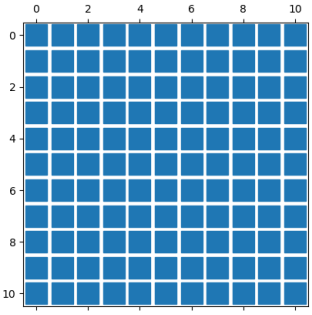
Figure 2.28: Convergence study for Bernstein and Lagrange polynomial bases with P1 (a) to P4 (d) elements.



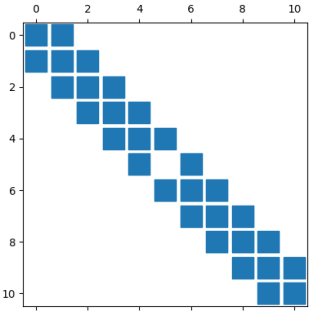
(a) $\hat{D}_{\hat{x_1}}$ for Lagrange and $N = 4$.



(b) $\hat{D}_{\hat{x_1}}$ for Bernstein and $N = 4$.



(c) $\hat{D}_{\hat{x_1}}$ for Lagrange and $N = 10$.



(d) $\hat{D}_{\hat{x_1}}$ for Bernstein and $N = 10$.

Figure 2.29: Sparsity comparison between Lagrange and Bernstein $\hat{D}_{\hat{x_1}}$ operator for $N = 4$ and $N = 10$.

Figure 2.29 clearly highlights the sparsity obtained by using Bernstein polynomial basis, but those results concern the local operator. To quantify the overall gain of the method, we propose to study the number of non-zero values (NZV) of the global volume term on a one hundred 1D cell problem for different orders of approximation. We display in Figure 2.30 the comparison of the NZV using Lagrange and Bernstein polynomial basis. In 1D, as the surface terms are trivial to compute (only one node as interface), the number of NZV of the volume operator gives an order of idea of the overall computational cost at each time step.

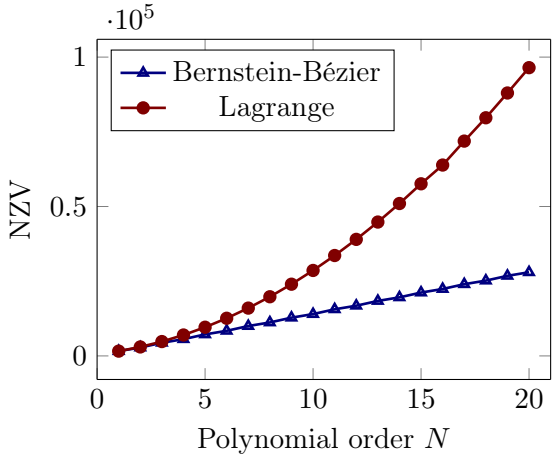


Figure 2.30: NZV of the global volume operator as a function of the polynomial order.

Figure 2.30 shows the superiority of Bernstein polynomial basis over Lagrange at high order N. In 1D, the graph attests of an interesting improvement starting at order $N = 5$. Below, Bernstein polynomials contribution is not significative in comparison with Lagrange polynomials.

Note that this heuristic results are only valid in 1D. For further comparison in 3D we refer to [Chan and Warburton, 2016], where an exhaustive comparison has been realized on GPUs. As far as this thesis is concerned, we are using Total DG propagator that we run on CPUs. The objective of the next section is to confirm the validity of the solver and to quantify the CPU time gain using Bernstein polynomial basis.

2.3.4 2D and 3D Numerical Experiments

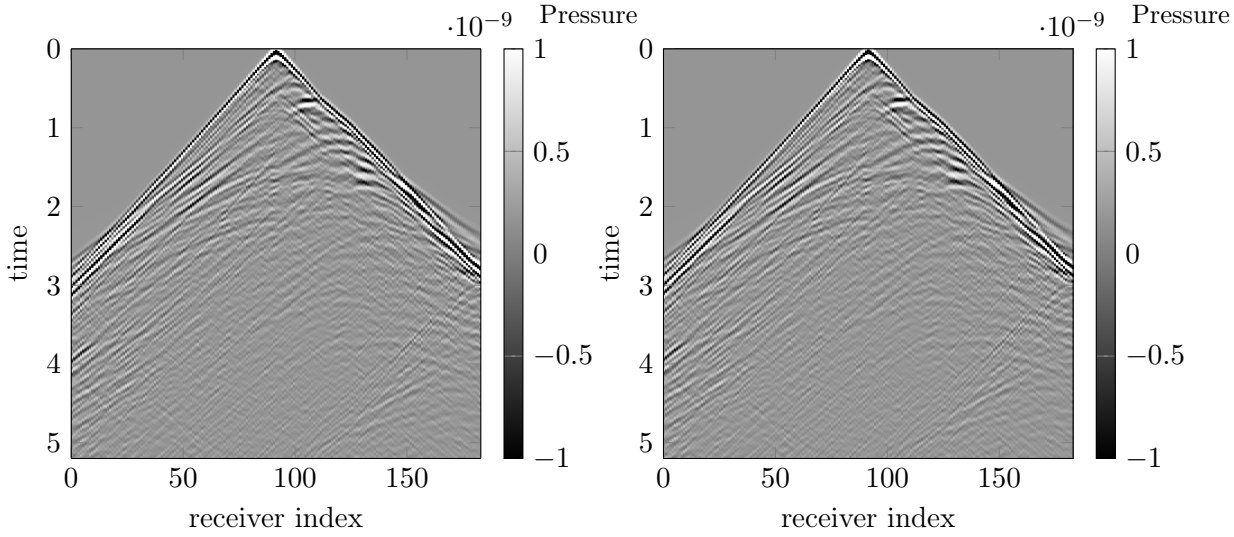
Before displaying CPU time comparison between Bernstein and nodal acoustic DG solver, we aim to show the accuracy of the solver whether we are using nodal or Bernstein polynomial basis. For that purpose, we will use again the Marmousi test introduced in Subsection 2.2.4 at page 71. In this section, we will compare the simulation performed with the Legendre nodal formulation with the same experiment using Bernstein-Bézier polynomial basis.

We remind that we study the propagation of a first order Ricker function perturbation on a source located at point (4550m,100m) in a domain of size 9200m $\times$ 3000m during 6.0 seconds. The Ricker function is determined by a $t_{peak} = 0.2$s and a $f_{peak} = 10$Hz. The wavespeed model $c(\boldsymbol{x})$ is the one from Marmousi while for the density we consider $\rho(\boldsymbol{x}) = 1000 kg.m^{-3}$. We will use a second order Runge Kutta time scheme with a time step Δt satisfying the CFL condition. The solutions will be compared at receivers that record the pressure perturbation over time. The 183 receivers used are placed in line at 50 meters depth from $x_1 = 50$m to $x_1 = 9150$m with a 50m gap in between each receiver. Concerning the boundary condition we used a free-surface condition at the top boundary ($p_{\Gamma_1} = 0$) and absorbing boundary condition

elsewhere $(\frac{\partial p}{\partial t}(t,\boldsymbol{x}) + c(\boldsymbol{x})\nabla p(t,\boldsymbol{x}).n = 0\,, \quad x \in \Gamma_2)$. The mesh generated for these tests is composed of 12980 triangles. For each element, we will use a polynomial approximation of order fourth.

We will then compare the seismogram obtained with the Bernstein polynomial basis with the reference seismogram and the one obtained with nodal simulation.

First of all, those seismograms look similar. We also display the trace associated to the 110^{th} receiver in Figure 2.32 for a better visualization. We can observe that the trace fit perfectly. As an indication, we also evaluate the relative L^2 norm of the error between the 2 seismograms. This error is equal to $1.6\,10^{-4}\%$, which highlights the high similarity between Bernstein and Lagrange computed solution with a ridiculous small relative error close to machine precision between them.



(a) Nodal seismogram.

(b) Bernstein seismogram.

Figure 2.31: Seismograms comparison between: (a) Numerical solution using Nodal polynomial basis, (b) Numerical solution using Bernstein polynomial basis.

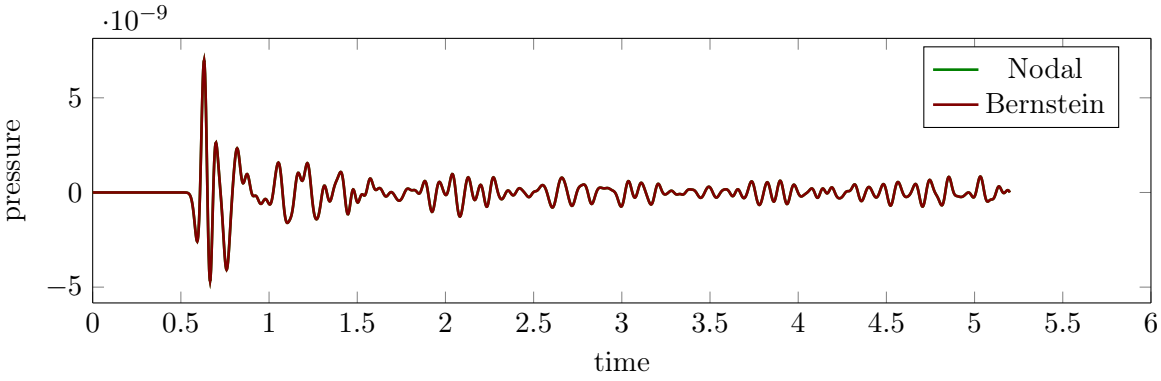


Figure 2.32: Trace comparison between Nodal and Modal numerical solution recorded by the 110^{th} receiver.

Since the Bernstein polynomial basis has been numerically validated, we define here an

experimental setup to compare the computational cost of DG modeling of acoustic wave equation using the two different polynomial bases.

Those comparisons will be held on three experimental setups. Two 2D cases are considered, one on Marmousi model and a second on Sigsbee model [Sig]. One 3D study is also treated on a truncated section of the SEAM Foothills model.

2D test on Marmousi model

Here we describe the experimental setup employed on Marmousi for nodal and modal comparisons:

- **Domain size:** 9200m $\times$ 3000m;
- **Wavespeed model:** c of Marmousi model (Figure 2.33);
- **Density model:** Constant $\rho = 1000$ kg.m^{-3};
- **Source:**
  - $n_{src} = 1$;
  - location : $x_{src} = (4550.0, 0.0, 10.0)$;
  - kind of sources : First order Ricker function $f_{peak} = 8$Hz, $t_{peak} = 0.2$s;
- **Simulation time:** $T_f = 4.0$s;
- **CPUs:** $n_{proc} = 120$.

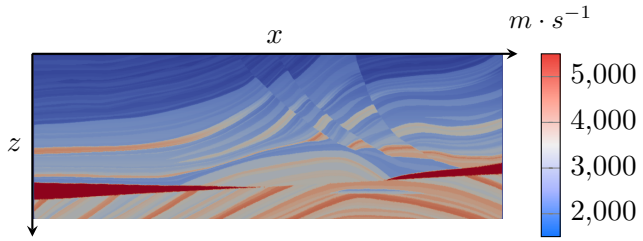


Figure 2.33: Marmousi wavespeed model (m.s^{-1}).

We compare, for different orders, the global computational time during a simulation lasting 4s, achieved with nodal and Bernstein polynomial bases. The considered global time is the total CPU time of the simulation where the communication times has been ignored. Then, it is principally defined by the addition of the computational time of the volume and surface terms. For each approximation order from 1 to 5, we define an adapted mesh, and we run the same experiment for both polynomial bases with the same mesh. In Table 2.5, we display the number of elements (N_e) and the global computational time for each simulation.

Marmousi	P1	P2	P3	P4	P5
Number of elements (N_e)	53077	23461	11804	6343	3455
Nodal CPU time (s)	1873	1459	1412	1401	1298
Bernstein CPU time (s)	3910	2676	1920	1423	1030
Ratio total CPU time (BB/Nodal)	2.1	1.8	1.4	1.0	0.8

Table 2.5: Global CPU time from nodal and Bernstein simulations on the Marmousi model.

The last row synthetises the ratio between the global computational time using Bernstein polynomial basis over the computational time using the nodal polynomial basis. It is clear that, for this 2D experiment, Bernstein basis is more expensive at low order (ratio > 1). The ratio is getting better for higher orders. It seems that we have comparable efficiency from P4 approximation and getting better after P4.

In order to verify that those figures do not depend on the current experiment, we provide the same comparison on a simulation made on the Sigsbee model.

2D test on Sigsbee model

We describe here the experimental setup employed on the Sigsbee model for nodal and modal comparisons:

- **Domain size:** 24348m $\times$ 9144m;
- **Wavespeed model:** c of Sigsbee model (see Figure 2.34);
- **Density model:** Constant $\rho = 1000\text{kg.m}^{-3}$;
- **Source:**
  - $n_{src} = 1$;
  - location : $x_{src} = (12000.0, 0.0, 10.0)$;
  - kind of sources : First order Ricker function $f_{peak} = 8\text{Hz}$, $t_{peak} = 0.2\text{s}$;
- **Simulated time:** $T_f = 4.0\text{s}$;
- **CPUs:** $n_{proc} = 120$.

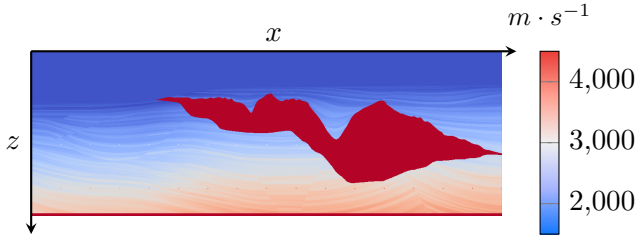


Figure 2.34: Sigsbee wavespeed model (m.s^{-1}).

We repeat the same experiment on the Sigsbee model. The global computational time is displayed in Table 2.6. The computational time is much more important than the one obtained with the Marmousi model but this is explained by the domain size that is much larger for the Sigsbee model.

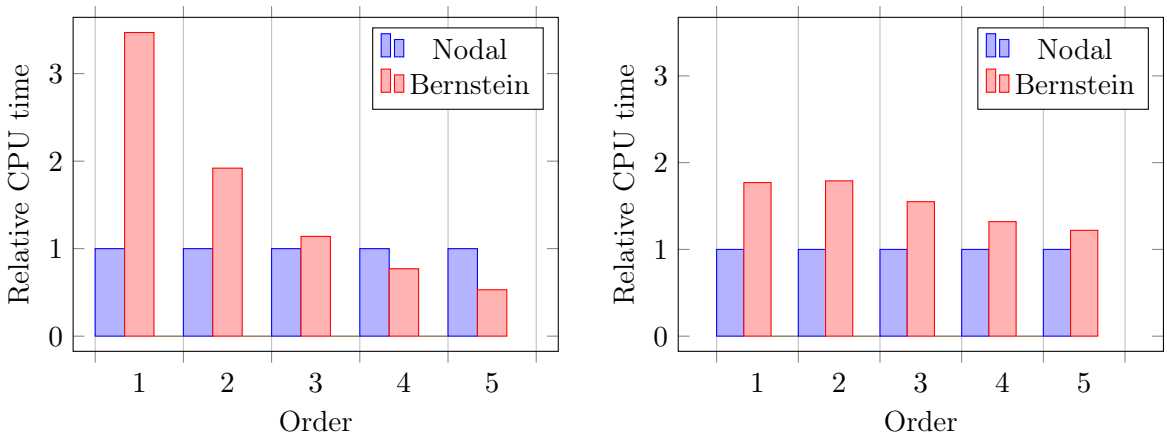
Sigsbee	P1	P2	P3	P4	P5
Number of elements (N_e)	569522	243636	121235	66644	35929
Nodal time computation (s)	20628	15938	14518	14649	13471
Bernstein Bézier time computation (s)	41976	27660	19944	15492	10960
Ratio total CPU time (BB/Nodal)	2.0	1.7	1.4	1.1	0.8

Table 2.6: Global CPU time from nodal and Bernstein simulations on the Sigsbee model.

Once again, the last row contains the main information of this table. The time ratios we get here are really similar to the ones obtained with the Marmousi experiment. We conclude that this behavior is symptomatic of 2D simulations performed with the solver we are using.

We can conclude that, in 2D, there is no advantage in using Bernstein polynomial basis instead of nodal basis for order below 5. Above this order, a computational gain is observed and is theoretically getting better for higher orders.

To figure out what is going on, we decompose the total computational time into the time associated to volume (Figure 2.35a) and surface terms (Figure 2.35b) in the two following histograms. We scaled the values by the CPU time from nodal experiments.



(a) Comparative histogram for volume terms time computation.

(b) Comparative histogram for surface terms time computation.

Figure 2.35: Comparative histograms for 2D volume and surface terms time computation.

Those histograms in Figure 2.35 compare the computational time between the two polynomial bases for the volume and the surface operators. Concerning the volume terms, it is clear that the Bernstein volume operator benefits from the sparsity offered by higher polynomial approximations. The higher the order is, the shorter the calculation time is. From P4 approximation, the 2D volume term has an interesting speed up in comparison with nodal operators.

Concerning the surface terms, the improvement by order elevation is less efficient. For every order studied, the computational cost of the surface terms using Bernstein polynomial is always more expensive.

For these 2D experiments, we showed that Bernstein polynomial basis can offer a speed up for higher order approximation. This enhancement is still moderate in view of the polynomial order considered (1 to 5).

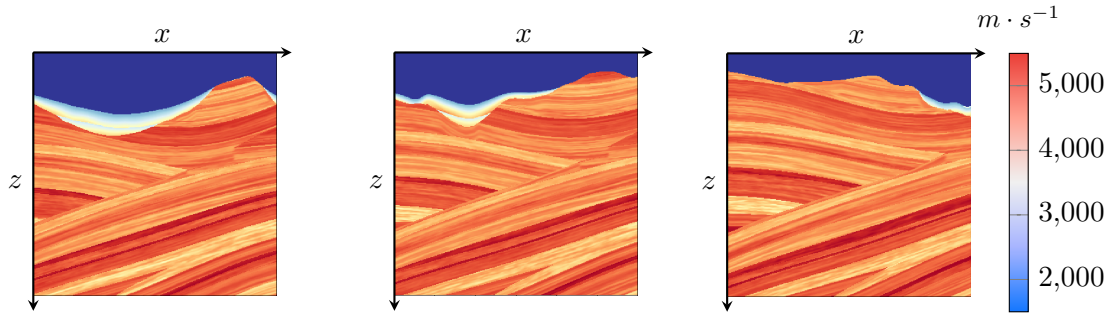
In the following, we address a 3D case to see if our conclusions still hold.

3D test on SEAM Foothills model

We describe here the experimental setup employed on the truncated SEAM Foothills model for nodal and modal comparisons:

- **SEAM Foothills model restricted to** : $(5000m, 5000m, 1500m)\times(8000m, 8000m, 5000m)$ subcube;
- **Wavespeed model:** c Seam Foothills model (see Figure 2.36);

- **Density model:** Constant $\rho = 1000$ kg.m^{-3};
- **Source:**
  - $n_{src} = 1$;
  - location : $x_{src} = (7000.0m, 7000.0m, 2002.0m)$;
  - kind of sources : First order Ricker function $f_{peak} = 8$Hz, $t_{peak} = 0.2$s;
- **Simulation time:** $T_f = 4.0$s.



(a) Section $y = 5000.0m$. (b) Section $y = 6500.0m$. (c) Section $y = 8000.0m$.

Figure 2.36: Seam-Foothills wavespeed model ($m.s^{-1}$).

In 3D, we aim to reproduce the same benchmark we did in 2D. We produce an adapted mesh, for the different polynomial orders (from 1 to 5). We display in Table 2.7 the resulting CPU time study.

Seam Foothills (reduced)	P1	P2	P3	P4	P5
Number of elements (N_e)	401424	122806	45932	20378	11299
Nodal CPU time (s)	42280	43080	46920	54120	69960
Bernstein CPU time (s)	100080	62640	43560	34200	33600
Ratio total CPU time (BB/Nodal)	2.2	1.5	0.9	0.6	0.5

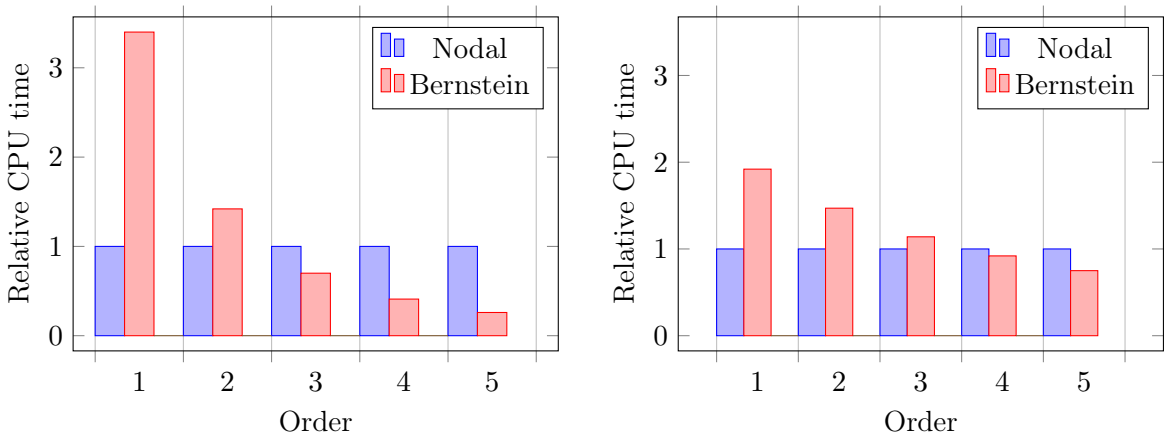
Table 2.7: Global CPU time from nodal and modal simulations on the reduced Seam foothills model.

The last row in Table 2.7, presents an interesting speed up from order 3. Below the order 3, the computational time using Bernstein polynomial basis is comparable or even worse than the one obtained using nodal basis. For P4 or P5 approximation, the overall computational time is twice shorter. To narrow the origin of the speed up, we display in the histogram (Figure 2.37) the relative time comparison between Bernstein and nodal basis for volume and source terms.

These histograms look the same in 2D and in 3D but the time savings is even stronger in 3D. The surface terms are slowly getting cheaper and cheaper with the polynomial order whereas the volume terms computation drops literally from 3 times more expensive for P1 elements to one quarter cheaper for P5 elements when using Bernstein polynomial basis.

Conclusion

We showed in this section that Bernstein polynomial bases have properties that enable to compute DG operators by combination of sparse matrices. For high order, such basis seems



(a) Comparative histogram for volume terms time computation.

(b) Comparative histogram for surface terms time computation.

Figure 2.37: Comparative histograms for 3D volume and surface terms time computation.

efficient to reduce the computational time of DG solvers. We led then some investigations on 1D, 2D and 3D acoustic problem to quantify the time savings that Bernstein polynomial basis can bring in comparison with nodal basis. Benchmarks have been realized using the industrial DG solver. They highlight, at least for the acoustic solver, that Bernstein-Bézier polynomial bases are interesting starting from P5 elements in 2D and from P3 in 3D. It is in 3D that Bernstein polynomial basis proved its true potential. Deeper investigations show that the sparse computation of the volume terms is the main asset which allows such performances.

2.4 Weight Adjusted Discontinuous Galerkin

To deal with physical parameters varying inside the elements, Mercerat and Glinsky [2015] proposed to use a weighted mass matrix. The weighted mass matrix coefficients are computed through quadrature rules and it does not affect stiffness matrices or source terms. Importantly, their technique is energy stable and high order accurate. However, because the wavespeed varies from element to element, each local weighted mass matrix is different. Thus, weighted mass matrices have to be inverted over each element for time-explicit schemes, which significantly increases storage costs. The idea of a weighted mass matrix was first introduced in [Koutschan et al., 2012] and later on in [Chan et al., 2017]. It is worth noting that Weight Adjusted DG (WADG) approximation only modifies the local mass matrix. Hence, much of the structure of DG methods can be reused from any prior DG implementation. WADG implementation distinguishes itself from DG one with the computation of a quadrature-based polynomial L^2 projection.

The main interest of considering a model varying by element is that we can avoid fine meshes for a decent approximation of the physical media. Methods such as WADG allows using coarser meshes with high order polynomial approximations which could reduce the overall computational cost while having a fine parameterized model.

In this section, we will develop the key idea of WADG and show how this can be easily implemented into an existing DGm solver. We will then show some results on a simple bilayered case for pedagogical purposes. Finally, we will discuss its efficiency in terms of computational time.

2.4.1 WADG formulation

In Subsection 2.2.2 describing the DG formulation of the acoustic wave equation, we introduced the local semi-discrete systems (2.33):

$$\begin{cases} \frac{1}{\kappa} M^K \frac{\partial \boldsymbol{P_h}^K}{\partial t} + \sum_{d=1}^{dim} S_{x_d}^K \boldsymbol{V_h}_d{}^K + \sum_{\Gamma} M_\Gamma^K \mathcal{F}_p^\Gamma(\boldsymbol{P_h}, \boldsymbol{V_h}) = \frac{1}{\kappa} M^K \boldsymbol{F_h}^K, \\ \rho M^K \frac{\partial \boldsymbol{V_h}_d^K}{\partial t} + S_{x_d}^K \boldsymbol{P_h}^K + \sum_{\Gamma} M_\Gamma \mathcal{F}_{v\,d}^\Gamma(\boldsymbol{P_h}, \boldsymbol{V_h}) = 0 \qquad (\text{for } d = 1 \text{ to } dim)\,, \end{cases} \tag{2.33}$$

that has to be solved for each element K.

In order to reduce the computational load, in particular by saving a large number of storage in memory, it is usual to assume that the physical parameters (ρ,κ) are constant per element K. Indeed, under this assumption, one can express the local mass matrix M^K as a function of a reference mass matrix $\hat{M}$ as follows:

$$M^K = |det(J_{T_K})| \hat{M}\,.$$

Such hypothesis defines a poor approximation of the model as illustrated in Figure 2.38. In particular, the model is discontinuous at the interface of the element creating artificial spurious reflections. One option to increase the resolution is to reduce the size of the elements which will increase drastically the computational cost. Moreover, it will prohibit the opportunity for speed-up offered by the use of coarse high order elements. There is thus a clear interest in accounting for physical parameters that vary inside the elements and this is what WADG formulations propose.

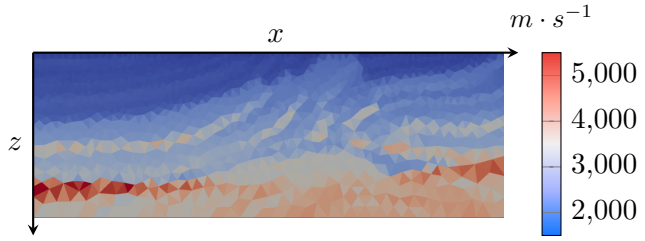


Figure 2.38: Illustration of the Marmousi wavespeed model represented on a 2000 elements unstructured mesh.

Let us denote by M_γ^K the local mass matrix weighted by the variable coefficient γ:

$$[M_\gamma^K]_{i,j} = \int_K \gamma(\boldsymbol{x}) \varphi_i^K(\boldsymbol{x}) \varphi_j^K(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.34}$$

In the acoustic case, γ can be either ρ or $\frac{1}{\kappa}$. Mercerat and Glinsky [2015] have considered the idea of computing all the local mass matrices defined in (2.34) for all elements. This approach has been proven to be energy stable but requires an expensive computational burden by inverting on the fly the local mass matrices besides an important memory use needed to store all of them.

WADG formulation consists in replacing the weighted mass matrices of Mercerat and Glinsky by approximate matrices defined through a weight-adjusted operator preserving the low-storage advantage of DGm. Chan et al. [2017] have proposed the following weight-adjusted approximation $\underset{\sim}{M}_\gamma^K$ of size $DoF \times DoF$:

$$M_\gamma^K \approx \tilde{M}{}_\gamma^K = M^K (M_{1/\gamma}^K)^{-1} M^K.$$

The inverse of the approximate operator is then:

$$(M_\gamma^K)^{-1} \approx (\tilde{M}{}_\gamma^K)^{-1} = (M^K)^{-1} M_{1/\gamma}^K (M^K)^{-1}.$$

This formula shows that for computing the inverse of M_γ^K, we can use $M_{1/\gamma}^K$. By plugging this approximation in the semi-discrete equation and introducing the reference mass matrix defined above, we have:

$$\begin{cases} \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} = -(M^K)^{-1} M_\kappa^K \left(\displaystyle\sum_{k=1}^{dim} \sum_{d=1}^{dim} [J_{T_K}^{-\top}]_{k,d} \hat{M}^{-1} \hat{S}_{\hat{x_d}} \boldsymbol{V_h}_d - \sum_{d=1}^{dim} \sum_{\Gamma \in K} \frac{|T_\Gamma|}{|T_K|} \hat{M}^{-1} \hat{M}_\Gamma \mathcal{F}_p{}^\Gamma \right) + \boldsymbol{F_h}^K \\ \dfrac{\partial \boldsymbol{V_h}_d{}^K}{\partial t} = -(M^K)^{-1} M_{1/\rho}^K \left(\displaystyle\sum_{k=1}^{dim} [J_{T_K}^{-\top}]_{k,d} \hat{M}^{-1} \hat{S}_{\hat{x_d}} \boldsymbol{P_h}^K - \sum_{\Gamma} \frac{|T_\Gamma|}{|T_K|} \hat{M}^{-1} \hat{M}_\Gamma \mathcal{F}_{v_d}^\Gamma \right), \end{cases} \tag{2.35}$$

for $d = 1$ to dim.

Let us now express each component of the local weight adjusted matrix using an exact quadrature for polynomials of order $2N+1$. We denote by $\hat{\boldsymbol{x}}_q$ and $\hat{\omega}_q$ the quadrature point location and its associated weight on the reference element. Then, we have:

$$\begin{aligned} [M_{\frac{1}{\gamma}}^K]_{i,j} &= \int_K \frac{1}{\gamma}(\boldsymbol{x}) \varphi_i^K(\boldsymbol{x}) \varphi_j^K(\boldsymbol{x}) d\boldsymbol{x}, \\ &= |T_K| \int_{\hat{K}} \frac{1}{\gamma}(T_K(\hat{\boldsymbol{x}})) \hat{\varphi}_i(\hat{\boldsymbol{x}}) \hat{\varphi}_j(\hat{\boldsymbol{x}}) d\hat{\boldsymbol{x}}, \\ &= |T_K| \sum_{q=1}^{n_q} \hat{\omega}_q \frac{1}{\gamma}(T_K(\hat{\boldsymbol{x}}_q)) \hat{\varphi}_i(\hat{\boldsymbol{x}}_q) \hat{\varphi}_j(\hat{\boldsymbol{x}}_q), \end{aligned}$$

where T_K is the bijection from the reference element $\hat{K}$ to the element K. We refer to Subsection 2.2.2 in page 62. We denote by n_q the number of quadrature points on the reference element $\hat{K}$. We introduce:

- $\boldsymbol{\gamma}^K$: vector of size n_q where $\boldsymbol{\gamma}_q^K$ represents $\gamma(T_K(\hat{\boldsymbol{x}}_q))$, where $\hat{\boldsymbol{x}}_q$ is the location of the q^{th} quadrature point on the reference element ;
- $\bar{Q}$: Matrix of size $DoF \times n_q$ such as $[Q]_{i,q} = \hat{\varphi}_i(\hat{\boldsymbol{x}}_q)$;
- $diag(\hat{\boldsymbol{\omega}} \frac{1}{\boldsymbol{\gamma}^K})$: diagonal matrix of size $n_q \times n_q$ where $diag(\hat{\boldsymbol{\omega}} \frac{1}{\boldsymbol{\gamma}})_{q,q} = \hat{\omega}_q \frac{1}{\gamma_q^K}$.

From these new notations, the weight-adjusted matrix can be computed as follows:

$$M_{\frac{1}{\gamma}}^K = |T_K| \, \bar{Q} \, diag(\hat{\boldsymbol{\omega}} \frac{1}{\boldsymbol{\gamma}^K}) \, \bar{Q}^\top \tag{2.36}$$

Considering the extra operator introduced in (2.35), we have:

$$(M^K)^{-1} M_{\frac{1}{\gamma}}^K = \hat{M}^{-1} \, \bar{Q} \, diag(\hat{\boldsymbol{\omega}} \frac{1}{\boldsymbol{\gamma}^K}) \, \bar{Q}^\top$$

Since $\bar{Q}$ and $\hat{\boldsymbol{\omega}}$ are local operators, the main memory burden of this method comes from the parameter values γ^K that requires $O(N^{dim})$ storage per element. This is what Guo and

Chan [2019] mention as a matrix-free fashion way to compute the inverse of the mass matrix containing the parameter information on each element. On the contrary, storing the full weighted mass matrix of Mercerat and Glinsky would imply an $O(N^{2dim})$ storage requirement.

The new semi-discrete system per element can be edited in such a way:

$$\begin{cases} \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} = -\hat{M}^{-1}\,\bar{Q}\,diag(\hat{\boldsymbol{\omega}}\boldsymbol{\kappa}^K)\,\bar{Q}^\top \left(\displaystyle\sum_{k=1}^{dim}\sum_{d=1}^{dim} [J_{T_K}^{-\top}]_{k,d}\hat{M}^{-1}\hat{S}_{\hat{x_d}}\boldsymbol{V_h}_d - \sum_{d=1}^{dim}\sum_{\Gamma\in K} \frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma \mathcal{F}_p{}^\Gamma \right) + \boldsymbol{F_h}^K \\ \dfrac{\partial \boldsymbol{V_h}_d{}^K}{\partial t} = -\hat{M}^{-1}\,\bar{Q}\,diag(\hat{\boldsymbol{\omega}}\frac{\mathbf{1}}{\boldsymbol{\rho}}^K)\,\bar{Q}^\top \left(\displaystyle\sum_{k=1}^{dim} [J_{T_K}^{-\top}]_{k,d}\hat{M}^{-1}\hat{S}_{\hat{x_d}}\boldsymbol{P_h}^K - \sum_{\Gamma} \frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma \mathcal{F}_{v_d}{}^\Gamma \right), \end{cases}$$

for $d = 1$ to dim.

This framework differs from classical DG one only by the model application that consists, in that case, in applying a scalar on each element. With WADG, we only have to replace the scalar multiplication by the matrix operator defined in (2.36). It is then relatively easy to plug the WADG parameterization in an existing DG solver. This operator requires $O(N^{2dim})$ extra operations per element, which should not be neglected in the overall computational process. The computation of the volume and surface terms are model independent.

2.4.2 Quadrature points

The quadrature formula used for WADG implementation is inspired from [Cools, 1999]. To ensure good accureacy for each component of the mass matrix:

$$[M^K_{\frac{1}{\gamma}}]_{i,j} = \int_K \frac{1}{\gamma}(\boldsymbol{x})\varphi_i^K(\boldsymbol{x})\varphi_j^K(\boldsymbol{x})d\boldsymbol{x}\,,$$

where $\varphi_i^K(\boldsymbol{x})$ and $\varphi_j^K(\boldsymbol{x})$ are polynomials of order N and by supposing that the model parameters is at least a linear function on K, it is required to choose a quadrature of order $N_q = 2N + 1$. This result is confirmed with numerical experiments in [Chan et al., 2017].

Therefore, the number of physical parameters strongly increases with the polynomial order approximation N. We display in Table 2.8 the number of quadrature points n_q required on the reference element $\hat{K}$ for several quadrature orders N_q.

N_q	1	2	3	4	5	6	7	8	9	10
n_q in 2D	1	3	6	6	7	12	15	16	19	25
n_q in 3D	1	4	6	11	14	23	31	44	57	74

Table 2.8: Number of quadrature points for several quadrature orders in 2D and 3D reference element (triangle and tetrahedron).

For high quadrature order, the amount of quadrature points can become important leading to increase both the memory and the computational burden of the solver more particularly in 3D. We illustrate the location of quadrature points displayed in 2D and 3D reference element in Figure 2.39 for quadrature order $N_q = 6$.

It is worth noting that WADG method is particularly interesting when addressing large scale problems as it is done for geophysical applications. It indeed offers the opportunity of using coarse meshes without loosing information on the physical parameters.

In the next part, we will present few numerical results of wave acoustic propagation based upon WADG formulation.

(a) Quadrature points on 2D reference elements. (b) Quadrature points on 3D reference elements.

Figure 2.39: Quadrature points on 2D and 3D reference element for $N_q = 6$.

2.4.3 Experimental results using WADG method

The WADG formulation was implemented in order to accurately account for the physical parameters describing the propagation domain when the propagation domain is discretized by a mesh composed of large cells and the parameters vary within the cells. This avoids numerical artifacts created by artificial reflections that are inherent to the coarse representation of the model by constant parameters per element. We propose to start this numerical study by comparing the seismograms obtained on a bilayered medium in three different configurations.

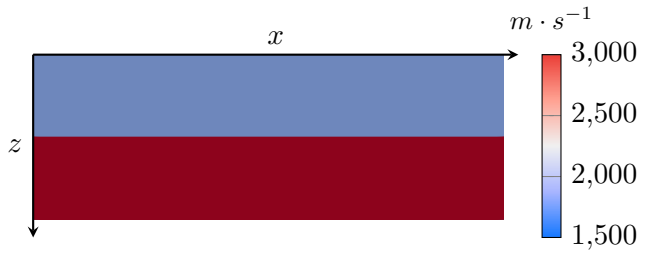
- **Case 1**: we use a mesh respecting the interface, and we use a model represented by constant parameters per element.
- **Case 2**: it is not always feasible to have a mesh following the interfaces, this case is using an isotropic mesh with no prior information of the interface. The model is represented as a constant model per element.
- **Case 3**: this case is using the same mesh from **case 2** but the model will be represented using WADG method.

For all those simulations, we will consider P4 elements. For the WADG implementation, we will use a quadrature of order 9 in order to respect the $2N + 1$ accuracy criterion. One source located at (4600m,60m) is perturbing the pressure wavefield with a first order Ricker function ($t_{peak} = 0.2$s, $f_{peak} = 10$Hz). A set of 183 receivers has been equally spaced on the X-axis from 50m to 9150m on a domain of size (9200m $\times$ 3000m). The interface is located at 1500m depth separating one wavespeed medium of 1500m.s^{-1} and an other of 3000m.s^{-1}.

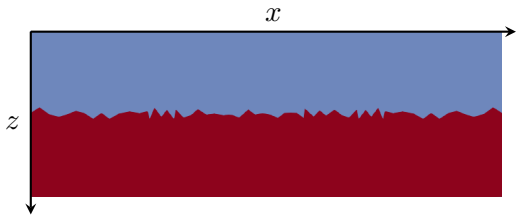
We display in Figure 2.40 an illustration of the different model representation for the three configurations.

In inverse problems, the location of the interfaces is most of the time not known precisely. In practice, we never have ideal meshes that fit the interface as the one used in Figure 2.40a. We clearly see in Figure 2.40b that large elements, may hamper the interface representation. Figure 2.40c shows the improvement of the interface shape which is very much closer to the plane (real) interface than in the case Figure 2.40b. Using WADG formulation clearly allows to improve the definition of the interface, hence reducing the chance of artificial reflections in the simulation. The mesh used for the two last configurations (**case 2** and **case 3**) is the same and comprises 2984 elements. Then, those two configurations are represented respectively by 2984 and 56696 parameters. For **case 1**, the mesh following the interface is constituted of 3500 elements.

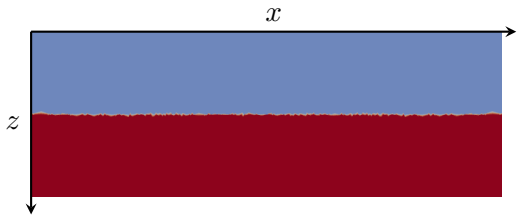
The simulation of the propagation is pictured in Figure 2.41. This experiment highlights the effects of the artificial reflectors induced by the misrepresentation of the model. In Figure 2.41b, the second, and expected, reflection is polluted by the extra artificial reflectors.



(a) BiLayEred wavespeed model for **case 1**.



(b) BiLayEred wavespeed model for **case 2**.



(c) BiLayEred wavespeed model for **case 3**.

Figure 2.40: Illustration of the wavespeed model for three different configurations: (a) **case 1**, (b) **case 2**, (c) **case 3**.

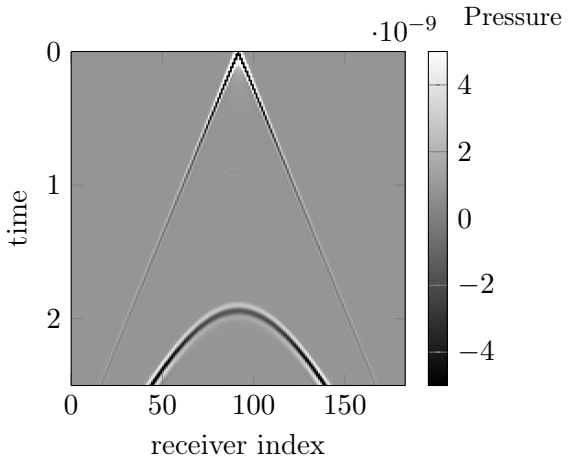
Even with a geometry as simple as an interface, the mesh does not always allow for a good approximation of the model. We can see in Figure 2.41c that the solution computed with WADG method is closer to the one obtained when the interface is perfectly modeled (Figure 2.41a) despite a coarse mesh. We observe that the WADG solution still suffers from residual oscillations coming from the coarseness of the mesh but those oscillations are attenuated in comparison with the ones in the solution computed with the mesh constructed with parameters constant per element. This new tool offers thus a possibility to mitigate errors coming from misrepresentation of the model due to space discretization.

Avoiding the artificial reflectors is crucial for achieving accurate FWI results. Indeed, FWI is based on the evaluation of a cost function that measures the difference between seismograms coming from observations and simulations. Hence, there is a clear interest in having accurate model approximation. Indeed, the underlying iterative process of minimization should be eased by avoiding spurious reflectors, since waves are a physical phenomenon very sensitive to the variations of the medium. This is clearly in favor of Weight Adjusted Discontinuous Galerkin approximations which provides a way to use a more accurate representation of the propagation domain. But WADG method is not the only way to improve the model representation.

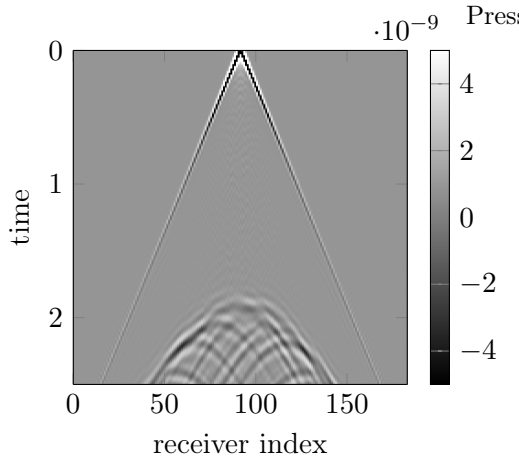
Instead of increasing the number of existing parameters per element (WADG), you can also simply have a finer mesh. This handling on the mesh is what is called h-adaptivity, which is made possible by the flexibility of DGm. However, reducing the element size would also affect the computational cost of the simulation and it will be not possible to benefit from the computational advantage of using larger elements at higher orders. In the following, we continue our analysis of the WADG method by carrying out numerical tests to evaluate the calculation times required by this technique.

2.4.4 Computational time study of WADG method

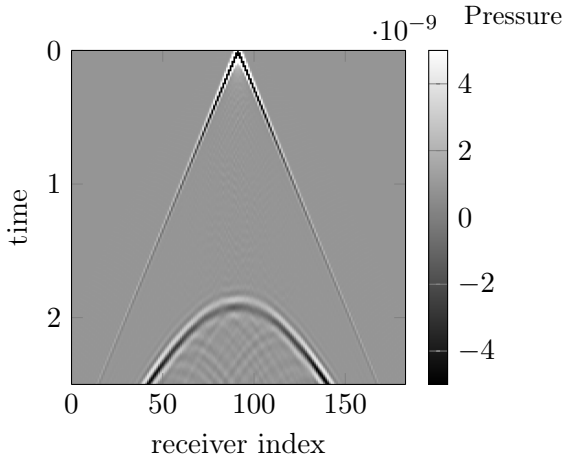
To quantify the computational cost of WADG method, given that this approach conists in providing a better reresentation of the propagation model, we will estimate the proportion



(a) Seismogram obtained using classical DG P4 elements with adapted mesh at the interface.



(b) Seismogram obtained using classical P4.



(c) Seismogram obtained using P4 Q9 elements.

Figure 2.41: Seismograms in a bilayered model represented in three different ways: (a) with **case 1** configuration, (b) with **case 2** configuration, (c) with **case 3** configuration.

time of calculation dedicated to model application. For that purpose, we repeat numerical experiments in the bi-layered domain described just above with P2 to P5 polynomial approximations in DG and WADG formulations. For a polynomial approximation of order N, we use a quadrature formula of order $N_q = 2N + 1$ as recommended in [Chan et al., 2017].

We recall that the problem to be solved reads:

$$\begin{cases} \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} = -\hat{M}^{-1}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\kappa}^K)\bar{Q}^\top \left(\displaystyle\sum_{k=1}^{dim}\sum_{d=1}^{dim} [J_{T_K}^{-\top}]_{k,d}\hat{M}^{-1}\hat{S}_{\hat{x}_d}\boldsymbol{V_h}_d \right. \\ \qquad\qquad \left. - \displaystyle\sum_{d=1}^{dim}\sum_{\Gamma\in K} \frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma \mathcal{F}_p^{\ \Gamma} \right) + \boldsymbol{F_h}^K, \\ \dfrac{\partial \boldsymbol{V_h}_d{}^K}{\partial t} = -\hat{M}^{-1}\bar{Q}diag(\hat{\boldsymbol{\omega}}\frac{\mathbf{1}}{\boldsymbol{\rho}}^K)\bar{Q}^\top \left(\displaystyle\sum_{k=1}^{dim} [J_{T_K}^{-\top}]_{k,d}\hat{M}^{-1}\hat{S}_{\hat{x}_d}\boldsymbol{P_h}^K - \sum_{\Gamma} \frac{|T_\Gamma|}{|T_K|}\hat{M}^{-1}\hat{M}_\Gamma \mathcal{F}_{vd}^{\ \Gamma} \right), \end{cases} \tag{2.37}$$

with $d = 1$ to dim.

Apart from the calculation of the source, we can see that the evaluation of the time derivative of the wave-field is step-halved:

- the evaluation of the volume and surface terms for all elements (terms in the parentheses);
- the model application, which consists in evaluating $y = -\hat{M}^{-1}\,\bar{Q}\,diag(\hat{\boldsymbol{\omega}}\frac{\mathbf{1}}{\boldsymbol{\gamma}}^K)\,\bar{Q}^\top x$ for all elements.

Remark. When classical DG is used (constant model per element), the application of the model is replaced by the product of the parenthesis by a scalar.

In the remaining of this section, we provide the proportion of computational time of WADG method compared with the one of DG method, in 2D and 3D. Most of the simulation computation time is spent on the evaluation of the right hand side of (2.37). We then export only the computational time needed to evaluate the temporal derivative of the wavefield.

Computational time assesment of WADG method in 2D

As we have just seen, Eqs. (2.37) show that the computational time is defined in two pieces:

- volume and surface terms evaluation (constant whatever we are using classical DG or WADG);
- model application.

In Table 2.9, we display the CPU time of these two key steps for different polynomial approximations and quadrature formulas in 2D.

Remark. It is worth noting that a quadrature formula of order 1 is equivalent to a classical DG method applied in a model parameterized with constant parameters per element.

Polynomial Order	2	3	4	5
CPU Time for $N_q = 1$ model computation (s)	22	15	8	5
CPU Time for $N_q = 2N + 1$ model computation (s)	86	111	76	67
Ratio CPU time (WADG/No WADG)	3.9	7.4	9.5	13.4
Proportion over total time ($N_q = 1$)	17.4%	13.4%	9.5%	7.1%
Proportion over total time ($N_q = 2N + 1$)	44.8%	53.3%	50.0%	50.7%

Table 2.9: CPU times comparison between constant and WADG model application in 2D.

This table clearly highlights the additional calculation cost caused by the 2D WADG operator. In the fourth row, we give the multiplicative factor to express the WADG calculation time as a function of the DG time. Depending on the polynomial order considered, the implementation of WADG operator is 4 to 13 times more expensive than considering a constant model per element. However, this ratio must be put into perspective of the total cost required for the simulation, i.e, here, in the evaluation of the wavefield time derivative.

When the model is constant per element, its application takes between 17% and 7% of the overall computational cost of the time derivative term, which is a reasonable proportion. Moreover, this fraction seems to decrease with the polynomial order. Which is explained by the fact that volume and surface terms are getting more and more expensive with the increase of the polynomial order. In contrast, the CPU time of WADG operator represents between 45% and 53% of the CPU time dedicated to compute the time derivative. This means that we increase the overall computational time and approximately 50% of this time is spent in the WADG operator. In addition, the cost of those operators does not seem to decrease with the polynomial approximation order. This is explained by the fact that the number of quadrature point is increasing with the polynomial approximation order.

The computational cost of the volume and surface terms does not depend on the quadrature formula used for the model approximation. The associated computational time is therefore the same whether you use the DG or WADG method. We display in the following histogram (see Figure 2.42), the repartition CPU of the two key steps for solving the problem. We scaled the values by the CPU time from classical DG experiments.

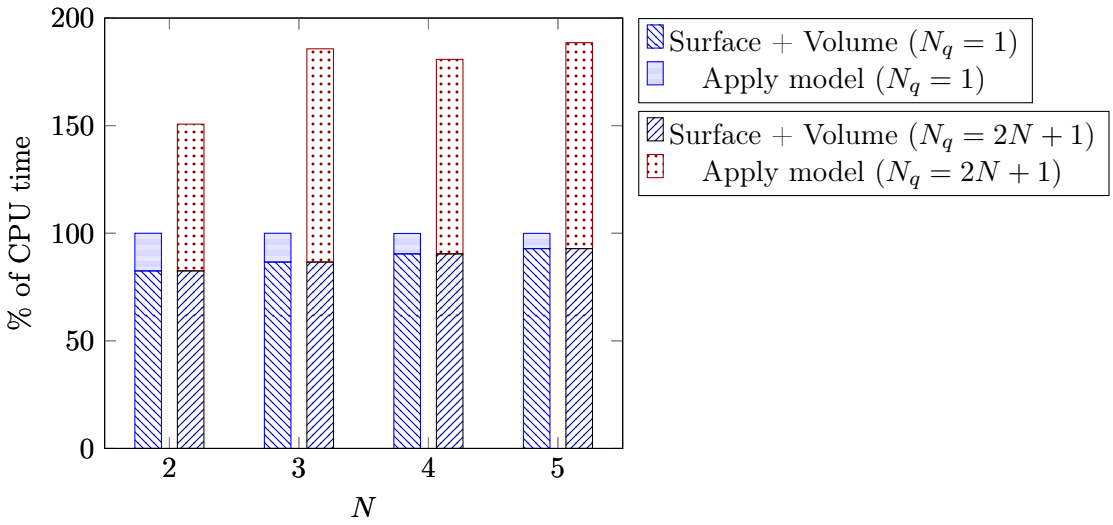


Figure 2.42: Histogram illustrating the computational proportion of the model operators in the time derivative evaluation in 2D.

We see that the WADG application in 2D contributes to increase significantly the CPU time, we actually have an augmentation of the CPU time of about 50% to 80%. In the next part, we propose the same study in 3D.

Computational time study of WADG method in 3D

We use again the SEAM Foothills setup we introduced in the previous section dealing with Bernstein-Bézier polynomial basis. We refer to page 91 for more details concerning the configuration of this experiment. The objective is the same as the one made in 2D. We performed

P2 to P4 simulations using nodal Legendre polynomial bases. We display in Table 2.10 the CPU time so obtained.

Polynomial Order	2	3	4
CPU Time for $N_q = 1$ model computation (s)	49	30	22
CPU Time for $N_q = 2N + 1$ model computation (s)	789	1062	1439
Ratio CPU time (WADG/No WADG)	16.1	35.4	65.4
Proportion over total time ($N_q = 1$)	3.0%	1.6%	0.9%
Proportion over total time ($N_q = 2N + 1$)	32.3%	36.8%	39.4%

Table 2.10: CPU times comparison between constant and WADG model application in 3D.

We arrive at the same conclusion as in 2D. If the model is constant per element, the proportion of the computation dedicated to the model application is small and represents between 1% to 3% of the overall computational time. The application of WADG operators takes between 16 and 65 times more CPU time depending on the polynomial and quadrature orders. Hence, the WADG application represents 32% to 39% of the overall computation in 3D for polynomial approximation orders varying from 2 to 4.

As before, we picture the changes brought by 3D WADG model application on the solution. In 3D, the time dedicated to volume and surface terms are much more important but does not depend on the model representation. In the histogram depicted in Figure 2.43, we display the computational time to compute the WADG solution scaled with the time obtained using constant model per element for P2 to P4 elements.

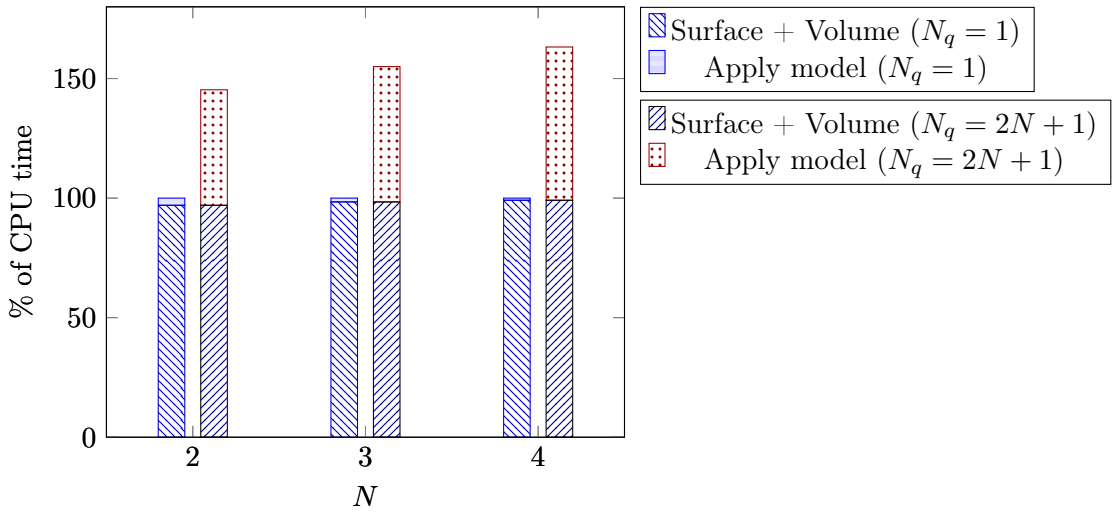


Figure 2.43: Histogram illustrating the computational proportion of the model operators in the time derivative evaluation in 3D.

As in 2D, the CPU time spent for WADG model application in 3D is a substantial part of the overall computational time. In addition, this part is getting more and more important while the polynomial approximation order increases.

Conclusion

In the time domain, the simulation consists in evaluating successively the time derivative of the wavefield. The corresponding CPU estimation we made on this evaluation is then repre-

sentative of the overall computational cost of the direct problem. If a WADG implementation ensures a better model definition, it requires non-negligible additional computational cost. We must, however, acknowledge that we conducted our study on CPUs and that the WADG method was designed for calculations on GPUs. Indeed, WADG implementation is based upon products of small matrices times local vectors, which is in favor to GPU architectures. But the objective of this thesis is to develop a time-domain FWI procedure that runs on CPUs. It could therefore be concluded that in this context, WADG method is not advisable. However, this method remains a very efficient way to improve the representation of the propagation model, which is, let us not forget, the solution to the inverse problem which is at the heart of this thesis. By being able to handle variable parameters in the elements, it also allows to use large cells, which represents a potential gain in computational time since the cost of implementing the WADG method does not seem to really increase with the order of polynomial approximation. There is then a trade-off between refinement and quadrature points in order to perform efficient simulations with accurate model approximation. We propose in Chapter 5, by the use mesh adaptation, to estimates this trade-off in the course of the FWI iterations. The inverse problem requires successive direct problem computations and it is then important to optimize the efficiency of the simulation more particularly in an industrial context.

As a perspective, we could try to change the basis functions in order to decrease the computational cost of WADG. For instance, Guo and Chan [2019] showed that WADG operator (2.36) can be expressed from combinations of sparse operators based upon Bernstein polynomial basis. Such an optimization would enhance the WADG complexity from $O(N^{2dim})$ to $O(N^{dim+1})$ using Bernstein polynomial basis. This feature is not implemented yet in the industrial code we are contributing to, but such a possible improvement seems quite conceivable since both Bernstein basis and WADG technique are already developed in the code.

Chapter 3

Characterization and numerical study of the adjoint state

We have seen in the first chapter of this manuscript how to solve an inverse problem with the adjoint state method. In the second chapter, we have defined the forward problem and more precisely its discretization using Discontinuous Galerkin method (DGm). In this chapter, we will define the adjoint state associated with the acoustic wave equation, which is either formulated as a first-order system or a second-order scalar equation. There are two standard approaches, namely the numerical adjoint state, which is the solution of the discrete adjoint problem or a more physical adjoint, which is the discrete adjoint problem. In [Chavent, 2010], the two strategies are respectively called: *Optimize then Discretize* and *Discretize then Optimize*, with the respective acronyms DtO and OtD. We display in Figure 3.1 a diagram showing how to switch from one formulation of the adjoint state to the other.

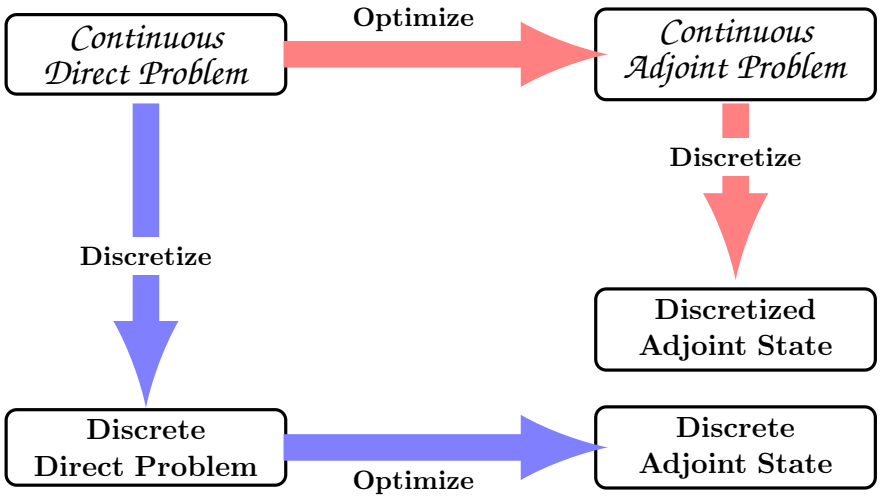


Figure 3.1: DtO and OtD diagram.

The objective of this chapter is to characterize the adjoint state and the resulting gradient and to investigate these two possibilities numerically. We will provide a numerical comparison of the two approaches to explain what led us to favor the *Optimize then Discretize* strategy for this thesis.

3.1 Characterization of the continuous adjoint state

In this section, we will focus on determining the continuous adjoint state. We will consider the second-order formulation of the acoustic wave equation, and we will deduce the continuous adjoint velocity-pressure field from the equivalence between both direct and adjoint formulations.

3.1.1 Second order scalar equation

We have seen in the previous chapter (see page 48) that the acoustic wave equation can be expressed either as a first-order equation system or as a second-order scalar equation. In this subsection, we will first construct the continuous adjoint state associated with the second-order scalar equation, which is defined as follows: find $p := p(t, \boldsymbol{x})$ the pressure field satisfying:

$$\begin{cases} \dfrac{1}{\kappa}\dfrac{\partial^2 p}{\partial t^2} - \nabla \cdot \left(\dfrac{1}{\rho}\nabla p\right) = g\,, & \text{in } [t_0, T_f] \times \Omega\,, \\ p(t_0, \boldsymbol{x}) = 0\,, \qquad \dfrac{\partial p}{\partial t}(t_0, \boldsymbol{x}) = 0\,, & \text{in } \Omega\,, \\ p = 0, & \text{in } [t_0, T_f] \times \Gamma_1\,, \\ \dfrac{1}{c}\dfrac{\partial p}{\partial t} + \partial_{\boldsymbol{n}} p = 0, & \text{in } [t_0, T_f] \times \Gamma_2\,, \end{cases} \tag{P_2}$$

where Γ_1 and Γ_2 represent respectively the upper and inner boundaries of the subsoil domain of interest Ω. In Figure 3.2, we display the domain Ω and its boundaries Γ_1 and Γ_2.

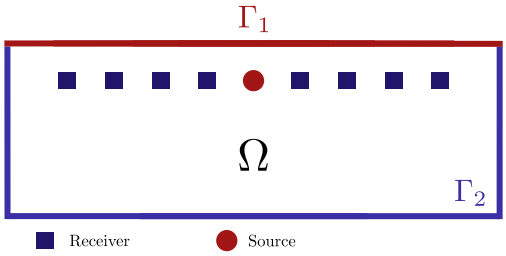


Figure 3.2: Truncated infinite domain.

The source g is defined with compact support in Ω and is assumed to be in $C^0([t_0, T_f]; L^2(\Omega))$. On the boundary Γ_2, one imposes an absorbing boundary condition and on Γ_1, which represents the terrestrial surface ($x_3 = 0$), one chooses a free surface condition $p = 0$.

We note d_{obs} the set of observations, and we introduce the cost function $\mathcal{J}$ given by:

$$\mathcal{J}(m) = \frac{1}{2}\int_{t_0}^{T_f}\int_{\Omega} (Qp(m) - d_{obs})^2 dx dt,$$

where $p(m)$ is the solution of the continuous problem (P_2) according to the set of parameter (m). The operator Q is the restriction operator from solution pressure field space to the observation space, this corresponding to the pressure fields recorded at the receivers. Numerically, we use the following formula:

$$(Qp(m) - d_{obs})^2 = \sum_{j=1}^{n_{rcv}} (Qp(m)_j - d_{obs\,j})^2\,,$$

where n_{rcv} corresponds to the number of receivers used for recording the observations.

We are interested in the following optimization problem:

$$\min_m \mathcal{J}(m), \qquad \text{under the constraint "}p \text{ is satisfying } (P_2)\text{".}$$

As we saw in Chapter 1, we can express this problem of minimization under constraint as a Lagrangian $\mathcal{L} := \mathcal{L}(p, m, p^*)$, with:

$$\mathcal{L}(p, m, p^*) = \hat{\mathcal{J}}(p) + \langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla \cdot (\frac{1}{\rho}\nabla p) - g, p^* \rangle \,, \tag{3.1}$$

where the inner product $\langle\ ,\ \rangle$ designates a scalar product on $L^2([t_0, T_f], \Omega)$. The simplest and widely used scalar product is defined as follows:

$$\forall (f, g) \in (L^2([t_0, T_f], \Omega))^2 \qquad \langle f, g \rangle = \int_{t_0}^{T_f} \int_\Omega fg dx dt.$$

It is worth noting that in the definition of the Lagrangian (3.1), we implicitly consider that p does not satisfy (P_2) since this is considered as a constraint. Hence, there is no reason for assuming that p depends on the physical parameters m. This is the reason why we define a new cost function $\hat{\mathcal{J}}$ that only depends on p:

$$\hat{\mathcal{J}}(p) = \frac{1}{2}\int_{t_0}^{T_f} \int_\Omega (Qp - d_{obs})^2 dx dt.$$

The minimum of $\mathcal{L} := \mathcal{L}(p, m, p^*)$ is necessary solution to:

$$\frac{\partial \mathcal{L}}{\partial p}(p, p^*, m) = 0\,, \qquad \frac{\partial \mathcal{L}}{\partial p^*}(p, p^*, m) = 0\,, \qquad \frac{\partial \mathcal{L}}{\partial m}(p, p^*, m) = 0\,.$$

Then, we have:

- By taking the derivative with respect to the adjoint state in the direction δ_{p^*}, at first glance, we can see that:

$$\frac{\partial \mathcal{L}}{\partial p^*}(p, m, p^*)\delta_{p^*} = \langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla \cdot (\frac{1}{\rho}\nabla p) - g, \delta_{p^*} \rangle \,,$$

  which shows that $\frac{\partial \mathcal{L}}{\partial p^*}(p, m, p^*) = 0$ ensures that, if (p, m, p^*) is a minimum, then p is solution of (P_2).

- Let us compute the derivative of $\mathcal{L}$ with respect to p, in the direction δ_p:

$$\frac{\partial \mathcal{L}}{\partial p}(p, m, p^*)\delta_p = \frac{\partial \hat{\mathcal{J}}}{\partial p}(p)\delta_p + \frac{\partial}{\partial p}\langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla \cdot (\frac{1}{\rho}\nabla p) - g, p^* \rangle \delta_p \,.$$

  Then we have:

$$\begin{aligned} \frac{\partial \hat{\mathcal{J}}}{\partial p}(p)\delta_p &= \frac{1}{2}(\langle Qp - d_{obs}, Q\delta_p \rangle + \langle Q\delta_p, Qp - d_{obs} \rangle)\,, \\ &= \langle Qp - d_{obs}, Q\delta_p \rangle\,, \\ &= \langle Q^*(Qp - d_{obs}), \delta_p \rangle\,. \end{aligned}$$

The derivative of the cost function $\hat{\mathcal{J}}$ with respect to the state variable p is given by the following expression:

$$\frac{\partial \hat{\mathcal{J}}}{\partial p}(p)\delta_p = \langle Q^*(Qp - d_{obs}), \delta_p \rangle \,. \tag{3.2}$$

To keep computing the derivative with respect to p, it is convenient to modify the second term in (3.1) in order to isolate p itself. For that purpose, we do integrate by parts:

$$\begin{aligned}
&\left\langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla\cdot\left(\frac{1}{\rho}\nabla p\right) - g, p^*\right\rangle = \int_{t_0}^{T_f}\int_\Omega \left(\frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla\cdot\left(\frac{1}{\rho}\nabla p\right) - g\right) p^* dxdt\,,\\
&= \int_\Omega \left[\frac{1}{\kappa}\partial_t p p^*\right]_{t_0}^{T_f} dx - \int_{t_0}^{T_f}\int_\Omega \frac{1}{\kappa}\partial_t p \partial_t p^* dxdt\\
&+ \int_{t_0}^{T_f}\int_\Omega \frac{1}{\rho}\nabla p\cdot\nabla p^* dxdt - \int_{t_0}^{T_f}\int_{\partial\Omega} \partial_{\boldsymbol{n}} p p^* dsdt\\
&- \int_{t_0}^{T_f}\int_\Omega g p^* dxdt\,,\\
&= \int_\Omega \left[\frac{1}{\kappa}\partial_t p p^*\right]_{t_0}^{T_f} dx - \int_\Omega \left[\frac{1}{\kappa} p\partial_t p^*\right]_{t_0}^{T_f} dx\\
&+ \int_{t_0}^{T_f}\int_\Omega \frac{1}{\kappa}\partial_t^2 p^* p dxdt - \int_{t_0}^{T_f}\int_\Omega \nabla\cdot\frac{1}{\rho}\nabla p^* p dxdt\\
&- \int_{t_0}^{T_f}\int_{\partial\Omega} \partial_{\boldsymbol{n}} p p^* dsdt + \int_{t_0}^{T_f}\int_{\partial\Omega} p\partial_{\boldsymbol{n}} p^* dsdt\\
&- \int_{t_0}^{T_f}\int_\Omega g p^* dxdt\,.
\end{aligned}$$

Since we have $p(t_0) = 0$ and $\dfrac{\partial p}{\partial t}(t_0) = 0$, we get:

$$\begin{aligned}
&\left\langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla\cdot\left(\frac{1}{\rho}\nabla p\right) - g, \delta_{p^*}\right\rangle = \left\langle \frac{1}{\kappa}\frac{\partial^2 p^*}{\partial t^2} - \nabla\cdot\left(\frac{1}{\rho}\nabla p^*\right) - g, p\right\rangle\\
&- \int_{t_0}^{T_f}\int_{\partial\Omega} \partial_{\boldsymbol{n}} p p^* + \int_{t_0}^{T_f}\int_{\partial\Omega} p\partial_{\boldsymbol{n}} p^*\\
&+ \int_\Omega \left(\frac{1}{\kappa}\partial_t p(T_f)p^*(T_f) - \frac{1}{\kappa}p(T_f)\partial_t p^*(T_f)\right) dx\,.
\end{aligned}$$

In addition, using the boundary conditions on Γ_2 yields:

$$\begin{aligned}
&\int_{t_0}^{T_f}\int_{\partial\Omega} \partial_{\boldsymbol{n}} p p^* dsdt = \int_{t_0}^{T_f}\int_{\Gamma_1} \partial_{\boldsymbol{n}} p p^* dsdt + \int_{t_0}^{T_f}\int_{\Gamma_2} \partial_{\boldsymbol{n}} p p^* dsdt\,,\\
&= \int_{t_0}^{T_f}\int_{\Gamma_1} \partial_{\boldsymbol{n}} p p^* dsdt - \int_{t_0}^{T_f}\int_{\Gamma_2} \frac{1}{c}\partial_t p p^* dsdt\,,\\
&= \int_{t_0}^{T_f}\int_{\Gamma_1} \partial_{\boldsymbol{n}} p p^* dsdt + \int_{t_0}^{T_f}\int_{\Gamma_2} \frac{1}{c} p\partial_t p^* dsdt - \left[\int_{\Gamma_2}\frac{1}{c} p p^* ds\right]_{t_0}^{T_f}\,,
\end{aligned}$$

and since $p = 0$ at time $t = t_0$, we have:

$$\int_{t_0}^{T_f}\int_{\partial\Omega} \partial_{\boldsymbol{n}} p p^* ds dt = \int_{t_0}^{T_f}\int_{\Gamma_1} \partial_{\boldsymbol{n}} p p^* ds dt + \int_{t_0}^{T_f}\int_{\Gamma_2} \frac{1}{c} p \partial_t p^* ds dt - \int_{\Gamma_2} \frac{1}{c} p(T_f) p^*(T_f) ds\,.$$

Then, if we assume that p^* satisfies:

$$\begin{cases} p^*(T_f, \boldsymbol{x}) = 0\,, & \text{in } \Omega\,, \\ \partial_t p^*(T_f, \boldsymbol{x}) = 0\,, & \text{in } \Omega\,, \\ p^* = 0\,, & \text{in } [t_0, T_f] \times \Gamma_1\,, \end{cases}$$

we end up with the following relation:

$$\begin{aligned} \left\langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla\cdot\left(\frac{1}{\rho}\nabla p\right) - g, p^*\right\rangle = \left\langle \frac{1}{\kappa}\frac{\partial^2 p^*}{\partial t^2} - \nabla\cdot(\frac{1}{\rho}\nabla p^*) - g, p\right\rangle \\ + \int_{t_0}^{T_f}\int_{\Gamma_2} p\left(\frac{1}{c}\partial_t p^* - \partial_{\boldsymbol{n}} p^*\right) ds dt\,, \end{aligned} \tag{3.3}$$

whose derivative with respect to p is easy to compute. By gathering results from (3.2) and (3.3), we obtain the derivative of the Lagrangian with respect to the state p in the direction δ_p:

$$\begin{aligned} \frac{\partial \mathcal{L}(p, m, p^*)}{\partial p}\delta_p &= \frac{\partial \hat{\mathcal{J}}}{\partial p}(p)\delta_p + \frac{\partial}{\partial p}\left\langle \frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla\cdot\left(\frac{1}{\rho}\nabla p\right) - g, p^*\right\rangle \delta_p\,, \\ &= \left\langle \frac{1}{\kappa}\frac{\partial^2 p^*}{\partial t^2} - \nabla\cdot(\frac{1}{\rho}\nabla p^*) + Q^*(Qp - d_{obs}), \delta_p\right\rangle \\ &+ \int_{t_0}^{T_f}\int_{\Gamma_2} \delta_p\left(\frac{1}{c}\partial_t p^* - \partial_{\boldsymbol{n}} p^*\right) ds dt\,. \end{aligned}$$

Then, we can define p^* such that $\frac{\partial \mathcal{L}}{\partial p}(p, m, p^*) = 0$ is the solution to the adjoint state problem:

$$\begin{cases} \dfrac{1}{\kappa}\dfrac{\partial^2 p^*}{\partial t^2} - \nabla\cdot\left(\dfrac{1}{\rho}\nabla p^*\right) = -Q^*(Qp - d_{obs})\,, & \text{in } [t_0, T_f] \times \Omega\,, \\ p^*(T_f, \boldsymbol{x}) = 0\,, \qquad \partial t p^*(T_f, \boldsymbol{x}) = 0\,, & \text{in } \Omega\,, \\ p^* = 0\,, & \text{in } [t_0, T_f] \times \Gamma_1\,, \\ -\dfrac{1}{c}\dfrac{\partial}{\partial_t} p^* + \partial_{\boldsymbol{n}} p^* = 0\,, & \text{in } [t_0, T_f] \times \Gamma_2\,. \end{cases}$$

We see that p^* is solution to a backward problem since its values are given at the final time T_f. However, it can be rewritten in the same form as the forward problem (P_2), by introducing the change of variable $t' \mapsto T_f + t_0 - t$ and the auxiliary unknown: $\tilde{p^*}(t', \boldsymbol{x}) = p^*(T_f + t_0 - t, x)$. Hence, we obtain:

Proposition 1. The adjoint state p^* such that $\dfrac{\partial \mathcal{L}}{\partial p}(p, m, p^*) = 0$ satisfies the adjoint state problem as follows:

$$\begin{cases} \dfrac{1}{\kappa}\dfrac{\partial^2 \tilde{p^*}}{\partial t'^2} - \nabla \cdot \left(\dfrac{1}{\rho}\nabla \tilde{p^*}\right) = -Q^*(Q((p - d_{obs})(T_f + t_0 - t'))) \,, & \text{in } [t_0, T_f] \times \Omega \,, \qquad (P_2^*) \\ \tilde{p^*}(t_0, \boldsymbol{x}) = 0 \,, \qquad \partial t' \tilde{p^*}(t_0, \boldsymbol{x}) = 0 \,, & \text{in } \Omega \,, \\ \tilde{p^*} = 0 \,, & \text{in } [t_0, T_f] \times \Gamma_1 \,, \\ \dfrac{1}{c}\partial_{t'} \tilde{p^*} + \partial_{\boldsymbol{n}} \tilde{p^*} = 0 \,, & \text{in } [t_0, T_f] \times \Gamma_2 \,. \end{cases}$$

It is worth noting that the continuous equations defined in (P_2) and (P_2^*) are identical except for the source term. This is a very interesting property that enable us to solve the forward and the backward problems with the same solver. Obtaining the adjoint state in the strategy *Optimize then Discretize* can therefore be done using the discrete operators constructed to approximate (P_2). This might be particularly interesting to avoid the development of adjoint operators that can be complicated to obtain.

Remark. The adjoint state solution of (P_2^*) is defined by the change of variable $t \mapsto T_f + t_0 - t$, which is equivalent to considering the evolution of p^* going from T_f to t_0. This is why the adjoint field is commonly called the "Backward" wavefield.

In this subsection, we have constructed an adjoint state as a solution to a boundary value problem that differs from the original problem only in the definition of the right-hand-side. In the numerical library we are developing, we solve the acoustic wave equation as a first-order system. This is actually convenient since this formulation gives access to the pressure and velocity directly. This is why, in the following, we construct an adjoint state for the first-order formulation.

3.1.2 First order system of equations

We recall the first order formulation of the acoustic wave equation:

$$\begin{cases} \dfrac{1}{\kappa}\dfrac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} = f \,, & \text{in } [t_0, T_f] \times \Omega \,, \\ \rho \dfrac{\partial \boldsymbol{v}}{\partial t} + \nabla p = 0 \,, & \text{in } [t_0, T_f] \times \Omega \,, \\ p = 0 \,, & \text{in } [t_0, T_f] \times \Gamma_1, \\ p - \rho c \boldsymbol{v} \cdot \boldsymbol{n} = 0 \,, & \text{in } [t_0, T_f] \times \Gamma_2, \\ p(t_0, \boldsymbol{x}) = 0 \,, \quad \boldsymbol{v}(t_0, \boldsymbol{x}) = 0 \,, & \text{in } \Omega. \end{cases} \qquad (3.4)$$

Contrary to the second-order equation, which is scalar, the first-order equation system governs a vector unknown whose components are the pressure field p and the velocity field $\boldsymbol{v}$.

We then introduce the global wavefield state variable $\boldsymbol{u}$ given by

$$\boldsymbol{u} = \begin{pmatrix} p \\ \boldsymbol{v} \end{pmatrix}$$

and defined in $L^2([t_0, T_f], \Omega))^{dim+1}$ as a solution to problem (3.1.2). As in the scalar case, we define the adjoint state variable $\boldsymbol{\lambda} \in (L^2([t_0, T_f], \Omega))^{dim+1}$ as the vector:

$$\boldsymbol{\lambda} = \begin{pmatrix} p^* \\ \boldsymbol{v^*} \end{pmatrix} .$$

The Lagrangian of the problem can then be expressed as follows:

$$\mathcal{L}(\boldsymbol{u}, m, \boldsymbol{\lambda}) = \hat{\mathcal{J}}(\boldsymbol{u}) + \left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} - f \\ \rho \frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \begin{pmatrix} p^* \\ \boldsymbol{v^*} \end{pmatrix} \right\rangle ,$$

where

$$\hat{\mathcal{J}}(\boldsymbol{u}) = \frac{1}{2}\int_{t_0}^{T_f}\int_{\Omega}(Qp - d_{obs})^2 dxdt. \tag{3.5}$$

The scalar product in $(L^2([t_0, T_f], \Omega))^{dim+1}$ is defined by:

$$\forall (\boldsymbol{f}, \boldsymbol{g}) \in (L^2([t_0, T_f], \Omega))^{dim+1}, \qquad \langle \boldsymbol{f}, \boldsymbol{g} \rangle = \sum_{i=1}^{dim+1}\int_{t_0}^{T_f}\int_{\Omega} f_i g_i dxdt = \sum_{i=1}^{dim+1}\int_{t_0}^{T_f}\int_{\Omega} \boldsymbol{f}\cdot\boldsymbol{g} dxdt\,.$$

As we did in the previous section, we will calculate the different derivatives of the Lagrangian with respect to $\boldsymbol{u}$, m, and $\boldsymbol{\lambda}$. Indeed, we seek for the minimum of a Lagrangian, which must satisfy the necessary conditions of having each partial derivative equal to zero.

- First of all, let us compute the derivatives with respect to the adjoint state:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}}(\boldsymbol{u}, m, \boldsymbol{\lambda})\delta_{\boldsymbol{\lambda}} = \left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} - f \\ \rho \frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \delta_{\boldsymbol{\lambda}} \right\rangle .$$

  Then we end up with the same formula as in the scalar case, which shows that $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}}(\boldsymbol{u}, m, \boldsymbol{\lambda}) = 0$ is sufficient to guarantee that $\boldsymbol{u}$ is solution to (3.1.2).

- Next, we compute the derivative of the Lagrangian with respect to the wavefield $\boldsymbol{u}$, which leads to:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}}(\boldsymbol{u}, m, \boldsymbol{\lambda})\delta_{\boldsymbol{u}} = \frac{\partial \hat{\mathcal{J}}(\boldsymbol{u})}{\partial \boldsymbol{u}}\delta_{\boldsymbol{u}} + \frac{\partial}{\partial \boldsymbol{u}}\left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} - f \\ \rho \frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \begin{pmatrix} p^* \\ \boldsymbol{v^*} \end{pmatrix} \right\rangle \delta_{\boldsymbol{u}}\,.$$

On the one hand, by computing the derivative of $\hat{\mathcal{J}}$, we get:

$$\frac{\partial \hat{\mathcal{J}}(\boldsymbol{u})}{\partial \boldsymbol{u}}\delta_{\boldsymbol{u}} = \langle \nabla_{\boldsymbol{u}}\hat{\mathcal{J}}, \delta_{\boldsymbol{u}} \rangle\,,$$

where

$$\nabla_{\boldsymbol{u}}\hat{\mathcal{J}} = \begin{pmatrix} \frac{\partial \hat{\mathcal{J}}}{\partial p} \\ \frac{\partial \hat{\mathcal{J}}}{\partial \boldsymbol{v}} \end{pmatrix} = \begin{pmatrix} Q^*(Qp - d_{obs}) \\ \mathbf{0} \end{pmatrix} .$$

On the other hand, we proceed as in the scalar case: by integrating by parts, we can isolate the state $\hat{\mathcal{J}}$. We have:

$$\left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{v} - f \\ \rho\frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \begin{pmatrix} p^* \\ \boldsymbol{v}^* \end{pmatrix} \right\rangle = \int_{t_0}^{T_f}\int_\Omega \left(\frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{v} - f\right) p^* dxdt$$

$$+ \int_{t_0}^{T_f}\int_\Omega \left(\rho\frac{\partial \boldsymbol{v}}{\partial t} + \nabla p\right)\cdot\boldsymbol{v}^* dxdt\,,$$

$$= -\int_{t_0}^{T_f}\int_\Omega \frac{1}{\kappa} p\frac{\partial p^*}{\partial t} dxdt + \left[\int_\Omega \frac{1}{\kappa} pp^* dxdt\right]_{t_0}^{T_f}$$

$$- \int_{t_0}^{T_f}\int_\Omega \boldsymbol{v}\cdot\nabla p^* dxdt + \int_{t_0}^{T_f}\int_{\partial\Omega} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt$$

$$- \int_{t_0}^{T_f}\int_\Omega f p^* dxdt$$

$$- \int_{t_0}^{T_f}\int_\Omega \rho\boldsymbol{v}\cdot\frac{\partial \boldsymbol{v}^*}{\partial t} dxdt + \left[\int_\Omega \rho\boldsymbol{v}\cdot\boldsymbol{v}^* dxdt\right]_{t_0}^{T_f}$$

$$- \int_{t_0}^{T_f}\int_\Omega p\nabla\cdot\boldsymbol{v}^* dxdt + \int_{t_0}^{T_f}\int_{\partial\Omega} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt.$$

On the boundary $\partial\Omega$, we have:

$$\int_{t_0}^{T_f}\int_{\partial\Omega} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\partial\Omega} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt = \int_{t_0}^{T_f}\int_{\Gamma_1} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\Gamma_1} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt$$
$$+ \int_{t_0}^{T_f}\int_{\Gamma_2} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\Gamma_2} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt.$$

Given that $p_{|\Gamma_1} = 0$ and assuming that $\boldsymbol{\lambda}$ satisfies ${p^*}_{|\Gamma_1} = 0$, we get:

$$\int_{t_0}^{T_f}\int_{\partial\Omega} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\partial\Omega} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt \tag{3.6}$$
$$= \int_{t_0}^{T_f}\int_{\Gamma_2} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\Gamma_2} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt$$

Since on Γ_2 the absorbing boundary condition is given by:

$$p - c\rho\boldsymbol{v}.\boldsymbol{n} = 0\,,$$

then, relation (3.6) becomes:

$$\int_{t_0}^{T_f}\int_{\partial\Omega} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\partial\Omega} p\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt = \int_{t_0}^{T_f}\int_{\Gamma_2} \boldsymbol{v}\cdot\boldsymbol{n} p^* dxdt + \int_{t_0}^{T_f}\int_{\Gamma_2} c\rho\boldsymbol{v}\cdot\boldsymbol{n}\boldsymbol{v}^*\cdot\boldsymbol{n} dxdt,$$

$$= \int_{t_0}^{T_f}\int_{\Gamma_2} \boldsymbol{v}\cdot\boldsymbol{n}(p^* + c\rho\boldsymbol{v}^*\cdot\boldsymbol{n}) dxdt.$$

Now, if we choose $\boldsymbol{\lambda}(T_f) = 0$ and ${p^*}_{|\Gamma_1} = 0$, we end up with:

$$\left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{v} \\ \rho\frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \begin{pmatrix} p^* \\ \boldsymbol{v}^* \end{pmatrix} \right\rangle = \left\langle \begin{pmatrix} p \\ \boldsymbol{v} \end{pmatrix}, \begin{pmatrix} -\frac{1}{\kappa}\frac{\partial p^*}{\partial t} - \nabla\cdot\boldsymbol{v}^* \\ \rho\frac{\partial \boldsymbol{v}^*}{\partial t} + \nabla p^* \end{pmatrix} \right\rangle \tag{3.7}$$
$$- \int_{t_0}^{T_f}\int_\Omega f p^* dxdt + \int_{t_0}^{T_f}\int_{\Gamma_2} \boldsymbol{v}\cdot\boldsymbol{n}(p^* + c\rho\boldsymbol{v}^*\cdot\boldsymbol{n}) dxdt\,.$$

By combining the calculation from (3.5) and (3.7), and choosing $\boldsymbol{\lambda}$ such that:

$$p^* + c\rho \boldsymbol{v}^* \cdot \boldsymbol{n} = 0\,,$$

then the derivative of the Lagrangian with respect to $\boldsymbol{u}$ reads:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{u}}\mathcal{L}(\boldsymbol{u}, m, \boldsymbol{\lambda})\delta_{\boldsymbol{u}} &= \frac{\partial}{\partial \boldsymbol{u}}\hat{\mathcal{J}}(m)\delta_{\boldsymbol{u}} + \frac{\partial}{\partial \boldsymbol{u}}\left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} \\ \rho\frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \begin{pmatrix} p^* \\ \boldsymbol{v}^* \end{pmatrix} \right\rangle \delta_{\boldsymbol{u}}\,, \\
&= \frac{\partial}{\partial \boldsymbol{u}}\left\langle \boldsymbol{u}, \begin{pmatrix} Q^*(Qp - d_{obs}) \\ \mathbf{0} \end{pmatrix} \right\rangle \delta_{\boldsymbol{u}} + \frac{\partial}{\partial \boldsymbol{u}}\left\langle \boldsymbol{u}, \begin{pmatrix} -\frac{1}{\kappa}\frac{\partial p^*}{\partial t} - \nabla \cdot \boldsymbol{v}^* \\ \rho\frac{\partial \boldsymbol{v}^*}{\partial t} + \nabla p^* \end{pmatrix} \right\rangle \delta_{\boldsymbol{u}}\,, \\
&= \left\langle \delta_{\boldsymbol{u}}, \begin{pmatrix} -\frac{1}{\kappa}\frac{\partial p^*}{\partial t} - \nabla \cdot \boldsymbol{v}^* + Q^*(Qp - d_{obs}) \\ -\rho\frac{\partial \boldsymbol{v}^*}{\partial t} - \nabla p^* \end{pmatrix} \right\rangle .
\end{aligned}$$

Thus, an adjoint state $\boldsymbol{\lambda}$ satisfying $\frac{\partial}{\partial \boldsymbol{u}}\mathcal{L}(\boldsymbol{u}, m, \boldsymbol{\lambda}) = 0$ can be defined as a solution to the following boundary value problem:

$$\begin{cases}
\frac{1}{\kappa}\frac{\partial p^*}{\partial t} + \nabla \cdot \boldsymbol{v}^* = Q^*(Qp - d_{obs})\,, & \text{in } [t_0, T_f] \times \Omega\,, \\
\rho\frac{\partial \boldsymbol{v}^*}{\partial t} + \nabla p^* = 0\,, & \text{in } [t_0, T_f] \times \Omega\,, \\
p^* = 0\,, & \text{in } [t_0, T_f] \times \Gamma_1\,, \\
p^* + \rho c \boldsymbol{v}^*.\boldsymbol{n} = 0\,, & \text{in } [t_0, T_f] \times \Gamma_2\,, \\
p^*(T_f, \boldsymbol{x}) = 0\,, \quad \boldsymbol{v}^*(T_f, \boldsymbol{x}) = 0\,, & \text{in } \Omega.
\end{cases}$$

We then introduce the change of variable $t' \mapsto T_f + t_0 - t$ and the auxiliary unknown $\tilde{\boldsymbol{\lambda}}(t', \boldsymbol{x}) = \boldsymbol{\lambda}(T_f + t_0 - t, x)$ and we obtain the following result:

Proposition 2. The adjoint state $\hat{\boldsymbol{\lambda}} = \begin{pmatrix} \tilde{p^*} \\ -\tilde{\boldsymbol{v}^*} \end{pmatrix}$ can be defined as a solution to:

$$\begin{cases}
\frac{1}{\kappa}\frac{\partial \hat{p^*}}{\partial t'} + \nabla \cdot \hat{\boldsymbol{v}^*} = -Q^*(Q((p - d_{obs})(T_f + t_0 - t'))\,, & \text{in } [t_0, T_f] \times \Omega\,, \\
\rho\frac{\partial \hat{\boldsymbol{v}^*}}{\partial t'} + \nabla \hat{p^*} = 0\,, & \text{in } [t_0, T_f] \times \Omega\,, \\
\hat{p^*} = 0\,, & \text{in } [t_0, T_f] \times \Gamma_1\,, \\
\hat{p^*} - c\rho \hat{\boldsymbol{v}^*} \cdot \boldsymbol{n} = 0\,, & \text{in } [t_0, T_f] \times \Gamma_2\,, \\
\hat{p^*}(T_f, \boldsymbol{x}) = 0\,, \quad \hat{\boldsymbol{v}^*}(T_f, \boldsymbol{x}) = 0\,, & \text{in } \Omega.
\end{cases}$$

As in the scalar case, we can define an adjoint state that is solution to a boundary value problem that differs from the original one only with the right-hand-side. Considering the *Optimize then discretize* strategy is thus very interesting regarding its numerical implementation since we can use the same discretized operator for both the forward and backward simulations.

We have defined the adjoint state in a formal way. To achieve its characterization, it is necessary to establish that the underlying boundary value problem is well-posed. This is the subject of the following section.

3.1.3 Wellposedness of first order formulation of the acoustic wave equation

In the two previous subsections, we have shown formally that the adjoint state can be defined as a solution of a boundary value problem that differs from the original one with the right-hand-side. Here, we are going to prove that the boundary value problem is well-posed and in this way, we will prove that the so-called adjoint state exists and is unique. We begin with the scalar wave equation and then address the case of the first-order formulation.

Existence and uniqueness of the solution to second order boundary value problem

We first consider the acoustic wave equation formulated as a second order scalar equation with the pressure field p as unknown. The problem we are interested in reads: for a given source g with support in $\Omega \times [t_0, T_f]$, and given initial data (p_0, p_1) defined at time $t = t_0$ in Ω, find p solution to:

$$\begin{cases} \dfrac{1}{\kappa}\dfrac{\partial^2 p}{\partial t^2} - \nabla \cdot \left(\dfrac{1}{\rho}\nabla p\right) = g\,, & \text{on } [t_0, T_f] \times \Omega\,, \\ p(t_0, \boldsymbol{x}) = p_0(\boldsymbol{x})\,, \qquad \dfrac{\partial p}{\partial t}(0, \boldsymbol{x}) = p_1(\boldsymbol{x})\,, & \text{on } \Omega\,, \\ p = 0\,, & \text{on } [t_0, T_f] \times \Gamma_1\,, \\ \dfrac{1}{c}\dfrac{\partial p}{\partial t} + \partial_{\boldsymbol{n}} p = 0\,, & \text{on } [t_0, T_f] \times \Gamma_2\,, \end{cases} \tag{P_2}$$

The computational domain Ω is an open bounded domain in $\mathbb{R}^{dim}$ with boundary $\partial\Omega$. The domain Ω is assumed being regular enough to have a normal vector field $\boldsymbol{n}$ at any point of $\partial\Omega$. For now, it is assumed that $\partial\Omega = \Gamma_1 \cup \Gamma_2$ and $\Gamma_1 \cap \Gamma_2 = \varnothing$.

The source g is given in $C^0([t_0, T_f], L^2(\Omega))$. The initial data (p_0, p_1) belong to H where $H = H^1_{0,\Gamma_1}(\Omega) \times L^2(\Omega)$ with $H^1_{0,\Gamma_1}(\Omega) = \{\phi \in H^1(\Omega), \phi = 0 \text{ on } \Gamma_1\}$. We provide H with the graph norm and the associated scalar product:

$$\forall V = (V_1, V_2) \in H, \qquad \| V \|^2_H = \int_\Omega \frac{1}{\rho}|\nabla V_1|^2 dx + \int_\Omega (V_2)^2 dx\,,$$

$$\forall (U, V) \in H \times H, \qquad \langle U, V \rangle_H = \int_\Omega \frac{1}{\sqrt{\rho}}\nabla U_1 \cdot \frac{1}{\sqrt{\rho}}\nabla V_1 dx + \int_\Omega U_2 V_2 dx\,,$$

It is worth noting that in $H^1_{0,\Gamma_1}(\Omega)$, the semi-norm $\| \frac{1}{\sqrt{\rho}}\nabla V_1 \|^2_{L^2}$ is a norm equivalent to the usual norm of $H^1(\Omega)$, assuming that $\rho > 0$. This result is a consequence of the Poincaré-Wirtinger's theorem.

Then, we obtain the following theorem:

Theorem 3.1.1. Let $g \in L^2([t_0, T_f], L^2(\Omega))$ and $(p_0, p_1) \in H$. Problem (P_2) admits a unique solution p and $p \in C^2([t_0, T_f], L^2(\Omega)) \cap C^1([t_0, T_f], H^1_{0,\Gamma_1}(\Omega)) \cap C^0([t_0, T_f], H^2(\Omega))$.

Sketch of the proof. This result has been demonstrated in [Barucq, 1993]. We resume the main steps of the demonstration based on Hille-Yosida's theorem (see [Dautray and Lions, 2012]). We introduce the operator A defined on H by:

$$A = \begin{pmatrix} 0 & I \\ \nabla \cdot \left(\frac{1}{\rho}\nabla\right) & 0 \end{pmatrix}.$$

Its domain $D(A)$ is defined in H by $D(A) = \{U = (U_1, U_2) \in H, AU \in H, \frac{1}{c}U_2 + \partial_{\boldsymbol{n}} U_1 = 0 \text{ on } \Gamma_2\}$.

Then, problem (P_2) can be rewritten as follows:

$$\frac{1}{\sqrt{\kappa}}\frac{\partial}{\partial t}P = AP + G\,, \tag{3.8}$$

where

$$P = \begin{pmatrix} p \\ \frac{1}{\sqrt{\kappa}}\frac{\partial p}{\partial t} \end{pmatrix}, \qquad G = \begin{pmatrix} 0 \\ g \end{pmatrix}.$$

To prove the existence and uniqueness of the solution of (3.8), we verify that the operator A is a maximal monotone operator in H:

(i) $\forall V \in D(A), \langle AV, V\rangle_H \leq 0$,

(ii) $\exists \lambda \in \mathbb{R}$ such that $A + \lambda I$ is surjective on H.

- point (i) can be demonstrated using Green formula, we have:

$$\begin{aligned} \forall V \in D(A),\ \langle AV, V\rangle_H &= \int_\Omega \frac{1}{\sqrt{\rho}}\nabla V_1 \cdot \frac{1}{\sqrt{\rho}}\nabla V_2 dx + \int_\Omega \left(\nabla \cdot \frac{1}{\rho}\nabla V_1\right) V_2 dx\,, \\ &= \int_{\partial\Omega} \frac{1}{\rho}\partial_{\boldsymbol{n}} V_1 V_2\,, \\ &= -\int_{\Gamma_2} \frac{1}{\rho c} V_2^2\,. \end{aligned}$$

- point (ii) results from the application of Lax-Milgram theorem, with any $\lambda < 0$.

We can remark that in general, Ω is a rectangle in 2D or a parallelepiped in 3D. Typically, Γ_1 is the upper boundary of Ω and we do not have $\Gamma_1 \cap \Gamma_2 = \varnothing$, as Γ_2 is the inner-surface introduced to truncate the propagation medium to get a bounded computational domain (see Figure 3.2). In practice, as we use discontinuous approximation spaces, the degrees of freedom at the junction between the two boundaries are different which amounts having $\Gamma_1 \cap \Gamma_2 = \varnothing$ and Theorem 3.1.1 remains valid. Now that we have demonstrated the existence and uniqueness of the solution of the second order boundary value problem, let us demonstrate the equivalence between the first and second order formulations.

Equivalence between first and second order boundary value problems

For our simulations, we use the first-order formulation using pressure and velocity fields. We recall its expression:

$$\begin{cases} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} = f\,, & \text{on } [t_0, T_f] \times \Omega\,, \\ \rho \frac{\partial \boldsymbol{v}}{\partial t} + \nabla p = 0\,, & \text{on } [t_0, T_f] \times \Omega\,, \\ p = 0\,, & \text{on } [t_0, T_f] \times \Gamma_1\,, \\ p - c\rho \boldsymbol{v} \cdot \boldsymbol{n} = 0\,, & \text{on } [t_0, T_f] \times \Gamma_2\,, \\ p(t_0, \boldsymbol{x}) = 0\,, \quad \boldsymbol{v}(t_0, \boldsymbol{x}) = 0\,, & \text{on } \Omega. \end{cases} \tag{P_1}$$

We will first assume that the initial conditions are zero. We seek to establish the equivalence between the two formulations ((P_2) and (P_1)). For that purpose, we will first show that if $(\boldsymbol{v}, p)$ satisfies (P_1), then p satisfies (P_2).

- If $(\boldsymbol{v}, p)$ is solution to (P_1), by deriving the first equation with respect to the time, we obtain:

$$\frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} + \nabla \cdot \frac{\partial \boldsymbol{v}}{\partial t} = \frac{\partial f}{\partial t}\,, \qquad \text{on } [t_0, T_f] \times \Omega\,.$$

  By injecting the expression of $\frac{\partial \boldsymbol{v}}{\partial t}$, we get:

$$\frac{1}{\kappa}\frac{\partial^2 p}{\partial t^2} - \nabla \cdot \frac{1}{\rho}\nabla p = \frac{\partial f}{\partial t}\,.$$

  Then, the equation that governs (P_2) can be found by imposing $g = \frac{\partial f}{\partial t}$.

  Regarding the initial conditions of (P_2), if we consider $(\tilde{p_0}, \boldsymbol{v}_0)$ designating the initial state of (P_1), one must impose:

$$p_0 = \tilde{p_0}, \qquad p_1 = -\kappa \nabla \cdot \boldsymbol{v}_0\,, \qquad \text{on } \Omega\,.$$

  The boundary condition at the boundaries Γ_2 can be written in the following way by taking the time derivative:

$$\begin{aligned} \frac{\partial}{\partial t}(p - c\rho \boldsymbol{v} \cdot \boldsymbol{n}) &= 0\,, \\ \frac{\partial p}{\partial t} - c\rho \frac{\partial \boldsymbol{v}}{\partial t} \cdot \boldsymbol{n} &= 0\,, \\ \frac{\partial p}{\partial t} + c\partial_{\boldsymbol{n}} p &= 0\,. \end{aligned}$$

  We then recover the expression of the second order absorbing boundary condition.

  Let us move on to the converse, which consists in checking that if p is solution of (P_2), then we can construct a vector field $\boldsymbol{v}$ such that $(\boldsymbol{v}, p)$ is solution to (P_1).

- Let p be a solution to (P_2). We integrate in time the volume equation. Assuming that the initial data are zero, we have:

$$\frac{1}{\kappa}\frac{\partial p}{\partial t} - \int_{t_0}^{t} \nabla \cdot \frac{1}{\rho}\nabla p = \int_{t_0}^{t} g(s, x) ds\,.$$

  We choose:

$$g(t,x) = \int_{t_0}^{t} f(s,x)ds \text{ and } v(t,x) = \frac{1}{\rho}\int_{t_0}^{t} \nabla p(s,x)ds\,.$$

We then obtain:

$$\frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla\cdot\boldsymbol{v} = f\,,\ \text{on } [t_0,T_f]\times\Omega\,.$$

In addition, by construction we have:

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \nabla p = 0\,,\ \text{on } [t_0,T_f]\times\Omega\,.$$

It has therefore been proven that, if we set:

$$\boldsymbol{v}(t,x) = -\frac{1}{\rho}\int_{t_0} \nabla p(s,x)ds\,,$$

then problem (P_2) is equivalent to problem (P_1), under the assumption that the initial conditions are zero and that the source terms f and g are linked by the relation $g = \dfrac{\partial f}{\partial t}$.

As a conclusion, if the initial data are zero and if p is the solution to (P_2), then the couple $(\boldsymbol{v},p)$ is solution to (P_1) with:

$$v(t,x) = -\frac{1}{\rho}\nabla p(s,x)ds\,,\qquad f(t,x) = \int_{t_0}^{t} g(s,x)ds\,,\qquad \text{on } [t_0,T_f]\times\Omega\,. \tag{3.9}$$

Conversely, if $(\boldsymbol{v},p)$ is solution to (P_1) then we have (3.9) and p is solution to (P_2) with $g(t,x) = \dfrac{\partial f}{\partial t}$.

If the initial conditions are non zero, we can notice that:

$$\begin{cases} \dfrac{1}{c}\dfrac{\partial p}{\partial t} + \partial_{\boldsymbol{n}} p = 0\,, & \text{on } [t_0,T_f]\times\Gamma_2\,,\\ \boldsymbol{v}(t,x) = -\dfrac{1}{\rho}\displaystyle\int_{t_0}^{t} \nabla p(s,x)ds\,, \end{cases}$$

implies

$$\frac{1}{c}p + \boldsymbol{v}\cdot\boldsymbol{n} = \frac{1}{c}p_0 + \boldsymbol{v}_0\cdot\boldsymbol{n}\,,\ \text{on } [t_0,T_f]\times\Gamma_2\,.$$

Hence, if p_0 and $\boldsymbol{v}_0$ have a support strictly included in Ω, we have

$$\frac{1}{c}p_0 + \boldsymbol{v}_0\cdot\boldsymbol{n} = 0\,,\ \text{on } \Gamma_2\,,$$

so that both problems (P_2) and (P_1) are equivalent.

Finally, by equivalence of the two problems and as a corollary of Theorem 3.1.1, we get:

Corollary 3.1.1. Let $f \in C^1([t_0,T_f];L^2(\Omega))$. Let $(p_0,\boldsymbol{v}_0) \in H^1_{0,\Gamma_1}\times H(div,\Omega)$. Problem ($P_1$) admits a unique solution $(p,\boldsymbol{v})$ such that p is solution to the second-order equation (P_2) and

$$v(t,x) = -\frac{1}{\rho}\nabla p(s,x)ds\,,\ \text{on } [t_0,T_f]\times\Omega\,.$$

Corollary 3.1.1 states the well-posedness of the first order boundary value problem, which defines both the forward and backward state with different right-hand sides.

Now that we have defined the continuous adjoint problem and have demonstrated its well-posedness, we propose in the following section to do an analogue work for the *Discretize then Optimize* strategy.

3.2 Characterization of the discrete adjoint state

In the previous section, we characterized the adjoint state by considering first the forward continuous problem (*Optimize then Discretize*). The adjoint state is then defined exactly through a backward continuous problem and this approach is convenient because both the forward and the backward problems have the same continuous expression, with different right-hand sides. This is an important feature regarding numerical implementation since it requires the discretization of the same set of operators for the two problems.

Another approach called *Discretize then Optimize* allows to obtain an exact gradient of the discrete problem. The objective of this section is to present the key ideas on how to define the discrete adjoint problem from the discrete direct problem constructed with DGm, in order to see what are the difficulties arising from the discretization.

3.2.1 Definition of the global linear system

In the previous chapter, we introduced the DG space discretization of the first-order formulation of the acoustic wave equation. We have introduced a set of notation, mainly in pages 53 and 63, that are necessary for the understanding of this section. As a reminder, the discrete problem is defined as follows: find

- the approximate pressure wavefield p_h,
- the approximate velocity wavefield $\boldsymbol{v_h}$,

belonging to their respective approximation spaces $\mathcal{Q}_h^N$ and $\boldsymbol{\mathcal{W}_h^N}$ that are defined as follows:

$$\mathcal{Q}_h^N = \{q \in L^2(\Omega),\, q|_K \in P^N(K),\, \forall K \in \mathcal{T}_h\}\,,$$
$$\boldsymbol{\mathcal{W}_h^N} = \{\boldsymbol{w} \in L^2(\Omega)^{dim},\, \boldsymbol{w}|_K \in (P^N(K))^{dim},\, \forall K \in \mathcal{T}_h\}\,.$$

The unknowns of the discrete problem are thus the coefficients of the approximate fields ${p_h}^K$ and ${\boldsymbol{v_h}}^K$ given in each element K as a polynomial written as follows:

$${p_h}^K = \sum_{j=1}^{DoF} {P_{h_j}}^K \varphi_j^K\,, \qquad ({\boldsymbol{v_h}}^K)_d = \sum_{j=1}^{DoF} {\boldsymbol{V_h}_{dj}}^K \varphi_j^K\,, \quad \text{for } d = 1 \text{ to } dim\,,$$

where φ_j^K is part of a polynomial basis of $P^N(K)$. Then, the approximate field is solution to the following local system (see page 60):

$$\begin{cases} \dfrac{1}{\kappa} M^K \dfrac{\partial {\boldsymbol{P_h}}^K}{\partial t} + \displaystyle\sum_{d=1}^{dim} S_{x_d}^K {\boldsymbol{V_h}_d}^K + \sum_{\Gamma} M_\Gamma^K {\bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}}^\Gamma = \frac{1}{\kappa} M^K {\boldsymbol{F_h}}^K, \\ \rho M^K \dfrac{\partial {\boldsymbol{V_h}_d}^K}{\partial t} + S_{x_d}^K {\boldsymbol{P_h}}^K + \displaystyle\sum_{\Gamma} M_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}d}^{\,\Gamma} = 0 \qquad (\text{for } d = 1 \text{ to } dim)\,. \end{cases}$$

and the global system is given by:

$$\begin{cases} M \dfrac{\partial}{\partial t} \boldsymbol{U} + S\boldsymbol{U} = M\boldsymbol{F}\,, \\ \boldsymbol{U}_K = ({\boldsymbol{P_h}}^K, {\boldsymbol{V_h}_1}^K, \ldots, {\boldsymbol{V_h}_{dim}}^K)^\top\,, \\ \boldsymbol{U} = (\boldsymbol{U}_{K_1}, \boldsymbol{U}_{K_2}, \ldots, \boldsymbol{U}_{K_{N_e}})^\top\,, \\ \boldsymbol{F}_K = ({\boldsymbol{F_h}}^K, 0, \ldots, 0)^\top \qquad \text{by assuming that: } f|_K \approx f_h^K = \displaystyle\sum_{i=1}^{DoF} {\boldsymbol{F_h}_i}^K \varphi_i^K\,, \\ \boldsymbol{F} = (\boldsymbol{F}_{K_1}, \boldsymbol{F}_{K_2}, \ldots, \boldsymbol{F}_{K_{N_e}})^\top\,. \end{cases}$$

We can rewrite it by introducing $A = -M^{-1}S$:

$$\frac{\partial}{\partial t}\boldsymbol{U} = A\boldsymbol{U} + \boldsymbol{F}\,.$$

Now, let us consider the time discretization. We denote by $\bar{\boldsymbol{U}}^i$ an approximation of the wavefield $\boldsymbol{U}$ at time $i\Delta t$, that is to say:

$$\bar{\boldsymbol{U}}^i \approx \boldsymbol{U}(i\Delta t)\,.$$

The global space and time unknown $\bar{\boldsymbol{U}}$ is then defined as:

$$\bar{\boldsymbol{U}} = (\bar{\boldsymbol{U}}^0, \bar{\boldsymbol{U}}^1, \ldots, \bar{\boldsymbol{U}}^{N_t})^\top\,,$$

where N_t denotes the number of time samples in the time discretization.

Property 6. For the time schemes considered in this thesis, the discrete direct problem can always be expressed in the following way:

$$L\bar{\boldsymbol{U}} = E\bar{\boldsymbol{F}}\,,$$

where L is a $[(N_t+1)(dim+1)N_eDoF] \times [(N_t+1)(dim+1)N_eDoF]$ matrix. The sizes of the vector $\bar{\boldsymbol{F}}$ and of the matrix E depend on the time scheme we are using.

As stated in the above property, the global discrete problem has the same structure whatever the time scheme is. Nevertheless, we will give hints on the proof by considering a particular scheme. By this way, we will end up with the expressions of L and E.

Proof for Runge Kutta scheme of order 2 (RK2).

For RK2 scheme, the global space and time source term $\bar{\boldsymbol{F}}$ is evaluated on half time steps:

$$\bar{\boldsymbol{F}} = (\bar{\boldsymbol{F}}^0, \bar{\boldsymbol{F}}^{\frac{1}{2}}, \ldots, \bar{\boldsymbol{F}}^{N_t-\frac{1}{2}}, \bar{\boldsymbol{F}}^{N_t})^\top,.$$

The RK2 time scheme reads:

$$\bar{\boldsymbol{U}}^{i+1} = \bar{\boldsymbol{U}}^i \Delta t k_2\,, \tag{3.10}$$

where

$$\begin{aligned} k_1 &= A\bar{\boldsymbol{U}}^i + \bar{\boldsymbol{F}}^i\,,\\ k_2 &= A(\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}k_1) + \bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,. \end{aligned}$$

Then, by removing k_1, we get

$$\begin{aligned} \bar{\boldsymbol{U}}^{i+1} &= \bar{\boldsymbol{U}}^i \Delta t k_2\,,\\ &= (I + \Delta t A + \frac{\Delta t^2}{2}A^2)\bar{\boldsymbol{U}}^i + \frac{\Delta t^2}{2}A\bar{\boldsymbol{F}}^i + \Delta t\bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,. \end{aligned}$$

Hence, one time iteration of RK2 time scheme can be summed up as follows:

$$\bar{\boldsymbol{U}}^{i+1} = B\bar{\boldsymbol{U}}^i + C_0\bar{\boldsymbol{F}}^i + C_{\frac{1}{2}}\bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,,$$

with

$$
\begin{aligned}
B &= I + \Delta t A + \frac{\Delta t^2}{2} A^2 \,, \\
C_0 &= \frac{\Delta t^2}{2} A \,, \\
C_{\frac{1}{2}} &= \Delta t I \,.
\end{aligned}
$$

Then, the matrix L of size $[(N_t+1)(dim+1)N_eDoF] \times [(N_t+1)(dim+1)N_eDoF]$ is given by:

$$
L = \begin{pmatrix}
I & & & & \\
-B & I & & & \\
 & -B & I & & \\
 & & \ddots & \ddots & \\
 & & & -B & I
\end{pmatrix} .
$$

The matrix E of size $[(N_t+1)(dim+1)N_eDoF] \times [2((N_t+1)(dim+1)N_eDoF) - 1]$ is given by:

$$
E = \begin{pmatrix}
0 & & & & & \\
C_0 & C_{\frac{1}{2}} & & & & \\
 & & C_0 & C_{\frac{1}{2}} & & \\
 & & & \ddots & \ddots & \\
 & & & & C_0 & C_{\frac{1}{2}} & 0
\end{pmatrix} .
$$

Proof for Runge Kutta time scheme of order 4 (RK4).

For RK4 time scheme, the global space and time source term $\bar{\boldsymbol{F}}$ is also evaluated on half time steps:

$$
\bar{\boldsymbol{F}} = (\bar{\boldsymbol{F}}^0, \bar{\boldsymbol{F}}^{\frac{1}{2}}, \dots, \bar{\boldsymbol{F}}^{N_t - \frac{1}{2}}, \bar{\boldsymbol{F}}^{N_t})^\top .
$$

The RK4 scheme writes:

$$
\bar{\boldsymbol{U}}^{i+1} = \bar{\boldsymbol{U}}^i \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4) \,,
$$

where

$$
\begin{aligned}
k_1 &= A\bar{\boldsymbol{U}}^i + \bar{\boldsymbol{F}}^i\,,\\
k_2 &= A(\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}k1) + f^{i+\frac{1}{2}}\,,\\
&= A\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A(A\bar{\boldsymbol{U}}^i + \bar{\boldsymbol{F}}^i) + \bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,,\\
&= A\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A^2\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A\bar{\boldsymbol{F}}^i + \bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,,\\
k_3 &= A(\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}k2) + \bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,,\\
&= A(\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}(A\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A^2\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A\bar{\boldsymbol{F}}^i + \bar{\boldsymbol{F}}^{i+\frac{1}{2}})) + \bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,,\\
&= A\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A^2\bar{\boldsymbol{U}}^i + \frac{\Delta t^2}{4}A^3\bar{\boldsymbol{U}}^i + \frac{\Delta t^2}{2}A^2\bar{\boldsymbol{F}}^i + \frac{\Delta t}{2}A\bar{\boldsymbol{F}}^{i+\frac{1}{2}} + \bar{\boldsymbol{F}}^{i+\frac{1}{2}}\,,\\
k_4 &= A(\bar{\boldsymbol{U}}^i + \Delta t k3) + \bar{\boldsymbol{F}}^i\,,\\
&= A(\bar{\boldsymbol{U}}^i + \Delta t(A\bar{\boldsymbol{U}}^i + \frac{\Delta t}{2}A^2\bar{\boldsymbol{U}}^i + \frac{\Delta t^2}{4}A^3\bar{\boldsymbol{U}}^i + \frac{\Delta t^2}{2}A^2\bar{\boldsymbol{F}}^i + \frac{\Delta t}{2}A\bar{\boldsymbol{F}}^{i+\frac{1}{2}} + \bar{\boldsymbol{F}}^{i+\frac{1}{2}})) + \bar{\boldsymbol{F}}^i\,,\\
&= A\bar{\boldsymbol{U}}^i + \Delta t A^2\bar{\boldsymbol{U}}^i + \frac{\Delta t^2}{2}A^3\bar{\boldsymbol{U}}^i + \frac{\Delta t^3}{4}A^4\bar{\boldsymbol{U}}^i + \frac{\Delta t^3}{4}A^3\bar{\boldsymbol{F}}^i + \frac{\Delta t^2}{2}A^2\bar{\boldsymbol{F}}^{i+\frac{1}{2}} + \Delta t A\bar{\boldsymbol{F}}^{i+\frac{1}{2}} + \bar{\boldsymbol{F}}^{i+1}\,.
\end{aligned}
$$

Then,

$$
\begin{aligned}
\bar{\boldsymbol{U}}^{i+1} &= \bar{\boldsymbol{U}}^i + \frac{dt}{6}(k_1 + 2k_2 + 2k_3 + k_4)\,,\\
&= (I + \Delta t A + \frac{\Delta t^2}{2}A^2 + \frac{\Delta t^3}{6}A^3 + \frac{\Delta t^4}{24}A^4)\bar{\boldsymbol{U}}^i\\
&+ \frac{1}{6}(\Delta t I + \Delta t^2 A + \frac{\Delta t^3}{2}A^2 + \frac{\Delta t^4}{4}A^4)\bar{\boldsymbol{F}}^i\\
&+ \frac{1}{6}(4\Delta t + 2\Delta t^2 A + \frac{\Delta t^3}{2}A^2)\bar{\boldsymbol{F}}^{i+1/2}\\
&+ \frac{\Delta t}{6}\bar{\boldsymbol{F}}^{i+1}\,.
\end{aligned}
$$

and one iteration of RK4 time scheme can be summed up in that way:

$$
\bar{\boldsymbol{U}}^{i+1} = B\bar{\boldsymbol{U}}^i + C_0\bar{\boldsymbol{F}}^n + C_{\frac{1}{2}}\bar{\boldsymbol{F}}^{i+\frac{1}{2}} + C_1\bar{\boldsymbol{F}}^{i+1}\,,
$$

where

$$
\begin{aligned}
B &= I + \Delta t A + \frac{\Delta t^2}{2}A^2 + \frac{\Delta t^3}{6}A^3 + \frac{\Delta t^4}{24}A^4\,,\\
C_0 &= \frac{1}{6}(\Delta t I + \Delta t^2 A + \frac{\Delta t^3}{2}A^2 + \frac{\Delta t^4}{4}A^4)\,,\\
C_{\frac{1}{2}} &= \frac{1}{6}(4\Delta t + 2\Delta t^2 A + \frac{\Delta t^3}{2}A^2)\,,\\
C_1 &= \frac{\Delta t}{6}I\,.
\end{aligned}
$$

Then, the matrix L of size $[(N_t+1)(dim+1)N_eDoF]\times[(N_t+1)(dim+1)N_eDoF]$ is given by:

$$L=\begin{pmatrix} I & & & & \\ -B & I & & & \\ & -B & I & & \\ & & \ddots & \ddots & \\ & & & -B & I \end{pmatrix}.$$

The matrix E of size $[(N_t+1)(dim+1)N_eDoF]\times[2((N_t+1)(dim+1)N_eDoF)-1]$ is given by:

$$E=\begin{pmatrix} 0 & & & & & \\ C_0 & C_{\frac{1}{2}} & C_1 & & & \\ & & C_0 & C_{\frac{1}{2}} & C_1 & \\ & & & \ddots & \ddots & \ddots \\ & & & & C_0 & C_{\frac{1}{2}} & C_1 \end{pmatrix}.$$

Proof for Adams Bashforth of order 3 (AB3).

For AB3 time scheme, the global space and time source terms is defined as follows:

$$\bar{\boldsymbol{F}}=(\bar{\boldsymbol{F}}^0,\bar{\boldsymbol{F}}^1,\dots,\bar{\boldsymbol{F}}^{N_t})^\top.$$

AB3 time scheme then reads:

$$\begin{aligned}\bar{\boldsymbol{U}}^{i+1}=&(I+\Delta t\beta_{0,2}A)\bar{\boldsymbol{U}}^i+\Delta t\beta_{1,2}A\bar{\boldsymbol{U}}^{i-1}+\Delta t\beta_{2,2}A\bar{\boldsymbol{U}}^{i-2}\\&+\Delta t\beta_{0,2}\bar{\boldsymbol{F}}^i+\Delta t\beta_{1,2}\bar{\boldsymbol{F}}^{i-1}+\Delta t\beta_{2,2}\bar{\boldsymbol{F}}^{i-2}.\end{aligned}\tag{3.11}$$

We can identify the global matrix L of size $[(N_t+1)(dim+1)N_eDoF]\times[(N_t+1)(dim+1)N_eDoF]$ which is given by:

$$L=\begin{pmatrix} I & & & & & & \\ -(I+\beta_{0,2}A) & I & & & & & \\ -\Delta t\beta_{1,2}A & -(I+\Delta t\beta_{0,2}A) & I & & & & \\ -\Delta t\beta_{2,2}A & -\Delta t\beta_{1,2}A & -(I+\Delta t\beta_{0,2}A) & I & & & \\ \ddots & \ddots & \ddots & \ddots & & & \\ & & -\Delta t\beta_{2,2}A & -\Delta t\beta_{1,2}A & -(I+\Delta t\beta_{0,2}A) & I \end{pmatrix}.$$

The matrix E, of size $[(N_t+1)(dim+1)N_eDoF]\times[(N_t+1)(dim+1)N_eDoF]$, is defined by:

$$E=\begin{pmatrix} 0 & & & & & \\ \Delta t\beta_{0,2} & & & & & \\ \Delta t\beta_{1,2} & \Delta t\beta_{0,2} & & & & \\ \Delta t\beta_{2,2} & \Delta t\beta_{1,2} & \Delta t\beta_{0,2} & & & \\ \ddots & \ddots & \ddots & \ddots & & \\ & & \Delta t\beta_{2,2} & \Delta t\beta_{1,2} & \Delta t\beta_{0,2} & 0 \end{pmatrix}.$$

Whatever the explicit time scheme used, matrix L is a block lower triangular. The discrete direct problem can therefore be written as follows: For any given $\bar{\boldsymbol{U}}^0$, find $\bar{\boldsymbol{U}}^i$ for $i=1$ to N_t such that:

$$L\bar{\boldsymbol{U}} = E\bar{\boldsymbol{F}}\,. \tag{3.12}$$

The solution to the discrete problem exists because L is a real lower triangular matrix with non-zero diagonal coefficients, and is unique because the field at time $i+1$ is deduced from the previous snapshots with an initial field at time $t=0$ set to be zero.

Remark. The block by block resolution of the discrete system (3.12) gives us back the induction formula given by the explicit time schemes (see (3.10)-(3.11)).

In this subsection, we have used the discretization defined in detail in the previous chapter to formulate a global time and space system that define the discrete state (3.12).

Now that an overall formulation of the discrete problem is given, we will define in the next subsection the associated constrained optimization problem and thus define the adjoint state for the *Discretize then Optimize* strategy.

3.2.2 Discrete Adjoint state

In the previous section, we have constructed the space-time discrete system that has the following general form, whether RK2, RK4 or AB3 is used:

$$L\bar{\boldsymbol{U}} = E\bar{\boldsymbol{F}}\,, \tag{3.13}$$

In the above expression, $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{F}}$ represent respectively the discrete unknown wavefield vector and the source term in space and time. In the following, we introduce the minimization problem exactly as in the previous section but, here, we consider the discrete problem as a constraint:

$$\min_m \mathcal{J}_h(m), \qquad \text{under the constraint "}\bar{\boldsymbol{U}}\text{ is satisfying (3.13)".}$$

The cost function is here defined as follows:

$$\mathcal{J}_h(m) = \frac{1}{2}\sum_{i=0}^{n_t}\sum_{r=1}^{n_{rcv}}([Q_h\bar{\boldsymbol{U}}(m)]_{i,r} - [d_{obs}]_{i,r})^2\,.$$

where Q_h is the restriction of the approximate pressure field evaluated at the receivers and interpolated on the same time sampling as the observations. Indeed, the time sampling of the discrete wavefield is not the same as the one of the observations.

The minimization problem can therefore be written in the form of a discrete Lagrangian $\mathcal{L}_h$:

$$\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}) = \hat{\mathcal{J}}_h(\bar{\boldsymbol{U}}) + \langle L\bar{\boldsymbol{U}} - E\bar{\boldsymbol{F}}, \bar{\boldsymbol{\Lambda}}\rangle\,, \tag{3.14}$$

where $\bar{\boldsymbol{\Lambda}}$ is the discrete adjoint state built on the same pattern that $\bar{\boldsymbol{U}}$, that is to say:

- $\bar{\boldsymbol{\Lambda}} = (\bar{\boldsymbol{\Lambda}}^0, \bar{\boldsymbol{\Lambda}}^1, ...\bar{\boldsymbol{\Lambda}}^{N_t})^\top$represents the global space time adjoint wavefield, where:
- $\bar{\boldsymbol{\Lambda}}^i = (\bar{\boldsymbol{\Lambda}}^i_{K_1}, \bar{\boldsymbol{\Lambda}}^i_{K_2}, ...\bar{\boldsymbol{\Lambda}}^i_{K_{Ne}})^\top$represents the adjoint wavefield at the i^{th} time step, where:
- $\bar{\boldsymbol{\Lambda}}^i_{K_j} = ((P_h)^{*i}_{K_j}, (\boldsymbol{V_h}1)^{*i}_{K_j}, .., (\boldsymbol{V_h}_{dim})^{*i}_{K_j})^\top$ represents the adjoint wavefield on the j^{th} element, where:
- $(P_h)^{*i}_{K_j}, (\boldsymbol{V_h}_d)^{*i}_{K_j}$for d=1 to dim, represent respectively the coefficient of the polynomial approximation of the adjoint pressure and velocity field on the j^{th} element at the i^{th} time step.

The scalar product used in (3.14) is usually chosen to be the canonical scalar product on $\mathbb{R}^{(N_t+1)N_eDoF}$. Then, the adjoint of a matrix operator is its transposition. For generalization purposes, we will note L^* the adjoint operator of any matrix operator L.

$\hat{\mathcal{J}}_h$ is the discrete cost function that is expressed only as a function of the state $\bar{\boldsymbol{U}}$ and is defined as follows:

$$\hat{\mathcal{J}}_h(\bar{\boldsymbol{U}}) = \frac{1}{2}\sum_{i=0}^{n_t}\sum_{r=1}^{n_{rcv}}([Q_h\bar{\boldsymbol{U}}]_{i,r} - [d_{obs}]_{i,r})^2\,.$$

Hence, we seek for $(\bar{\boldsymbol{U}}, \bar{\boldsymbol{\Lambda}}, m)$ satisfying:

$$\frac{\partial}{\partial\bar{\boldsymbol{U}}}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}) = 0\,, \qquad \frac{\partial}{\partial\bar{\boldsymbol{\Lambda}}}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}) = 0\,, \qquad \frac{\partial}{\partial m}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}) = 0\,,$$

which are necessary conditions on $\bar{\boldsymbol{U}}, \bar{\boldsymbol{\Lambda}}, m$ to minimize the Lagrangian. Then, we end up with the same types of conclusion as with the continuous problem:

- If we look at the derivative with respect to the discrete adjoint state in the direction $\delta_{\bar{\boldsymbol{\Lambda}}}$ we have:

$$\begin{aligned}\frac{\partial}{\partial\bar{\boldsymbol{\Lambda}}}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}})\delta_{\bar{\boldsymbol{\Lambda}}} &= \frac{\partial}{\partial\bar{\boldsymbol{\Lambda}}}\hat{\mathcal{J}}_h(\bar{\boldsymbol{U}})\delta_{\bar{\boldsymbol{\Lambda}}} + \frac{\partial}{\partial\bar{\boldsymbol{\Lambda}}}\langle L\bar{\boldsymbol{U}} - E\bar{\boldsymbol{F}}, \bar{\boldsymbol{\Lambda}}\rangle\delta_{\bar{\boldsymbol{\Lambda}}}\,,\\ &= \langle L\bar{\boldsymbol{U}} - E\bar{\boldsymbol{F}}, \delta_{\bar{\boldsymbol{\Lambda}}}\rangle\,.\end{aligned}$$

  which is zero if and only if $\bar{\boldsymbol{U}}$ is the solution to the direct discrete problem (3.13).

- If we compute the Lagrangian derivative with respect to the state variable $\bar{\boldsymbol{U}}$ in any direction $\delta_{\bar{\boldsymbol{U}}}$, we get:

$$\begin{aligned}\frac{\partial}{\partial\bar{\boldsymbol{U}}}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}})\delta_{\bar{\boldsymbol{U}}} &= \frac{\partial}{\partial\bar{\boldsymbol{U}}}\hat{\mathcal{J}}_h(\bar{\boldsymbol{U}})\delta_{\bar{\boldsymbol{U}}} + \frac{\partial}{\partial\bar{\boldsymbol{U}}}\langle L\bar{\boldsymbol{U}} - E\bar{\boldsymbol{F}}, \bar{\boldsymbol{\Lambda}}\rangle\delta_{\bar{\boldsymbol{U}}}\,,\\ &= \frac{\partial}{\partial\bar{\boldsymbol{U}}}\frac{1}{2}\langle Q_h\bar{\boldsymbol{U}} - d_{obs}, Q_h\bar{\boldsymbol{U}} - d_{obs}\rangle + \frac{\partial}{\partial\bar{\boldsymbol{U}}}\langle L\bar{\boldsymbol{U}}, \bar{\boldsymbol{\Lambda}}\rangle\,,\\ &= \langle Q_h^*(Q_h\bar{\boldsymbol{U}} - d_{obs}) + L^*\bar{\boldsymbol{\Lambda}}, \delta_{\bar{\boldsymbol{U}}}\rangle\,.\end{aligned}$$

Hence, in order to have $\dfrac{\partial}{\partial\bar{\boldsymbol{U}}}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}) = 0$, $\bar{\boldsymbol{\Lambda}}$ must be a solution to the following linear system:

$$L^*\bar{\boldsymbol{\Lambda}} + Q_h^*(Q_h\bar{\boldsymbol{U}} - d_{obs}) = 0\,, \tag{3.15}$$

where $Q_h^*(Q_h\bar{\boldsymbol{U}} - d_{obs})$ is the adjoint source that we will note $\bar{\boldsymbol{G}}$ for simplicity. Then, we define the discrete adjoint state as the solution to:

$$L^*\bar{\boldsymbol{\Lambda}} + \bar{\boldsymbol{G}} = 0\,.$$

Matrix L^* is block upper triangular. From the definition of the global operator L for the different time schemes we use, we can easily compute the adjoint operator L^*. For Runge Kutta time schemes, we can show that the adjoint operator L^* is given by:

$$L^* = \begin{pmatrix} I & -B^* & & & \\ & I & -B^* & & \\ & & \ddots & \ddots & \\ & & & I & -B^* \\ & & & & I \end{pmatrix}.$$

where

$$B^* = I + \Delta t A^* + \frac{\Delta t^2}{2} A^{*2}, \qquad \text{for RK2}\,,$$
$$B^* = I + \Delta t A^* + \frac{\Delta t^2}{2} A^{*2} + \frac{\Delta t^3}{6} A^{*3} + \frac{\Delta t^4}{24} A^{*4}, \qquad \text{for RK4}\,.$$

Concerning the AB3 time scheme, we get:

$$L^* = \begin{pmatrix} I & -(I+\delta\beta_{0,2}A^*) & -\Delta t\beta_{1,2}A^* & -\Delta t\beta_{2,2}A^* & \\ & \ddots & \ddots & \ddots & \ddots \\ & I & -(I+\delta\beta_{0,2}A^*) & -\Delta t\beta_{1,2}A^* & -\Delta t\beta_{2,2}A^* \\ & & I & -(I+\delta\beta_{0,2}A^*) & -\Delta t\beta_{1,2}A^* \\ & & & I & -(I+\delta\beta_{0,2}A^*) \\ & & & & I \end{pmatrix}.$$

The block upper triangular structure of the matrix L^* allows to solve the adjoint system (3.15) by substitution from bottom to top, i.e., backward in time. We retrieve numerically the change of variable performed to establish the continuous adjoint state. We also remark that such systems require that $\bar{\mathbf{\Lambda}}^{N_t} = \bar{\boldsymbol{G}}^{N_t} = 0$, since we assume that the acquisition is long enough to not have any information recorded at time $t = T_f$. Moreover, the matrix E inherited from the time scheme no longer appears. It is therefore not possible to use the same time scheme routines for the calculation of the direct state and the adjoint state.

Regarding the time scheme, the *Discretize then Optimize* strategy then requires the development of specific time schemes routines to solve the discrete adjoint state problem. We have shown previously that such a development is not required when considering the *optimization* prior the *discretization*. We then propose the following adaptation of the time schemes to solve the adjoint discrete problem:

Proposition 3. Adaptation of RK2 time scheme for discrete adjoint problem:

$$\bar{\mathbf{\Lambda}}^{i-1} = \bar{\mathbf{\Lambda}}^{i} + \Delta t k_2 - \bar{\boldsymbol{G}}^{i}\,,$$

where

$$k_1 = A^* \bar{\mathbf{\Lambda}}^{i}\,,$$
$$k_2 = A^*(\bar{\mathbf{\Lambda}}^{i} + \frac{1}{2}\Delta t k_1)\,.$$

Proposition 4. Adaptation of RK4 time scheme for discrete adjoint problem:

$$\bar{\mathbf{\Lambda}}^{i-1} = \bar{\mathbf{\Lambda}}^{i} + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) - \bar{\boldsymbol{G}}^{i}\,,$$

where

$$k_1 = A^* \bar{\mathbf{\Lambda}}^{i}\,,$$
$$k_2 = A^*(\bar{\mathbf{\Lambda}}^{i} + \frac{1}{2}\Delta t k_1)\,,$$
$$k_3 = A^*(\bar{\mathbf{\Lambda}}^{i} + \frac{1}{2}\Delta t k_2)\,,$$
$$k_4 = A^*(\bar{\mathbf{\Lambda}}^{i} + \Delta t k_3)\,.$$

Proposition 5. Adjoint AB3 time scheme:

$$\bar{\boldsymbol{\Lambda}}^{i-1} = \bar{\boldsymbol{\Lambda}}^{i} + \Delta t(\beta_{0,2} A^* \bar{\boldsymbol{\Lambda}}^{i} + \beta_{0,2} A^* \bar{\boldsymbol{\Lambda}}^{i} + \beta_{0,2} A^* \bar{\boldsymbol{\Lambda}}^{i}) + \bar{\boldsymbol{G}}^{i} .$$

It now remains to determine the operator A^* that is the adjoint of the discrete space operator. We remind that the operator A was previously defined as:

$$\frac{\partial \boldsymbol{U}}{\partial t} = A\boldsymbol{U} + \boldsymbol{F} ,$$

where

$$A = M^{-1} S ,$$

and A, M and S are of size $N_e DoF \times N_e DoF$. Then we have:

$$A^* = S^* M^{-1^*} .$$

If we add access to the complete matrix A, it would be easy to build A^* and to fully characterize the equation system defining the discrete adjoint state in time and in space. Unfortunately, in the framework of DG method, the space operator A is never completely built. We indeed solve a local problem on each element. We recall that on one element, the direct problem is expressed in the following way:

$$\begin{cases} \dfrac{1}{\kappa} M^K \dfrac{\partial \boldsymbol{P_h}^K}{\partial t} + \displaystyle\sum_{d=1}^{dim} S_{x_d}^K \boldsymbol{V_h}_d{}^K + \sum_{\Gamma} M_\Gamma^K \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{p}}^{\ \Gamma} = \frac{1}{\kappa} M^K \boldsymbol{F_h}^K, \\ \rho M^K \dfrac{\partial \boldsymbol{V_h}_d^K}{\partial t} + S_{x_d}^K \boldsymbol{P_h}^K + \displaystyle\sum_{\Gamma} M_\Gamma \bar{\boldsymbol{\mathcal{F}}}_{\boldsymbol{v}d}^{\ \Gamma} = 0 \qquad (\text{for } d = 1 \text{ to } dim) . \end{cases}$$

In order to understand how to compute A^*, it is necessary to link these local problems with the global matrices M and S. It is clear that the mass matrix M, is composed of square submatrices of size $DoF(dim+1)$ defined on each element as:

$$M_{|_K} = \begin{pmatrix} \frac{1}{\kappa} M^K & 0 & 0 & 0 \\ 0 & \rho M^K & 0 & 0 \\ 0 & 0 & \rho M^K & 0 \\ 0 & 0 & 0 & \rho M^K \end{pmatrix} .$$

Thus, the transpose of M can also be expressed as the transpose of the submatrices $M_{|_K}$:

$$M_{|_K}{}^{-*} = \begin{pmatrix} \kappa M^{K-*} & 0 & 0 & 0 \\ 0 & \frac{1}{\rho} M^{K-*} & 0 & 0 \\ 0 & 0 & \frac{1}{\rho} M^{K-*} & 0 \\ 0 & 0 & 0 & \frac{1}{\rho} M^{K-*} \end{pmatrix} .$$

Matrix S can be separated into two parts:

$$S = S_\Omega + S_\Gamma .$$

Similarly to M, S_Ω is defined elementwise as

$$S_{\Omega|_K} = \begin{pmatrix} 0 & S_{x_1}^K & S_{x_2}^K & S_{x_3}^K \\ S_{x_1}^K & 0 & 0 & 0 \\ S_{x_2}^K & 0 & 0 & 0 \\ S_{x_3}^K & 0 & 0 & 0 \end{pmatrix} ,$$

and its transpose can also be constructed element by element:

$$S_{\Omega|_K}^* = \begin{pmatrix} 0 & S_{x_1}^{K*} & S_{x_2}^{K*} & S_{x_3}^{K*} \\ S_{x_1}^{K*} & 0 & 0 & 0 \\ S_{x_2}^{K*} & 0 & 0 & 0 \\ S_{x_3}^{K*} & 0 & 0 & 0 \end{pmatrix}.$$

Matrix S_Γ accounts for the calculation of surface terms, i.e., the calculation of fluxes and boundary conditions, and link each element with its neighbors. It cannot be expressed as easily as the previous matrices and its adjoint is much more complicated to determine. This matrix is actually never built in the code and the corresponding matrix-vector product is rather computed thanks to a succession of operations.

The structure of the DG code inexorably requires additional development to define the adjoint operators of the discrete adjoint problem. One possibility would then be to turn to the automatic differentiation tools [Griewank and Walther, 2008] to obtain directly the adjoint functions or even directly the gradient. Unfortunately, the functions linked to flux computation must manage the MPI communications between the subdomains of the simulation. Using automatic differentiation tools in parallel environment is challenging [Tröltzsch, 2010] and will give slow and poorly optimized functions.

Industrial context

In addition to the parallel environment, another technical difficulty is that the code is split into three libraries and one application. This hierarchical organization (see Chapter 4) makes automatic differentiation software even more complex to employ.

To conclude, we have described in this chapter, step by step, the construction of the system of equations whose discrete adjoint state is the solution. Unfortunately, the difficulties of the *Discretize then Optimize* strategy are technical. Indeed, it requires the development of the adjoint of a huge amount of modular functions inherited from the DG structure that is solved element by element.

However, even if the adjoint operator requires some extra developments, it exists a way to validate the implementation via an adjoint test.

3.2.3 Adjoint test

We have seen before that the discrete forward problem as well as its adjoint can be expressed synthetically in the following way:

$$L\bar{\boldsymbol{U}} = E\bar{\boldsymbol{F}}\,,$$

$$L^*\bar{\boldsymbol{\Lambda}} = \bar{\boldsymbol{G}}\,.$$

The difficulty lies in obtaining L^* knowing L, since L is never entirely constructed in the code. According to the discretization, it can be tough to obtain the adjoint operator. Nevertheless, dealing with the adjoint of the discrete problem allows setting up a test proving that the adjoint operator L^* is well constructed. This test is commonly called the adjoint test.

Let us consider two randomly generated source terms $\tilde{f}$ and $\tilde{g}$, we now choose $\bar{\boldsymbol{U}}$ and $\bar{\boldsymbol{\Lambda}}$ as:

$$L\bar{\boldsymbol{U}} = \tilde{f}\,, \qquad L^*\bar{\boldsymbol{\Lambda}} = \tilde{g}\,. \tag{3.16}$$

Then, by definition, L^* is the adjoint of L if and only if:

$$\langle L\bar{\boldsymbol{U}}, \bar{\boldsymbol{\Lambda}} \rangle = \langle \bar{\boldsymbol{U}}, L^* \bar{\boldsymbol{\Lambda}} \rangle \,,$$

which writes, according to (3.16),:

$$\langle \tilde{f}, \bar{\boldsymbol{\Lambda}} \rangle = \langle \bar{\boldsymbol{U}}, \tilde{g} \rangle \,. \tag{3.17}$$

This is what we call "adjoint test", it consists in verifying (3.17) to the roundoff machine error. Validating this test is a proof that L^* provides a way to properly calculate the adjoint state of the considered problem (provided that the adjoint source is also correctly implemented). This test is only available when considering the strategy *Optimize then Discretize*.

In this section, we described two approaches for computing the adjoint states according to whether one decides to *optimize* before *discretizing* or *discretize* before *optimize*. The two resulting adjoint operators are different. On the one hand, when we *optimize* the continuous problem, the continuous adjoint state is expressed in the same way as the direct state. It is therefore possible to use the same *discretization* for both problems, saving the development of an additional solver.

On the other hand, the strategy that consists in *discretizing* before *optimizing* requires additional development in order to define the adjoint discrete problem.

Concerning the discrete adjoint operators, they can be tested thanks to the adjoint test. Unfortunately, this test is only available considering the *Discretize then Optimize* strategy. Indeed, we only have access to an approximation of the continuous states, so that the test will fail because of the error from the discretization, which is much greater than the round-off error.

The adjoint state is only an intermediate step in the calculation of the gradient of the physical model. We propose in the next section to define the calculation of the gradient according to the *Optimize then Discretize* and *Discretize then Optimize* strategies.

3.3 The gradient expressions

In this section, we will define the expression of the gradient of the cost function according to the physical parameters of the continuous or discrete problem. We will first establish the gradient of the continuous problem. This gradient is thus used in the inversion algorithm after discretization, and we will call the resulting gradient the *discretized gradient*. In a second approach, we will discretize the problem before computing its gradient. Then, the resulting gradient will be called the *discrete gradient*.

The gradient is relative to the parameters representing the model and several choices of parameterization are possible. For instance, for the acoustic wave equation one can choose $(\frac{1}{\kappa}, \rho)$, $(c, \frac{1}{\rho})$ etc. In [Faucher, 2017], it has been shown that the conditioning of the associated Jacobian may be hindered for some choices of parameterization while other improve the reconstruction. For instance $(\frac{1}{\kappa}, \rho)$ is a good choice of parameterization for the acoustic wave equation.

For the sake of generalization, we will denote the parameterization by (α, β). In what follows, we will define the *discretized gradient* expression for any parameterization (α, β).

3.3.1 Discretized gradient

The gradient is obtained by considering the derivative of the Lagrangian with respect to the physical parameters. Here, we will consider the first order equation system we defined in (3.1.2). Let us consider the derivative with respect to the parameter α:

$$\frac{\partial \mathcal{J}}{\partial \alpha}(m)\delta_\alpha = \frac{\partial \mathcal{L}}{\partial \alpha}(\boldsymbol{u}, m, \boldsymbol{\lambda})\delta_\alpha = \frac{\partial}{\partial \alpha}\hat{\mathcal{J}}(m)\delta_\alpha + \frac{\partial}{\partial \alpha}\left\langle \begin{pmatrix} \frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} - f \\ \rho \frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \end{pmatrix}, \begin{pmatrix} p^* \\ \boldsymbol{v^*} \end{pmatrix} \right\rangle \delta_\alpha \,,$$

$$= \frac{\partial}{\partial \alpha}\int_\Omega \int_{t_0}^{T_f} \left[\left(\frac{1}{\kappa}\frac{\partial p}{\partial t} + \nabla \cdot \boldsymbol{v} - f \right) p^* + \left(\rho \frac{\partial \boldsymbol{v}}{\partial t} + \nabla p \right) \cdot \boldsymbol{v^*} \right] dt dx \delta_\alpha,$$

$$= \int_\Omega \delta_\alpha \int_{t_0}^{T_f} \left[\frac{\partial}{\partial \alpha}\frac{1}{\kappa}\frac{\partial p}{\partial t} p^* + \frac{\partial}{\partial \alpha}\rho \frac{\partial \boldsymbol{v}}{\partial t} \cdot \boldsymbol{v^*} \right] dx dt \,.$$

It should be noted that parameters $\frac{1}{\kappa}$ and ρ are functions of α and β. We could then write them down as $\frac{1}{\kappa}(\alpha, \beta)$ $\rho(\alpha, \beta)$. But for the sake of conciseness, we will keep the notation $\frac{1}{\kappa}$, ρ. We warn the reader not to forget that these parameters are functions of α and β.

By identification, we may express the continuous gradient with respect to the parameterization α as follows:

$$\nabla_\alpha \mathcal{J} = \int_{t_0}^{T_f} \left(\frac{\partial}{\partial \alpha}\frac{1}{\kappa}\frac{\partial p}{\partial t} p^* + \frac{\partial}{\partial \alpha}\rho \frac{\partial \boldsymbol{v}}{\partial t} \cdot \boldsymbol{v^*} \right) dx dt \,. \tag{3.18}$$

Since we are considering any parameterization (α, β), then the gradient with respect to the parameter β is obtained by replacing α by β in (3.18).

$$\nabla_\beta \mathcal{J} = \int_{t_0}^{T_f} \left(\frac{\partial}{\partial \beta}\frac{1}{\kappa}\frac{\partial p}{\partial t} p^* + \frac{\partial}{\partial \beta}\rho \frac{\partial \boldsymbol{v}}{\partial t} \cdot \boldsymbol{v^*} \right) dx dt \,.$$

Since we do not have access to the exact pressure field, we keep computing the gradient by replacing p and $p*$ by their approximations p_h and p_h*. In the case where the parameters are constant per element, we define the parameterization α (or β) by the set of parameter $\alpha_{1 \le l \le n_p}$. Here we have $n_p = N_e$. Let us consider the approximation of the derivative of the continuous cost function by the l^{th} parameter α_l. We have:

$$\left(\frac{\partial}{\partial \alpha_l}\mathcal{J}(m) \right)_h \approx \frac{\partial}{\partial \alpha_l}\int_{t_0}^{T_f}\int_\Omega \left(\frac{1}{\kappa}\frac{\partial p_h}{\partial t}(p^*)_h + \rho \frac{\partial \boldsymbol{v_h}}{\partial t} \cdot (\boldsymbol{v^*})_h \right) dx dt \,,$$

$$\approx \int_{t_0}^{T_f}\int_\Omega \left(\frac{\partial}{\partial \alpha_l}\frac{1}{\kappa}\frac{\partial p_h}{\partial t}(p^*)_h + \frac{\partial}{\partial \alpha_l}\rho \frac{\partial \boldsymbol{v_h}}{\partial t} \cdot (\boldsymbol{v^*})_h \right) dx dt \,,$$

$$\approx \int_{t_0}^{T_f} \left[\sum_j^{N_e} \int_{K^l} \left(\frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_j}\frac{\partial p_h}{\partial t}(p^*)_h + \frac{\partial}{\partial \alpha_l}\rho_j \frac{\partial \boldsymbol{v_h}}{\partial t} \cdot (\boldsymbol{v^*})_h \right) dx \right] dt \,.$$

Since $\frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_j} = 0$ and $\frac{\partial}{\partial \alpha_l}\rho_j = 0$ for $j \neq l$, then:

$$\left(\frac{\partial}{\partial \alpha_l}\mathcal{J}(m) \right)_h \approx \int_{t_0}^{T_f}\int_{K^l} \left(\frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}\frac{\partial p_h}{\partial t}(p^*)_h + \frac{\partial}{\partial \alpha_l}\rho_l \frac{\partial \boldsymbol{v_h}}{\partial t} \cdot (\boldsymbol{v^*})_h \right) dx dt \,.$$

We can then consider the forward and backward approximate states as a polynomial approximation on the l^{th} element:

$$\frac{\partial}{\partial t}{p_h}^{K^l} = \sum_{j=1}^{DoF} (\frac{\partial}{\partial t}P_{h_j})^{K^l} \varphi_j^{K^l} \,, \qquad {(p^*)_h}^{K^l} = \sum_{j=1}^{DoF} {(P^*)_h}_j{}^{K^l} \varphi_j^{K^l} \,,$$

$$\frac{\partial}{\partial t}{v_h}_d{}^{K^l} = \sum_{j=1}^{DoF} (\frac{\partial}{\partial t}{V_h}_{d_j})^{K^l} \varphi_j^{K^l}\,, \qquad (\boldsymbol{v^*})_h{}^{K^l} = \sum_{j=1}^{DoF} (V^*)_{h_{d_j}}{}^{K^l} \varphi_j^{K^l}\,.$$

Then, by introducing the time discretization, the approximate derivative of the cost function with respect to the l^{th} component of $\frac{1}{\kappa}$ can be written as follows:

$$\begin{aligned}
\left(\frac{\partial}{\partial \alpha_l}\mathcal{J}(m)\right)_h &\approx \int_{t_0}^{T_f}\int_{K^l} \frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}\frac{\partial p_h}{\partial t}(p^*)_h dx dt + \int_{t_0}^{T_f}\int_{K^l} \frac{\partial}{\partial \alpha_l}\rho_l \frac{\partial \boldsymbol{v_h}}{\partial t}\cdot(\boldsymbol{v^*})_h dx dt,\\
&\approx \sum_{n=0}^{N_t}\Delta t \int_{K^l}\left(\sum_{i=1}^{DoF}\frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}(\frac{\partial}{\partial t}{P_h}_i^n)^{K^l}\varphi_i^{K^l}\sum_{j}^{DoF}(({P^*)_h}_j^n)^{K^l}\varphi_j^{K^l}\right)dx\\
&+ \sum_{n=0}^{N_t}\Delta t \frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}\int_{K^l}\left(\sum_{d=1}^{dim}\sum_{i=1}^{DoF}\frac{\partial}{\partial \alpha_l}\rho_l(\frac{\partial}{\partial t}{V_h}_{di}^n)^{K^l}\varphi_i^{K^l}\sum_{j}^{DoF}(({V^*)_h}_{dj}^n)^{K^l}\varphi_j^{K^l}\right)dx\,,\\
&\approx \sum_{n=0}^{N_t}\Delta t \int_{K^l}\left(\sum_{i=1}^{DoF}(\frac{\partial}{\partial t}{P_h}_i^n)^{K^l}\varphi_i^{K^l}\sum_{j}^{DoF}(({P^*)_h}_j^n)^{K^l}\varphi_j^{K^l}\right)dx\\
&+ \sum_{n=0}^{N_t}\Delta t \frac{\partial}{\partial \alpha_l}\rho_l\sum_{d=1}^{dim}\int_{K^l}\left(\sum_{i=1}^{DoF}(\frac{\partial}{\partial t}{V_h}_{di}^n)^{K^l}\varphi_i^{K^l}\sum_{j}^{DoF}(({V^*)_h}_{dj}^n)^{K^l}\varphi_j^{K^l}\right)dx\,,\\
&\approx \Delta t \frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}\sum_{n=0}^{N_t}\sum_{i=1}^{DoF}\sum_{j=1}^{DoF}(\frac{\partial}{\partial t}{P_h}_i^n)^{K^l}\int_{K^l}\varphi_i^{K^l}\varphi_j^{K^l}(({P^*)_h}_j^n)^{K^l}dx\\
&+ \Delta t \frac{\partial}{\partial \alpha_l}\rho_l\sum_{n=0}^{N_t}\sum_{d=1}^{dim}\sum_{i=1}^{DoF}\sum_{j=1}^{DoF}(\frac{\partial}{\partial t}{V_h}_{di}^n)^{K^l}\int_{K^l}\varphi_i^{K^l}\varphi_j^{K^l}(({V^*)_h}_{dj}^n)^{K^l}dx\,,\\
&\approx \Delta t \frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{P_h}^n)^{K^l\top}M^{K^l}(\boldsymbol{(P^*)_h}^n)^{K^l}\\
&+ \Delta t \frac{\partial}{\partial \alpha_l}\rho_l\sum_{n=0}^{N_t}\sum_{d=1}^{dim}(\frac{\partial}{\partial t}\boldsymbol{V_{hd}}^n)^{K^l\top}M^{K^l}(\boldsymbol{(V^*)_{hd}}^n)^{K^l}\,,\\
&\approx \Delta t \frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}|T_{K^l}|\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{P_h}^n)^{K^l\top}\hat{M}(\boldsymbol{(P^*)_h}^n)^{K^l}\\
&+ \Delta t \frac{\partial}{\partial \alpha_l}\rho_l|T_{K^l}|\sum_{d=1}^{dim}\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{V_{hd}}^n)^{K^l\top}\hat{M}(\boldsymbol{(V^*)_{hd}}^n)^{K^l}\,,
\end{aligned}$$

where M^K, $\hat{M}$ and $|T_{K^l}|$ are the DG operators defined in Chapter 2, page 63.

We have thus computed the l^{th} component of the discretized gradient of the cost function with respect to any parameter α representing the l^{th} element, and we get:

Proposition 6. For any parameterization (α, β) constant per element, the discretized gradi-

ent of the continuous first order acoustic problem relative to (α, β) is given by:

$$\left(\frac{\partial}{\partial \alpha_l}\mathcal{J}(m)\right)_h \approx \Delta t \frac{\partial}{\partial \alpha_l}\frac{1}{\kappa_l}(\alpha_l,\beta_l)|T_{K^l}|\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{P_h}^n)^{K^l\top}\hat{M}((\boldsymbol{P^*})_{\boldsymbol{h}}{}^n)^{K^l} + \Delta t \frac{\partial}{\partial \alpha_l}\rho_l|T_{K^l}|\sum_{d=1}^{dim}\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{V_{hd}}^n)^{K^l\top}\hat{M}((\boldsymbol{V^*})_{\boldsymbol{h_d}}{}^n)^{K^l}. \tag{3.19}$$

$$\left(\frac{\partial}{\partial \beta_l}\mathcal{J}(m)\right)_h \approx \Delta t \frac{\partial}{\partial \beta_l}\frac{1}{\kappa_l}(\alpha_l,\beta_l)|T_{K^l}|\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{P_h}^n)^{K^l\top}\hat{M}((\boldsymbol{P^*})_{\boldsymbol{h}}{}^n)^{K^l} + \Delta t \frac{\partial}{\partial \beta_l}\rho_l|T_{K^l}|\sum_{d=1}^{dim}\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{V_{hd}}^n)^{K^l\top}\hat{M}((\boldsymbol{V^*})_{\boldsymbol{h_d}}{}^n)^{K^l}. \tag{3.20}$$

Industrial context

To simplify the computation and implementation of the gradient for any parameterization, we may notice that the gradient expression is way more simplified when choosing $(\alpha, \beta) = (\frac{1}{\kappa}, \rho)$. Indeed, we have:

$$(\frac{\partial}{\partial \frac{1}{\kappa_l}}\mathcal{J}(m))_h \approx \Delta t|T_{K^l}|\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{P_h}^n)^{K^l\top}\hat{M}((\boldsymbol{P^*})_{\boldsymbol{h}}{}^n)^{K^l},$$

and

$$(\frac{\partial}{\partial \rho_l}\mathcal{J}(m))_h \approx \Delta t|T_{K^l}|\sum_{d=1}^{dim}\sum_{n=0}^{N_t}(\frac{\partial}{\partial t}\boldsymbol{V_{hd}}^n)^{K^l\top}\hat{M}((\boldsymbol{V^*})_{\boldsymbol{h_d}}{}^n)^{K^l}.$$

In the code, since this parameterization gives a very simple expression of the gradient, we use it as a reference, and we compute the gradient relative to other parameterizations by applying the derivation chain rule. Hence, for any (α, β), we have:

$$\frac{\partial}{\partial \alpha}\mathcal{J} = \frac{\partial \frac{1}{\kappa}}{\partial \alpha}\frac{\partial \mathcal{J}}{\partial \frac{1}{\kappa}} + \frac{\partial \rho}{\partial \alpha}\frac{\partial \mathcal{J}}{\partial \rho},$$

$$\frac{\partial}{\partial \beta}\mathcal{J} = \frac{\partial \frac{1}{\kappa}}{\partial \beta}\frac{\partial \mathcal{J}}{\partial \frac{1}{\kappa}} + \frac{\partial \rho}{\partial \beta}\frac{\partial \mathcal{J}}{\partial \rho}.$$

Eqs. (3.19) and (3.20) show how the approximate gradient is computed when considering a model constant per element. However, the DG solver we are using also has Weight Adjusted Discontinuous Galerkin method that allows to set multiple parameters per element. In this case, the parameterization (α, β) is defined by a set of parameters $(\alpha_{l,q}\}\beta_{l,q}\}))$, where $1 \leq l \leq N_e$ and $1 \leq q \leq n_q$. Using the WADG method, the total number of parameters n_p is defined such that: $n_p = n_q N_e$. The quantity $\alpha_{l,q}$ or $\beta_{l,q}$ represents the parameter evaluated at the q^{th} quadrature point on the l^{th} element. Let us compute the derivative with respect to

parameter $\alpha_{l,q}$. It reads:

$$\left(\frac{\partial}{\partial\alpha_{l,q}}\mathcal{J}(m)\right)_h \approx \frac{\partial}{\partial\alpha_{l,q}}\int_{t_0}^{T_f}\int_\Omega\left(\frac{1}{\kappa}\frac{\partial p_h}{\partial t}(p^*)_h+\rho\frac{\partial \boldsymbol{v_h}}{\partial t}\cdot(\boldsymbol{v^*})_h\right)dxdt\,,$$
$$\approx \frac{\partial}{\partial\alpha_{l,q}}\int_{t_0}^{T_f}\left[\sum_{k=1}^{N_e}\int_{K^k}\left(\frac{1}{\kappa}\frac{\partial p_h}{\partial t}(p^*)_h+\rho\frac{\partial \boldsymbol{v_h}}{\partial t}\cdot(\boldsymbol{v^*})_h\right)dx\right]dt\,,$$

By using the WADG quadrature formerly defined in page 95, we can approximate the integral on the l^{th} element K^l and the approximate gradient becomes:

$$\left(\frac{\partial}{\partial(\alpha_{l,q})}\mathcal{J}(m)\right)_h \approx \sum_{n=0}^{N_t}\Delta t\frac{\partial}{\partial\alpha_{l,q}}|T_{K^l}|\sum_{m=1}^{n_q}\hat{\omega}_m\left[\left(\frac{1}{\kappa}\right)_{l,m}p_h(\boldsymbol{x}_m)(p^*)_h(\boldsymbol{x}_m)+\rho_{l,m}\boldsymbol{v_h}(\boldsymbol{x}_m)\cdot(\boldsymbol{v^*})_h(\boldsymbol{x}_m)\right],$$

$$\begin{aligned}
&\approx \sum_{n=0}^{N_t}\Delta t\frac{\partial}{\partial\alpha_{l,q}}|T_{K^l}|{\boldsymbol{P_h}^{nK^l}}^\top M_{\frac{1}{\kappa}}^{K^l}{(\boldsymbol{P^*})_{\boldsymbol{h}}}^{nK^l}\\
&+\sum_{n=0}^{N_t}\Delta t\frac{\partial}{\partial\alpha_{l,q}}|T_{K^l}|\sum_{d=1}^{dim}{\boldsymbol{V_h}_d^{nK^l}}^\top M_\rho^{K^l}{(\boldsymbol{V^*})_{\boldsymbol{h_d}}}^{nK^l},\\
&\approx \Delta t|T_{K^l}|\sum_{n=0}^{N_t}{\boldsymbol{P_h}^{nK^l}}^\top\bar{Q}\frac{\partial}{\partial\alpha_{l,q}}diag\left[\hat{\boldsymbol{\omega}}\left(\frac{1}{\kappa}\right)^l\right]\bar{Q}^\top{(\boldsymbol{P^*})_{\boldsymbol{h}}}^{nK^l}\\
&+\Delta t|T_{K^l}|\sum_{n=0}^{N_t}\sum_{d=1}^{dim}{\boldsymbol{V_h}^{nK^l}}^\top\bar{Q}\frac{\partial}{\partial\alpha_{l,q}}diag(\hat{\boldsymbol{\omega}}\rho^l)\bar{Q}^\top{(\boldsymbol{V^*})_{\boldsymbol{h_d}}}^{K^l},\\
&\approx \Delta t|T_{K^l}|\frac{\partial}{\partial\alpha_{l,q}}\left(\frac{1}{\kappa}\right)_{l,q}\sum_{n=0}^{N_t}{\boldsymbol{P_h}^{nK^l}}^\top\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top{(\boldsymbol{P^*})_{\boldsymbol{h}}}^{nK^l}\\
&+\Delta t|T_{K^l}|\frac{\partial}{\partial\alpha_{l,q}}\rho_{l,q}\sum_{n=0}^{N_t}\sum_{d=1}^{dim}{\boldsymbol{V_h}^{nK^l}}^\top\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top{(\boldsymbol{V^*})_{\boldsymbol{h_d}}}^{K^l}.
\end{aligned}$$

To facilitate the reading of these formulas, we recall the notations related to the implementation of WADG method:

- n_q represents the number of quadrature points on each element;
- $\bar{Q}$ is the matrix of size $DoF\times n_q$ such that $[\bar{Q}]_{i,q}=\hat{\varphi}_i(\hat{\boldsymbol{x}}_q)$;
- for a set of parameters γ^l on the l^{th} element K^l we denote by $diag(\hat{\boldsymbol{\omega}}\boldsymbol{\gamma}^l)$ the diagonal matrix of size $n_q\times n_q$, where $diag(\hat{\boldsymbol{\omega}}\boldsymbol{\gamma}^l)_{q,q}=\hat{\omega}_q\boldsymbol{\gamma}_{l,q}$;
- $diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})$ is a matrix of size $n_q\times n_q$ whose entry is zero everywhere except at the intersection of the q^{th} row and the q^{th} column, where it is equal to $\hat{\omega}_q$.

Hence we have:

Proposition 7. Let us assume that the parameterization is given by $(\alpha_{l,q}\}\beta_{l,q}\}))$, where $1 \leq l \leq N_e$ and $1 \leq q \leq n_q$, the discretized gradient is given by:

$$\left(\frac{\partial}{\partial \alpha_{l,q}}\mathcal{J}(m)\right)_h \approx \Delta t|T_{K^l}|\frac{\partial}{\partial \alpha_{l,q}}\left(\frac{1}{\kappa}\right)_{l,q}\sum_{n=0}^{N_t}\boldsymbol{P_h}^{nK^l\top}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top(\boldsymbol{P^*})_{\boldsymbol{h}}{}^{nK^l}$$
$$+\Delta t|T_{K^l}|\frac{\partial}{\partial \alpha_{l,q}}\rho_{l,q}\sum_{n=0}^{N_t}\sum_{d=1}^{dim}\boldsymbol{V_h}^{nK^l\top}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top(\boldsymbol{V^*})_{\boldsymbol{h_d}}{}^{K^l},$$

and

$$\left(\frac{\partial}{\partial \beta_{l,q}}\mathcal{J}(m)\right)_h \approx \Delta t|T_{K^l}|\frac{\partial}{\partial \beta_{l,q}}\left(\frac{1}{\kappa}\right)_{l,q}\sum_{n=0}^{N_t}\boldsymbol{P_h}^{nK^l\top}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top(\boldsymbol{P^*})_{\boldsymbol{h}}{}^{nK^l}$$
$$+\Delta t|T_{K^l}|\frac{\partial}{\partial \beta_{l,q}}\rho_{l,q}\sum_{n=0}^{N_t}\sum_{d=1}^{dim}\boldsymbol{V_h}^{nK^l\top}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top(\boldsymbol{V^*})_{\boldsymbol{h_d}}{}^{K^l}.$$

Industrial context

Once again, to simplify the developpement of the gradient expression in the industrial code, we compute a reference gradient using the parameterization $(\alpha, \beta) = (\frac{1}{\kappa}, \rho)$ leading to the following expression:

$$\left(\frac{\partial}{\partial \frac{1}{\kappa}_{l,q}}\mathcal{J}(m)\right)_h \approx \Delta t|T_{K^l}|\sum_{n=0}^{N_t}\boldsymbol{P_h}^{nK^l\top}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top(\boldsymbol{P^*})_{\boldsymbol{h}}{}^{nK^l},$$

and

$$\left(\frac{\partial}{\partial \rho_{l,q}}\mathcal{J}(m)\right)_h \approx \Delta t|T_{K^l}|\sum_{n=0}^{N_t}\sum_{d=1}^{dim}\boldsymbol{V_h}^{nK^l\top}\bar{Q}diag(\hat{\boldsymbol{\omega}}\boldsymbol{\delta_q})\bar{Q}^\top(\boldsymbol{V^*})_{\boldsymbol{h_d}}{}^{K^l}.$$

Once these quantities computed, we can, as explained before, express the gradient for any parameterization (α,β).

$$\frac{\partial}{\partial \alpha}\mathcal{J} = \frac{\partial \frac{1}{\kappa}}{\partial \alpha}\frac{\partial \mathcal{J}}{\partial \frac{1}{\kappa}} + \frac{\partial \rho}{\partial \alpha}\frac{\partial \mathcal{J}}{\partial \rho},$$

$$\frac{\partial}{\partial \beta}\mathcal{J} = \frac{\partial \frac{1}{\kappa}}{\partial \beta}\frac{\partial \mathcal{J}}{\partial \frac{1}{\kappa}} + \frac{\partial \rho}{\partial \beta}\frac{\partial \mathcal{J}}{\partial \rho}.$$

We have thus constructed the *discretized gradient* using DG discretization when the model is constant per element and WADG discretization in case of multiple parameters per element. It is worth noting that the discretized gradient is an approximation of the gradient associated with the continuous problem. It is thus inaccurate regarding the continuous problem. This is the reason why the strategy *Optimize then Discretize* is quite criticized [Bonnans, 2006]. Indeed, potential inaccuracies can lead to convergence issues of the optimization algorithms [Gunzburger, 2002]. Non-exact gradient can actually disturb quasi-Newton methods such as L-BFGS algorithm, which is widely used for inverse problems in geophysics.

Moreover, the so-called discretized gradient is not unique. [Sei and Symes, 1994]. Indeed, It depends on the scalar product applied in the functional space and the discretization method used. For instance, the gradient obtained with DG discretization and computed over a model constant per element is not the same as the one obtained with WADG discretization and one quadrature point. The difference is due to the fact that in the first case, the integral is computed on an element by integrating the polynomial basis functions while in the second case, the integral is only approached by a one point quadrature.

We have defined in this section how to compute each component of the discretized gradient according to any parameterization (α, β) of the acoustic first order problem. Its expression varies according to whether we consider constant physical parameters per element or WADG parameterization.

In what follows, we will define the key steps to compute the *discrete gradient*.

3.3.2 Discrete gradient

The *discrete gradient* is the gradient obtained by considering the minimization problem written directly with the discrete problem. We recall that we have determined a synthetic writing of the discrete problem using space-time matrix operators. The associated Lagrangian is then expressed as follows:

$$\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}) = \hat{\mathcal{J}}_h(\bar{\boldsymbol{U}}) + \langle L\bar{\boldsymbol{U}} - E\bar{\boldsymbol{F}}, \bar{\boldsymbol{\Lambda}} \rangle \,.$$

Let us consider the calculation of the derivative for any parameterization (α, β). We will consider the derivative by the kth component and see what are the required operator to compute the discrete gradient whether we are using constant model per element or WADG parameterization. The calculations are the same using β this why we restrict the study to α parameter in the subsection.

Let us consider the derivative by the kth component of the parameter field α:

$$\begin{aligned}\frac{\partial}{\partial \alpha_k}\mathcal{J}_h(m) &= \frac{\partial}{\partial \alpha_k}\mathcal{L}_h(\bar{\boldsymbol{U}}, m, \bar{\boldsymbol{\Lambda}}),\\ &= \frac{\partial}{\partial \alpha_k}\hat{\mathcal{J}}_h(m) + \frac{\partial}{\partial \alpha_k}\langle L\bar{\boldsymbol{U}} - E\bar{\boldsymbol{F}}, \bar{\boldsymbol{\Lambda}} \rangle \,.\end{aligned}$$

Given that only operator L depends on the physical parameters, we get:

$$\frac{\partial}{\partial \alpha_k}\mathcal{J}_h(m) = \left\langle \frac{\partial}{\partial \alpha_k} L\bar{\boldsymbol{U}}, \bar{\boldsymbol{\Lambda}} \right\rangle \,.$$

Then it is important to see that the discrete gradient depends on the time scheme as L corresponds to a space-time discretization. Hence, the discrete gradient will inevitably change with the time scheme. Hence, by replacing L by its expression and summing over all the time steps, we obtain respectively for each time scheme the expression of the gradient regarding the parameter of interest. For RK2 and RK4, the discrete gradient is given by:

$$\frac{\partial}{\partial \alpha_k}\mathcal{J}_h(m) = \Delta t \sum_{n=2}^{N_t-1} \left\langle \frac{\partial}{\partial \alpha_k}(I - B)\bar{\boldsymbol{U}}^n, \bar{\boldsymbol{\Lambda}}^n \right\rangle \,,$$

where

$$\begin{aligned} B &= I + \Delta t A + \frac{\Delta t^2}{2}A^2 \,, && \text{for RK2,}\\ B &= I + \Delta t A + \frac{\Delta t^2}{2}A^2 + \frac{\Delta t^3}{6}A^3 + \frac{\Delta t^4}{24}A^4 \,, && \text{for RK4.}\end{aligned}$$

For AB3, the discrete gradient is:

$$\frac{\partial}{\partial \alpha_k}\mathcal{J}_h(m) = \Delta t \sum_{n=2}^{N_t-1} \left\langle -\frac{\partial}{\partial \alpha_k}\Delta t A \bar{\boldsymbol{U}}^n, \bar{\boldsymbol{\Lambda}}^n \right\rangle .$$

We recall that we have defined A as the global DG operator in space as follows:

$$A(m) = M^{-1}(m)(S_\Omega + S_\Gamma(m)) .$$

Hence, in each case, it is necessary to compute the derivative of operator A with respect to the model parameter. In the above formula, the physical parameters are involved in (M^{-1} and also in S_Γ). Indeed, the matrix S_Γ contains the flux terms that we defined at page 61. These flux terms are functions of the physical parameters. Unlike the *discretized gradient*, the *discrete* one cannot be computed element by element since the flux terms are global operators. It is therefore necessary to develop the operator $\frac{\partial}{\partial \gamma_k} S_\Gamma$, where γ_k denotes the k^{th} parameter for all parameterization (α, β). The fluxes are global operators and will therefore necessarily involve MPI communications, making the calculation of the *discrete gradient* more complex. Indeed, the fluxes are exclusively numerical and do not appear in the continuous problem. We are thus facing one difficulty related to DG discretization: the calculation of the *discrete gradient* is really more difficult than the computation of the *discretized gradient*, as it was formerly observed by Wilcox et al. [2013].

Once again, as for the calculation of the adjoint state of the discrete adjoint problem, the associated gradient also requires the development of additional operators. Its construction is very sensitive to the DG fluxes and turns out to be very difficult to handle. As a conclusion, the implementation of DtO approach seems much more tricky that OtD one. On the one hand, the *Optimize then Discretize* strategy allows us to see the direct problem as a black box that we use almost identically to define the adjoint state. The resulting gradient is then an approximation of the continuous gradient formulation and is therefore computed by exploiting all the DG operators already implemented. On the other hand, the *Discretize then Optimize* strategy allows us to have an exact discrete gradient but requires the development and implementation of the discrete adjoint problem and its discrete gradient.

In order to convince ourselves on which strategy to adopt in the industrial code, we have performed a numerical study in 1D. This is the purpose of the next section.

3.4 1D numerical comparison

In this section, we aim at comparing the performance of the *Optimize then Discretize* (OtD) and *Discretize the Optimize* (DtO) solution methodologies in a simplified FWI 1D test case. For this test, we propose to adapt the 1D solver developed at the beginning of the thesis for pedagogical reasons, into an application solving the inverse problem in 1D. We recall that convergence tests validating this code have been shown in Subsection 2.3.3, page 83.

Having a full mastery of the code sources, it was easy to formulate all the operators as matrices and thus formulate the discrete adjoint problem. Forming the global space-time matrix of the problem is of course to be avoided in production code for reasons of efficiency and memory.

3.4.1 1D FWI problem

Let Ω be a domain of 1000m length. We use a source and a receiver respectively placed at $x_s = 10m$ and $x_r = 300m$ (see Figure 3.3).

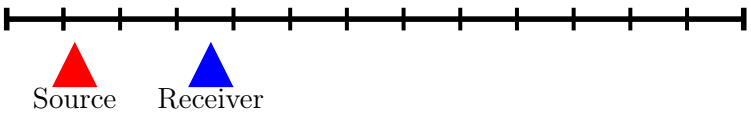


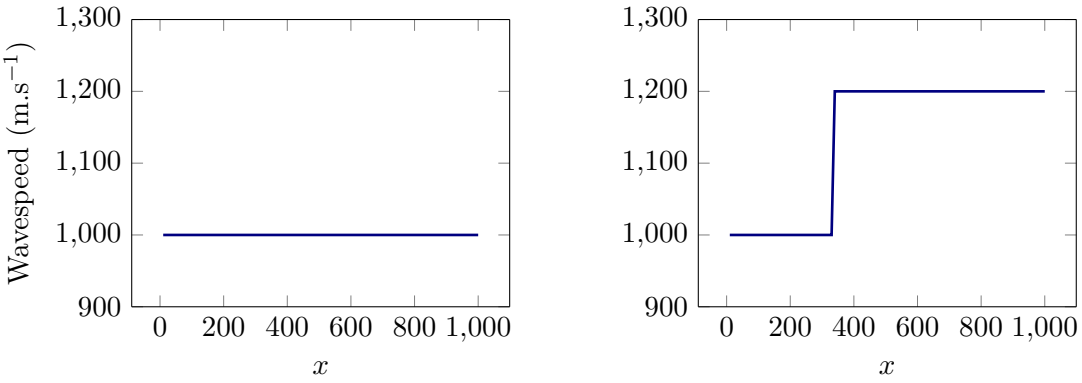
Figure 3.3: 1D domain Ω.

The source f is a first order Ricker function, which is expressed as follows:

$$f(t) = (t - t_{peak}) \exp^{-(\pi f_{peak}(t-t_{peak}))^2} .$$

For this experiment, we choose $f_{peak} = 1Hz$ and $tpeak = 0.5s$.

We generate data from a homogeneous density model with $\rho(x) = 1000\text{kg.m}^{-3}$ but a bi-layer wavespeed medium with $c(x)$=$1000m.s^{-1}$ for 0m$< x <$ 333m and $c(x) = 1200\text{m.s}^{-1}$ for 333m$< x <$ 1000m. The data are obtained by using a 3s-long simulation ($t_0 = 0$s, $T_f = 3$s) For this reconstruction, we propose a homogeneous initial model with $c(x)$=1000m.s^{-1} and $\rho(x)$=1000kg.m^{-3}. We represent the initial model and the target model in Figure 3.4.



(a) Initial wavespeed model.

(b) Target wavespeed model.

Figure 3.4: Initial (a) and Target model (b) for the 1D FWI.

For this test, we used a 100-element grid on which the pressure and velocity fields were approximated by Bernstein-Bézier second order polynomials. The physical model is described by parameters constant per element.

For the sake of simplicity, we have used centered fluxes, thus avoiding the derivation of flux terms with respect to the physical parameters for the discrete gradient calculation. Indeed, as we have previously pointed out, the upwind fluxes, used in Total's code, are functions of the physical parameters. They must therefore be taken into account in the calculation of the DtO gradient, which makes the implementation of the gradient very complicated. We also made sure that the adjoint state resulting from the discrete problem satisfied the adjoint test as introduced in Subsection 3.2.3.

3.4.2 1D Gradient validation

Before comparing the convergence of the optimization algorithm using either the *Optimize then Discretize* (OtD) or *Discretize then Optimize* (DtO) strategy, we aim to verify that the gradient thus obtained by the adjoint state method is valid. First, we propose to compare the gradient obtained via these two strategies with a reference gradient obtained by finite difference (3.21).

The gradient calculated by finite difference for the i^{th} parameter can be calculated as follows:

$$\{\nabla \mathcal{J}(m^0)\}_i = \frac{\mathcal{J}(m^0 + \epsilon\delta^i) + \mathcal{J}(m^0 - \epsilon\delta^i)}{2\epsilon}, \tag{3.21}$$

where $\epsilon\delta^i$ corresponds to a perturbation made on the initial set of parameters m^0 on the i^{th} parameter.

N.B.: Here the problem is one-dimensional, which makes the evaluation of the gradient for all the parameters, by this method, feasible. It would be inconceivable in higher dimensions due to the increasing number of parameters but above all, due to the exponential computational cost. In this example, the gradient obtained by finite difference using a centered scheme is obtained with $2n_p$ evaluations of the cost function against an equivalent cost of two evaluations when using the adjoint state method.

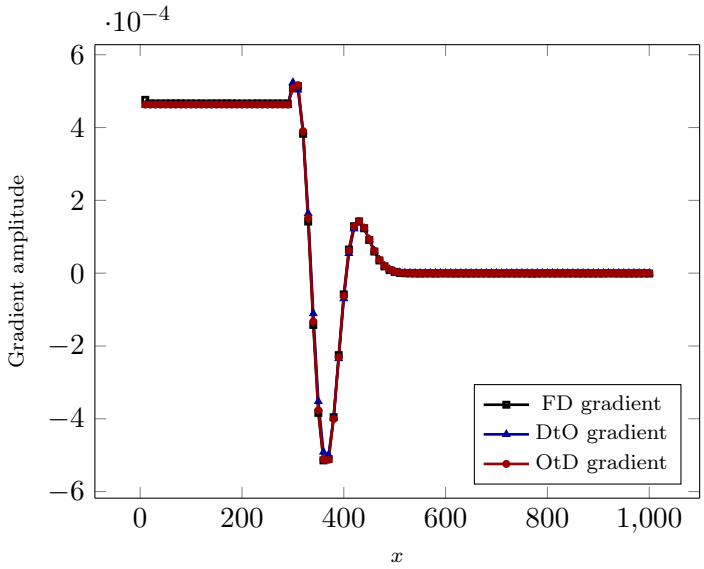


Figure 3.5: Gradient obtained using OtD and DtO strategies compared with gradient computed using FD.

We display in Figure 3.5 the comparison of the gradient obtained by finite difference which serves as reference, as well as those obtained by adjoint state method with the OtD and DtO strategies. All three methods give a similar result. Moreover, we do not observe any significant difference between the gradients obtained by the OtD or DtO strategy.

For further validation of the gradient, we know that for any scalar α, we have the Taylor expansion of the cost function $\mathcal{J}$ which can be expressed as follows:

$$\mathcal{J}(m^0 + \alpha dm) = \mathcal{J}(m^0) + \alpha\langle\nabla\mathcal{J}(m^0), dm\rangle + O(\alpha^2), \tag{3.22}$$

where dm corresponds to a random perturbation of the set of parameters m^0.

We then study the variation of this quantity for α being smaller and smaller:

$$\frac{|\mathcal{J}(m^0 + \alpha dm) - \mathcal{J}(m^0) - \alpha\langle\nabla\mathcal{J}(m^0), dm\rangle|}{|\mathcal{J}(m^0)|} = O(\alpha^2). \tag{3.23}$$

We displayed in Figure 3.6 the convergence curves of this 1D experiment with both gradients computed using OtD and DtO strategies.

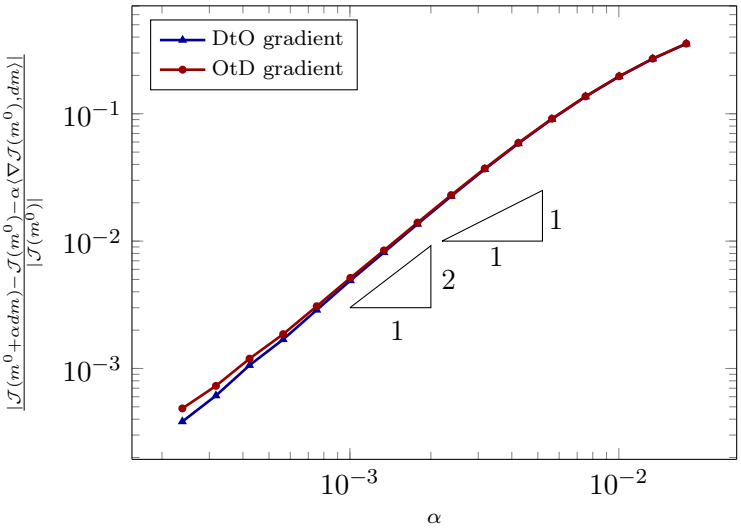


Figure 3.6: Convergence study of the cost function $\mathcal{J}(m^0 + \alpha dm)$ as a function of α.

We can observe that, whatever the chosen strategy (DtO or OtD), we obtain a convergence of order two. This is the result expected if the gradient is well-defined in view of the Taylor expansion of the cost function expressed in equations (3.22) and (3.23). If the slope of convergence is of order one then the gradient obtained is wrong and the convergence is only due to the continuity of the observed quantity for α going to zero. Note that, if the convergence order is null then it exists a trivial bug in the optimization loop. This test is then useful to assess the gradient computed but also to verify the validity of the optimization algorithm.

3.4.3 1D Results and convergence comparisons

We propose to study the evolution of the wavespeed model over 400 optimization iterations using the L-BFGS method. We display in Figure 3.7 the comparison of the final models thus reconstructed. We see that both methods manage to reconstruct the two-layer medium. The strategy *Optimize then Discretize* seems to work in spite of an optimization based on the computation of an approximate gradient and thus not exact in the sense of the continuous problem.

To further compare the two approaches, we propose to compare the speed of convergence of the cost function according to the method used. We display the evolution curves in Figure 3.8.

Finally, we can observe that both strategies lead to similar results. It is difficult to assert the superiority of one method over the other, at least in this case. Convergence is not faster with one method than another. This numerical comparison does not make it possible to assert the superiority of one strategy over the other, notwithstanding the fact that the OtD strategy performs the optimization with an approximation of the continuous gradient that can alter the convergence of the optimization algorithm.

3.5 Conclusion

Before proceeding to implement the inverse problem in the Total code, it is important to determine the strategy to be followed. Since the problem is formulated as a constrained

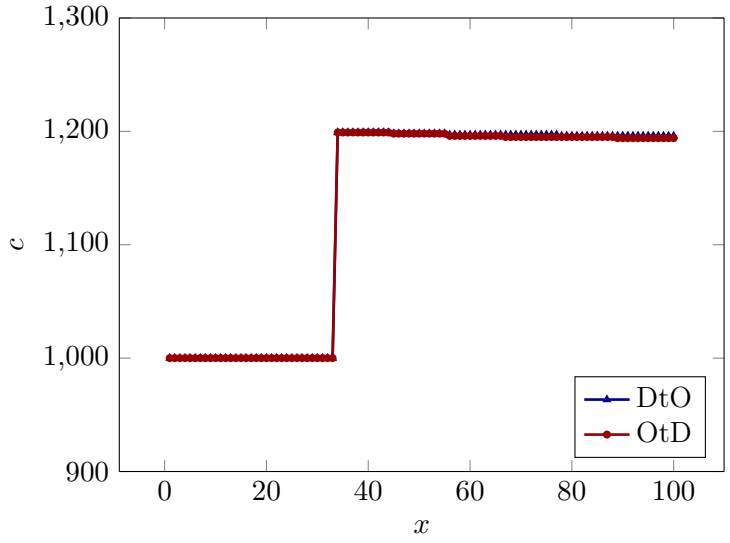


Figure 3.7: Final 1D wavespeed model reconstructed in 400 L-BFGS iterations using OtD and DtO strategy.

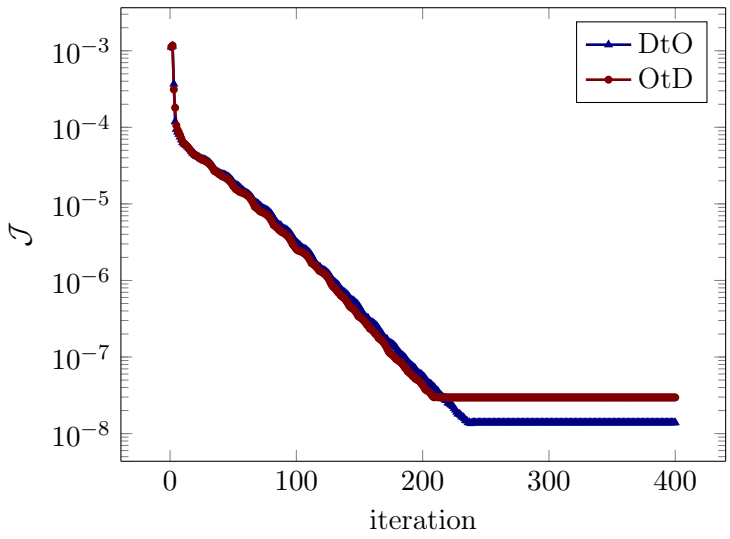


Figure 3.8: Evolution of the cost function through 400 L-BFGS iterations using OtD and DtO strategy.

minimization problem solved by the adjoint state method, should we proceed with the optimization of the continuous problem and then discretize it or on the contrary, implement the optimization on the discrete problem? It would seem that a consensus [Bonnans, 2006, Chavent, 2010, Kern, 2016] chooses the strategy *Discretize then Optimize*. Indeed, with the strategy *Optimize then Discretize*, the gradient of the cost function with respect to the parameters to be reconstructed is only an approximation of the gradient of the continuous problem. This approximation can then lead to imprecisions on the gradient, which can be detrimental to the efficiency of local optimization routines [Gunzburger, 2002] and all the more when one tries to approach the Hessian of the cost function with methods such as L-BFGS. On the other hand, the gradient obtained by the *Discretize then Optimize* is correct in the sense of the discrete problem and does not suffer from approximation penalizing optimization.

We have shown in this chapter that, depending on the used strategy, the needs for development are not the same. Indeed, if we consider the optimization of the continuous problem, the associated adjoint state can be written as a solution of the same continuous problem with another right hand side expressed in terms of the solution of the direct problem. It is then possible, with this strategy, to use the same discretization for the forward problem as for the backward problem. This allows to avoid a specific development of the adjoint problem that is necessary in the strategy *Discretize then Optimize*.

The difficulty resulting from the implementation of the discrete adjoint problem is relative to the discretization of the direct problem and to the architecture of the code. Even if automatic differentiation tools are used to obtain the desired functions, these are often poorly optimized and difficult to include into the parallelism, if it exists. In the industrial context of this thesis, it seems to us impossible to use this type of tool. Indeed, we would have to consider another problem given by the associated adjoint problem that would take a very long time to solve. It is actually necessary within the framework of the strategy *Discretize then Optimize* to develop the discrete adjoint problem. In this chapter, we have reviewed the necessary developments to show how many efforts are required. For a time domain DG solver, these developments can be complicated by the complex interplay of operators expressing themselves either globally (time and fluxes) or locally within a mesh (mass and stiffness matrix). Once the discrete adjoint state has been determined, the discrete gradient must still be computed. The latter is also complex to determine in DG framework if upwind fluxes are used. Indeed, it involves the physical parameters and must therefore be derived for the calculation of the exact gradient in the sense of the discrete problem [Wilcox et al., 2013]. In practice, this derivation is ignored [Wang, 2017] and an approximate gradient is used.

Nevertheless, we can cite the case of solvers that are more conducive to the expression of the adjoint state. In FEM/SEM solvers, all operators are more simply decomposed than DG ones into local matrices. Their adjoint operators are then easier to determine than the adjoint operators of DG fluxes. For example, the AxiSEM3D code, which is a solver combining SEM and Fourier pseudo-spectral method for the azimuthal wavefield, has been shown to be self-adjoint [Szenicer et al., 2020].

Even if the construction of the discrete gradient in DG formulation seems very complex, we wanted to implement it on a 1D case. The objective of this numerical study was to compare discrete and discretized gradients in a simple inversion case. We have decided to use the L-BFGS algorithm, which was expected to perform better with DtO approach. We observed that the two calculations lead to similar results.

In the following, we opt for the implementation strategy privileging the method *Optimize then Discretize*, reassured by a simple 1D case. Since we have to develop from scratch the minimization problem in Total's code, there is no structure managing the adjoint state, the gradient or even the optimization routines. It is therefore easier to set up all the necessary

structures by taking the forward operators to calculate the backward field. Getting the discrete adjoint state requires a sharp knowledge of the implementation of the industrial direct solver and, after that, a consequent development time while the entire optimization structure does not exist yet. Starting with the strategy *Optimize then Discretize* will therefore allow us to set up everything we need, and build the skeleton of the FWI while leaving the possibility, for later, to calculate the discrete exact gradient.

In the next chapter, we will describe the industrial framework in which this thesis has been developed.

Chapter 4

Time Domain Full Waveform Inversion in an industrial context

The objective of this thesis is to implement a FWI algorithm using a time-domain DG solver provided by the industrial partner Total.

The choice of the DGm has already been motivated in Chapter 2. The DGm actually offers a great flexibility to handle complex subsoil models (topography, discontinuities) and allows a simplified parallelism as it works by communicating only fluxes between each element.

FWI has been used in the frequency-domain for a long time. Its frequency formulation is indeed very practical since it is limited to solving the direct problem with different right-hand sides, depending on whether one calculates the forward or backward solution. If one has a solver that is able to exploit the multi right-hand side structure of the discrete inverse problem, one reduces the computational burden considerably and the method can operate in 3D [Pocock and Walker, 1998, Operto et al., 2014, Faucher, 2017]. Another advantage of the frequency approach is that wavefields need to be stored only for a discrete set of frequencies that are not in large numbers. The efficiency of the inversion method is also considerably increased by a multi-frequency approach, but it is not easy to determine which frequencies to work with. FWI in the time domain requires more computational resources, especially because all wave fields must be stored. Shoja et al. [2018] performed a study in 2D on a simple model (four parameters) that shows that the time approach is more robust than the frequency approach and that this robustness can be explained by the possibility to take into account a full band of frequencies. In [Singh et al., 2018], a comparison of the two approaches is also carried out using the Marmousi model as a target. This study highlights the following points:

- the time method is more efficient when it comes to solving the direct problem in large propagation domains; the frequency method may indeed be limited by the ability of solvers to solve large-scale problems;
- the frequency method is less expensive than the time method: computational costs are proportional to the number of frequencies used if the multi right-hand-side option is implemented; in time, computational costs are proportional to the number of sources;
- the frequency method is very sensitive to the choice of the initial model and the data must have a very low frequency content to obtain accurate results.

Given that the time solver requires less memory than the frequency one, one option consists in performing time-dependent forward simulations on 3D large cases in time-domain and then

to perform the optimization in the frequency-domain [Sirgue et al., 2010, Brossier et al., 2014]. Unfortunately, this approach requires using discrete Fourier transforms over the whole wavefield which turns out to be very expensive.
In an industrial context, the choice of a full time-domain FWI is motivated by the will to overcome the memory limitations of frequency solvers. Even in 2D, the forward problem is solved in a much more efficient way than in time domain. More importantly, in 3D, frequency solvers require a significant amount of computer memory and often have difficulty solving large-scale problems [Operto et al., 2007]. One way to deal with this problem is to use an iterative solver [Warner et al., 2008]. However, iterative methods require good preconditioners. Moreover, iterative solvers do not deal with multiple right-hand sides which increase drastically the solution of the inverse problem. Hence, the resolution of 3D problems solved in frequency-domain comes to a bottleneck and as our industrial partner has to deal with 3D problems, we chose to develop an FWI in the time domain. It is worth noting that the use of a time-domain solver allows to limit the storage of the forward field, which is necessary to solve the inverse problem. Checkpointing strategies [Griewank, 1992] or storage of the solution only at the interfaces [Clapp, 2008] can be used to limit the storage requirement for the cost of additional calculations.

In this chapter we will describe the industrial environment in which the development of the FWI has been carried out. We will then present the different results of the reconstructions obtained.

4.1 Industrial architecture and contributions

In this section, we will describe the industrial environment in which the FWI code was developed.

First, we will introduce the development environment. Then, we will describe the resources and clusters we had access to for solving the inverse problems. Finally, we will explain the different levels of parallelism we had to deal with to realize massively parallel inversion code.

4.1.1 Description of the development environment

The industrial code in which the development of the thesis took place is a Fortran code separated into three libraries and a main application. These libraries are subject to a hierarchical dependency. We can then sketch out the whole code as in Figure 4.1.

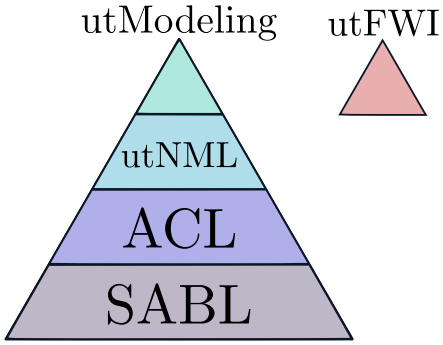


Figure 4.1: Illustration of the hierarchical Total's code.

Going from the base to the top of the pyramid that is the application we have:

- **SABL**: Seismic Application Base Library is a library that contains a set of packages commonly used in different seismic applications. Mainly, it is this library that will manage parallelism, errors, as well as the reading of physical parameters, seismic traces and acquisition configuration (sources and receivers locations).
- **ACL**: Application Core Library is an application that will manage everything concerning space discretization. Indeed, we will find there the management of the mesh, the decomposition in sub-domains as well as the creation of the local DG operators according to the various polynomial bases.
- **utNML**: Unstructured Time-domain Numerical Methods Library contains various propagators on unstructured meshes (acoustic isotropic, elastic isotropic, elastic TTI and coupled elastic-acoustic isotropic propagators). The time schemes are developed in this library and it is also where local DG operators, built thanks to **ACL** to construct these different propagators, are assembled. In this library we may also find the types characterizing the physical parameters and seismic data, which are key for FWI.
- **utModeling**: Unstructured Time-domain Modeling is the main application. It will read the input parameters provided by the user, initialize the parallelism and simulate the time domain wave propagation with the propagator provided by the user.

At the beginning of this thesis, only the application **utModeling** existed. Then, the new application **utFWI**, on the model of **utModeling**, has been developed to perform FWI in time domain with DG propagators. The **utFWI** application must provide a generic FWI workflow compatible with all existing propagators. Basically, the **utFWI** application follows the global FWI workflow defined earlier and illustrated in Figure 4.2.

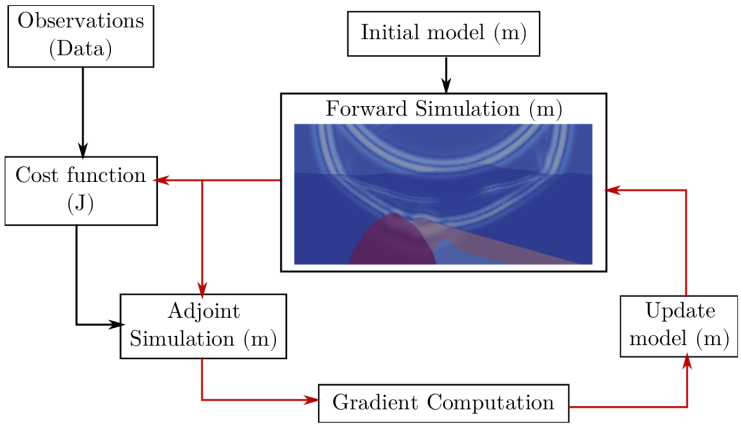


Figure 4.2: General FWI workflow.

For specific functions induced by the propagator, it is necessary to develop the library below, referred to as **utNML**.

Each of the codes comprising the FWI workflow is followed by its own git repository thus splitting the development library by library. The majority of the developments in this thesis have been done in **utFWI** and in **utNML**.

We have developed specific applications compatible with the libraries provided by the industrial partner. We have also enriched these libraries with functions specific to inversion,

especially in the acoustic propagator.

Now that we have briefly described the development environment, we propose in the rest of this section to make an overview of the HPC framework in which the thesis took place.

4.1.2 HPC environnement

The very high cost to perform forward modeling, associated to a large number of data (number of sources), makes FWI a complex High Performance Computing (HPC) problem.

To solve either forward or inverse problem, we use a parallelism developed in Open MPI [Gabriel et al., 2004]. This formalism allows to separate the calculations on cores. One important issue is the development of communications that allow data exchange between cores when necessary.

In the developed code, we have preserved and adapted the existing parallelism for solving FWI problems. There are two levels of parallelism:

- one parallelism by subdomain decomposition;
- and one parallelism per shot.

What we call a shot, is the resolution of the direct problem for a given source. To carry out a shot, it is possible to divide the domain of calculation Ω in subdomains (see Figure 4.3).

Figure 4.3: Illustration of a potential domain decomposition to solve one forward problem.

Subdomain decomposition is widely used for solving partial derivative equations either using finite difference or finite element methods [Smith et al., 2004]. The advantage with DGm is that it drastically reduces the cost of communication between each subdomain thanks to numerical fluxes that are part of the numerical method. In fact, in DGm, it is enough to communicate with the direct neighbor regardless of the order of approximation in space, while other solvers using continuous finite element methods require communication with more distant elements. Having a high amount of information to be exchanged during communication can hammper the scalability of the problem, this is an additional reason to favour DG methods. The subdomain decomposition displayed in Figure 4.3 has been obtained by the Total code which uses the external library PARMETIS [Karypis et al., 1997]. It is important to have an appropriate decomposition to have the best load balancing possible.

To highlight the scalability property provided by DG technology, we displayed in Figure 4.4 the strong Speed-up curve obtained by computing the acoustic wave on a large 3D domain of 5km×9.1km×3.2km. The spatial discretization is based on a mesh of 72 526 tetrahedra where the solution is approximated by polynomials of order 4. For this experiment, we are using a first order Ricker pressure source with a frequency peak of 15Hz.

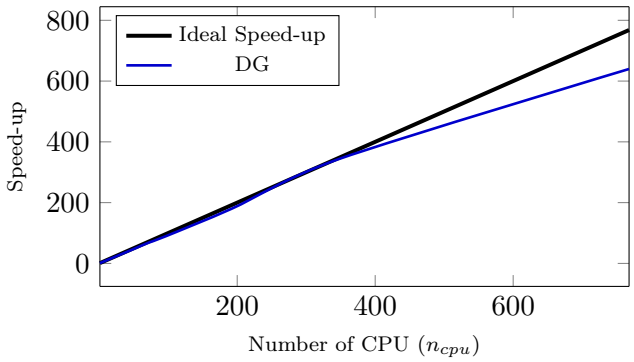


Figure 4.4: Speed-up analysis using 3D DG acoustic solver.

The speed up is computed as follows:

$$\text{Speed-up}(n_{cpu}) = \frac{t_1}{t_{n_{cpu}}},$$

where:

- t_1 represents the computational time required for one CPU core to compute the simulation,
- $t_{n_{cpu}}$ represents the computational time required for n_{cpu} CPU cores to compute the simulation.

In the ideal case, we expect to have simulations n_{cpu} times faster when using n_{cpu} cores (Speed-up$(n_{cpu}) = n_{cpu}$). As formerly mentioned in many publications, we can observe in Figure 4.4 that DG solvers have interesting HPC properties, since the experimental curve is close to the ideal one. This is due to the low communication cost of DG solvers. Indeed, the discontinuous finite element methods only required flux communication between each subdomain [Biswas et al., 1994]. Note that for high numbers of subdomains, the amount of communication increases and the speed-up is getting less competitive. This is why, in what follows, we will propose another level of parallelism that consists in computing shots separately.

We have just described the parallelism defined for one shot. However, for the inverse problem, the resolution of the direct problem has to be done over several shots. It is then intuitive to compute the shots in parallel. For large problems, it is necessary to couple these two levels of parallelism. If only subdomain decomposition is used, then, for a large number of cores, communications would kill the scalability of the problem. Conversely, if only shot parallelism is used, only one core must have enough memory to carry out the forward modeling over the entire domain Ω.

We denote by *nb_cluster* the number of shots performed in parallel. In order to simplify the communications, we consider that each shot has the same subdomain decomposition. Let *nb_domain* be the number of subdomains to compute each shot. We call a *cluster* a set of *nb_domain* cores. Then each *cluster* has to compute a gradient, which we will denote as a subgradient since it is computed with the subset of shot attributed to it. Each *cluster* has its own subgradient, which is itself divided into the different *nb_domain* subdomains. In this way, the number of cores to allocate for the job is necessarily given by *nb_cluster* $\times$ *nb_domain*. For the same reasons, we favor that the total number of sources considered is a multiple of the number of *cluster*: *nb_cluster*.

The global gradient for the considered full set of sources is then the sum of the subgradients calculated on each *cluster*. We have developed an additional communication in order to adapt the FWI to the existing parallelism already implemented for the direct problem. Once all the subgradients are computed, we have to sum the contribution of each *cluster* subdomain by subdomain. The difficulty of such a communication is that it is necessary to distinguish the global rank of a core from its local rank in each *cluster*. This information is contained and managed in types defined in the **ACL** library. We illustrate the computation of the global gradient using the two levels of parallelism, we just described, in Figure 4.5.

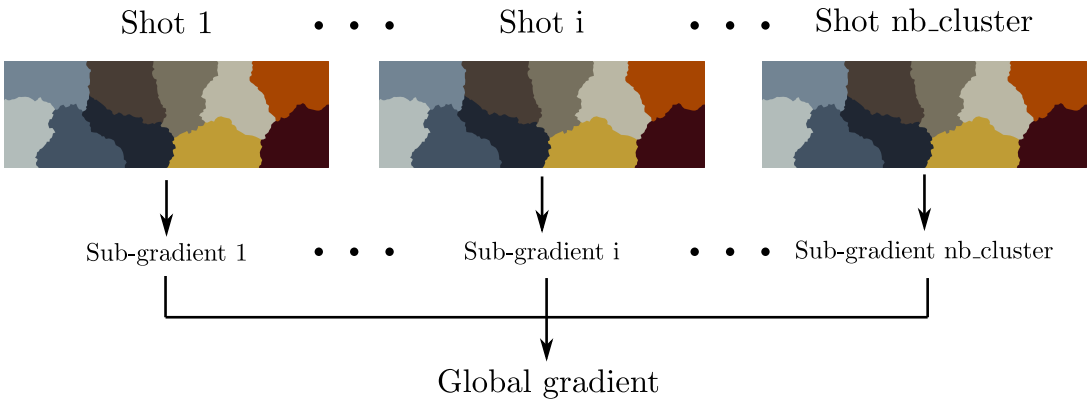


Figure 4.5: Illustration of shot parallelism for gradient computation.

The choice of $nb\_cluster$ and nb_domain must then take into consideration the size of the computational domain and the number of available shots. To illustrate the choice of those two quantities, we studied on a 2D model the computational time of 20 shots for different ($nb\_cluster$,nb_domain) configurations (see Figure 4.6).

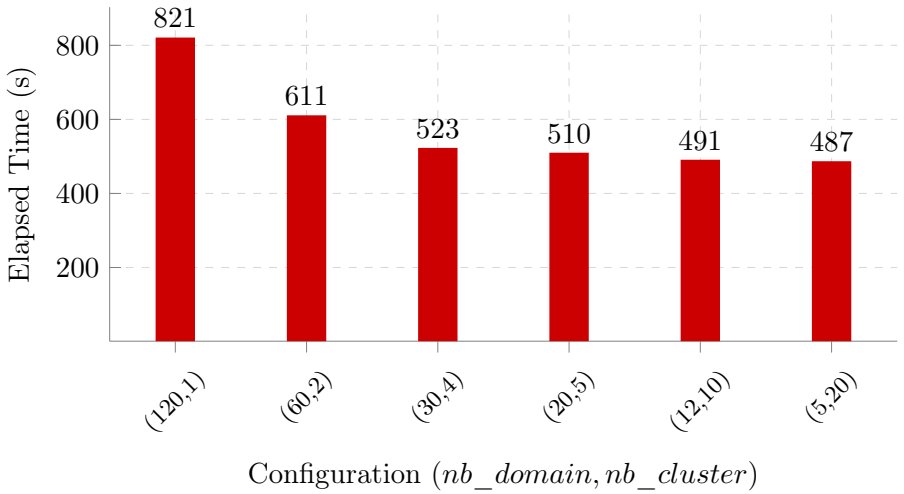


Figure 4.6: Computational time for 20 shots simulations for several ($nb\_domain, nb\_cluster$) configurations.

In this experiment, we are timing the calculation of the 20 shots propagating on the Marmousi 2D model. This consists in the simulation of a 5s-long acoustic propagation caused by 20 pressure sources modeled by a first order Ricker. The elapsed times reported in the Figure 4.6 allow to highlight the influence of ($nb\_domain, nb\_cluster$) configurations employed.

This choice has to take into account the geometry and the discretization of the simulated problem. In this example (≈7500 P4 triangles), the choice of parallelism allows, for a fixed number of cores (here 120 CPU cores), to reduce the computational time by almost 40%.

To carry out the reconstructions that will follow, we had the opportunity to use different computer clusters provided by Total. We had access to the Mesquite and Hickory clusters located in Houston and the Pangea2 cluster located in Pau. Most of the 2D reconstructions were launched on Mesquite while the 3D reconstructions required using Hickory or Pangea2 as they provide greater computing capacity.

We give in Table 4.1 the number of cores available on each of these clusters.

Cluster	Location	Nodes	Cores
Mesquite	Houston	24 CPU	480
Hickory[1]	Houston	48 CPU + 64 CPU/GPU 176 KNL	13632
Pangea2	Pau	4608 CPU	110592

Table 4.1: Cluster information.

In this section, we have described the industrial environment in which we have developed the FWI time-domain code. We have also briefly described the used parallelism and the clusters provided to perform our reconstructions.

In the rest of the chapter, we will test the inverse problem with the different options proposed by the industrial solver. Then, we will define what is a multiscale reconstruction, and we will show some results using this process for the reconstruction of different 2D and 3D models.

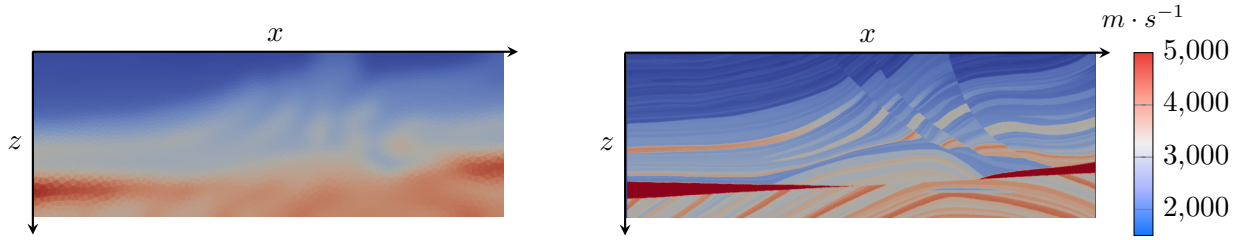
4.2 Reconstruction analysis regarding the discretizations and the optimization

Whether we solve the direct or the inverse problems, we have several discretizations in the code. One can choose the time scheme, the polynomial basis used for the DG method, the type of parameterization to describe the model that can be defined either by a set of constant parameters per element or by variable parameters which are managed by the WADG method. In this section, we will look at the influence of these choices on the reconstruction performed by the developed FWI.

Throughout this section, we aim at reconstructing the Marmousi wavespeed model from an initial guess obtained with a smoothed version of the true model. The initial model is constructed by using a Gaussian filter with a standard deviation choosen to be equal to 40. This filter is applied to the true model defined as regular grid of resolution 2301×751. Concerning the density, we will consider it as known and homogeneous all over the domain: $\rho = 1000$ kg.m^{-3}. We display the initial guess and the target model in Figure 4.7.

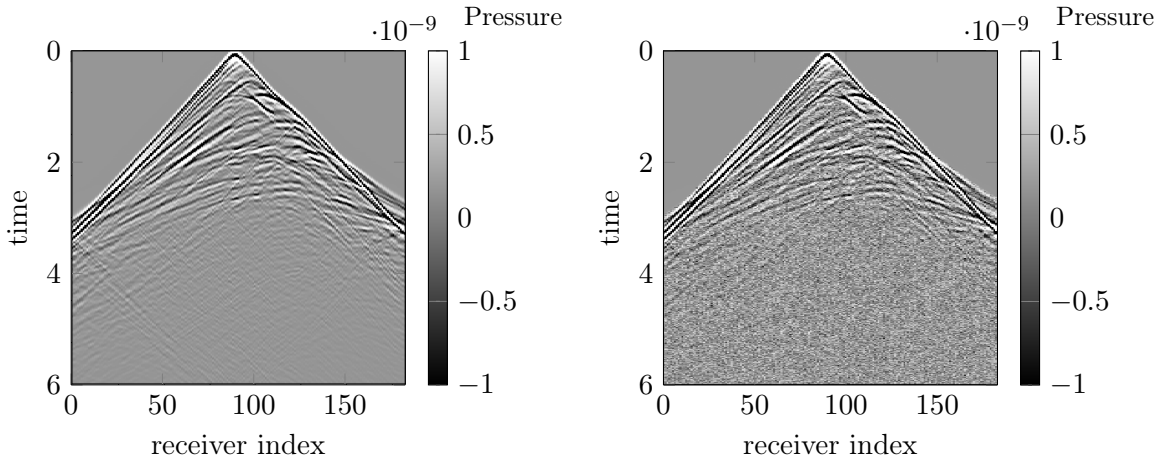
For the reconstruction we will use data generated with the true model by using 20 sources inducing a first order Ricker pressure perturbation ($t_{peak} = 0.2s$, $f_{peak} = 10Hz$). These sources are evenly located on the X-axis from 550m to 8150m with 400m in between at 10m depth. We use 183 receivers at a depth of 100m positioned from 50m to 9150m on the X-axis with 50m between each other. The data so obtain are then noised with a white Gaussian noise with a Signal Noise Ratio (SNR) of 10 (see Figure 4.8c).

[1]Hickory has been shut down in september 2020.

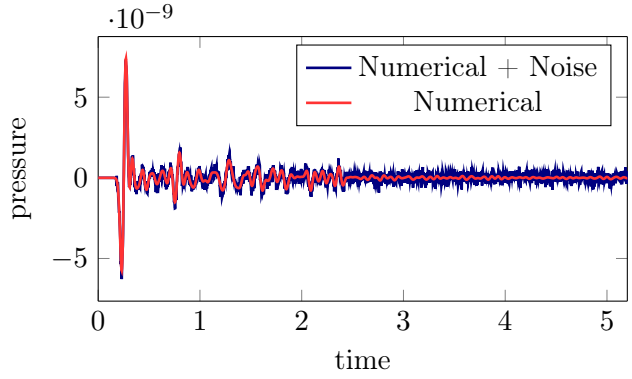


(a) Initial velocity model (m.s^{-1}) (b) Target Marmousi velocity model (m.s^{-1})

Figure 4.7: Initial and target model.



(a) Sismograph obtained by simulation of the 10^{th} source. (b) Sismograph obtained after adding white Gaussian noise (SNR=10).



(c) Comparison of the trace on the 80^{th} receiver with and without noise.

First, we will test the different optimization methods that have been implemented in the industrial code. Then, we will study the influence of discretization, i.e., the time scheme, the polynomial basis and the parameterization of the model depending on whether we use WADG method.

4.2.1 FWI reconstruction with different search direction method.

The optimization algorithm is carried out through an iterative process that ends up with updating the physical model m^i at the i^{th} iteration into m^{i+1} in order to make the cost

function $\mathcal{J}$ decrease "*well enough*". Such update follows the relation:

$$m^{i+1} = m^i + \alpha^i d^i,$$

where α is the line search step length coefficient that we described in Subsection 1.4.4 and d is the search direction.

We introduced in Chapter 1 different search directions. In the industrial environment, we have implemented the following ones:

- Steepest descent method
- NonLinear conjugate gradient (NLCG) [Nocedal and Wright, 2006] with several formulations:
  - Fletcher-Reeves (FR)
  - Polak-Ribière (PR)
  - Hestenes-Stiefel (HS)
  - Dai-Yuan (DY)
- Limited-BFGS (L-BFGS)

We choose to perform 50 iterations using the full frequency component of the observed data. For the reconstruction, we will use a mesh composed of 10977 P3 elements using nodal polynomial basis. The model is parameterized with a set of constant parameters per element which means that we have 10977 velocity parameters to retrieve.

We display in Figure 4.9 the behavior of the cost function during the 50 iterations.

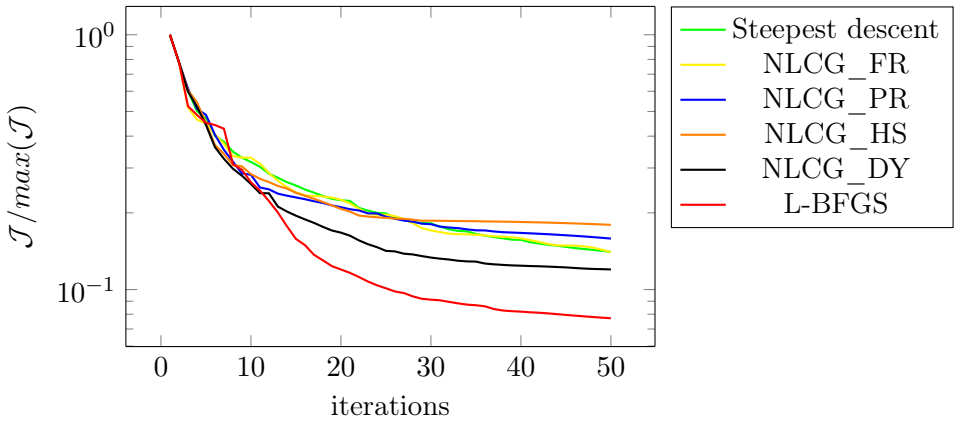


Figure 4.9: Cost function evolution for different search direction strategies.

First of all, we can see that all the optimization algorithms converge. For this case of study, it seems that steepest descent and NLCG methods have quite the same behavior. As expected, the Limited-BFGS method gives the best convergence as it is a quasi-Newton method that aims to give an estimate of the Hessian.

NLCG methods can be enhanced by adding a restart condition as suggested in [Nocedal and Wright, 2006]. A restart consists in using the steepest descent direction instead of the one computed with NLCG methods in order to refresh information from previous old iterations. We recompute the reconstruction using a restart all n iterations with $n = 10$ in this case. We display in Figure 4.10 the corresponding behavior of the cost function with restart.

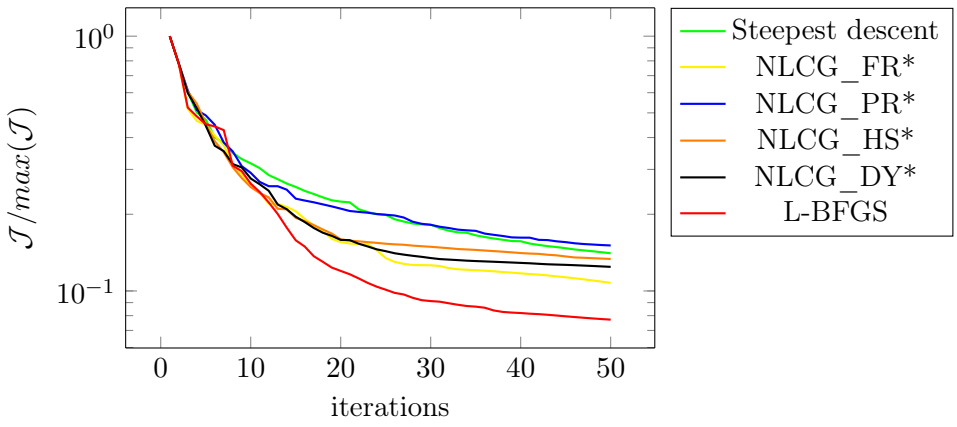


Figure 4.10: Cost function evolution for different search direction strategies; NLCG methods have been performed with a restart all $n = 10$ iteration.

Figure 4.10 clearly shows the advantage to use seldom restarts in the NLCG search direction. In this test, all optimizers give better convergence than the steepest descent except for Polak-Ribière NLCG formulation which remains close to it. The improvement is particularly impressive when comparing the cost functions in Figure 4.9 and Figure 4.10 around the twentieth iteration.

In Figure 4.11, we compare the final velocity model reconstructed for different search direction methods.

The optimization methods implemented in Total's environment accurately recover the velocity Marmousi model with a comparable accuracy and computational time. The different layers of the domain are well located with the appropriate wave speed values. It is worth noting that in addition to obtain the better cost function, the Limited-BFGS approach is also capable of retrieving the deepest structures of the target medium. To illustrate this important feature of L-BFGS algorithm, we display in Figure 4.12 a one dimensional profile of the reconstructed velocity model using Limited-BFGS algorithm in comparison with the initial guess and target results. This profile is established on the segment defined by the vertical axis $x_1 = 4500m$.

The profile obtained with the L-BFGS method is close to the true profile we are looking for. We notice that the reconstructed model is more accurate near the surface than at depth. The deeper a structure is, the more difficult it is to reconstruct it accurately.

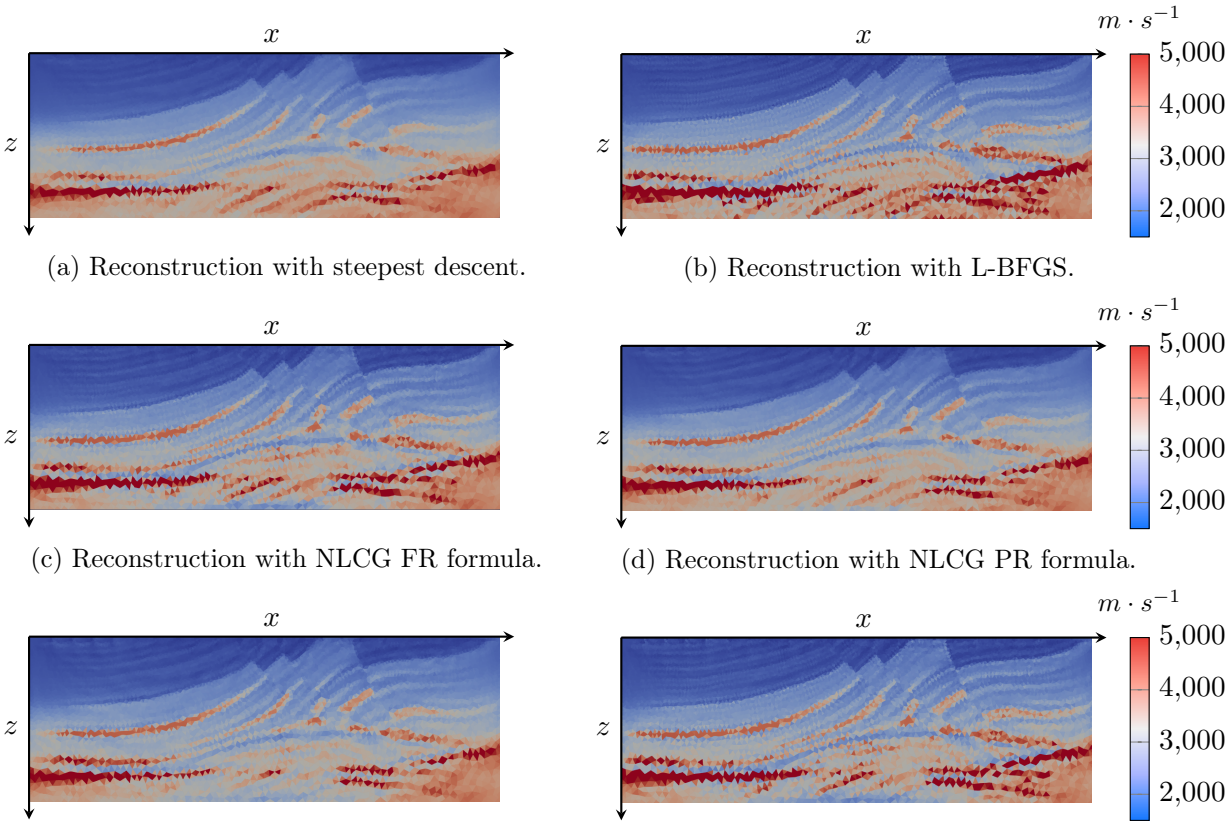
We presented in this subsection the different convergence curves obtained for the different quasi-Newton algorithms implemented in the code. All the methods are able to reconstruct pretty well the targeted model. For this experiment, Limited-BFGS gives the best convergence results as expected since this method aims to approximate the Hessian of the cost function. This method is well known in geophysical community for its efficiency and in addition its reasonable memory burden. Other examples where the L-BFGS algorithm has been successfully applied can be found in [Virieux and Operto] and [Brossier et al., 2010].

For the rest of the thesis, the optimization process will be exclusively performed using Limited-BFGS search direction.

4.2.2 Influence of the time scheme

In the industrial solver, we can use three different time schemes

- a second order Runge-Kutta time scheme (RK2);



(a) Reconstruction with steepest descent.

(b) Reconstruction with L-BFGS.

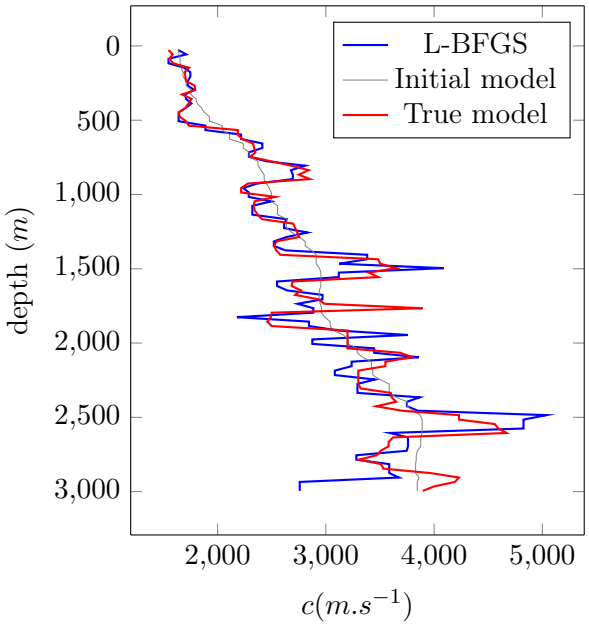
(c) Reconstruction with NLCG FR formula.

(d) Reconstruction with NLCG PR formula.

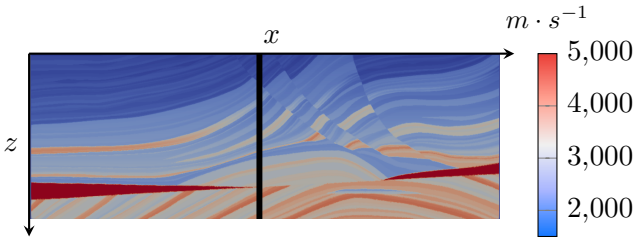
(e) Reconstruction with NLCG HS formula.

(f) Reconstruction with NLCG DY formula.

Figure 4.11: Final reconstruction of the Marmousi velocity model obtained with an initial smoothed model using the entire frequency component of the source and the observed data for different search directions.



(a) 1D profile.



(b) Location of the 1D profile.

Figure 4.12: 1D profile of the Marmousi velocity model reconstructed with L-BFGS search direction compared with initial and target model.

- a fourth order Runge-Kutta time scheme (RK4);
- the third order Adams Basforth time scheme (AB3).

They have been introduced in Subsection 2.2.3 page 71.

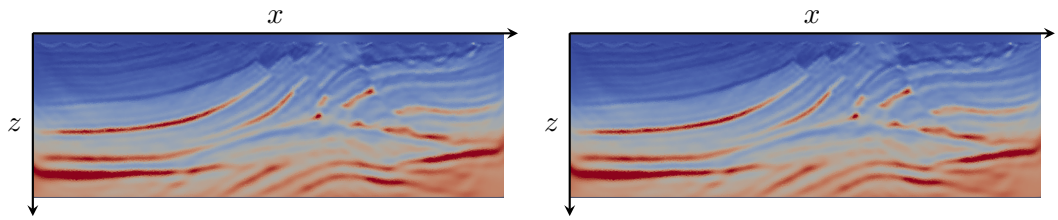
The objective of this subsection is to analyze the influence of the time scheme on the reconstruction of the Marmousi velocity model previously described in 4.2. For these reconstructions, we used a P1 mesh with 47000 elements. The time step is adapted at each iteration of the FWI, following the update of the wavespeed model, according to the criterion established with the CFL condition.

Industrial context

At the beginning of the thesis, Total's code was meant to solve the direct problem and the time step was computed manually by dichotomy. This approach is perfectly adapted to simulations with a single velocity model represented with parameters constant per elements. However, when performing a FWI, the model is likely to evolve. Hence, a time step Δt given as input may no longer be correct. This could ruin the job and require a manual restart.

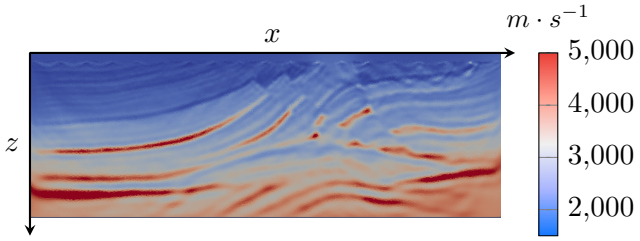
It was then imperative to experimentally define an agile CFL condition and to implement it in the solver. This is mandatory to have a scheme that keeps stable at any FWI iteration where the model keeps evolving. In addition, respecting the CFL gives a reasonable time step size, which prevents from hampering the computational time. We have developed in the solver an automatic calculation of the time step that takes into account the mesh, the time scheme and the velocity model. We give the definition of CFL that we have developed on page 69.

After 30 iterations of the FWI, we get, for each time scheme, the reconstructed Marmousi velocity models depicted in Figure 4.13. We do not observe any difference between the reconstructed models obtained with a time scheme or another. This observation is corroborated by the graph in Figure 4.14, which shows a strictly similar evolution of the cost function during iterations.



(a) Marmousi velocity model (m.s^{-1}) after 30 iterations using RK2.

(b) Marmousi velocity model (m.s^{-1}) after 30 iterations using RK4.



(c) Marmousi velocity model (m.s^{-1}) after 30 iterations using AB3.

Figure 4.13: Marmousi velocity model (m.s^{-1}) reconstructed using several time schemes.

However, if we observe the same accuracy of the reconstruction, it might have a difference in calculation times. We have thus displayed the computational time associated with each reconstruction in Table 4.2.

	RK2	RK4	AB3
Elapsed time	3h15	4h30	5h10

Table 4.2: Comparison of the computational time for Marmousi wavespeed model reconstruction using different time schemes.

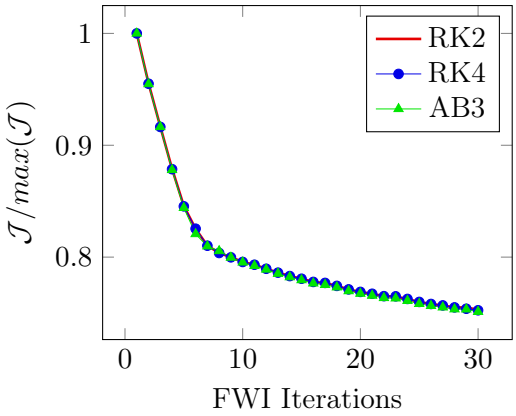


Figure 4.14: Evolution of the cost function using different time schemes.

Indeed, we are observing that the choice of the time scheme has an impact on the global computational time. We also notice that, despite the fact that RK4 scheme is a 4^{th} order time scheme, it is faster than AB3 scheme, which is a 3^{rd} order scheme only. Indeed, forward and backward computation represent the main part of the FWI computational burden. Relaxing the discretization allows then to accelerate the reconstruction. In addition, high lovel of accuracy is unnecessary regarding the uncertainties on the data.

For this reason, for all the other reconstructions performed during this thesis, we have chosen to use the RK2 scheme which gives the best performance in terms of computational time.

In the next section, we propose to make a similar study comparing a reconstruction using either the nodal polynomial basis or the Bernstein-Bézier modal basis.

4.2.3 Influence of the polynomial basis

In this section, we will compare the reconstruction of the Marmousi model when using the nodal polynomial basis [Hesthaven and Warburton, 2007] or the Bernstein-Bézier modal basis [Chan and Warburton, 2016]. We already had the opportunity to show, in Chapter 2, page 76, that a simulation performed with one or the other of these bases is equivalent. There is therefore, in theory, no difference to be expected from such a comparison when addressing the solution of the inverse problem. However, comparing the two reconstructions allows to validate the gradient formula developed in the previous chapter (see page 128) regardless the considered polynomial used.

For this reconstruction, we will use the same discretization as in the previous test, i.e. a mesh of 47000 P1 elements and a model constant per element. The time scheme employed is RK2.

As we can see in the graph in Figure 4.15, the cost function evolves in the same way whatever the polynomial basis used. This similarity validates the implementation of the discrete gradient whether a nodal or modal basis is used.

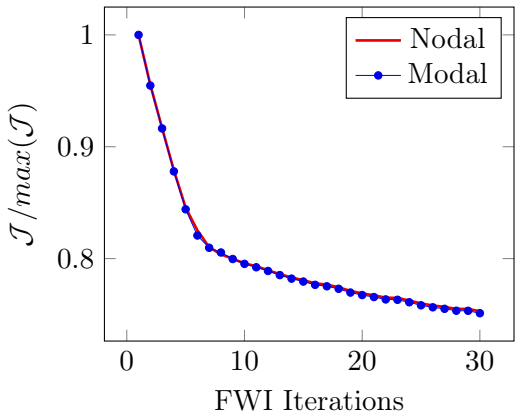
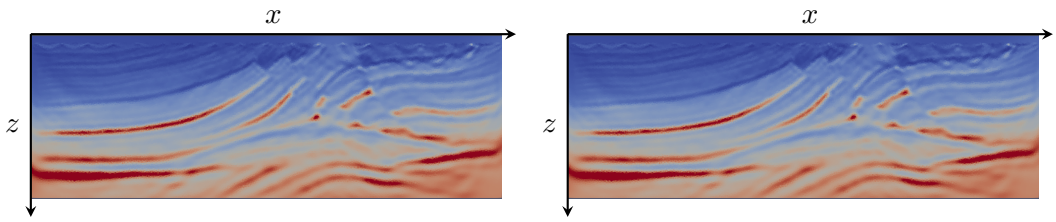


Figure 4.15: Evolution of the cost function using nodal or modal polynomial bases.

We can see that the reconstructed model is strictly identical for both polynomial bases. We display the obtained velocity models in Figure 4.16.

However, the polynomial basis has an impact on the computational time. We display in Table 4.3 the value of the computational time for the reconstruction performed with the two polynomial bases.

With P1 elements, we see that using Bernstein-Bézier elements takes 1.9 times longer than when using the nodal polynomial basis. This factor is close to the one obtained during the



(a) Marmousi velocity model (m.s^{-1}) after 30 iterations using RK2 and Nodal basis.

(b) Marmousi veocity model (m.s^{-1}) after 30 iterations using RK2 and Modal basis.

Figure 4.16: Marmousi velocity model (m.s^{-1}) reconstructed using nodal and modal polynomial base.

	Nodal	Modal
Elapsed time	3h15	6h20

Table 4.3: Comparison of the computational time of a Marmousi velocity model reconstruction using different polynomial bases (using 120 CPU cores).

numerical tests carried out on page 90 of Chapter 2. Hence, it seems that the computational time for solving the direct problem with a given discretization gives a reliable hint on the computational time for solving the inverse problem with the same discretization.

Since we find a similar factor when performing FWI, we deduce that the factor obtained on the direct problem can be used to choose the most suitable polynomial base for the inverse problem.

For the sake of synthesis, we summarize these results in Table 4.4. Table 4.4 seems to

Polynomial order (N)	1	2	3	4	5
Ratio total CPU time (Modal/Nodal) **in 2D**	2.0	1.7	1.4	1.1	0.8
Ratio total CPU time (Modal/Nodal) **in 3D**	2.2	1.5	0.9	0.6	0.5

Table 4.4: Ratio of the global CPU time between Modal and Nodal simulations in 2D and 3D.

indicate that it is more interesting to use a nodal polynomial basis for polynomials of order lower than 5 in 2D and 3 in 3D. Also, in the following, we will use the Bernstein-Bézier polynomial basis only if $N \geq 5$ in 2D and $N \geq 3$ in 3D.

4.2.4 Conclusion and observations

In this section, we were able to perform the very first reconstruction tests with the FWI code developed in Total's environment. These tests have allowed us to validate the optimization routines that were developed during the thesis. Thanks to these tests, we were also able to test the different choices related to discretization, whatever the choice of the time scheme or the choice of the polynomial basis. The computational time is a very important factor to be taken into account during the FWI which is very expensive because several simulations are carried out successively. For reasons of calculation cost, we have decided in this chapter that we will use the RK2 time scheme. Concerning the choice of the polynomial basis, Bernstein-Bézier basis should be used only if the polynomial order is high enough ($N \geq 5$ in 2D and $N \geq 3$ in 3D).

In the following we will see how to perform a reconstruction with a WADG parameterization of the problem.

4.3 FWI with Weight Adjusted Discontinuous Galerkin Method

In the previous tests, we considered the model to be constant per element. We implemented the Weight Adjusted Discontinuous Galerkin method previously introduced by Chan et al. [2017], which allows taking into account variable physical parameters within the elements. This approach has been introduced page 93 and we have already shown that it allows to improve the representation of the model. We propose here to use it in the inversion algorithm.

As we have already said, this technique is very interesting because it avoids using fine mesh while keeping a sufficient amount of parameter to describe the physical model. It then gives the possibility to use large meshes which allows high order polynomials that contributes to reduce the number of degrees of freedom and thus to reduce computational costs.

This technology was introduced in the industrial code during the thesis and it was also necessary to implement an adapted visual interface. Before presenting reconstructions using this new parameterization, we will describe the tools that have been developed to visualize the physical parameters expressed at WADG quadrature points.

4.3.1 Visualization of WADG model parameters

Visualization is a feature that is essential for FWI. Hence, FWI is meant to provide a map of the subsoil as an output that will be analyzed next by geologists. In the WADG formulation, the physical parameters are expressed on a set of quadrature points that are located inside the elements. This is a very different way of defining the physical parameters and, for their visualization, we need tools working on point cloud. The objective of this subsection is to discuss the different strategies we have investigated to make possible the visualization of the physical parameters. The solution methodologies proposed in this subsection will be tested on the Marmousi velocity wavefield. The objective is to provide a visualization that is able to give a reliable rendering of the true Marmousi model that we display again in Figure 4.17.

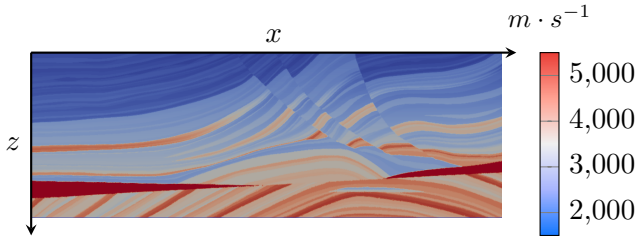


Figure 4.17: True Marmousi velocity model.

Vizualize as a constant model per element

A first attempt was to transform the WADG representation in such a way the model can be visualized with existing tools. At the very beginning, we dealt with piecewise constant model per elements, which are easily displayed on each element. Hence, to mimic this idea, one approach consisted in taking the average value of the physical parameters on each element to obtain a piecewise constant model that could be visualized straightforwardly with existing routines.

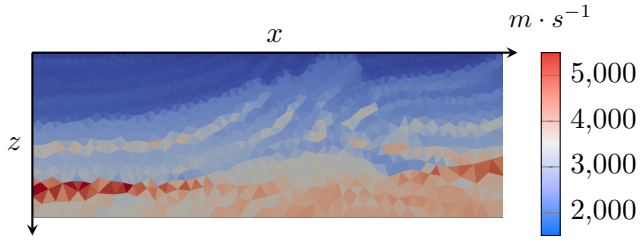


Figure 4.18: Ilustration of the Marmousi velocity model on 2000 P4-Q9 element mesh.

In the example displayed in Figure 4.18, we represent the visualization we obtained by averaging the velocity parameter on each element. In this case, we chose a mesh composed of 2000 elements and the space discretization is carried out with fourth order polynomials. With P4 WADG elements, it is recommended to choose a quadrature of order 9 ($N_q = 2N + 1$), which leads in 2D to have 19 parameters per cell [Chan et al., 2017]. In the figure caption, we have introduced the notation P4 Q9 element to indicate that the WADG approximation involves a polynomial approximation of order 4 and a quadrature formula of order 9. As expected, we can see that averaging the physical parameters provides model with a poor resolution. The idea of replacing the model with its representation based upon averaged values of the physical parameters is therefore non receivable, and we have to develop a more sophisticated method for visualizing the model. .

High-order visualization and projection into the wavefield interpolation nodes

The development of high-order finite element solvers raises questions concerning the vizualization of complex solutions at cell levels. Remacle et al. [2007] suggest subdividing the computational mesh into smaller elements on which the solution will be represented as a linear function. This is exactly what is done while outputting snapshots for wavefield visualization. We display an example in Figure 4.19, where a sub-triangulation is done in between each node supporting the nodal definition of the wavefield.

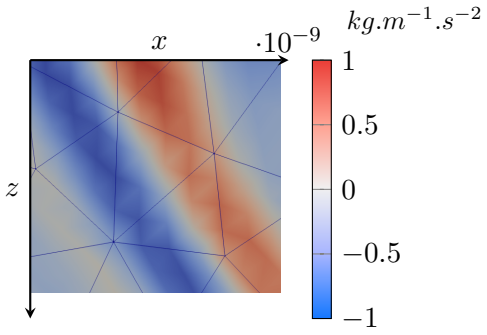
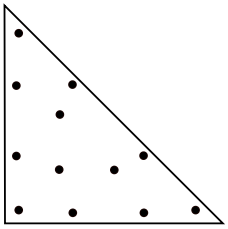


Figure 4.19: P4 pressure snapshot visualized using subdivided cells.

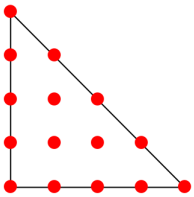
In this snapshot, we can see the wavefield that varies inside the element. Note that we can observe some in-print from the sub-triangulation managed by the output and Paraview software which is the software we are using for visualization [Ayachit, 2015].

Since such an output already exists for each wavefield, we investigate the interpolation of the parameter field (Figure 4.20a) into the node representing the wavefield (Figure 4.20b).

In the reference element $\hat{K}$, the wavefield $(p, \boldsymbol{v})$ is represented on DoF nodes located at points $\hat{\boldsymbol{x}}_j$ for $j = 1$ to DoF. In the following, we will use the generic letter γ to refer to one of



(a) Example of quadrature point on 2D reference element for $N_q = 6$.



(b) Example of interpolation node on a 2D reference element for $N = 4$.

Figure 4.20: Illustration of quadrature points defining the model parameter (a) and the interpolation node representing the wavefield in nodal polynomial basis (b).

the physical parameters (c, ρ, κ, etc.). We assume that γ is defined on n_q quadrature points located at points $\hat{\boldsymbol{x}}_q$ for $q = 1$ to n_q.

We aim to project the set of physical parameters, $\boldsymbol{\gamma}$, located at the quadrature points onto a new set, $\tilde{\boldsymbol{\gamma}}$, defined such that:

$$\sum_{j=1}^{DoF} \tilde{\boldsymbol{\gamma}}_j \hat{\varphi}_j(\hat{\boldsymbol{x}}_q) = \boldsymbol{\gamma}_q,$$

where $(\hat{\varphi})$ is the nodal polynomial basis on the reference element $\hat{K}$.

We have then the following over-determined matrix system of size $n_q \times DoF$:

$$A\hat{\boldsymbol{\gamma}} = \boldsymbol{\gamma}, \qquad [A]_{q,j} = \hat{\varphi}_j(\hat{\boldsymbol{x}}_q), \quad \text{for} \quad q = 1 \text{ to } n_q, \quad \text{and for} \quad j = 1 \text{ to } DoF. \tag{4.1}$$

We aim to find $\hat{\boldsymbol{\gamma}}$ that minimizes the L2 norm of the residual of (4.1) ($\inf_{\hat{\boldsymbol{\gamma}}} \parallel A\hat{\boldsymbol{\gamma}} - \boldsymbol{\gamma} \parallel_2$).

The solution of this least square problem can be obtained by solving the following normal equation:

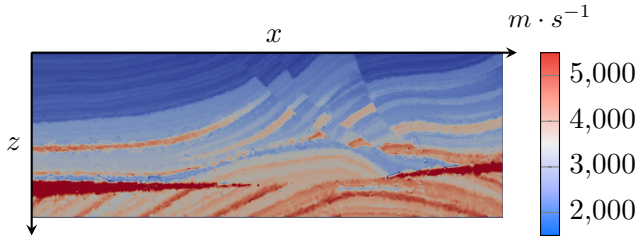
$$A^\top A\hat{\boldsymbol{\gamma}} = A^\top \boldsymbol{\gamma},$$

$$\hat{\boldsymbol{\gamma}} = (A^\top A)^{-1} A^\top \boldsymbol{\gamma}.$$

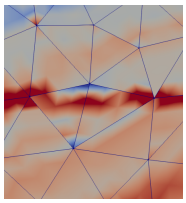
The operator $(A^\top A)A^\top$ is local and it can be computed only once and saved with a small memory burden. After implementing this extra operator, we are able to project the model expressed on quadrature points on the nodal points. The corresponding L2 projection allows us to exploit a routine already developed for wavefield visualization but for WADG model visualization. We have then adapted an existing routine for wavefield output into a new one that enables to visualize WADG parameter field on the wavefield interpolation point. In Figure 4.21a, we display the visualization of the velocity model Marmousi defined with 38000 parameters located on 19 quadrature points on a 2000 elements mesh. We observe a great improvement in comparison with what we get in Figure 4.18.

It is possible to plot the variation of the model inside each element. Unfortunately, the interpolation we performed in between quadrature points and nodal points generate Runge effects (see illustration page 65) and deteriorate the quality of the visualization where there are high model discontinuities. We display a zoom of such areas in Figure 4.21b showing some inaccuracies in the model representation.

To improve the visualization performed by Paraview software, we have addressed the option *pixel by pixel rendering* develop by Inria team GAMMA3 [Loseille and Feuillet]. Based on



(a) Marmousi velocity model visualized using nodal points.



(b) Zoom on area with high discontinuities.

Figure 4.21: Illustration of the Marmousi model on 2000 P4 Q9 elements mesh using projection on nodal points.

OpenGL 4.0 graphic pipeline, mainly used in animation or CAD (Computer Aided Design), ViZiR enables to rewrite Lagrange polynomial solution on each element as a Bézier curve. That way, the solution can be represented with an almost pixel-exact rendering. Such rendering is much more efficient than the one obtained with subdivided mesh. The subdivision can be expensive if it is too much refined or poorly represented if too coarse.

Unfortunately, the model parameters we aim to visualize are not defined as a Lagrange polynomial solution on each element. Hence, this interesting tool for high order visualization can not be used for physical parameters. However, it can be applied on the wavefield for a better rendering of the solution, but this is not the main purpose here.

Visualization mesh

We saw previously that there are two different ways to visualize high order solutions:

- subdivision of the mesh into linear interpolation on each sub-cells;
- almost pixel-exact rendering of Nodal polynomial solution on each element.

Since the second option is not compatible with the quadrature point representation in the WADG model, we aim to create a triangulation that links all quadrature points and visualize the model field with a linear interpolation on each element. We represent in Figure 4.22 a sample of the visualization mesh in 2D.

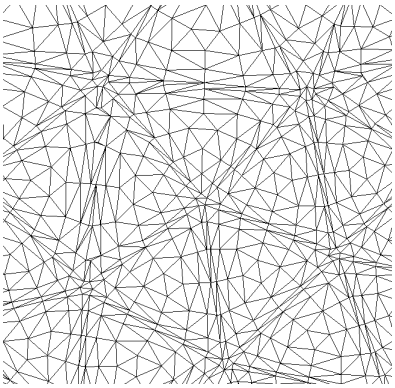
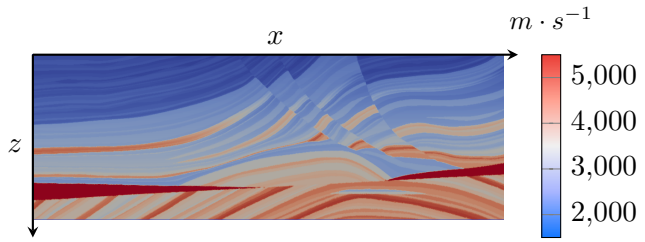


Figure 4.22: Zoom on 2D visualization mesh.

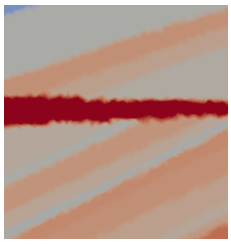
From a list of points and their coordinates, some software enables to generate a triangulation linking all the quadrature points. As far as this thesis is concerned, we used Mmg2D

[Mmg Software] for 2D triangulations and TetGen [Si, 2015] in 3D.

With these visualization meshes, we are able to render in Paraview a linearly interpolated solution on each element.



(a) Marmousi velocity model using visualization mesh.



(b) Zoom on area with high discontinuities.

Figure 4.23: Illustration of the Marmousi model on 2000 P4 Q9 elements using visualization mesh.

Note that we can clearly see an enhancement on the model rendering by comparing Figure 4.23b and Figure 4.21b. This strategy seems to be the more reliable one with respect to the targeted velocity model. This visualization ensures to represent exactly the model at quadrature points and does not generate Runge effects.

Once the visualization mesh is defined, it is straightforward to associate a value of the physical model to each vertex and have a linearly interpolate visualization on Paraview. More precisely, Mmg2D and TetGen software are able to manage VTK files (which is a compatible format with Paraview) describing the intermediate mesh which is useful to automatize post-processing script in order to visualize the physical model in 2D (see Marmousi wavespeed model at Figure 4.23a) or in 3D (see SEAM-foothills wavespeed model at Figure 4.24).

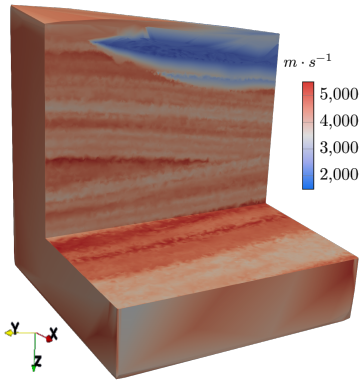


Figure 4.24: 3D visualization of truncated SEAM-foothills velocity model using 7K P4 Q9 WADG elements.

In this section, we discussed how we dealt with WADG parameters visualization. Since WADG is a new method implemented from scratch inside Total's code, there was no existing routine to visualize this new model architecture based on quadrature points.

For the visualization, we investigated three different strategies which are almost development free. We have either used existing tools (constant model output and high order wavefield output) or use external pieces of software as post-processing. These pieces of software create a visualization mesh on which the solution is seen as a linear function in between each vertex.

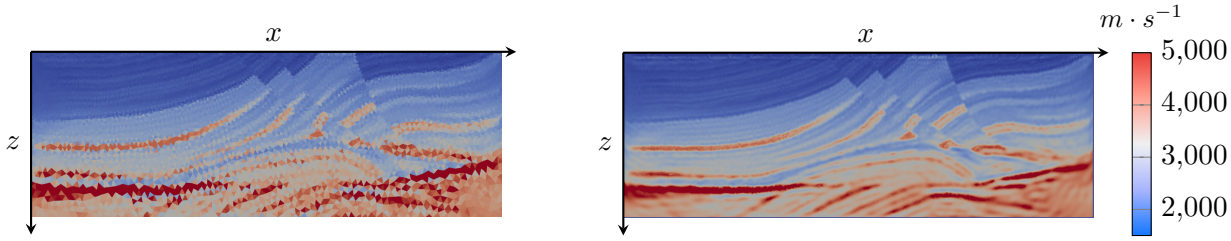
This last strategy gives the most appealing and reliable visualization of the model and is the one we definitely choose for representing physical parameters using WADG.

4.3.2 Reconstruction using WADG

Now that we have defined a tool to visualize a model defined with WADG method, we will proceed to the reconstruction of the Marmousi wavespeed model defined in the previous section (see Section 4.2 at page 150) using this new technology.

For this reconstruction, we propose to use the mesh of 10977 P3 elements using nodal polynomial basis. This mesh is the same as the one used for the tests in Subsection 4.2.1. We can then have a possible comparison between the reconstruction performed with WADG or with a constant model per element. Since we have elements of order 3, we have to choose a quadrature of order 7 giving then 15 coefficients defining the wavespeed model in each element. Without WADG we had 10977 parameters to reconstruct, while with WADG there are 164655 unknowns for the inverse problem.

We display in Figure 4.25 the final model reconstructed with 50 L-BFGS iterations.



(a) Reconstruction constant model per element. (b) Reconstruction with WADG parameterization.

Figure 4.25: Comparison of the reconstructions with and without WADG

This reconstruction allows to validate the adaptation of the FWI workflow to the WADG method. Definitely, the WADG parameterization allows a better appreciation of the model which varies within the elements. The result when using WADG is much smoother and is no longer subject to the discontinuities from constant parameterization per element.

Regarding the evolution of the cost function, we have a similar convergence between both methods. Indeed, as it can be seen in Figure 4.26, the two curves are very close to each other. As long as the parameterization is fine enough regarding the processed wavelength, it would seem that the developed FWI is not hindered by the large number of parameters due to the WADG parameterization.

However, the two methods do not have the same calculation cost. Indeed, we report in Table 4.5 the elapsed time for the 50 iterations performed during this test.

	Constant model	WADG
Number of parameters n_p	10977	164655
Elasped time (h)	2.9	4.9

Table 4.5: Comparison of the computational time of a Marmousi wavespeed model reconstruction using constant model per element or WADG technique (using 120 CPU cores).

The calculation cost with WADG method is, in this case, 1.7 times higher than the calculation cost with constant parameters by elements. This result is not surprising in view of the computational time study we made on WADG method on the direct problem (see page

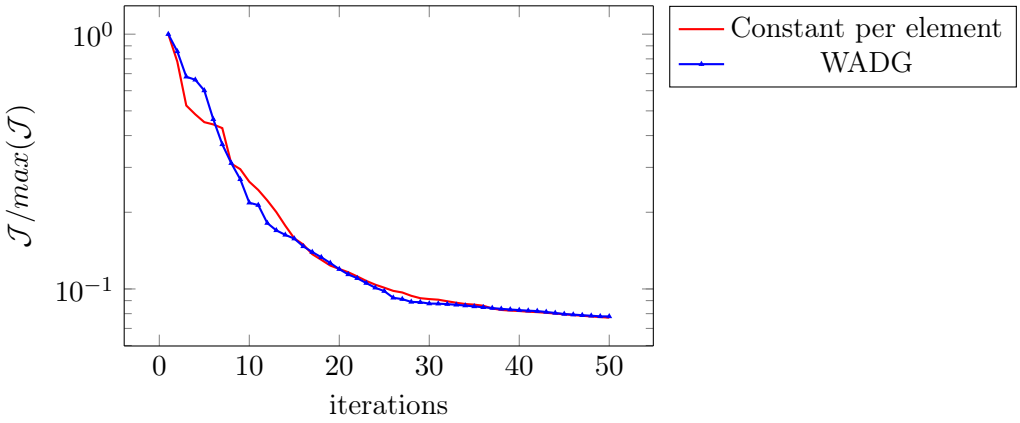


Figure 4.26: Comparison of the cost function evolution having a model constant per element or with WADG representation.

100). Despite the fact that WADG method can handle more complex models, it is nevertheless much more expensive. This method can be advantageous if the mesh size is very large compared to the wavelength of the emitted signal. This case requires high order polynomial approximation. In Chapter 5, we propose to define tools designing this type of discretization.

All the results shown here and in the previous sections are aimed at reconstructing the Marmousi velocity model from an initial model relatively close to the target model. This type of reconstruction favors the convergence of the optimization towards the global minimum of the problem. However, it is rare to have information on the model that we are trying to reconstruct. The objective, here, was to test the developed FWI with the different possibilities offered by the Total solver.

In the following section, we will work on reconstructing models for which we have no prior information, which is more complex.

4.4 Multiscale reconstructions

In the tests previously carried out on the Marmousi model, the initial model was a regularized version of the target velocity model. We could see that, hence, the optimization algorithm works very well. Indeed, the regularized model, by being deduced from the targeted model, contains enough information to prevent the cost function from stagnating on a local minimum, thus preventing the algorithm from converging. However, in practice, there is no prior information on the model to be reconstructed. Initialization is performed with a nearly blind model, with the help of geophysicists and geologists who interpret the acquisitions as best as they can. We present in Figure 4.27 a typical initial model which corresponds to a regularized version of a stratified model corresponding to continuous variations of velocity in the depth direction, the maximum and minimum values of the velocity corresponding to the maximum and minimum values of the velocities of the Marmousi model.

If we use this initial speed model to reconstruct the Marmousi velocity model, we get the reconstruction shown in Figure 4.28).

We can see that the optimization algorithm converges to a local minimum, victim of the non-convexity of the cost function that makes FWI difficult to implement. We observe a well-known phenomenon in wave equation inversion called cycle-skipping. This occurs when the observed and simulated data are out of phase by more than half a period of observed

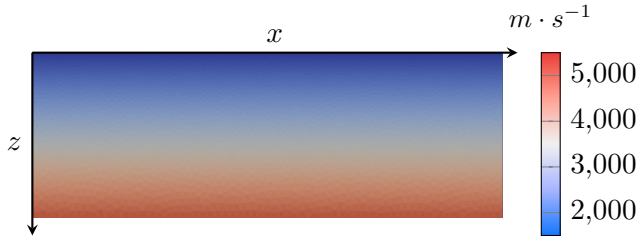


Figure 4.27: Initial velocity model (m.s^{-1}) without any prior information.

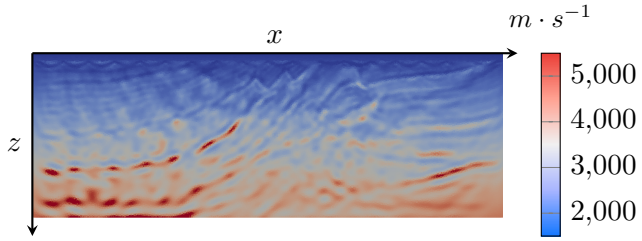
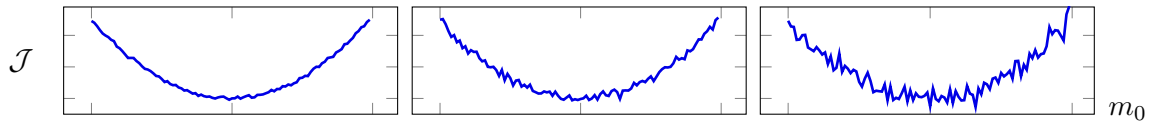


Figure 4.28: Velocity model (m.s^{-1}) reconstructed with an initial nearly blind velocity model.

input data. This phenomenon is all the more frequent as the initial model is far from the real model [Gauthier et al., 1986].

To avoid cycle-skipping, and thus ensure convergence towards the global minimum, it would be interesting to use a 'more convex' cost function. Recently, the Wasserstein norm based on the optimal transport theory has shown good performance and improved convergence to the global minimum [Yang et al., 2017]. This approach is quite recent and is not yet widespread in the geophysical community for reconstructing models with real data. Most often, multiscale methods are used. For example, Bunks et al. [1995], proposed to filter the data with a low-pass filter using the FIR Hamming-window function. In Figure 4.29, we illustrate heuristically the effect of low frequencies on an arbitrary 1D cost function. This method allows to reconstruct the model step by step from low to high frequencies and the presented results show that the optimization algorithm is more likely to converge towards the global minimum.

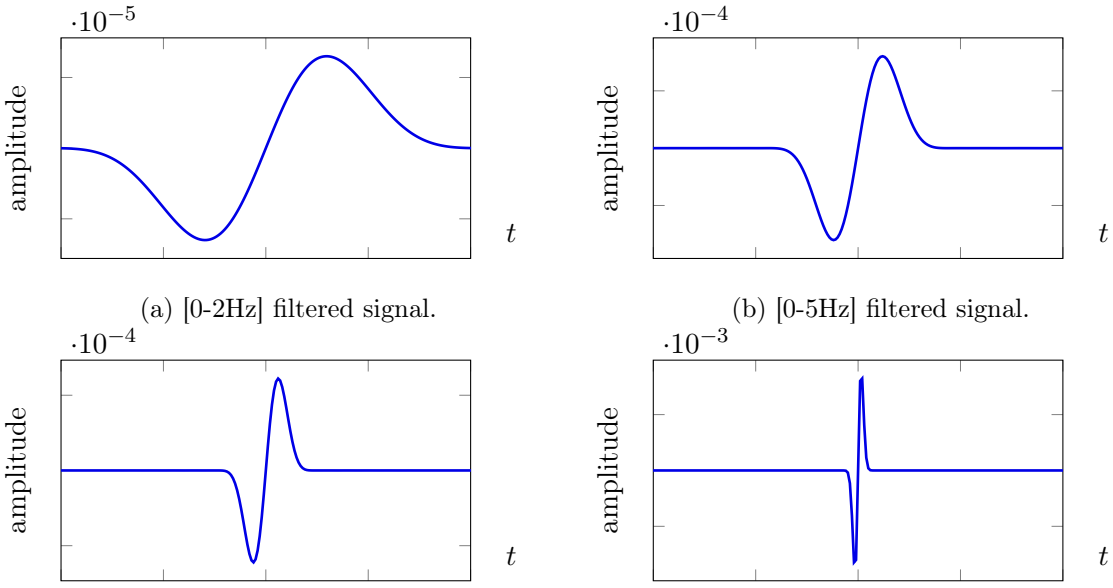


(a) Low frequency filter. (b) Medium frequency filter. (c) No filtering.

Figure 4.29: Heuristic representation of frequency filter influence on the misfit function.

To reconstruct the model by considering successive frequency values is an approach that is naturally implemented in frequency solvers. However, the choice of the frequencies to be considered is not obvious and can result in a significant increase in computing costs. To reduce these costs intelligently, Sirgue and Pratt [2004] proposed a strategy in the frequency domain to select the frequencies to be reconstructed. However, this strategy requires an adaptation in order to be usable in the time domain. In Boonyasiriwat et al. [2009], the leaky-low pass Hamming-window filter is replaced by more efficient filters that minimize the leaking of high-frequency components and the method for for choosing the frequency to filter in frequency-domain in [Sirgue and Pratt, 2004] is extended to time-domain.

Among the set of filters proposed by Boonyasiriwat et al. [2009], the Nutall-window function [Nuttall, 1981] caught our attention as it was available within Total's code. We will systematically use it to filter out the perturbations at the input of the direct problem as follows than the observed data. We display in Figure 4.30 the effect of the filter on the disturbance signal at a source simulated by a Ricker of order one with $f_{peak} = 15Hz$.



(a) [0-2Hz] filtered signal.

(b) [0-5Hz] filtered signal.

(c) [0-10Hz] filtered signal.

(d) No filtering.

Figure 4.30: First order Ricker signals filtered by using low-pass Nutall filter.

4.4.1 2D Multiscale reconstruction of Marmousi wavespeed model

Contrary to the previously presented Marmousi reconstructions, we seek to reconstruct the model without a priori information (Figure 4.32a). To do this, we will carry out a multiscale reconstruction, using the frequencies presented in Table 4.6.

Filter	[0,2Hz]	[0,5Hz]	[0,8Hz]	[0,12Hz]	[0,15Hz]	Total
Nb of iterations	20	20	20	20	20	100

Table 4.6: Multiscale strategy proposed to reconstruct Marmousi wavespeed model.

Since the frequency content of the signal emitted by the sources is mostly contained between 0 and 15Hz, we propose a sampling of frequency bands for multiscale reconstruction as indicated in the Table 4.6. This choice allows to progressively reconstruct a wide range of frequencies scaled between 2 and 15Hz.

The objective is to find the wavespeed model used for the generation of the input data that we displayed in Figure 4.31. For this reconstruction, we propose the following configuration:

- **Domain**: 9200m × 3000m;
- **Total time:** 5.2s (t_0 = 0s, T_f=5.2s);
- **Number of sources**: 20;

- **Source location**: Evenly located on the X-axis from 550m to 8150m with 400m in between at 10m depth;
- **Source type**: First order Ricker (f_{peak} = 10Hz, t_{peak}=0.2s);
- **Number of receivers**: 183;
- **Receiver location**: At a depth of 100m, positioned from 50m to 9150m on the X-axis with 50m between each other;
- **Parallelism**: 120 CPU; ($nb\_cluster = 20$, $nb\_subdomain = 5$)
- **Noise**: White Gaussian noise with a SNR of 10;
- **Polynomial basis**: Nodal;
- **Time scheme**: RK2;
- **Number of elements** N_e: 7218;
- **Polynomial order** N: 4;
- **WADG quadrature** N_q: 9;
- **Number of parameters** n_p: 137142;
- **Parameterization**: ($\frac{1}{\kappa}$,ρ) with known density ($\rho = 1000\text{kg.m}^{-3}$);
- **Optimizer**: Limited-BFGS.

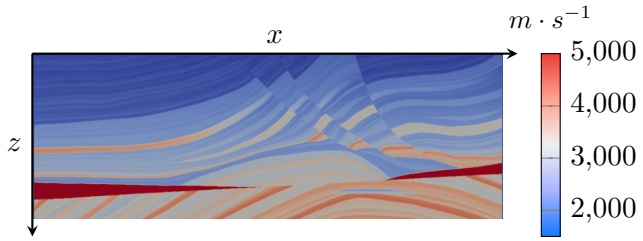
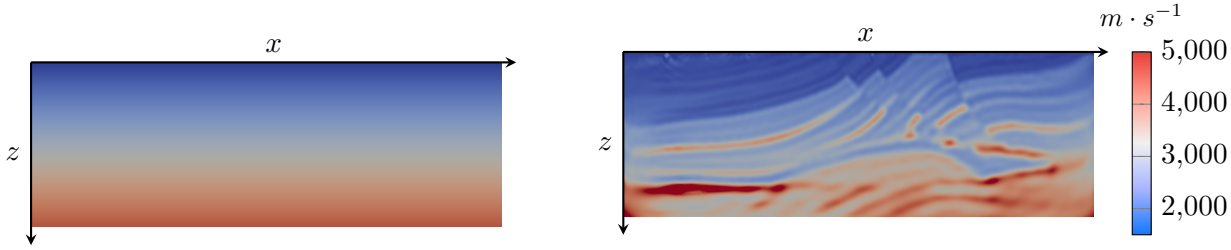


Figure 4.31: Target wavespeed model for Marmousi reconstruction.



(a) Initial wavespeed model.

(b) Reconstructed wavespeed model.

Figure 4.32: Initial and reconstructed wavespeed models.

After 100 iterations, which took a total computing elapsed time of 18 hours, we obtain the final result displayed in Figure 4.32b. The reconstruction is accurate even in depth where the

structures are also reconstructed. Despite the added noise, the implemented FWI allows to reconstruct the Marmousi wavespeed model accurately even without initial information (see Figure 4.32a).

4.4.2 2D Multiscale reconstruction of Overthrust wavespeed model

The Overthrust model presented in this sub-section is a 2D section extracted from the original Overthrust model which is a 3D model coming from SEG-EAGE 3D wavespeed model of Aminzadeh et al. [1994]. The objective is to reconstruct the 2D velocity model displayed in Figure 4.33, the latter comprises several layers in which wavespeed increases with depth.

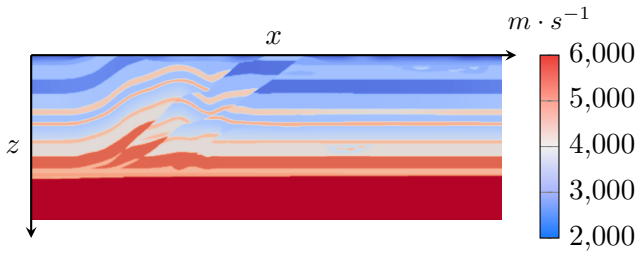


Figure 4.33: Target wavespeed model for 2D Overthrust reconstruction.

The initial model (see Figure 4.34a) being nearly blind, we propose to reconstruct the problem with a multiscale approach, according to the frequencies presented in Table 4.7.

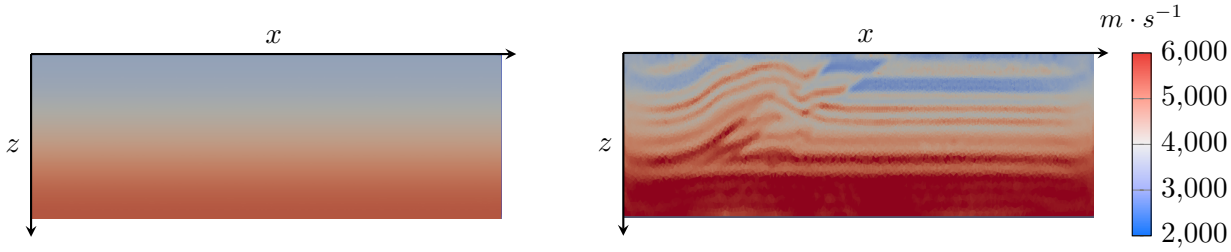
Filter	[0,2Hz]	[0,5Hz]	[0,8Hz]	[0,12Hz]	[0,15Hz]	Total
Nb of iterations	20	20	20	20	20	100

Table 4.7: Multiscale strategy proposed to reconstruct Overthrust 2D wavespeed model.

- **Domain**: 20km × 4.65km;
- **Total time:** 5.2s (t_0 = 0s, T_f=5.2s);
- **Number of sources**: 30;
- **Source location**: evenly located on the X-axis from 900m to 18300m with 600m in between at 50m depth;
- **Source type**: first order Ricker (f_{peak} = 10Hz, t_{peak}=0.2s);
- **Number of receivers**: 391;
- **Receiver location**: at a depth of 100m, positioned from 100m to 1975m on the X-axis with 50m between each other;
- **Parallelism**: 120 CPU ($nb\_cluster = 10$, $nb\_subdomain = 12$);
- **Polynomial basis**: nodal;
- **Time scheme**: RK2;
- **Number of elements** N_e: 20678;
- **Polynomial order** N: 2;

- **WADG quadrature** N_q: 1;
- **Number of parameters** n_p: 20678;
- **Parameterization**: $(\frac{1}{\kappa},\rho)$ with known density ($\rho = 1000\text{kg.m}^{-3}$);
- **Optimizer**: Limited-BFGS.

In Figure 4.34b, we present the result obtained after 100 iterations of FWI. We can see that the reconstruction is successful as we can see the different layers that constitute the Overthrust 2D model. For this reconstruction, we run the calculation on 120 CPU during 21 hours.



(a) Initial wavespeed model.

(b) Reconstructed wavespeed model.

Figure 4.34: Initial and reconstructed wavespeed models for 2D Overthrust reconstruction.

4.4.3 2D Multiscale reconstruction of Sigsbee wavespeed model

The Sigsbee model (see Figure 4.35) is a particularly challenging wavespeed model [Sig]. Indeed, in addition to having a larger size than the previously treated 2D models (i.e. 24.4km × 9.1km), this model includes a salt dome. This highly contrasted object corresponds to a propagation wavespeed of 4500m.s^{-1} while the background varies gently from 1500 to 3300m.s^{-1}.

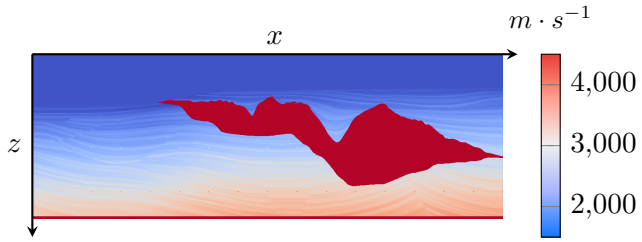


Figure 4.35: Target wavespeed model for Sigsbee reconstruction.

We propose to reconstruct this model from a model that has no prior information about the object (see Figure 4.36a). In this particular case, where we are looking for a high-contrasted object, the absence of initial information is very challenging because there is no structure in the initial model anticipating strong reflections coming from the searched object.

For the reconstruction, we used the frequencies presented in Table 4.8.

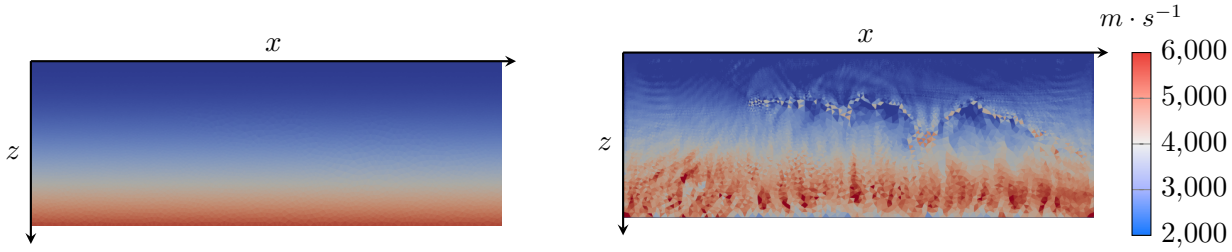
Filter	[0,2Hz]	[0,5Hz]	[0,7Hz]	[0,10Hz]	Total
Nb of iterations	30	20	15	10	75

Table 4.8: Multiscale strategy proposed to reconstruct Sigsbee wavespeed model.

The conditions of the reconstruction are given below:

- **Domain**: 24.4km × 9.1km;
- **Total time:** 7.2s (t_0 = 0s, T_f=7.2s);
- **Number of sources**: 72;
- **Source location**: evenly located on the X-axis from 1500m to 22800m with 300m in between at 10m depth;
- **Source type**: first order Ricker (f_{peak} = 10Hz, t_{peak}=0.2s);
- **Number of receivers**: 320;
- **Receiver location**: at a depth of 100m, positioned from 100m to 1975m on the X-axis with 50m between each other;
- **Parallelism**: 120 CPU ($nb\_cluster = 12$, $nb\_subdomain = 10$);
- **Polynomial basis**: Nodal;
- **Time scheme**: RK2;
- **Number of elements** N_e: 20454;
- **Polynomial order** N: 2;
- **WADG quadrature** N_q: 1;
- **Number of parameters** n_p: 20454;
- **Parameterization**: $(\frac{1}{\kappa},\rho)$ with known density ($\rho = 1000\text{kg.m}^{-3}$);
- **Optimizer**: Limited-BFGS.

The results have been obtained in 75 iterations, which were completed in 16 hours on 120 cores. Except for the surface of the object, which is rather well recovered, the reconstruction did not allow to find accurately the Sigsbee wavespeed model we were looking for. We can observe structures with high wavespeed values appearing in the lower part of the domain.



(a) Initial wavespeed model.

(b) Reconstructed wavespeed model.

Figure 4.36: Initial and reconstructed wavespeed models for Sigbsee reconstruction.

The obtained result is very similar to the one obtained in the thesis of Faucher [2017] using a frequency FWI code. Indeed, when reconstructing with only real frequencies, the final reconstruction also shows a vague approximation of the upper interface of the object and volutes with high velocities in the lower part of the domain.

However, to fill the initial information gap, it is possible in the frequency domain to use complex frequencies which will allow better determination of high-contrast salt domes. We refer to the work of Shin et al. [2010], Petrov and Newman [2014], Faucher [2017] for examples of applications where the benefits of using complex frequencies are illustrated. However, the data as such frequencies suffer from the noise and may be inaccessible in seismic (see [Faucher, 2017] page 164). Moreover, this technique cannot be exploited in time domain.

Once the upper interface part of the object was found, it was shown [Yang, 2019] that the L^2 cost function had difficulties to reconstruct the interior of such an obstacle. Then, it has been shown that Wasserstein norm seams more efficient that L^2 norm for thick high wavespeed value structures.

In this section, we have shown the results obtained with a multiscale approach. The latter allows to reconstruct the model in the absence of initial information and has been tested on three different 2D models. For Marmousi and Overthrust 2D models the reconstruction works properly but occurs difficulties on Sigsbee model where a salt dome without previous information complicates the reconstruction. However, it is in 3D where the performance of the FWI in time domain is expected. In the following section we propose to test the developed algorithm on a synthetic 3D case.

4.5 3D Reconstructions

In this section, we will present the 3D reconstructions performed in the thesis. First of all, we aim to test and validate the FWI workflow performed on a reduced 3D wavespeed model. We then choose to reconstruct the SEAM foothills within a simplified framework which will be described in details in what follows. Finally, we propose to reconstruct a model with a challenging dimension and representative of the scale of seismic exploration campaigns. This reconstruction will be done on the full 3D Overthrust model that was presented previously in 2D in section 4.4.2 at page 168.

4.5.1 3D reconstruction of truncated SEAM Foothills wavespeed model

In order to test and validate the code in 3D, we propose to reconstruct a truncated subcube of the SEAM foothills model. Since we want to show that the developed architecture also works in 3D, we will make a set of choices that will allow to reduce the amount of computation to perform this test. Because of the dramatically higher calculation costs in 3D, we have chosen to reconstruct the model from an initial model which is a smooth version of the targeted model. In this way, we can then avoid a multiscale reconstruction which would require several iterations of reconstruction for several frequency bands. In order to reduce the computational cost of the test, we have chosen to reconstruct the model by transmission and not by reflection as would be required for a classical representation of a seismic acquisition campaign. Reconstructing by transmission consists in carrying out the inverse problem by looking at the data recorded at receivers which are at the bottom of the domain of interest. Then, the simulation time can be reduced to the propagation time of a wave from the surface to the bottom of the domain. In the case of a reflective reconstruction, the receivers are on the surface, which means that the simulations reproduce waves traveling back and forth. Using the transmission approach, it is then possible to have a final simulation time about half as long as for a reflective reconstruction.

The 3D problem corresponds to the following configuration:

- **Domain**: (5000m, 5000m, 1500m) $\times$ (8000m, 8000m, 5000m) subcube;

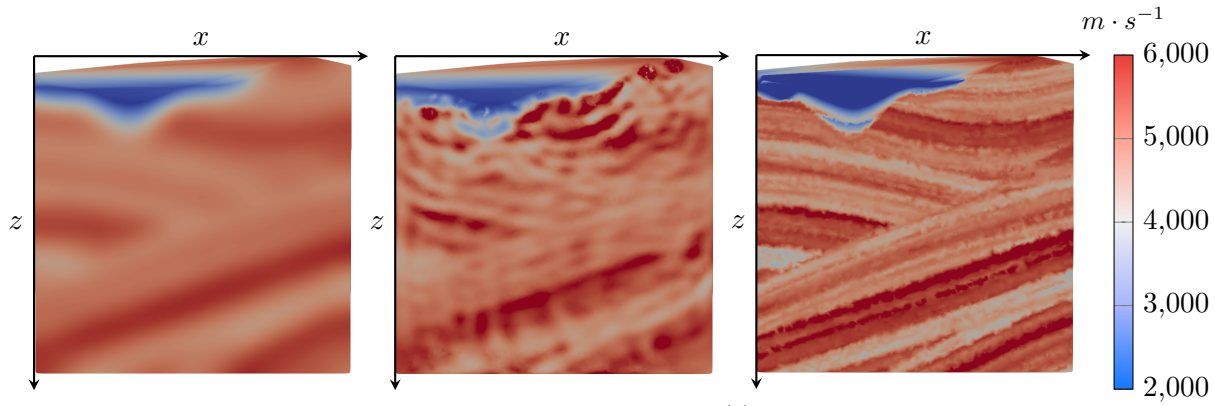
- **Total time:** 3.2s (t_0 = 0s, T_f=3.2s);
- **Number of sources**: 30;
- **Source location**: 3 lines of sources oriented in the Y direction. The sources are located at position $(x_{src}, y_{src}, z_{src})$, where x_{src} goes from 5250m to 7000m with a 200m step, y_{src} goes from 6250m to 6750m with a 250m step, and z_{src} is a constant depth of 50m, which represents the depth according to the topography;
- **Source type**: first order Ricker (f_{peak} = 8Hz, t_{peak}=0.2s);
- **Number of receivers**: 1225;
- **Receiver location**: positioned at points $(x_{rcv}, y_{rcv}, z_{rcv})$, where x_{rcv} and y_{rcv} go from 5200m to 7750m with a 75m step, and z_{rcv} = 4950m;
- **Parallelism**: 360 CPU ($nb\_cluster = 30$, $nb\_subdomain = 12$);
- **Polynomial basis**: Bernstein-Bézier;
- **Time scheme**: RK2;
- **Number of elements** N_e: 16544;
- **Polynomial order** N: 4;
- **WADG quadrature** N_q: 9;
- **Number of parameters** n_p: 943008;
- **Parameterization**: $(\frac{1}{\kappa},\rho)$ with density known ($\rho = 1000\text{kg.m}^{-3}$);
- **Optimizer**: Limited-BFGS.

For calculation cost reasons, we have chosen not to illuminate the whole domain. We chose to focus the sources along three lines aligned in the X direction in the center of the domain. In this way we expect an efficient reconstruction mostly in the X direction. We present the reconstruction results according to different planes depicted in Figure 4.37, where we compare the reconstructed wavespeed model with the initial and targeted model. We see that the reconstruction is much better on the plane y = 6530m, which allows to observe the slice plane located under the source lines.

We can see that on this plane, we are able to reconstruct some layers that did not exist in the initial model. The reconstruction gives better results near the receivers. Since here, the resolution is done by transmission, it is the lower part of the domain that is better imaged. We can however see some artifacts at the level of localized sources which are close to the topography.

Industrial context

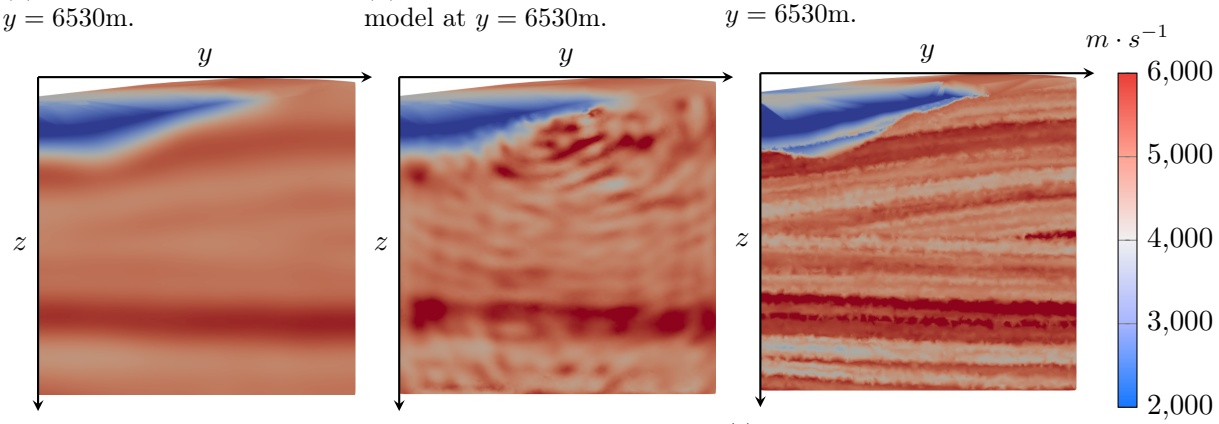
The topography turned out to be a challenge for the reconstruction of this model. Indeed, the location of the sources and receptors is done from the topography drawn by the mesh. Therefore, depending on the mesh used, the location of the sources and receivers changes, which strongly compromises the inverse problem if this location is approximative. In the absence of tools that allows for placing in an absolute way the position of the sources



(a) Initial wavespeed model at $y = 6530$m.

(b) Reconstructed wavespeed model at $y = 6530$m.

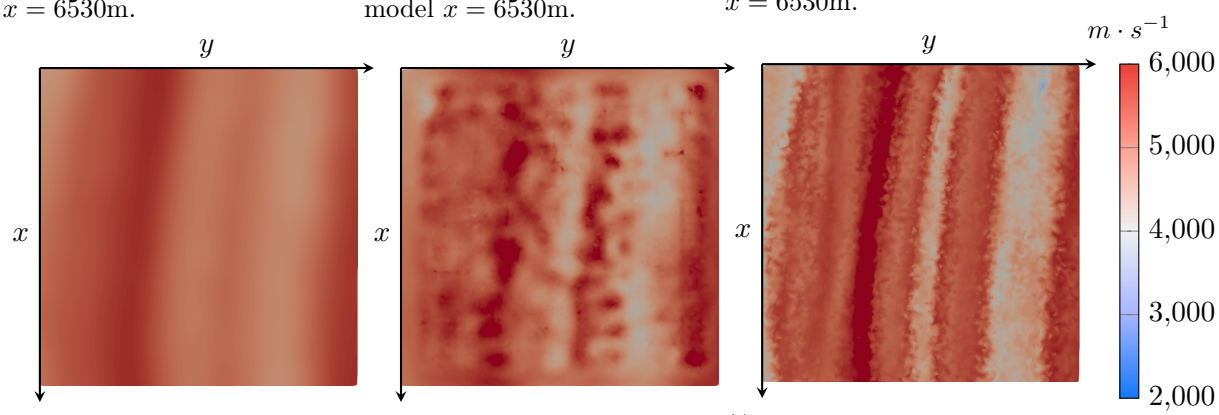
(c) Target wavespeed model at $y = 6530$m.



(d) Initial wavespeed model at $x = 6530$m.

(e) Reconstructed wavespeed model $x = 6530$m.

(f) Target wavespeed model at $x = 6530$m.



(g) Initial wavespeed model at $z = 4950$m.

(h) Reconstructed wavespeed model at $z = 4950$m.

(i) Target wavespeed model at $z = 4950$m.

Figure 4.37: Wavespeed models for truncated SEAM reconstruction (Initial model at the left, reconstructed model at the center and target model at the right).

and receivers according to the exact topography, we have bypassed this issue by using the same mesh for the inverse problem as the one used to generate the data.

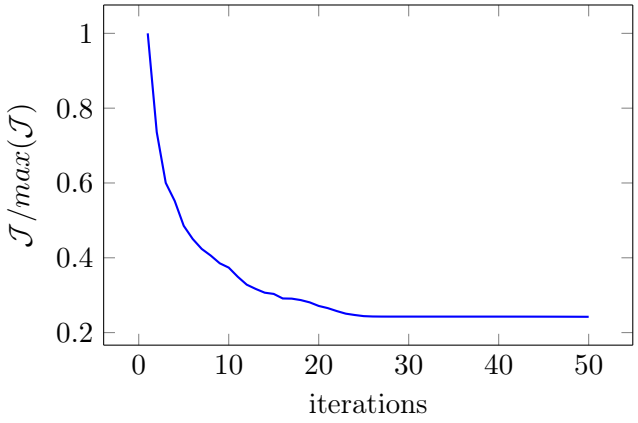


Figure 4.38: Cost function evolution for truncated SEAM wavespeed model reconstruction.

We display in Figure 4.38 the evolution of the cost function during the 50 optimization iterations. These iterations were performed in 68 hours of calculation on 360 cores. We can see the ability of the developed code to minimize the cost function on a 3D problem.

In this sub-section, we showed the ability of the developed code to reconstruct a 3D model. The problem proposed in 3D is deliberately created so that it can be reconstructed in a reasonable computational time.

However, the result of the reconstruction is mixed. Indeed, we manage to reconstruct some interfaces from which we had no prior information. But we were not able to reconstruct the whole model and there are multiple reasons to this. First, we have reduced the number of sources while deciding to privilege a direction of illumination. By undersampling the source fields, we surely contribute to hinder the behavior of the cost function.

We propose then, in the following subsection, to carry out a reconstruction on a model of size corresponding to the size of the seismic acquisition campaigns without any simplifying assumptions. In this way, we can perform an inversion in a realistic industrial configuration.

4.5.2 3D reconstruction of Overthrust wavespeed model

For this experiment, we wish to use the developed code to an industrial size problem. Indeed, we propose here to reconstruct the velocity model of the 3D synthetic Overthrust model. Contrary to the previous 3D experiment, the reconstruction will be performed by reflection, as usual, which will increase the total simulation time required. Moreover, the initial set of parameters chosen does not provide any information concerning the targeted model. Then, a multi-scale reconstruction is required to drive the convergence towards the global minimum. Due to its size (20km $\times$ 20km $\times$ 4.65km), the high number of sources required (3171) and the lack of knowledge concerning the targeted model, this reconstruction using DGm turns out to be a real HPC challenge.

The configuration of the reconstruction is as follows:

- **Domain**: (20km, 20km, 4.65km);
- **Total time:** 5s (t_0 = 0s, T_f=5s);
- **Number of sources**: 3171;

- **Source location**: 21 lines of sources oriented in the Y direction. The sources are located at position $(x_{src}, y_{src}, z_{src})$, where x_{src} goes from 25000m to 17500m with a 100m step, y_{src} goes from 2500m to 125000m with a 500m step, and z_{src} is a constant depth of 50m;
- **Source type**: first order Ricker (f_{peak} = 8Hz, t_{peak}=0.2s);
- **Number of receivers**: 3171;
- **Receiver location**: located at 100m depth, 50m below the sources;
- **Parallelism**: up to 2972 CPU;
- **Time scheme**: RK2;
- **Number of elements** N_e: up to 453612;
- **Parameterization**: ($\frac{1}{\kappa}$,ρ) with density known ($\rho = 1000\text{kg.m}^{-3}$);
- **Optimizer**: Limited-BFGS.

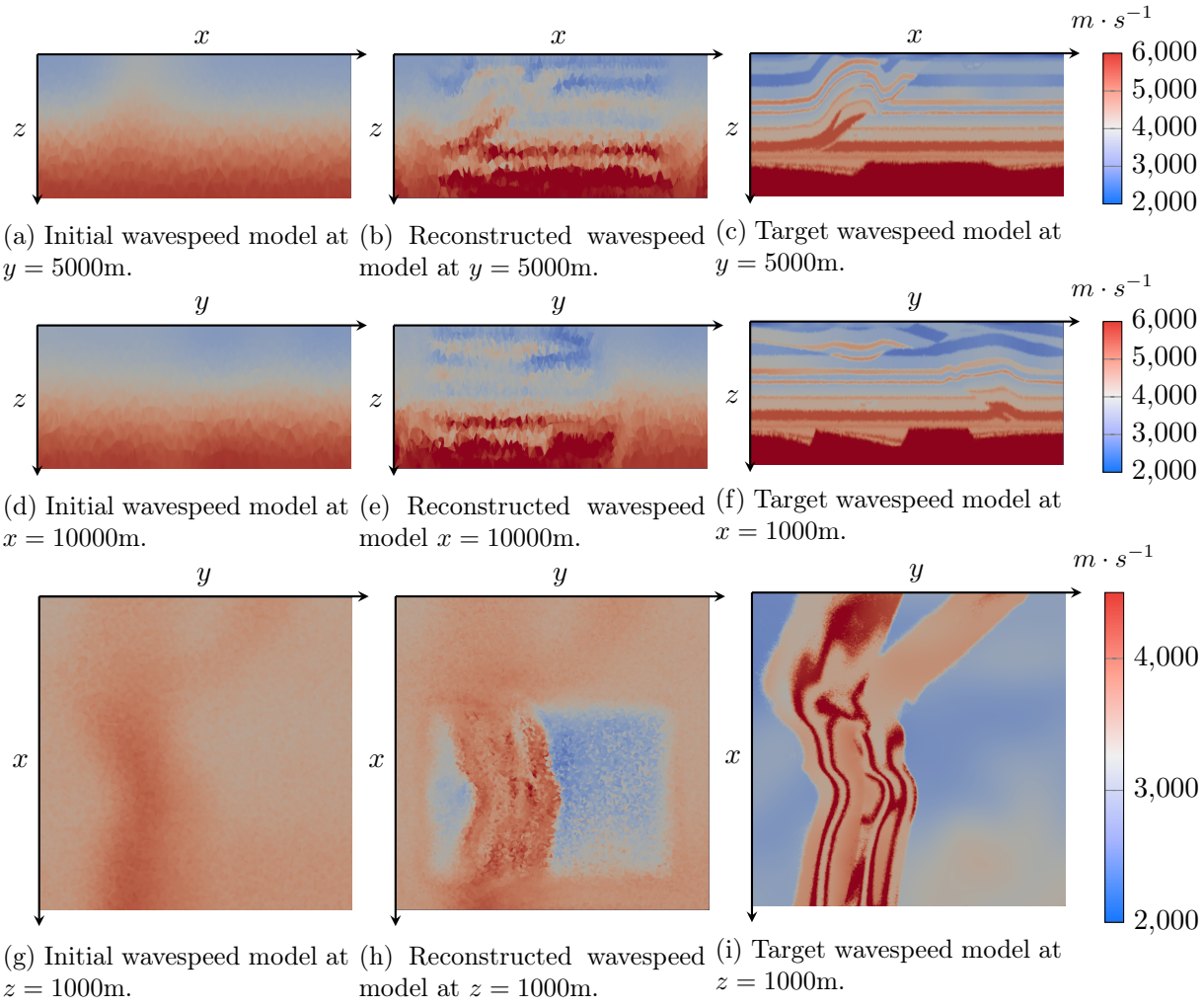
Contrary to the previous example, the lack of initial information on the velocity model requires to apply a multiscale method for the reconstruction. In view of the size of the problem and due to computational requirements, we then propose a discretization adapted to the current reconstructed frequencies. The choice of discretization is made with the help of criteria that we do not expose here. We refer the reader to the next chapter for more details. These criteria lead us to the choices specified in the Table 4.9 concerning the discretization, the frequency segmentation and the parallelism. We have chosen, here, to have a constant model per element for memory reasons.

Filters	[0-2.5Hz]	[0-5.0Hz]	[0-7.5Hz]
N_e	121755	121755	453612
% P1	8.5%		
% P2	90.5%		1%
% P3	1%	81.5%	99%
% P4		18.5%	
Number of unknowns	4670521	11091880	63191693
Polynomial Basis	Lagrange	Bernstein-Bézier	Bernstein-Bézier
CPU	1920	2976	2976
$n\_cluster$	40	62	31
n_domain	48	48	96
Number of iterations	17	20	8
Elapsed-time (days)	3.5	10.5	≈29

Table 4.9: Discretization and parallelism configuration for Overthrust 3D wavespeed model reconstruction.

The choices displayed in Table 4.9 allow us to significantly reduce the computational load for the first two frequency bands processed. Restarting the reconstruction in between two frequency bands enables to adapt the discretization (*hp*-adaptivity and polynomial basis choice) but also the parallelism configuration. Despite an adapted discretization, the computational cost, especially for the highest frequency band, is very important. The dedicated job was

stopped after 29 days for time allocation cluster reason. We display in the Figure 4.39 the final reconstructed velocity model that we compare with the initial and target ones.



(a) Initial wavespeed model at $y = 5000$m.
(b) Reconstructed wavespeed model at $y = 5000$m.
(c) Target wavespeed model at $y = 5000$m.
(d) Initial wavespeed model at $x = 10000$m.
(e) Reconstructed wavespeed model $x = 10000$m.
(f) Target wavespeed model at $x = 1000$m.
(g) Initial wavespeed model at $z = 1000$m.
(h) Reconstructed wavespeed model at $z = 1000$m.
(i) Target wavespeed model at $z = 1000$m.

Figure 4.39: Wavespeed models for 3D Overthrust reconstruction (Initial model at the left, reconstructed model at the center and target model at the right).

In the Figure 4.39 we see that the final result obtained allows to call structures which did not exist initially in the velocity model and which are present in the target velocity model. These details are however limited to the area illuminated by all the sources employed. This area is clearly visible in the visualization according to the XY plane. The plane privileged by the location of the sources being the XZ plane, it is this last one where layers are the most visible. Indeed, in spite of a resolution limited by the time and the capacities of calculation implemented, the visible results on the first line of image of the Figure 4.39 are convincing and very satisfactory in view of the technical challenge of this 3D reconstruction carried out using a Discontinuous Galerkin solver.

4.6 Conclusion and perspectives

In this chapter, we have described the industrial environment in which the developments of this thesis were established.

We have then tested and validated the optimization code developed within the Total environment. These tests have been performed on the reconstruction of the Marmousi wavespeed model from an initial smooth model which already has information on the target model. These tests have allowed to validate the optimization functions and in particular the Limited-BFGS algorithm that, as thought, happens to be the most efficient optimization method among the implemented ones. On this reconstruction, we were able to test the influence of the time scheme and the polynomial basis. Both did not influence the reconstruction except the associated computational load. In order to reduce this load, we decided to use the RK2 time-scheme and to privilege Bernstein-Bezier basis over Nodal [Chan and Warburton, 2016, Hesthaven and Warburton, 2007] when this basis is more efficient ($N \geq 5$ in 2D and $N \geq 3$ in 3D for the CPU architecture used).

We have also performed reconstructions where physical parameters are expressed using WADG technique [Chan et al., 2017]. As a recent technology in Total's code, it lacked an associated visualization tool. Hence, we have proposed one to observe a field of physical parameters defined by WADG method. Then, we have been able to observe and validate reconstructions performed by physical parameters expressed at quadrature points from the WADG method.

For more complex reconstructions, i.e., without prior information about the medium to be reconstructed, we have seen that it is possible to facilitate the convergence of optimization by multiscale methods as in [Bunks et al., 1995, Boonyasiriwat et al., 2009]. The latter consist in reconstructing the model by increasing frequency bands. This process allows reconstructing first the large structures of the model and thus to facilitate the convergence of the inverse problem towards the global minimum. It has been tested and validated on different 2D models: Marmousi, Overthrust and Sigsbee model. We had some difficulties to reconstruct the Sigsbee velocity model, which presents a salt dome generating reflected waves with high contrast in the input data. It seems that for this type of model the considered L^2 cost function, has difficulties to be minimized. The recent literature suggests to look for cost functions coming from the optimal transport theory, which seems to greatly improve the reconstructions of physical parameters representing models where there are large structures with high propagation speed [Yang, 2019]. Developing this type of cost functions is a very interesting perspective that should be addressed in the future.

We then tried to reconstruct 3D wavespeed models. First, we aim to retrieve the physical parameters coming from the SEAM Foothills problem truncated into a 3km $\times$ 3km $\times$ 3km cube. Despite mixed results due to simplified assumptions, this reconstruction enables to validate the capability of the program to deal with 3D problems and to make the objective function decrease. Finally, we aim to perform industrial scale inverse problem. For this reason, we manage to perform the reconstruction of the wavespeed model from the synthetic 3D Overthrust problem. This reconstruction turned out to be a real HPC challenge due to the size of domain and leads to satisfactory results. This technical inversion was only possible thanks to a discretization adapted to the different frequency band treated by using *hp*-adaptivity of the DG solver.

Such adaptation turns out to be necessary for challenging multiscale reconstructions. However, in the current state of the program described in this chapter, the multiscale re-

construction is based upon one single mesh which is inherited from the initial structure of the industrial code for solving the direct problem. This makes the multi-scale method very expensive since it requires meshing at low frequencies more than necessary, given that the mesh is built according to the highest frequency.

In order to best adapt the calculations necessary for the reconstruction of a given model in 2D or 3D, we propose in the next chapter to define a discretization in space adapted to the model evolving during the FWI, while taking into account the order of approximation in space desired by the user and the frequency of the used sources. As the physical model and the reconstruction frequency evolve during iterations, we will naturally turn to mesh adaptation methods.

Chapter 5

Mesh adaptation in FWI workflow

The mesh adaptation consists in adjusting the mesh size and the polynomial order approximation locally from *a posteriori* numerical errors providing insight into the accuracy of the scheme. The mesh is then locally refined and unrefined to reach an error threshold. Nowadays, the mesh adaptation is extensively used for solving boundary value problems, in particular in Computational Fluid Dynamics (CFD) field. Litterature on this subject is abundant, we can cite among others [Castro-Díaz et al., 1997, Schall et al., 2004] where the mesh is updated among the iterations of the simulation in order to control *a posteriori* errors of the numerical scheme [Gauci et al., 2019]. Two strategies are commonly adopted to adapt the mesh according to these numerical errors. A first option consists in determining an error estimation on a given mesh from the discrete solution. By introducing the adjoint state, it is possible to track areas to be refined in order to minimize *a posteriori* estimates of the error [Becker and Rannacher, 1996]. A second option consists in minimizing the continuous model error. For example, using a classical finite element method makes it possible to handle the approximation error by monitoring the interpolation error. The numerical solution converges towards the continuous solution as the mesh size decreases [Frey and Alauzet, 2005]. In this case, a metric based on the Hessian of the solution gives a size map that determines a guideline to adapt the mesh to reduce *a posteriori* errors on the numerical solution.

The objective of this chapter is to define a tool, inspired by classical mesh adaptation, that enables to redefine a mesh in keeping with the updated model parameters obtained through the FWI workflow. We remind that FWI is based upon an iterative minimization algorithm which is meant to converge towards the set of the physical parameters representing the subsoil. Minimization refers to a cost function which measures the difference (or error) between numerical data and observations (or real data).

With the current workflow, the mesh dedicated to the Forward and Adjoint simulations is driven by the initial model, which contains very little information in contrast with the intermediate reconstructions and obviously the targeted final model. In this context, the same mesh is used for any inversion loop. However, during the FWI process, there is no guarantee that the mesh keeps being adapted to the model that continues to evolve until convergence and this may contribute to increase the computational cost of the method. For instance, if the initial wavespeed model are taken far from the real values, for a fixed frequency, the initial mesh will have cells that do not have an appropriate size. It can be smaller than necessary resulting in an undesirable computational time or coarser which may hamper the accuracy of the forward and backward problem and then the quality of the reconstruction.

Adapted meshes are efficient to reduce the computational cost and the memory burden of the FWI. They have been recently investigated in [Thrastarson et al., 2020], for fitting the mesh with the geometry of the acquisition. More precisely, Thrastarson et al. [2020] propose

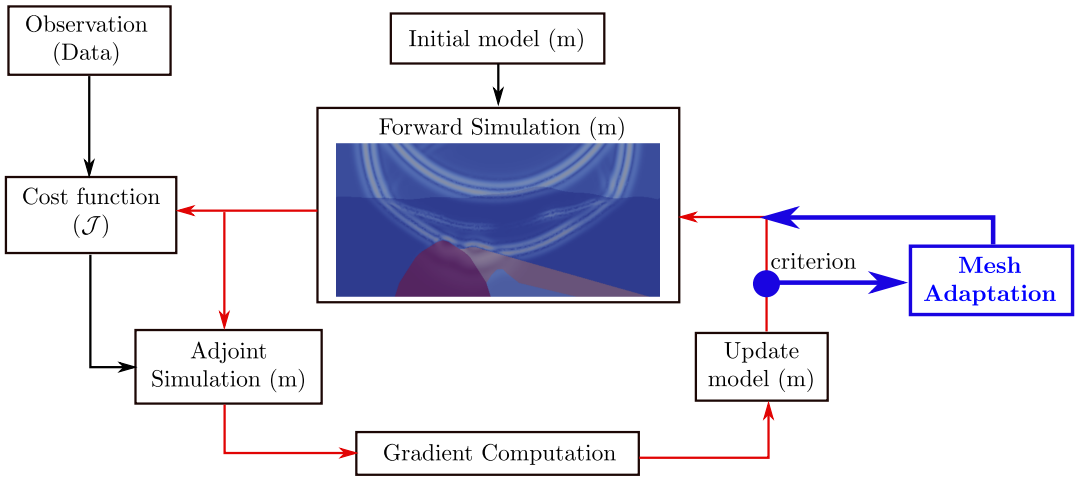
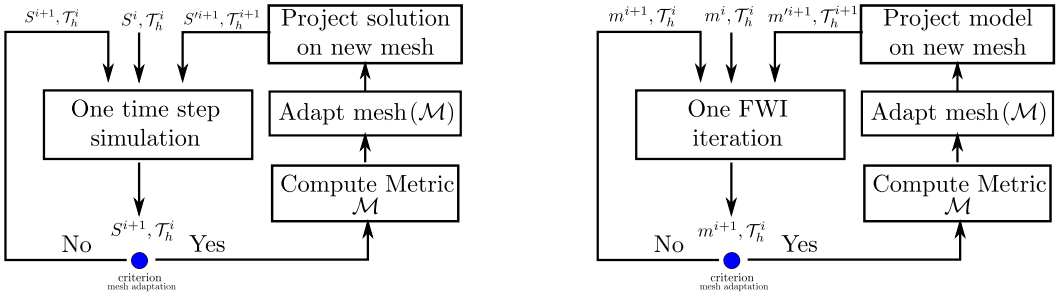


Figure 5.1: FWI workflow extended with mesh adaptation.

to use an anisotropic adaptive mesh refinement centered at each source. The underlying meshes designed for each source propagation fasten the computation. However, this strategy works if the model is smooth enough, which is rarely the case with the targeted model in seismic exploration. In this work, we investigate a mesh adaptation methodology regarding the physical parameters of the domain of interest.

The objective of this chapter is to provide a proof of concept of the mesh adaptation integrated in the FWI workflow. This implementation is carried out in 2D. We implement one extra step in this workflow to adapt the mesh in keeping with the model information. We represent in Figure 5.1 an extension of the current FWI workflow including mesh adaptation (in blue).

We intend to develop this additional module by mimicking the mesh adaptation scheme, usually included within the iterations of the simulation of the direct problem, by integrating it into the resolution of the inverse problem. Hence, here, we focus on the set of physical parameters m which are used for estimating the numerical error. We display in Figure 5.2 the classical mesh adaptation workflow with the one we aim to implement for the inversion problem.



(a) Classical mesh adaptation.

(b) Mesh adaptation in FWI context.

Figure 5.2: Mesh adaptation flowchart for direct problem simulation and FWI context.

The main difference between those two workflows is the metric that has to be defined as a function of the geophysical model, but also other parameters such as the frequency of the source or the polynomial approximation order N. The metric, also called the size map, is the guideline to update the mesh ${\mathcal{T}_h}^i$ at iteration i to a mesh ${\mathcal{T}_h}^{i+1}$ for next iteration.

Keeping the same mesh during all the FWI iterations requires working from the beginning with a sufficiently fine mesh to take into account a wide range of frequencies. This obviously implies unwelcomed computational costs, in particular when reconstructing low frequency information. Adapting the mesh frequently during the FWI process will allow us to fully exploit the *hp*-adaptivity of Discontinuous Galerkin method. This will give us the opportunity to optimize the computation of the Forward and Adjoint states, which are the keystones of the inversion process. Moreover, in a multiscale reconstruction framework, the mesh adaptation will reduce drastically the computational cost and the memory burden at low frequency reconstructions.

We will therefore implement a mesh adaptation tool that will be based on the physical parameters, contrary to what is classically done in mesh adaptation, which rather takes into account the unknown of the boundary problem under consideration. We will inspire ourselves from what is done in mesh adaptation by using and/or adapting tools and software dedicated to it. A first step consists in defining a metric $\mathcal{M}$ determining the size of an element in the triangulation of the computational domain Ω in order to meet an error criterion. There are programs to determine this size map as the Mshmet program of the Adapt_Tools software. [AdaptTools - Mshmet] that we have chosen to consider. We will have to adapt it to make it compatible with geophysical models. Motivation to use Mshmet, for the calculation of the size map, comes from the fact that it is an open source project which already has all the structures to integrate a metric compatible with the parameters of our geophysical models. With such already existing structures, we limit developments, which we believe is appropriate, our objective being to provide a proof of concept.

Industrial context

Since we are developing in an industrial context, it is more suitable to use external tools to make a proof of concept. A new method must be proven before it can be implemented. In addition a rigid framework would hamper the time of setting up all the mesh adaptation workflow.

Once the size map is built, we can use it to adapt the mesh and for this, we chose to use Mmg [Mmg Software], which is an open source remeshing software.

The remainder of this section is organized as follows. First of all, we will introduce some notions specific to the adaptation of meshes, and we will briefly describe the process. We will then construct the size map according to the velocity model and the DG discretization used. The resulting mesh adaptation will be validated by comparing several simulations applied to the Marmousi velocity model. The size map will allow implementing local mesh refinement which may be isotropic or anisotropic. Both types of refinement will be described and compared numerically. Finally, mesh adaptation will be integrated in the resolution of the inverse problem. Numerical results will also be presented.

5.1 Definitions

Unstructured meshes have demonstrated their effectiveness in improving the accuracy of numerical solutions, in particular to better capture physical phenomena. To adapt the mesh, it is important to estimate the error caused by the numerical scheme, the goal being to have a good accuracy with a reasonable calculation time. The discontinuous Galerkin method facilitates the h adaptation which exploits the relation between the local error estimated on an element and the size h of this element. This relation, given for each element, is used to

construct a metric or size map that guides the evolution of the ${\mathcal{T}_h}^i$ mesh to the ${\mathcal{T}_h}^{i+1}$ mesh in order to evenly distribute the error over the adapted mesh.

In practice, the mesh adaptation follows three steps:

- estimate the numerical error;
- define a metric field;
- adapt the mesh (h-adaptivity).

The two first items are most of the time based on the Hessian of the solution field of the treated PDE [Kunert, 2002]. The so obtained metric field $\mathcal{M}$ describes on each point of the computational domain Ω a Symmetrical Positive-Definite (SPD) matrix that defines a distance in the associated geometrical metric space. Mesh adaptation relies heavily on the notions of metrics and distance that we introduce in the following sub-section. In this context, the way the metric will be constructed is determined by a geometrical interpretation of the metric space.

It is then important to define the notions of metric and distance and to depict briefly how mesh adaptation, according to a metric field, is working. This is the purpose of the next subsection that will introduce all the notions involved in the remainder of this chapter.

5.1.1 Metrics and distance

Let us consider a continuous metric field $\mathcal{M}$ on the domain Ω. For any point $P \in \Omega$, we denote by $\mathcal{M}(P)$ the SPD matrix of size $dim \times dim$, where dim is the space dimension (2 in 2D, 3 in 3D).

We define the scalar product in $\mathbb{R}^{dim}$ with respect to the metrics $\mathcal{M}(P)$:

$$\langle u, v \rangle_{\mathcal{M}(P)} = \langle u, \mathcal{M}(P)v \rangle = u^\top \mathcal{M}(P) v \in \mathbb{R}\,.$$

The corresponding distance is given by:

$$\| u \|^2_{\mathcal{M}(P)} = \langle u, u \rangle_{\mathcal{M}(P)} = \langle u, \mathcal{M}(P)u \rangle = u^\top \mathcal{M}(P) u \in \mathbb{R}\,.$$

In the case where $\mathcal{M}(P) = I_{dim}$, we retrieve the classical euclidean distance.

Since the metrics $\mathcal{M}(P)$ is a real SPD matrix, it is similar to a positive diagonal matrix:

$$\mathcal{M}(P) = S \Lambda S^\top\,,$$

where Λ is the diagonal matrix of the eigenvalues of $\mathcal{M}(P)$ and S is the matrix composed on each column by the eigenvectors of $\mathcal{M}(P)$. Then :

$$\Lambda = \begin{pmatrix} \lambda_1 & & & & \\ & \ddots & & & \\ & & \lambda_i & & \\ & & & \ddots & \\ & & & & \lambda_{dim} \end{pmatrix} \qquad \text{and} \qquad S = (v_1 \,|\, \dots \,|\, v_i \,|\, \dots \,|\, v_{dim}),$$

where v_i is the eigenvector associated to the eigenvalue λ_i ($\mathcal{M} v_i = \lambda_i v_i$).

With these notations, it is possible to represent geometrically the unit ball defined by the metrics $\mathcal{M}(P)$ at a point P of Ω. For a point $M \in \Omega$, M belongs to the unit ball centered on P with respect to the metrics $\mathcal{M}(P)$ if and only if:

$$\| \overrightarrow{PM} \|_{\mathcal{M}(P)} = \sqrt{\overrightarrow{PM}^\top \mathcal{M}(P)\, \overrightarrow{PM}} = 1\,.$$

The set of points M satisfying the above relation is an ellipse in 2D, or ellipsoid in 3D, centered on P. The orientation and size of the unit ball are defined by the eigenvectors and their associated eigenvalues of the metrics $\mathcal{M}$. In 2D, for instance, the ellipse axes are oriented following the two eigenvectors $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ and the semi-axes are equal to $\frac{1}{\sqrt{\lambda_i}}$ with λ_i the eigenvalue associated to the eigenvector $\overrightarrow{v_i}$, as illustrated in Figure 5.3.

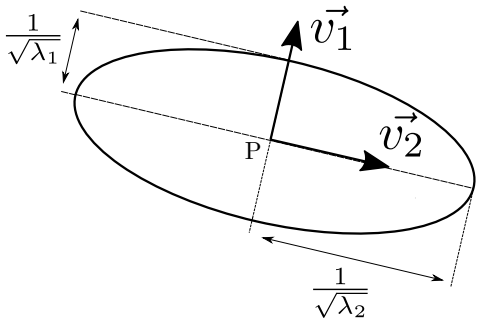


Figure 5.3: Unit ball centered on P with respect to the distance norm $\| \, . \, \|_{\mathcal{M}(P)}$.

In the case where we have one or several predominant eigenvalues, the metrics is naturally anisotropic. For example, in Figure 5.3 we have $\lambda_1 >> \lambda_2$ and we can clearly see that the unit ball is represented by an ellipse stretched in the direction given by the eigenvector $\overrightarrow{v_2}$. On the contrary, when all the eigenvalues are equal, the unit ball is a circle in 2D or a sphere in 3D (Figure 5.4). In this case, the metrics corresponds to an isotropic adaptation.

Remark. Considering isotropic metric fields, we generally assimilate the metrics $\mathcal{M}$ to the real factor corresponding to the radius of its unit ball:

$$\mathcal{M}(P) = \lambda I_{dim} \, ,$$
$$h(P) = \frac{1}{\sqrt{\lambda}} \, .$$

In this situation, we employ the term size map that corresponds to a scalar field instead of a metric field.

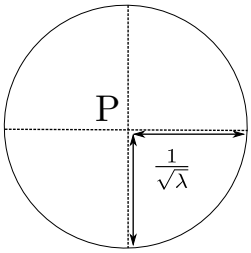


Figure 5.4: Unit ball centered on P with respect to the distance norm $\| \, . \, \|_{\mathcal{M}(P)}$ in isotropic case.

We have then defined a geometrical interpretation of the metrics $\mathcal{M}$ at point P. In the next subsection, we will show how the metric field is used in mesh adaptation.

5.1.2 Application of the metric field to mesh adaptation

It has been shown by Frey and Alauzet [2005] that it is possible to construct, on each element K of $\mathcal{T}_h$, a metrics $\bar{\mathcal{M}}(K)$ that quantifies the numerical error ε_K on K such that:

$$\varepsilon_K = \underset{\overrightarrow{e} \in E_K}{max} \langle \overrightarrow{e}, \bar{\mathcal{M}}(K) \overrightarrow{e} \rangle. \tag{5.1}$$

Here, E_K denotes the set of vertices of the element K. Relation (5.1) links the interpolation of the solution on K to the square of the longest edge of the element. In other words, it is possible to monitor the interpolation error by controlling the size of the mesh.

In the framework of mesh adaptation, the goal is to evenly distribute, on all the elements of the triangulation of Ω, the interpolation error to control the accuracy of the numerical simulation. Let us consider ε, the maximum error threshold authorized on each element of the adapted mesh. In mesh adaptation, we adjust the edge length so that the error level on each edge is controlled by ε. Then, we construct an optimal mesh $\mathcal{T}_h{}'$ from $\mathcal{T}_h$ by imposing to all edges:

$$\varepsilon = \langle \overrightarrow{e}, \bar{\mathcal{M}}(K) \overrightarrow{e} \rangle \,, \qquad \forall K \in \mathcal{T}_h{}' \,, \qquad \forall \overrightarrow{e} \in E_K.$$

Let us consider $\mathcal{M}(K) = \frac{1}{\varepsilon} \bar{\mathcal{M}}(K)$, we have:

$$\langle \overrightarrow{e}, \mathcal{M}(K) \overrightarrow{e} \rangle = 1 \,, \qquad \forall K \in \mathcal{T}_h{}' \,, \qquad \forall \overrightarrow{e} \in E_K.$$

This relation is the key of the mesh adaptation. We are just defining a criterion on each edge $\overrightarrow{e}$ of $\mathcal{T}_h$ in order to satisfy a tolerated error ε. The objective of the mesh adaptation is to update ${\mathcal{T}_h}^i$ into ${\mathcal{T}_h}^{i+1}$ where all the new edges have a unitary length with respect to the metrics.

By considering a continuous metric field, it is then possible to express the distance between two points with respect to the metrics as follows:

$$\| \overrightarrow{PM} \|_{\mathcal{M}} = \int_0^1 \sqrt{\overrightarrow{PM}^\top \mathcal{M}((1-t)P + tM) \overrightarrow{PM}} \, dt \,, \tag{5.2}$$

where $(1-t)P + tM$ for $t \in [0,1]$ describes all the points between P and M.

Such a distance is mainly computed using a quadrature formula during the mesh adaptation process. In practice, the metric field is never known continuously and is principally given at the vertices of the mesh. We refer to [Frey and Alauzet, 2005] for more explanation concerning how to interpolate the metrics in between two vertices.

The main steps of the mesh adaptation algorithm are then:

1. Scan all the edges $\overrightarrow{PM}$ and compute $\| \overrightarrow{PM} \|_{\mathcal{M}}$ using (5.2):
   - Split the current edge if too long;
   - Collapse the edge if too short.
2. Check quality of the elements:
   - Swap edges if "too bad elements" (see Figure 5.5);
   - Move points.
3. Go to 1. until convergence of the algorithm defined by the expected $\| \overrightarrow{PM} \|_{\mathcal{M}}$ values.

The above process is lightened for simplicity. Mesh adaptation is a complex subject that is not intended to be described in this thesis. It is worth noting that there are different ways to couple the split/collapse/swap/move steps, which are the main procedures of this algorithm.

We aim to construct an adapted mesh ${\mathcal{T}_h}^{i+1}$ comprised of edges that satisfy $\| \overrightarrow{PM} \|_{\mathcal{M}} = 1$. In practice, the strict equality is impossible to obtain and this is the reason why a tolerance

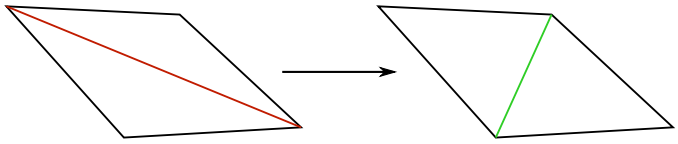


Figure 5.5: Illustration of the swap edge operator in 2D.

is allowed. For instance, the external tool we chose to perform the h-adaptation targets edge lengths such as:

$$\frac{\sqrt{2}}{2} \leq \| \overrightarrow{PM} \|_{\mathcal{M}} \leq \sqrt{2}\,. \tag{5.3}$$

Depending on the eigenvalues of the metrics, the element will be more or less stretched. We illustrate in Figure 5.6a and Figure 5.6b what we will call respectively isotropic and anisotropic cell in the rest of the chapter.

(a) Isotropic cell. (b) Anisotropic cell.

Figure 5.6: Illustration of isotropic and anisotropic cells.

We have then described the main keys of mesh adaptation which has been introduced for an abstract metric field. In the following part of this chapter, the objective is to define a metric field according to the geophysical model. We will first determine a heuristic relation between the size of the element and the parameters of the simulation. Since we are considering the physical model that is defined point-wise, we will first construct, for the sake of simplicity, a heuristic size map which will be naturally isotropic. This isotropic size map will be the first step before further investigations.

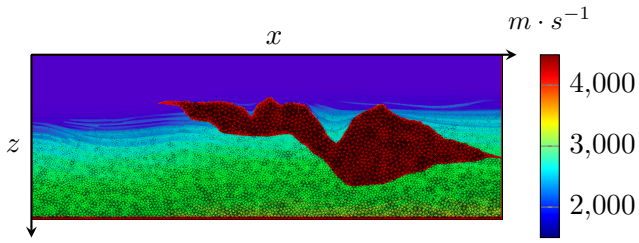
5.2 Size Map Computation

In this section, the objective is to adapt a mesh by taking into account the model parameters. As we are considering a DG acoustic wave solver, we are particularly interested in the wavespeed model c that is defined, in the DG solver we are using, as a point cloud in the computational domain Ω, as shown in Figure 5.7a. The idea of seeing the wavespeed model as a point cloud is inherited from the Weight Adjusted Discontinuous Galerkin (WADG) method (see Section 2.4). For each point P of the point cloud with coordinates (x_1, x_2) (in 2D), we match a value $c(P)$ of the wavespeed model. To facilitate the presentation of our approach, we will first assume that the model is known at all the vertices of the initial mesh that we aim to adapt. The set of all those vertices is actually defining a point cloud. We will assimilate for the rest of this section the physical model to its point cloud.

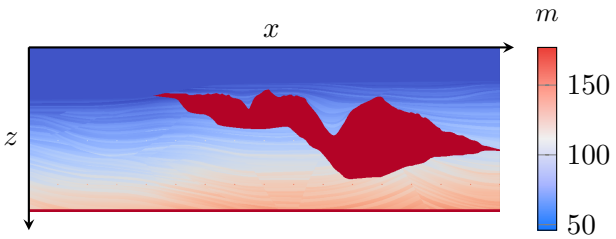
For DG PDEs solver, *a posteriori* error estimates [Grote and Schötzau, 2009] and mesh adaptation [Dolejší et al., 2018] have already been studied on the solution field in order to optimize the computational burden all along the time iterations of the simulation. Therefore, using classical mesh adaptation according to the wavefield would modify the mesh several

times in the forward and backward propagation. This strategy would make difficult the gradient computation and the model update. Here we require to keep the same mesh during one iteration of the FWI in order to have manageable operators on the physical parameters. Classically, the mesh is adapted in order to control the error *a posteriori* which is made when calculating the solution of a PDE. In the case of a DG solver, the calculation of the *a posteriori* error has been considered in [Grote and Schötzau, 2009] for the acoustic wave equation and a mesh adaptation method has been implemented in [Dolejší et al., 2018]. The objective of these two works was to optimize the computational burden throughout the time iterations. Adapting the mesh several times in the course of the time iterations performed in one optimization step, would complicate the gradient computation since the model parameterization would change with the mesh. Instead, we will implement a method that allows to use the same mesh during a FWI iteration.

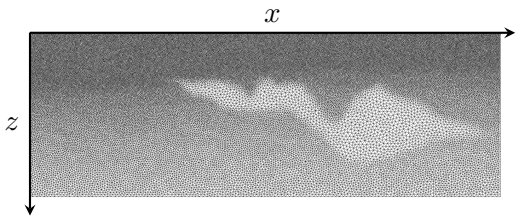
Instead of computing the mesh with respect to the PDEs solutions, we desire to adapt the mesh and to control the discretization error by taking into account the wavespeed model. Since the model is evolving in the FWI process, we want to perform mesh adaptation in the course of the FWI in order to solve forward and backward simulation accurately and efficiently. We need then to define a size map (Figure 5.7b) that will be the guideline to adapt the mesh for the next FWI iterations (Figure 5.7c).



(a) Wavespeed model ($m.s^{-1}$).



(b) Size map according to the wavespeed model.



(c) Mesh adapted with respect to the size map.

Figure 5.7: Meshing steps illustrated with Sigsbee wavespeed model. (a) set of points defining the wavespeed of the Sigsbee model. (b) Size map obtained for $p = 4$, $freq = 10Hz$, $n_{ppw} = 10$. (c) Resulting mesh using mesh adaptation in keeping with the size map.

The framework presented in Figure 5.7 requires the definition of the size map. As there is no existing criterion to define the more appropriate size of cell on each point of Ω, we aim, in the next subsection, to determine with a heuristic and experimental way an expression that provides an appropriate size map for a given wavespeed model for DG acoustic solver.

5.2.1 Heuristic expression of the isotropic size map

The objective of this subsection is to determine an expression of the size map as a function of the wavespeed model. As a first step, the size map defined here will be intrinsically isotropic. We will denote by $h(P)$ the scalar size prescribed at point P giving the suitable size h of the element as a function of the wavespeed model.

The notion of number of points per wavelength, n_{ppw}, is directly inherited from convergence studies of Finite Difference solvers [Alford et al., 1974]. This value determines the size of the mesh and is natural for regular grids. To obtain a criterion on the size of a 2D unstructured mesh grid, we introduce the notion of points per wavelength on a 2D triangulation $\mathcal{T}_h$. The objective is to determine how many elements should be fitted in a reference square of size λ, to reach a given accuracy. The parameter λ denotes the local wavelength of the simulated signal. As usual, $\lambda = c/f_{max}$ where f_{max} is the maximal frequency component of the source used in the current simulation.

The criterion we are looking for, determines the number of degrees of freedom that should be fitted in a reference area of size $\lambda \times \lambda$. The number of nodes as well as the element size are clearly dependent on the polynomial approximation order. We can then express a relation that defines n_{ppw} as a function of the area a of one element and the number of degrees of freedom DoF that in the reference square:

$$n_{ppw}^2 = \frac{\lambda^2}{a} DoF\,. \tag{5.4}$$

For instance with DG triangles, $DoF = \frac{(N+1)(N+2)}{2}$ and then relation (5.4) becomes:

$$n_{ppw}^2 = \frac{\lambda^2}{a} \frac{(N+1)(N+2)}{2}\,, \tag{5.5}$$

where N stands for the polynomial approximation order.

By assuming the mesh is isotropic, we can consider that all the triangles are equilateral. We then have this simple relation linking the area a of an element with its edge h:

$$a = \frac{\sqrt{3}}{4} h^2\,. \tag{5.6}$$

If we inject (5.5) in (5.6), it is thus possible to define a function defining the isotropic size map which gives the suitable length $h(P)$ for all points P of the point cloud. This function depends on the polynomial order N, the wavelength λ and the number of points per wavelength n_{ppw}. We get the following expression of the **heuristic isotropic size map**:

$$h(P) = 2\frac{\lambda(P)}{n_{ppw}} \sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}\,. \tag{5.7}$$

In this heuristic relation, only n_{ppw} remains undefined and is closely related to the polynomial approximation order. To automatize the size map calculation, an accuracy assessment is done in a homogeneous wavespeed model to determine the appropriate n_{ppw} regarding the polynomial approximation order N. This study is presented in the following subsection.

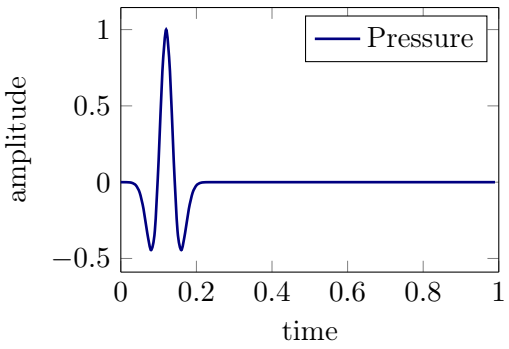
5.2.2 Numerical assessment of the isotropic size map

In this section, we aim to define the most appropriate element size h according to the wavespeed model for a fixed polynomial approximation order N. We will consider a simple homogeneous test case for which we will calculate the number of points per wavelength n_{ppw} for several orders of approximation to guarantee a relative error of at most 1%.

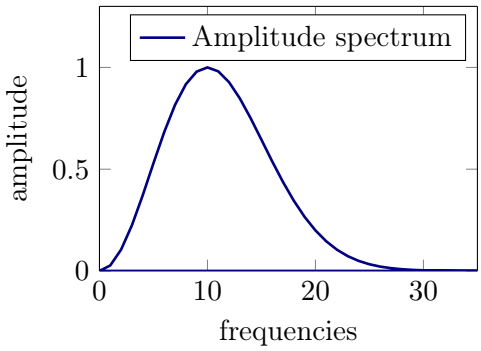
To establish the proper values n_{ppw} for different polynomial orders in the DG solver, an experimental protocol has been defined. We will consider a homogeneous model ($c = 1500\text{m.s}^{-1}$, $\rho = 1000\text{kg.m}^{-3}$) to avoid errors coming from the possible misrepresentation of the physical parameters induced by using constant parameters over each element. The

objective is to make a convergence study for several meshes constructed using the previously defined isotropic size map. Since we will modify the mesh size, it is important to maintain the same approximation of the subsurface model for all the different numerical experiments we are about to describe. This is actually the reason why we will consider for this study a homogeneous model.

A set of receivers has been placed in a 2D acoustic model of size 9200m by 3000m. This set contains 462 receivers spaced 50m apart each other, their locations are shown in Figure 5.9. We use a second order Ricker function source, located at (6100m,250m), defined by f_{peak}=10Hz and $t_{peak} = 0.12$s, which leads to have a maximum frequency $f_{max} = 30$Hz. Figure 5.8 represents the signal used and its amplitude spectrum. The receivers are located on wide horizontal lines, which guarantees measurements of the perturbation for wide incident wave angles. The experimental time is calibrated to last 2s. Since $f_{max} = 30$Hz, 2s is then 60 times longer than the smallest period of the signal we are using. Then, we will consider that 2s is long enough for the reliability of the experiment. The goal of this test is to figure out the behavior of the error depending on the size of the element h that we determined by using relation (5.7).



(a) Ricker source with $f_{peak} = 10$Hz



(b) Amplitude spectrum of the source

Figure 5.8: (a) Source defined as a Ricker perturbation. (b) Amplitude spectrum of the $f_{peak} = 10$Hz Ricker perturbation.

To evaluate the level of accuracy, we use a relative L^2 error defined as follows:

$$Error = \sqrt{\frac{\sum_{rcv}\sum_{t}^{T_f}(p_{ref}(rcv,t) - p_{num}(rcv,t))^2}{\sum_{rcv}\sum_{t}^{T_f}(p_{ref}(rcv,t))^2}}.$$

It measures the difference between the analytical pressure p_{ref}, generated with Gar6more software [Gar6more2D] and the numerical solution p_{num}, obtained with Elasticus software [Elasticus], which also uses DGm. The interest, here, is to quantify the space-time error observed for several meshes built for different values of n_{ppw}. The time step Δt is fixed with respect to the CFL condition associated to the Runge-Kutta 2 time scheme used in the experiment.

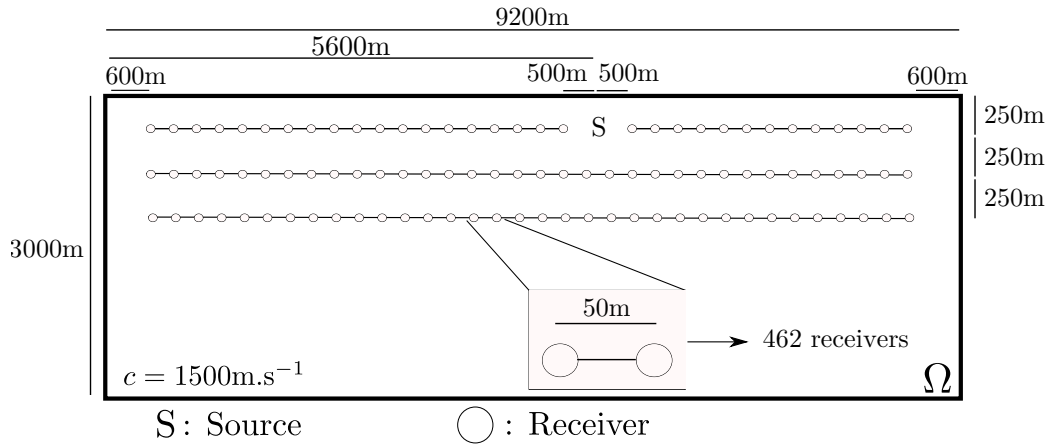


Figure 5.9: Experimental setup for the accuracy assessment.

The objective is to determine the appropriate element size h according to the isotropic size map we defined previously. To do so, we aim to numerically determine the appropriate n_{ppw} values as a function of the polynomial order. Then, for a fixed polynomial approximation order N, we will vary h by modifying the value n_{ppw} in the isotropic size map. We choose to satisfy a relative error of 1% for this experiment. Such accuracy is quite restrictive, in particular in FWI context, where the uncertainty of the input data can be larger than 1%. This criterion can then be relaxed when treating FWI. However, the objective, here, is to determine the size map that leads to an efficient and accurate direct problem simulation. By increasing, step by step, the number of points per wavelength, we are able to vary the size of the mesh and find the appropriate n_{ppw} that satisfies the 1% relative error for each order. Those tests enable us to complete the graphs presented in Figure 5.10.

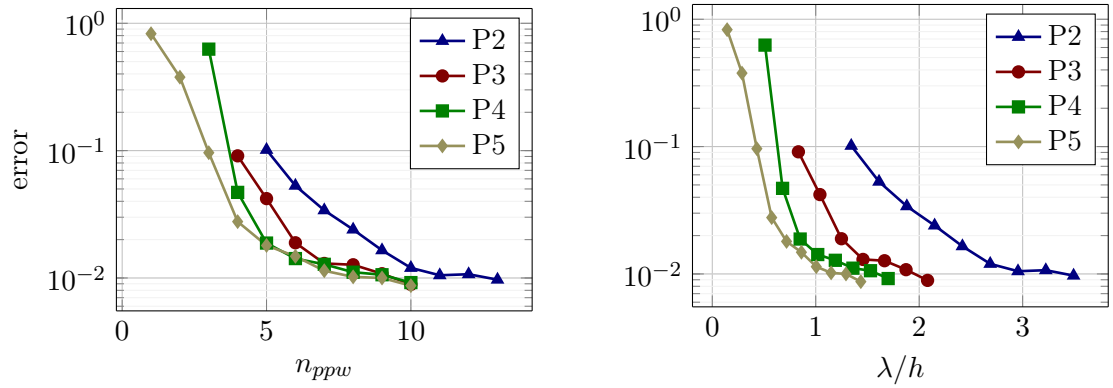


Figure 5.10: Error evolution as a function of n_{ppw} for different polynomial orders (a) and λ/h ratio (b). These graphs allow to link, via the size map, the error of the simulation in a homogeneous medium to the size of elements. According to the desired error criterion, it is possible to extract from these graphs an element size adapted to the simulation using several polynomial order approximations.

We choose to determine the element size h that satisfies the 1% relative error. This criterion is satisfied for the following values of n_{ppw} we display in Table 5.1. These values are not so far from the ones typically taken around five to ten grid points per wavelength as

mentioned in [Fichtner, 2010] and [Igel, 2017] for Finite Element methods. To give an idea of the size of the elements in comparison with the wavelength, we also display in the same table the corresponding ratio λ/h. This ratio is computed by replacing the value of n_{ppw} in (5.7).

Polynomial order (N)	2	3	4	5	6
$\boldsymbol{n_{ppw}}$	13	10	10	9	9
Ratio $\boldsymbol{\lambda/h}$	3.49	2.08	1.70	1.29	1.12
Number of Element (N_e)	303283	105788	69813	40012	29894
Computational time (s)	5230	2764	1956	2017	1789

Table 5.1: Summary table for 1% relative error for 2D DG acoustic solver on triangular grid.

The simulation has been performed using Elasticus software on eigth open mpi threads. The obtained computational time shows the efficiency of p-adaptivity over h-adaptivity as described in [Hesthaven and Warburton, 2007], as long as the physical parameters are correctly approximated. Indeed, using large high order elements is computationally efficient but does not guarantee that the model is accurately defined. In the ideal case, we aim to use large high order elements where the model occurs weak variations and smaller elements with a lower order of approximation (for computational time savings) in regions of the computational domain where the parameters significantly vary.

Industrial context

To determine n_{ppw} for each polynomial order, to get an accurate and efficient modeling is a must-have feature to control the computational time of simulations In addition, these values are required for the automation of the h or p-adaptivity process. The objective is to obtain a mesh adaptation (combining h and p-adaptivity) tool that conforms to the input model, whatever its complexity. The robustness of the process will then allow to adapt the mesh to the physical parameter between any iteration of the FWI.

In this subsection, we determined through numerical experiments, the number of point per wavelength n_{ppw} required for several polynomial orders we displayed in Table 5.1. The numerical experiments have been carried out on a very specific numerical configuration (see Figure 5.9). The so obtained n_{ppw} has been determined in order to get a final global error less than 1% on a 2s long experiment made on homogeneous model.

In what follows, we aim to validate this criterion on a more complex experiment realized on Marmousi model.

5.2.3 Validation on Marmousi model

In this subsection, we aim at checking that the n_{ppw} values we determined on homogeneous model remain valid in a more complicated test case. The isotropic size map relation we defined previously can be used for two different purposes:

- if the maximal polynomial approximation is fixed, then the size map gives a guideline for h-adaptivity,

- if the mesh is given, then we can evaluate the appropriate polynomial approximation order at each element thanks to the isotropic size map formula (p-adaptivity).

It is difficult to compare accuracy results for different h-adaptivity on a heterogeneous model because the error due to the misrepresentation of the model largely dominates the error in which the discretization error itself is hidden. For the validation, we propose the second option that consists in adapting the polynomial order on each element (p-adaptivity) of a given mesh, using the values of the ratio λ/h previously obtained and summed up in Table 5.1.

The mesh used in this validation is composed of 12809 triangles (Figure 5.11c) and is locally refined at the main interfaces of the geophysical model (Figure 5.11d). (*The refinement has been achieved with refinement techniques we describe in Subsection 5.2.5*). We choose such a mesh in order to test our criterion on several configurations of cell size and wavespeed model. We could have used a uniform isotropic mesh but this mesh ensures a wider diversity of polynomial orders and is an interesting case mixing h and p-adaptivity.

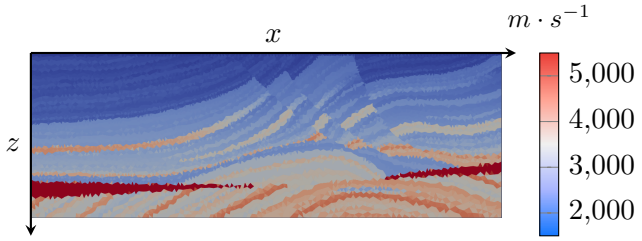
To assess if the choice of the value of n_{ppw} as a function of N is valid or at least agreeable, we compare the traces obtained after a direct problem simulation for one shot, which lasts 4.2s, for different polynomial order approximations. We will compare the efficiency of the p-adaptivity, with respect to the criterion we defined in Table 5.1, in three other configurations that consist in keeping the same polynomial order, from P2 to P4 for each element. A full P5 modeling is used as a reference solution. The time integration is performed with a Runge Kutta scheme of order 2. We are using a first order Ricker function source with a $f_{peak} = 6$Hz, that is to say a $f_{max} = 15$Hz. The application of the p-adaptivity, regarding the ratio λ/h we benchmarked previously, gives us for the given mesh displayed in Figure 5.11c, an order distribution composed of 11% P2, 62% P3 and 27% P4 elements. The corresponding p-adaptivity map is shown in Figure 5.11.

	p-adaptivity	Full P2	Full P3	Full P4	Full P5
L2 relative error	0.38%	17.40%	1.44%	0.30%	ref.
CPU Time (s)	815	502	1122	2244	3455

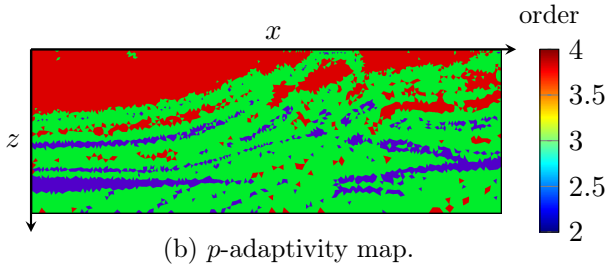
Table 5.2: Relative error on traces obtained for different p-adaptivity strategies.

In Table 5.2, we display the L^2 relative error between the traces obtained for the different polynomial approximation configurations previously stated. In light of the results in this table, it is obvious that applying the p-adaptivity, while considering the established values of λ/h, gives the best compromise between the computational cost and the accuracy. Even if the wavespeed model is more complex, the criterion obtained for the values of n_{ppw} on a homogeneous model seems to be still valid since we get an error less than 1%. Using, for this case, a full P2 order distribution gives a relative error of 17% on the traces, which is far over the 1% subjective error we addressed before. Using the p-adaptivity rule we benchmarked before gives an error close to the full P4 configuration, with a computational time that is 2.75 times lower. Beyond simulations in full P3, the CFL condition obtained with such a mesh and such a model starts to be too much restrictive, which leads to smaller and smaller time steps and explodes the cost of computation. Here, we observe the well-known interest of p-adaptivity that turns out to be a key tool for explicit time schemes. The p-adaptivity actually avoids reducing the time-step and thus contributes to limit the overall computational costs.

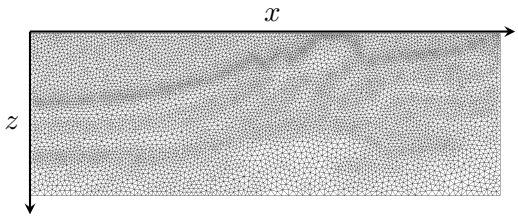
To summarize, the simulations performed with the p-adaptivity computed using the n_{ppw} and λ/h ratio obtained on homogeneous model, remain efficient on a more complex model such as Marmousi. The isotropic size map relation gives us either an efficient h-adaptivity



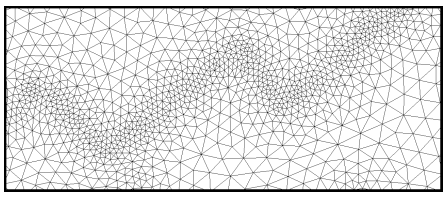
(a) Wavespeed model ($m.s^{-1}$).



(b) p-adaptivity map.



(c) Marmousi refined mesh at the interface.



(d) Zoom on interface mesh refinement.

Figure 5.11: Mesh and model used to check the validity of the isotropic size map on more complex media (Marmousi), (a) Piecewise constant Marmousi model on the mesh, (b) p-adaptivity map following the λ/h ratio summed-up in Table 5.1, (c) Mesh refined at the main interfaces of Marmousi wavespeed model. (d) Zoom on interface mesh refinement.

size map if the maximal polynomial order approximation is given or an efficient p-adaptivity criterion.

In the case where we aim to adapt the mesh according to a set of parameters, the size map expression defines an efficient guideline for the mesh adaptation. It is then possible to fully benefit of h-adaptivity, which has been shown to be very efficient for DG elements [Bernard et al., 2007]. In addition, as shown in the test performed in this subsection, it is still feasible, once we have a given h-adaptivity, to re-evaluate the p-adaptivity on each cell. Indeed, the maximal polynomial order requested by the user in the isotropic size map formula may not be always satisfied at each cell because of geometrical limitations (see (5.3)). Then, applying twice the size map expression (first for the h-adaptivity then for the p-adaptivity) enables us to exploit hp-adaptivity of DG methods for a given model.

Concerning the p-adaptivity, we show in the piece of program 5.1 a code that enables to attribute a polynomial order for each cell of a given mesh. The polynomial order is computed, here, in 2D with respect to the ratio λ/h we determined before (see Table 5.1).

```
# Ratio obtained experimentally
ratio_p1 = 5.00  # Chosen arbitrarily
ratio_p2 = 3.49
ratio_p3 = 2.08
ratio_p4 = 1.70
ratio_p5 = 1.29
ratio_p6 = 1.12

# Loop over all elements k
for k in range(nb_elem):
  c_min = np.min(c[:,k]) # Determine the smallest wavespeed inside k
  lambda = c_min/freq
```

```
max_edge = get_max_edge_length(mesh,k)
ratio = lambda/max_edge

if ratio>ratio_p1:
  p_map[k] = 1
  elif ratio <= ratio_p1 and ratio > ratio_p2:
  p_map[k] = 2
  elif ratio <= ratio_p2 and ratio > ratio_p3:
  p_map[k] = 3
  elif ratio <= ratio_P3 and ratio > ratio_P4:
  p_map[k] = 4
  elif ratio <= ratio_p4 and ratio > ratio_p5:
  p_map[k] = 5
  elif ratio <= ratio_p5 and ratio > ratio_p6:
  p_map[k] = 6
  else:
  p_map[k] = 7
```

Listing 5.1: Synthetic algorithm to determine p-adaptivity map.

Tests and validations performed in this section have been realized to automate the size map computation. The computation of the size map that we designed for geophysical application requires only few parameters:

- the values of wavespeed c expressed as a point cloud on the vertices of the associated mesh;
- f_{max}, the maximal frequency of the source;
- N, the global polynomial approximation order.

We first determined an heuristic expression of the isotropic size map:

$$h(P) = 2\frac{\lambda(P)}{n_{ppw}(N)}\sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}\,. \tag{5.8}$$

Then, using numerical experiments performed on a homogeneous model, we determined the optimal value of the parameter $n_{ppw}(N)$ to estimate a mesh size that ensures a given error for different polynomial orders (see Table 5.1 and Figure 5.10).

Finally, formula (5.8) has been validated on a more complex model (Marmousi). In the remainder of the chapter, the expression displayed in (5.8) will be referred to **default size map**, to say that the size map is used *by default* when no extra information is given.

In the next subsections, we will discuss how to fully use the flexibility offered by the size map computation for defining a local refinement in order to fit the main interfaces of the wavespeed model. But, first of all, we need to be able to detect the interfaces from the model information we have, which is the question addressed next.

5.2.4 Interfaces detection

Most often, for obvious reasons of simplicity, wavespeed models are displayed as constant parameters per cell. However, in an inversion procedure, the wavespeed model needs to be updated and one of the key elements of this update is the detection of true reflectors or interfaces. Basically, interfaces are the areas in the wavespeed model where the wavespeed

undergoes an abrupt change. Reducing the size of the mesh at locations where it is known that a seismic event may occur helps to catch the model and avoid artificial reflectors. The notion of interfaces is linked with the notion of the spatial gradient of the model. We do not have access to a continuous formulation of this gradient since the model is expressed as a point cloud (see Figure 5.12). However, it is possible to have a discrete approximation of this gradient using least square method. It turns out that this method is already implemented in Mshmet which gives an additional motivation to adapt this open source program to compute our metric field.

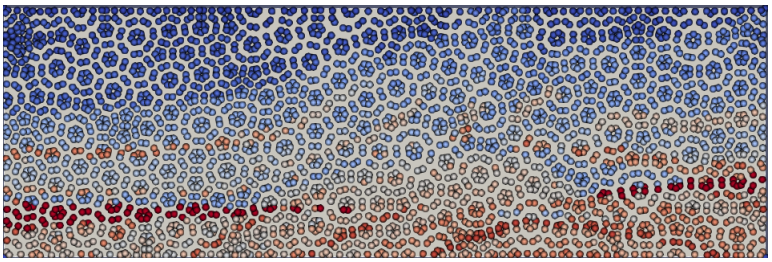
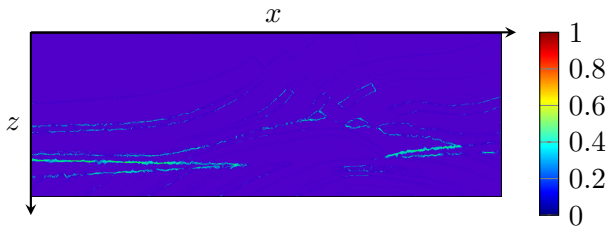


Figure 5.12: Illustration of a point cloud defining Marmousi wavespeed model. Each point represents the quadrature point of WADG method on each element.

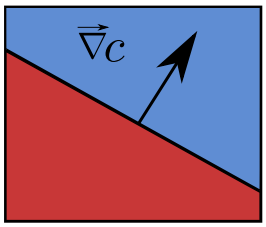
By computing the spatial gradient $\vec{\nabla}c$ of the wavespeed model, we get information of the interfaces (see Figure 5.13). The norm of $\vec{\nabla}c$ helps to localize the interfaces (see Figure 5.13a) and their orientation is given by $\vec{\nabla}c$ as pictured in Figure 5.13b. Using the gradient is then more intuitive than using the Hessian of the wavespeed model. It gives direct access to the location and orientation of the different interfaces. The objective is to add this information in the expression of the default isotropic size map we defined above (5.8) in order to better approximate geophysical interfaces.

We recall that $c(P)$ corresponds to the wavespeed at point P and $\vec{\nabla}c(P)$ is a vector, of dimension 2 in 2D (3 in 3D) that describes the spatial derivative of the wavespeed model at point P:

$$\vec{\nabla}c(P) = \begin{pmatrix} \dfrac{\partial c}{\partial x_1} \\ \dfrac{\partial c}{\partial x_2} \end{pmatrix}(P)\,.$$



(a) Marmousi $||\vec{\nabla}c||_2$ normalized between $[0, 1]$



(b) Sketch of $\vec{\nabla}c$ at an interface

Figure 5.13: P1 representation of the spatial gradient of the model. (a) Gradient amplitude to localize interfaces. (b) Illustration of gradient orientation at an interface.

For the sake of simplicity, we scale $||\vec{\nabla}c||_2$ between $[0, 1]$. Unfortunately, in this case of study, the discrete $||\vec{\nabla}c||_2$ poorly highlights the interfaces of the wavespeed model, as shown in Figure 5.13a. The reason is that the discrete $||\vec{\nabla}c||_2$ is sensitive to the point cloud

locations and the wavespeed model. Some high values of $||\vec{\nabla}c||_2$ are fading the interpretation of the interface location. This effect is obvious when studying the histogram that records the number of points from the point cloud as a function of the normalized gradient amplitude (see Figure 5.14). The point cloud we are using to define the Marmousi model is composed of 75532 points, and we can clearly see that the majority of the points are associated to a normalized value of the gradient that is comprised in between 0.0 and 0.1. Some outliers prevent the interface from being defined easily.

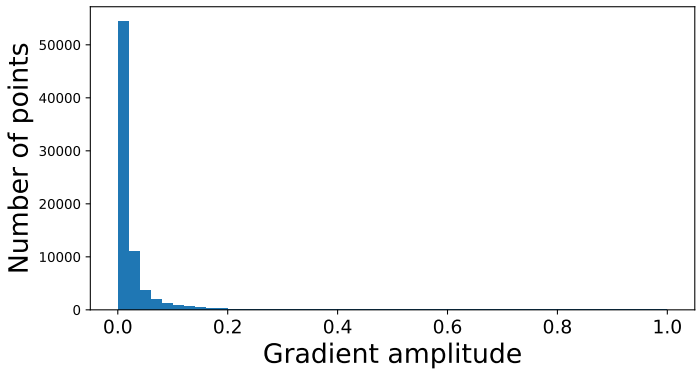


Figure 5.14: Histogram recording the number of points in the point cloud as a function of the normalized gradient amplitude

One way to resolve this is to define a threshold, $\epsilon \in [0, 1]$, over which we will apply the refinement. For each point P of the point cloud, we need to know if it belongs to an interface or not. For each point P, we define the Boolean $Interface(P)$ that satisfies:

$$Interface(P) = \begin{cases} 1, & \text{if} \dfrac{\|\vec{\nabla}c(P)\|}{\max\limits_{P}(\|\vec{\nabla}c(P)\|)} \geq \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{5.9}$$

Remark. This threshold is not unique as it can vary depending on the wavespeed model but also the point cloud which is determined by the mesh to be updated. For instance, two similar models represented on different meshes can have different least square gradients $||\vec{\nabla}c||_2$ leading to different thresholds. To illustrate this, we compare two histograms corresponding to the number of points that are associated with the values of the amplitude of the normalized gradient for two distinct point clouds defining the Marmousi wavespeed model (see Figure 5.15).

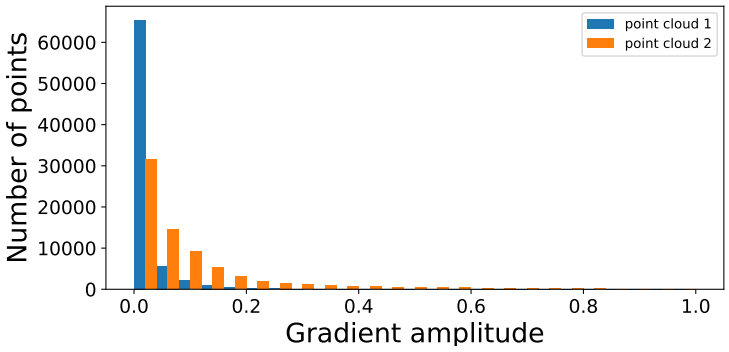


Figure 5.15: Histogram recording the number of points in the point cloud as a function of the normalized gradient amplitude for two different point clouds.

This histogram clearly shows that the distribution of the normalized amplitude is really different despite point clouds that contain a similar amount of points (75532 for point cloud 1 and 74720 for point cloud 2), and represent the same wavespeed model. At a glance, it is possible to conclude that the threshold, chosen in (5.9), to define the interface cannot be the same for these two point clouds.

We will then consider a given point cloud defining the Marmousi model. For this parameterization, we use a dichotomy over the threshold and a visual assessment of the interface to define a threshold of 0.08. Concerning the point cloud we are using, such threshold corresponds to the fact that 5% of the points are labeled as an interface. Then, the $Interface(P)$ map obtained is pictured in Figure 5.16.

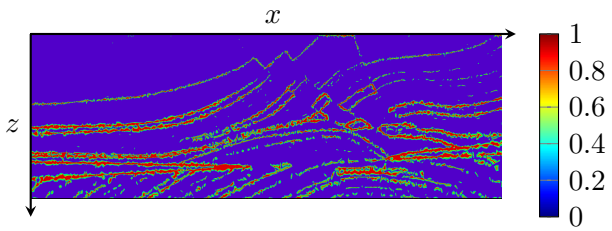
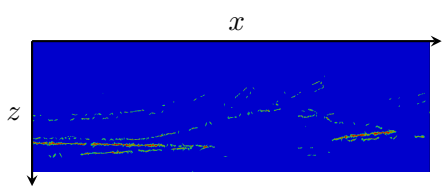


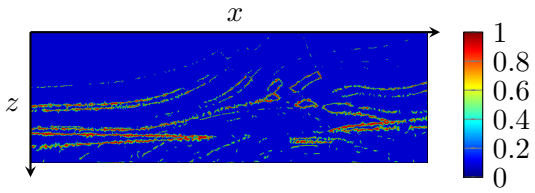
Figure 5.16: Highlighted interfaces.

The definition of the threshold depends on the wavespeed model but also on the point cloud. It is difficult to set a universal criterion that highlights all the interfaces for all kind of models defined on an unstructured grid. The way we are highlighting the interfaces can, without any doubt, be improved in particular with techniques borrowed from image processing or statistics by skipping outliers. The difficulty, here, relies on the fact that the information is not located on a Cartesian grid (see Figure 5.12). It is important to note that the interface detection is a complex task that is not automated yet. We brought in this subsection some hints to define the interface locations, which can be done by defining the threshold ϵ between 0 and 1 at the convenience of the user. To illustrate the choice of the threshold, we display in Figure 5.17 the resulting interfaces for several thresholds values. Note that we are using the point cloud that defines the histogram pictured in Figure 5.14.

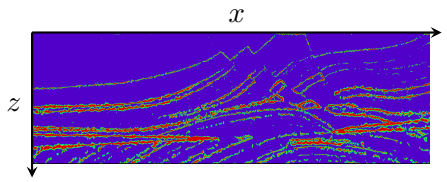
The objective of the two following subsections is to exploit the flexibility offered by a size map computation, by introducing interface information in the metrics. First, we will edit the isotropic size map regarding a point P is on an interface or not. Secondly, we will consider



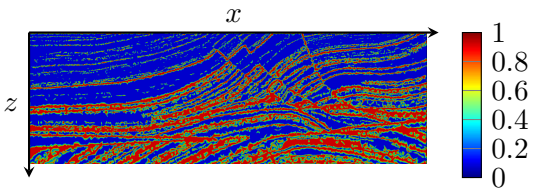
(a) Interfaces highlighted using $\epsilon = 0.2$.



(b) Interfaces highlighted using $\epsilon = 0.1$.



(c) Interfaces highlighted using $\epsilon = 0.08$.



(d) Interfaces highlighted using $\epsilon = 0.02$.

Figure 5.17: Marmousi highlighted interfaces for several threshold values.

anisotropic metrics at the interfaces in order to take the interface orientation into account. As it is a proof of concept, we will consider the Marmousi model and the interface map we just defined in Figure 5.16.

5.2.5 Isotropic refinement

Before defining what is an isotropic refinement, let us clarify what we are calling a refinement. A refinement consists in adapting the size map at a point P leading to construct smaller elements than the default isotropic metrics developed before.

Without taking into account the interface, the default isotropic size map has been defined by the scalar field h (5.10), where N is fixed and denotes the maximal polynomial order of approximation desired by the user:

$$h(P) = 2\frac{\lambda(P)}{n_{ppw}}\sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}\,. \tag{5.10}$$

Then, we redefine locally the size map where an interface is detected. At these locations, the size map will adapt the mesh to be locally reduced or shrunk from what it could have been with the default isotropic size map. In this section, we will consider an isotropic refinement: the size map at an interface is represented by a scalar, leading to reduce homogeneously the size of the cell in all directions. Basically, we define the scalar size map as follows:

$$h(P) = \begin{cases} 2\frac{\lambda(P)}{n_{ppw}}\sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}, & \text{if } Interface(P) = 0, \\ \frac{1}{r}2\frac{\lambda(P)}{n_{ppw}}\sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}},\ r \geq 1.0, & \text{if } Interface(P) = 1. \end{cases}$$

The isotropic case is the easiest one. The refinement is only described by a real factor r that stands for the reduction of the element area. We have adapted the interface of the software computing the size map in order to take into account the refinement factor r at the user convenience.

Remark. If we choose $r = 1$, then the refinement at the interfaces is ignored.

To compare the impact of the local refinement of the mesh near the interfaces, we conduct an error study on three different meshes. The objective is to quantify the error made by a

misrepresentation of the model induced by the mesh. We set up direct problem simulations on a shot in Marmousi model for a source represented by a first order Ricker function ($t_{peak} = 0.2s$, $f_{peak} = 6Hz$ and $f_{max} = 15Hz$). The first mesh is a mesh without any refinement. It is obtained by generating a mesh according to the default isotropic size map (5.10). The second and the third ones are respectively refined at the interfaces by a factor $r = \sqrt{2}$ and $r = 2$. A fourth mesh is used to have a reference solution. Its size map has been computed with a global refinement of $r = 2$ which is carried out by setting the threshold $\epsilon = 0.0$ and amounts considering all points of the point cloud to be located on an interface. In addition, in the fourth case, we are using Weight-Adjusted Discontinuous Galerkin (WADG) model description to generate a solution that is as little as possible polluted by errors from model approximation. We refer to the section dedicated to WADG method at page 93.

The parameters used to generate the size maps are summed up in Table 5.3.

	Order	Frequency	ϵ	r factor
mesh 1	4	15Hz	0.0	1.0
mesh 2	4	15Hz	0.08	$\sqrt{2}$
mesh 3	4	15Hz	0.08	2.0
mesh 4	4	15Hz	0.0	2.0

Table 5.3: Parameters to generate the size map.

Table 5.4 gives us the total number of elements of the generated meshes. After refinement at the interface, the sizes of elements under consideration are not following anymore the default size map which provides rules to adapt the size of the mesh according to the polynomial order requested by the user. In practice, since the elements are smaller after adaptation, we may reduce the polynomial order at the interfaces. Then, it makes sense to re-evaluate the p-adaptivity, at each cell, in the same way we proceeded in the validation subsection. We display in Table 5.4 the percentage of elements for each order of approximation after evaluating the p-adaptivity.

Remark. It is worth noting that 92% of the elements for the first mesh are P4 elements. This percentage highlights the capability of our workflow to adapt the mesh in keeping with the user instructions.

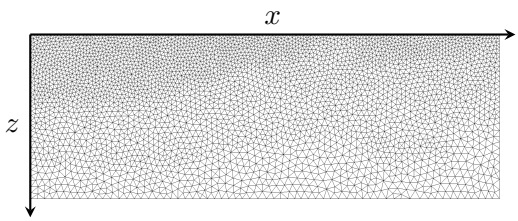
	Nb of Elements	P2 Elements	P3 Elements	P4 Elements
Mesh 1	6808	5 (<0.1%)	518 (8%)	6225 (92%)
Mesh 2	8434	180 (2%)	2127 (25%)	6127 (73%)
Mesh 3	12809	1424 (11%)	7981 (62%)	3404 (27%)
Mesh 4	26621	0 (0%)	0 (0%)	26621 (100%)

Table 5.4: Meshes information and polynomial order decomposition.

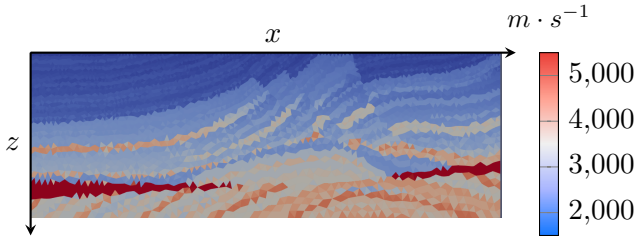
Figure 5.18 represents the mesh for four levels of refinement (left pictures) and the associated representation of the wavespeed model (right pictures). We can see that the refinement improves the quality of the interfaces in the wavespeed model. This helps to clean the medium from possible parasitic reflectors that can affect the wavefield to the point of giving erroneous results despite using a very precise numerical method.

To quantify the advantage brought by the refinement, we compute the relative L^2 error of the traces obtained on mesh 1 to 3 with the reference solution obtained on mesh 4. The

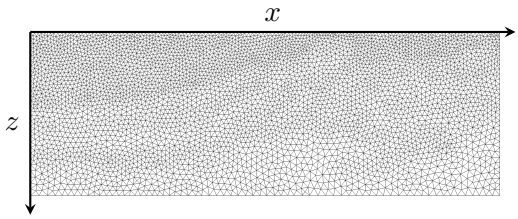
reference solution has been obtained using a P4 polynomial approximation on each cell and a WADG quadrature formula of order 9 resulting in 505799 points defining the model. This last configuration ensures we have a reference solution that avoids perturbations from model misrepresentation.



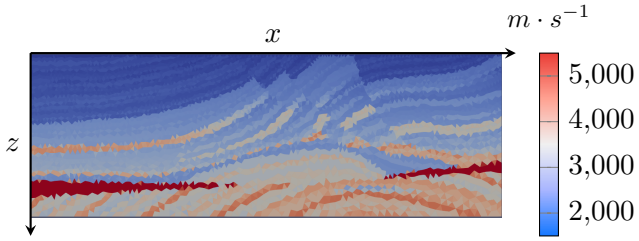
(a) Mesh 1.



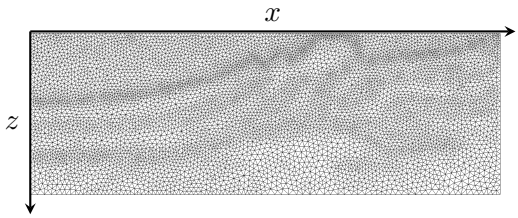
(b) Wavespeed model on mesh 1 ($m.s^{-1}$).



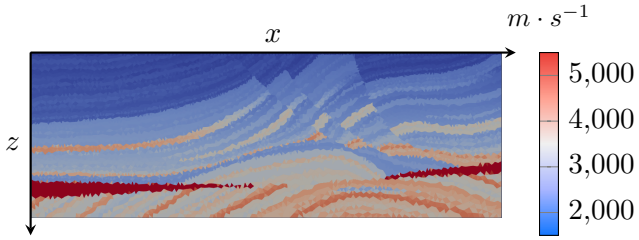
(c) Mesh 2.



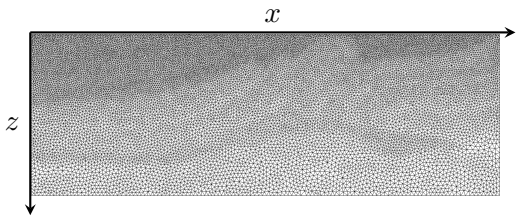
(d) Wavespeed model one mesh 2 ($m.s^{-1}$).



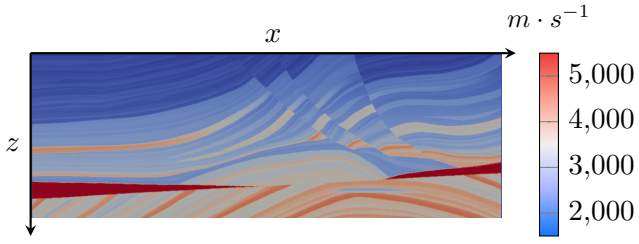
(e) Mesh 3.



(f) Wavespeed model on mesh 3 ($m.s^{-1}$).



(g) Mesh 4.



(h) Wavespeed model on mesh 4 ($m.s^{-1}$) using 9th order quadrature formula for WADG.

Figure 5.18: Visualization of the wavespeed model and the related mesh for different refinements. (a,b) for mesh 1, (c,d) for mesh 2, (e,f) for mesh 3, (g,h) for mesh 4.

After refinement, the computational burden obviously increases due to the growing number of elements. But this can be attenuated by applying the p-adaptivity. p-adaptivity after refinement is crucial in time domain, as it avoids to reduce too much the computational time step Δt which may increase unnecessarily the computational cost.

Table 5.5 shows the effect of local refinement near the interfaces of the model. We will assume that these errors are mainly induced by the poor approximation of the physical model and the numerical errors are here comparatively insignificant. We showed previously, in

Table 5.2, that the employed numerical scheme is accurate. The higher the refinement, the closer the solution is to the reference solution. These tests also bring to light the error induced by a misrepresentation of the physical parameters. The refinement is here a well recommended strategy to improve the model approximation. For instance, a refinement with a factor $r = 2$ can divide by three the error done without refinement at the interface.

	CPU time(s)	Relative L2 error
Mesh 1	379	15.0%
Mesh 2	463	10.5%
Mesh 3	815	5.2%
Mesh 4	4384	ref.

Table 5.5: Performances induced by the different refinement.

The possibility to design the size map computation as desired offers a lot of flexibility. Using the spatial gradient of the model is a way to highlight interfaces and to enhance the model approximation on those tough areas. We investigated in this section the asset brought by isotropic refinement. Without any, it is difficult to accurately reproduce the characteristics of the medium, especially in the vicinity of interfaces with sharp zones. The previous statement has been confirmed by the error study we conducted in 5.2.7.

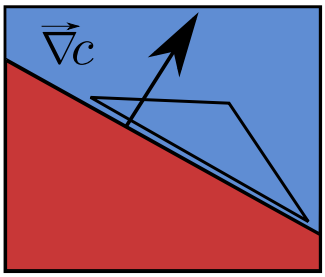
We have implemented a refinement method that is capable of locating and then remeshing problematic interfaces. The error of approximation can then be clearly reduced (Table 5.5).

The isotropic refinement is a straightforward way to refine the mesh since it is mainly implemented by applying a single reduction factor r to the elements. In the next subsection, we will define an anisotropic refinement at interfaces and see its efficiency in comparison with the isotropic refinement developed right here.

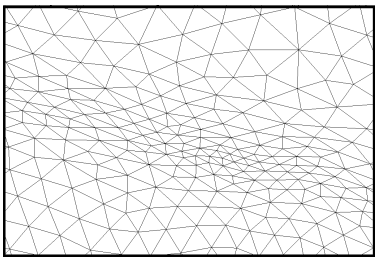
5.2.6 Anisotropic refinement

The key idea of the local refinement is to increase the model resolution near the interfaces by reducing locally the size of the mesh. Previously, we showed that the error can be reduced by applying such a refinement to areas where an interface has been detected. Isotropic refinement leads to locally reduce the size of the element in all directions.

In this subsection, we will extend the investigation conducted for isotropic refinement to anisotropic one. In the previous case, the size map was only defined by a scalar h that determines the size of the edge of the element at a given location. When dealing with anisotropy, the size map is defined as a discrete metric field on all points P of the point cloud. We have then to define a metrics $\mathcal{M}(P)$ at point P that describes the tolerated edge lengths in all directions. The idea of introducing this metrics is borrowed from error estimates that are classically computed on the numerical solution [Alauzet and Frey, 2003] of the PDE under consideration. The objective here, is to overcome the misrepresentation of the model at interfaces by increasing the density of elements at those areas. The anisotropic refinement consists in adapting the mesh to get elements that can be shrunk in the direction of spatial gradient ($\vec{\nabla}c$) if an interface is detected as shown in Figure 5.19a. This refinement is more natural than an isotropic refinement that reduces the size of the element in all directions. It will lead to have elements whose elongation will be favoured in the direction of the interface (see Figure 5.19b).



(a) Ilustration of an anisotropic cell at an interface.



(b) Zoom on anisotropic refinement.

Figure 5.19: Illustration of anisotropic refinement at an interface.

We showed at the beginning of this chapter, see Section 5.1, that we can define a metrics $\mathcal{M}(P)$ at each point P , which is a Symmetrical Positive-Definite (SPD) matrix. Such metrics defines a distance norm $\| \, . \, \|_{\mathcal{M}(P)}$.

The metrics $\mathcal{M}(P)$ can then be computed by taking into account the information of the model c, its spatial gradient $\vec{\nabla}c$ and the interface map, $Interface$ that has been previously introduced in Subsection 5.2.4.

Depending whether or not the point P is located on an interface, we propose to compute the metrics $\mathcal{M}(P)$ as follows:

$$Interface(P) = 0 \begin{cases} \mathcal{M}(P) = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}, \\ \lambda_1 = \lambda_2 = \dfrac{1}{h(P)^2}, \\ h(P) = 2\dfrac{\lambda(P)}{n_{ppw}} \sqrt{\dfrac{(N+1)(N+2)}{2\sqrt{3}}}, \end{cases}$$

$$Interface(P) = 1 \begin{cases} \mathcal{M}(P) = S\Lambda S^{\top}, \quad \Lambda = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \quad \text{and} \quad S = \begin{pmatrix} v_{1_x} & v_{2_x} \\ v_{1_z} & v_{2_z} \end{pmatrix}, \\ \lambda_1 = \dfrac{1}{h_r(P)^2} \quad \text{and} \quad \lambda_2 = \dfrac{1}{h(P)^2}, \\ h_r(P) = \frac{1}{r} 2\frac{\lambda(P)}{n_{ppw}} \sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}} \quad \text{and} \quad h(P) = 2\frac{\lambda(P)}{n_{ppw}} \sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}, \\ \text{with:} \ \ \vec{v_1} = \frac{\vec{\nabla}c}{\|\vec{\nabla}Vc\|_2} \quad \text{and} \quad \vec{v_2} \quad \text{such as} \quad {}^t\vec{v_1}\vec{v_2} = 0 \quad \text{and} \quad \| \, \vec{v_2} \, \|_2 = 1. \end{cases}$$

When there is no interface, we define locally the metrics in order to build isotropic cells satisfying the default isotropic size map. Else, we construct the metrics in order to shrink the element in the spatial gradient $\vec{\nabla}c$ direction. The lengths $h(P)$ and $h_r(P)$ are computed similarly to what it is done in the isotropic refinement section. The anisotropic refinement is also defined by a factor ($r \geq 1.0$) that can be set while computing the size map at the convenience of the user.

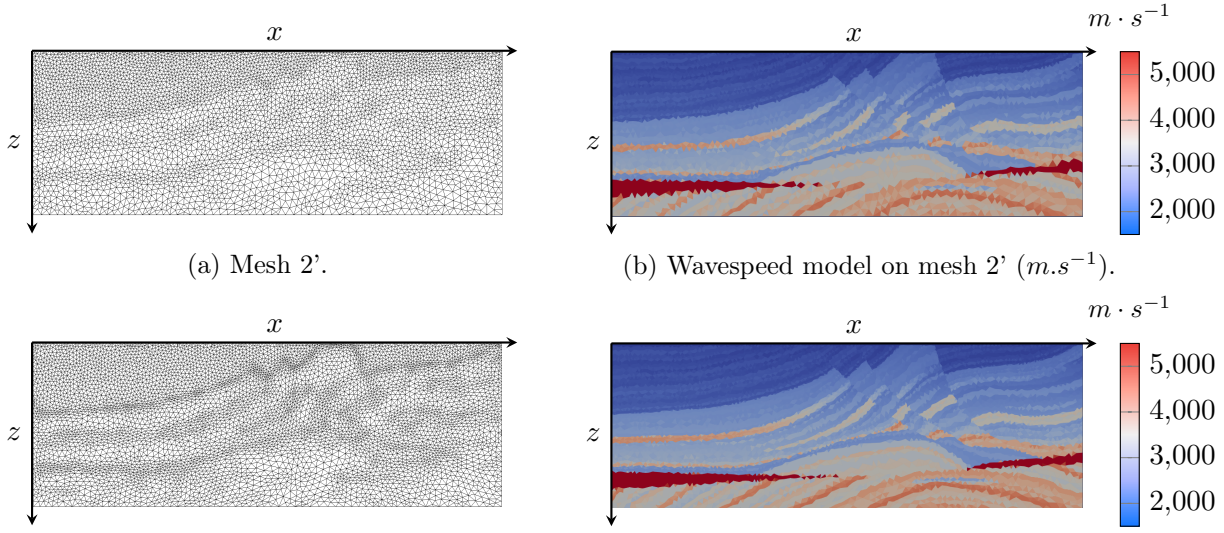
To analyze the improvement of the model approximation brought by using anisotropic refinement, we will compare the global error obtained with the same experimental setup used for isotropic refinement tests. First of all, based on Marmousi wavespeed model, we construct two new meshes denoted by mesh 2' and mesh 3' (Figure 5.20) where we apply an anisotropic refinement of factor $r = \sqrt{2}$ and $r = 2$ respectively. Those meshes are echoing with mesh

2 and mesh 3 of the previous section with the same respective refinement factors. Table 5.6 displays the different parameters used in order to generate the different meshes:

	Order	Isotropic	Freq	ϵ	r factor
Mesh 1	4	True	15Hz	0.0	1.0
Mesh 2'	4	False	15Hz	0.08	$\sqrt{2}$
Mesh 3'	4	False	15Hz	0.08	2.0
Mesh 4	4	True	15Hz	0.0	2.0

Table 5.6: Parameters to generate the size map.

For the same refinement factor r, the anisotropic refinement creates a triangulation that holds fewer elements than the one obtained with isotropic refinement. Instead of reducing elements in all directions, applying an anisotropic refinement results in adapting the size of the element only in the direction $\vec{\nabla}c$. It is worth noting that, despite having fewer elements with the anisotropic refinement, the computational time is shorter on isotropic meshes. This is because the CFL condition is computed with the radius of the inscribed circle, which is ineluctably smaller for anisotropic cells. We refer to Section 2.2.3 on time schemes, page 71 for further explanations on the CFL condition. Furthermore, anisotropic elements require also a high order polynomial approximation to accurately simulate the wavefield in the elongated direction. For those reasons, the CFL condition is even more restrictive on anisotropic cells. A small computational time step Δt inexorably increases the global computational cost.



(a) Mesh 2'. (b) Wavespeed model on mesh 2' ($m.s^{-1}$).

(c) Mesh 3'. (d) Wavespeed model on mesh 3' ($m.s^{-1}$).

Figure 5.20: Visualization of the wavespeed model and the related mesh for different anisotropic refinements. (a,b) for mesh 2', (c,d) for mesh 3'.

A reasonable refinement, such as the one used in mesh 2 and mesh 2', highlights that anisotropic meshes better catch the interfaces and thus help reducing the error. By comparing the results obtained with mesh 3 and mesh 3' in Table 5.7, applying p-adaptivity on the anisotropic mesh leads to have a higher fraction of high-order elements than with isotropic refinement (58% of P4 elements using anisotropic refinement against 27% for the isotropic one). We can see that the computational time is much more important using mesh 3' (1731s) than mesh 3 (815s) with a similar final error. We obtain the same error in both cases because

the refinement factor is similar, and we certainly reach a precision limit at the interfaces. The observed difference in computational time is explained by the fact that in the case of an anisotropic refinement, long edges are kept, which requires using high order of approximation and therefore requires more computations. In the case of an isotropic refinement, the size of the elements is uniformly decreased, which enables to reduce the order of approximation and therefore the amount of associated calculations.

	Nb of Elements	P2 Elements	P3 Elements	P4 Elements	CPU time(s)	Relative L2 error
Mesh 1	6808	5 (<0.1%)	518 (8%)	6225 (92%)	379	15.2%
Mesh 2	8434	180 (2%)	2127 (25%)	6127 (73%)	463	10.5%
Mesh 2'	8226	75 (1%)	1167 (14%)	6984 (85%)	602	8.5%
Mesh 3	12809	1424 (24%)	7981 (62%)	3404 (27%)	815	5.2%
Mesh 3'	11496	603 (5%)	4288 (37%)	6605 (58%)	1731	5.1%
Mesh 4	26621	0 (0%)	0 (0%)	26621 (100%)	4384	ref.

Table 5.7: Performance comparison between isotropic and anisotropic mesh refinement.

Unfortunately, anisotropic cells can strongly affect the stability condition of explicit time scheme. On the one hand, anisotropic cells require higher order polynomial approximation than isotropic ones, due to the preserved edge size in the tangent direction to the interface. On the other hand, we use an explicit time scheme where the time step is computed as a function of the inner radius of the element, so that the anisotropic ratio would degrade the CPU time.

Strategies such as explicit local time stepping [Diaz and Grote, 2015, Gödel et al., 2010] or even local implicit time stepping [N'Diaye, 2017, Barucq et al.] can reduce such effects and can improve the overall time computation. In the frequency domain, there is no such geometrical restrictions and anisotropic refinement is undoubtedly an interesting path to follow because it allows to improve the accuracy with a lower cost in memory in comparison with isotropic refinement.

An anisotropic refinement seems to be more efficient in terms of precision but with higher computational cost than with isotropic refinement. Thus, while working with explicit time schemes and without the implementation of variable time steps, it seems more appropriate to use isotropic refinement.

5.2.7 Conclusion

We have constructed a heuristic size map to include mesh adaption tools in an inverse problem framework in order to have accurate and computationally efficient Forward and Adjoint simulations. To sum up the overall section that concerns the computation of the size map, we first developed a heuristic expression of an isotropic size map in 2D (5.11):

$$h(P) = 2\frac{\lambda(P)}{n_{ppw}(N)}\sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}\,, \tag{5.11}$$

where:

- P is a point of the point cloud defining the model;
- λ is the local wavelength $\lambda = \frac{c(P)}{f_{max}}$;

- N is the maximal polynomial approximation order (fixed by the user);
- n_{ppw} is a heuristic definition of the number of points per wavelength, which depends on the polynomial approximation order.

Then, thanks to several numerical experiments, we have defined a value of n_{ppw} for different polynomial orders to generate meshes for accurate simulations.

We recall the n_{ppw} values and the corresponding ratio $\frac{\lambda}{h}$ that ensures a global error of 1% that we defined thanks to numerical experiments described on page 187:

Polynomial order (N)	2	3	4	5	6
$\boldsymbol{n_{ppw}}$	13	10	10	9	9
Ratio $\boldsymbol{\lambda/h}$	3.49	2.08	1.70	1.29	1.12

Table 5.8: Summary of the appropriate size map for accurate space discretization using 2D DGm.

After validation of the formula (5.11), we have used the calculation of a size map to define local isotropic and anisotropic refinements to improve the interface representation of a given geophysical model. In a first step, we had to detect these interfaces and for this, we used the gradient of the velocity model. It should be noted that the detection of interfaces is difficult to achieve. We have given in the subsection 5.2.4 a few hints to highlight areas where the model is subject to strong contrasts, but we have not been able to establish absolute criteria for locating the interfaces in a simple way. As expected, we have shown that an anisotropic refinement is more efficient than an isotropic refinement in terms of model accuracy. However, it is important to note that if an explicit time scheme is used, an isotropic refinement seems to be more preferable to keep reasonable computational times.

To process isotropic and anisotropic refinement, we have adapted the interface of Mshmet, which calculates the size map, by introducing the following new parameters:

- i, a boolean that defines whether the refinement is isotropic (T) or anisotropic (F) ;
- r, the refinement factor ($r \geq 1$);
- is *epsilon*, the threshold for highlighting the interfaces ($epsilonin[0.1]$).

It has to be emphasized that such local refinements are dependent on interface detection, which is difficult to achieve. We do not provide any absolute criterion to detect the interfaces, but we brought in Subsection 5.2.4 some hints to highlight areas where the model undergoes strong contrasts.

Having a better discretization allows for a better description of the interfaces and thus a better parameterization of the physical model. To finish this section, we propose a final test which consists in comparing the error obtained on simulations with meshes having a similar number of elements. We will compare the seismographs obtained with the adapted meshes with isotropic refinement proposed on page 199 (Meshes 1,2,3) with uniform isotropic meshes (Meshes 1*,2*,3*) on the same experiment.

Table 5.9 shows the gain in accuracy that can be achieved with meshes adapted to the wavespeed model. For a (approximately) fixed number of elements, we observe a better accuracy and therefore a better parameterization of the physical parameters. The computational time is also lower when the mesh is adapted. This can be explained by a geometry favorable to a less restrictive CFL.

	Nb of elements	CPU time(s)	Relative L2 error
Mesh 1	6808	379	15.0%
Mesh 2	8434	463	10.5%
Mesh 3	12809	815	5.2%
Mesh 4	26621	4384	ref.

	Nb of elements	CPU time(s)	Relative L2 error
Mesh 1*	7090	482	19.7%
Mesh 2*	8462	582	17.3%
Mesh 3*	12974	850	8.0%
Mesh 4	26621	4384	ref.

Table 5.9: Comparison of performances using adapted mesh with isotropic refinement (left) and uniform isotropic meshes with comparable number of elements (right).

For this test, we still used the p-adaptivity criterion developed in this chapter to make sure that the final inaccuracy is, at best, exclusively due to the representation of the model. It should be noticed that, before the study conducted in this chapter, we did not have clearly defined criteria to set the p-adaptivity, which would have resulted in either inaccuracies or excessive computational time.

The size map formulation adapted for acoustic wave model has been carefully studied in this section. We determined criteria that offered meshes giving an accurate space discretization that is adapted to the physical model in which we aim to perform simulations. To assess tools we developed, we have generated a mesh (Figure 5.21), which aims to give a reliable discretization for high frequency simulations ($f_{max} = 25$Hz, $N = 3$, $\epsilon = 0.08$, $r = 2$, $i = F$). The strength of the workflow we propose is its capability to describe an adapted space discretization with only few parameters, here: c, f_{max}, N, i, ϵ and r (where i, ϵ and r are optional).

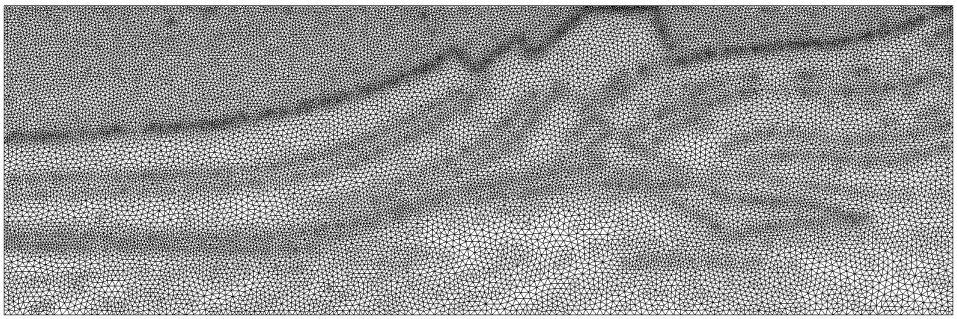


Figure 5.21: Example of adapted meshes obtained with respect to a size map and taking into account sharp interfaces (42K elements).

These meshes offered an efficient h-adaptivity in keeping with the physical parameters. In addition, the parameter determined in Table 5.8 allows re-evaluating on each cell the proper polynomial order (p-adaptivity). We have then elaborated tools that enable to adapt a mesh with respect to the physical model by exploiting the hp-adaptivity feature of DGm. Such optimized space discretization offered accurate simulation obtained in an optimal computational time.

5.3 FWI workflow extended with mesh adaptation

FWI is a computational intensive process as the direct problem is repeated, successively, to the rhythm of updates of the set of numerical parameters. It is then essential to reduce the calculation time and the memory burden of the direct problem while maintaining sufficient

accuracy. The computational loads are directly correlated to the number of elements that comprise the mesh and thus the size of the involved matrices. A significant reduction of that number helps to increase the performance of FWI, whatever the solver is. We can mention the mesh doubling democratized by Spectral Element methods [Komatitsch and Tromp, 2002] and which is used in the SPECFEM code. AxiSEM code is also based upon an impressive degrees of freedom reduction using Axisymmetric Spectral Element Method [Nissen-Meyer et al., 2014]. More recently, [van Driel et al., 2020] proposes to use anisotropic SEM adaptive mesh in an inversion workflow in order to construct wavefield adapted meshes for each shot propagation.

The objective of this section is to include the size map computation and the h-adaptation, previously described, in the FWI workflow. Before going into details, we give a brief overview of the inclusion of mesh adaptation in the FWI process in the scheme Figure 5.22.

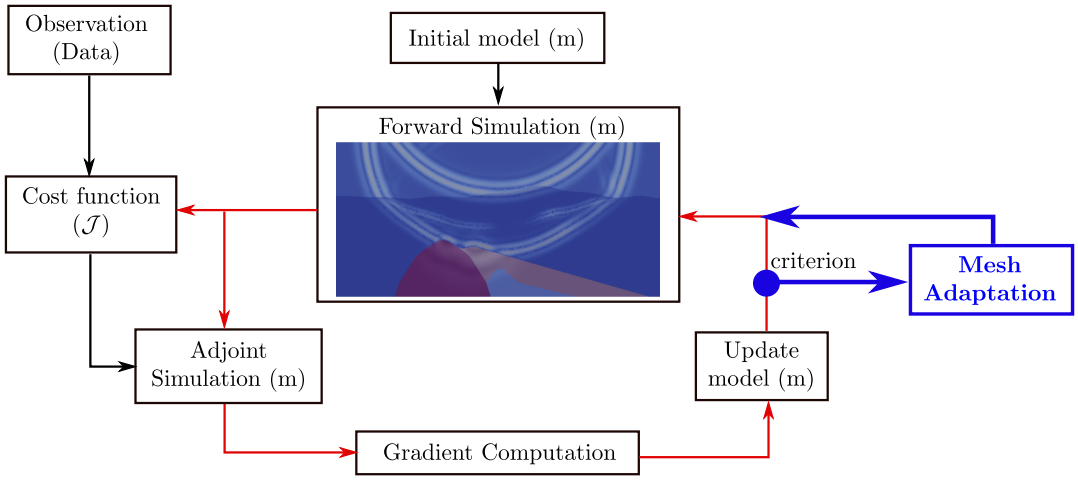


Figure 5.22: FWI workflow extended with mesh adaptation.

This section is organized as follows. We will first explain how the mesh adaptation has been included inside the FWI workflow. We will then discuss a criterion to decide if mesh adaptation must be on or off. Finally, we will present results of inversion, and we will compare reconstruction of wavespeed models with and without the re-meshing process.

5.3.1 Mesh adaptation in FWI workflow

As said previously, we have decided to adopt and possibly adjust tools and software dedicated to mesh adaptation to our FWI workflow. Concerning the size map, we have developed its expression in Mshmet piece of software. It is in this program where main adjustments have been developed in order to take into account the geophysical wavespeed parameters expressed as a point cloud. In the classical mesh adaptation process, this program takes as an input the mesh of the simulation at the i^{th} time step with the unknown field located on all the vertices of $\mathcal{T}_h{}^i$.

Let us now consider the FWI workflow extended with the mesh adaptation. Then, we quickly run into an issue. Indeed, we assumed in the previous section that the model is known at the vertices of the mesh to be updated and this is a prerequisite for computing the size map. But here, this is not the case as the physical parameters are located on the quadrature points formerly defined on page 93, and those points are not located at the vertices of the elements. We then propose to generate an intermediate triangulation $\mathcal{T}_h{}^{int}$ that is linking all points in the wavespeed point cloud. This intermediate mesh has two advantages: it

does not only serve for the calculation of the size map but it also allows a visualization of the wavespeed model at the point cloud (see Chapter 4, page 158). For constructing the intermediate triangulation $\mathcal{T}_h^{int}$, we are using Mmg software that provides a mesher in 2D. It is worth noting that the points are only linked together (without any node insertion, deletion or displacement), hence the resulting mesh is of poor quality but the pints in point cloud are now located at the vertices of the mesh. By using $\mathcal{T}_h^{int}$, the mesh adaptation as a key link of the resolution of the inverse problem diverges from the one traditionally followed in direct problems. The flowchart in Figure 5.23 illustrates how we deal with mesh adaption in the FWI process. It is fundamentally different because we are introducing an intermediate mesh to handle geophysical models.

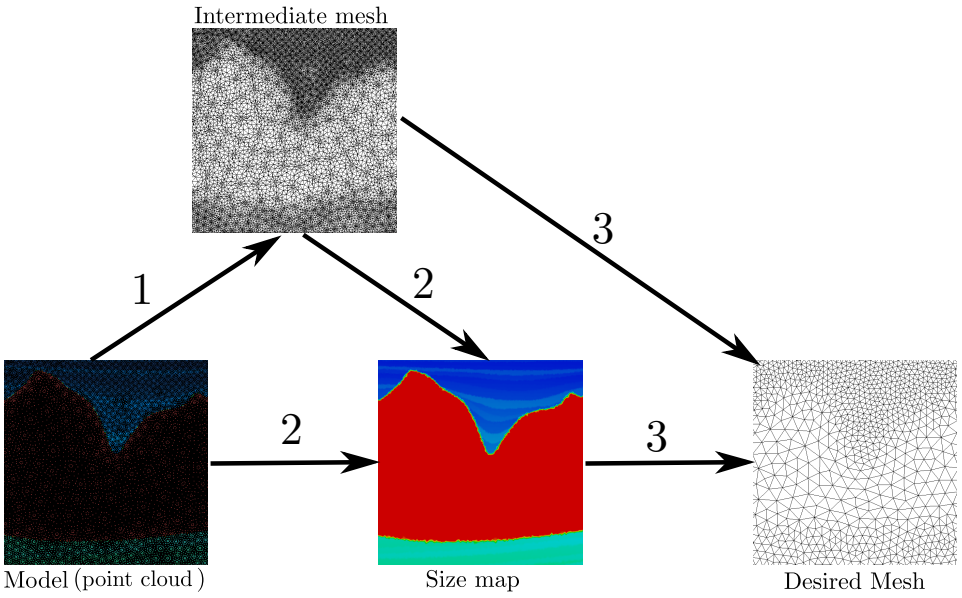


Figure 5.23: Flowchart representing steps for generating a mesh from wavespeed model as a point cloud. ((1) Triangulation, (2) Size map computation, (3) Remeshing)

The mesh adaptation s included in the FWI progressively by accomplishing the following steps:

- **Getting the size map:** ithas been described in 5.2 (see page 185).
- **Building the mesh:** this step is processed by Mmg software. The mesh adaptation algorithm is briefly presented in 5.1.2 (see page 183).
- **Apply preprocessing tools:** this final achievement consists in extracting all the information of the new mesh and the previous model to generate files required to restart the FWI process.

Once the mesh is created, the new coordinates of the model are computed in keeping of the quadrature order (1 for classical DG piecewise constant model, $\geq$2 for WADG). The projection step consists in projecting the values from the outputted model of the i^{th} iteration to the new model that initializes the $i+1^{th}$ iteration. This projection is naively done for each new model point by taking the value of the nearest point from the old model.

Concerning the creation of the polynomial order map, it can be computed thanks to the ratio λ/h we numerically determined previously (see Table 5.8 page 204). This ratio is computed by taking the smallest value of the wavespeed model and the longest edge of the

element. Depending on the value of this ratio, we are able to determine the polynomial order associated to each cell. We refer to the synthetic code displayed in the piece of program 5.1 (see page 192).

Let us consider that, at some point after the i^{th} iteration of the FWI, the decision is taken to adapt the mesh with the new model for next iterations. We sum up in the flowchart in Figure 5.24 the different steps that have been developed in order to have an automatic mesh adaptation in the FWI workflow. Figure 5.24 describes all required steps, in between two FWI iterations, to adapt the mesh with the new model parameters.

In the inversion process, we choose to only consider the default isotropic size map defined in the previous section by the following formula:

$$h(P) = 2\frac{\lambda(P)}{n_{ppw}}\sqrt{\frac{(N+1)(N+2)}{2\sqrt{3}}}\,. \tag{5.12}$$

Remark. We also showed before that it is possible to modify this formula in order to refine the mesh near the main interfaces of the medium. The detection of the interfaces is unfortunately defined with a criterion that is for the moment only defined by visual assessment. It is then not appropriate in the industrial framework, since the procedure is not automated. It is a track that we put aside for future enhancement.

By following the process pictured in Figure 5.24, we defined an automated remeshing procedure that exploits h and p-adaptivity of the DG solver. It is important to note that, once the h-adaptivity is defined, it is worth reevaluating the p-adaptivity on each element. In the case where the default isotropic map is used, most of the elements will satisfy the maximal order given by the user. However, since the mesh adaptation stops at a given threshold (5.3) and the physical model is slightly modified by the projection, we cannot assume that 100% of the elements have a size that satisfies the ratio λ/h for the polynomial order N requested by the user. For computational savings, we recommend to reevaluate the p-adaptivity map after defining the h-adaptivity.

Industrial context

One of the tedious task to develop this framework is to keep tracking the different formats required in between two steps. Making all the workflow displayed in Figure 5.24 working required also to automate the compatibility format between each step. All the dependencies are not expressed in the flowchart. Our objective here is to give a brief idea of the main steps required to adapt the mesh with the current model parameters obtained in the FWI.

Now that we have built an automated procedure to adapt the mesh from the information on the physical parameters, we need a criterion in order to go through this procedure or not. Several criteria can be considered. We can for instance, adapt the mesh every n iterations where n is an integer set by the user. But n has to be chosen to be aware of the optimization process. For instance, the L-BFGS optimizer we developed is computing the search direction using the eight previous iterations' information. In that case, we would not recommend changing the parameterization of the problem every n iterations as long as $n \leq 8$. If n is chosen too small, it can hamper the performances of the optimization process.

We can also impose a criterion on the computational time step. We recall the CFL condition we defined at page 70. For each element, we determine a time step Δt^K that is

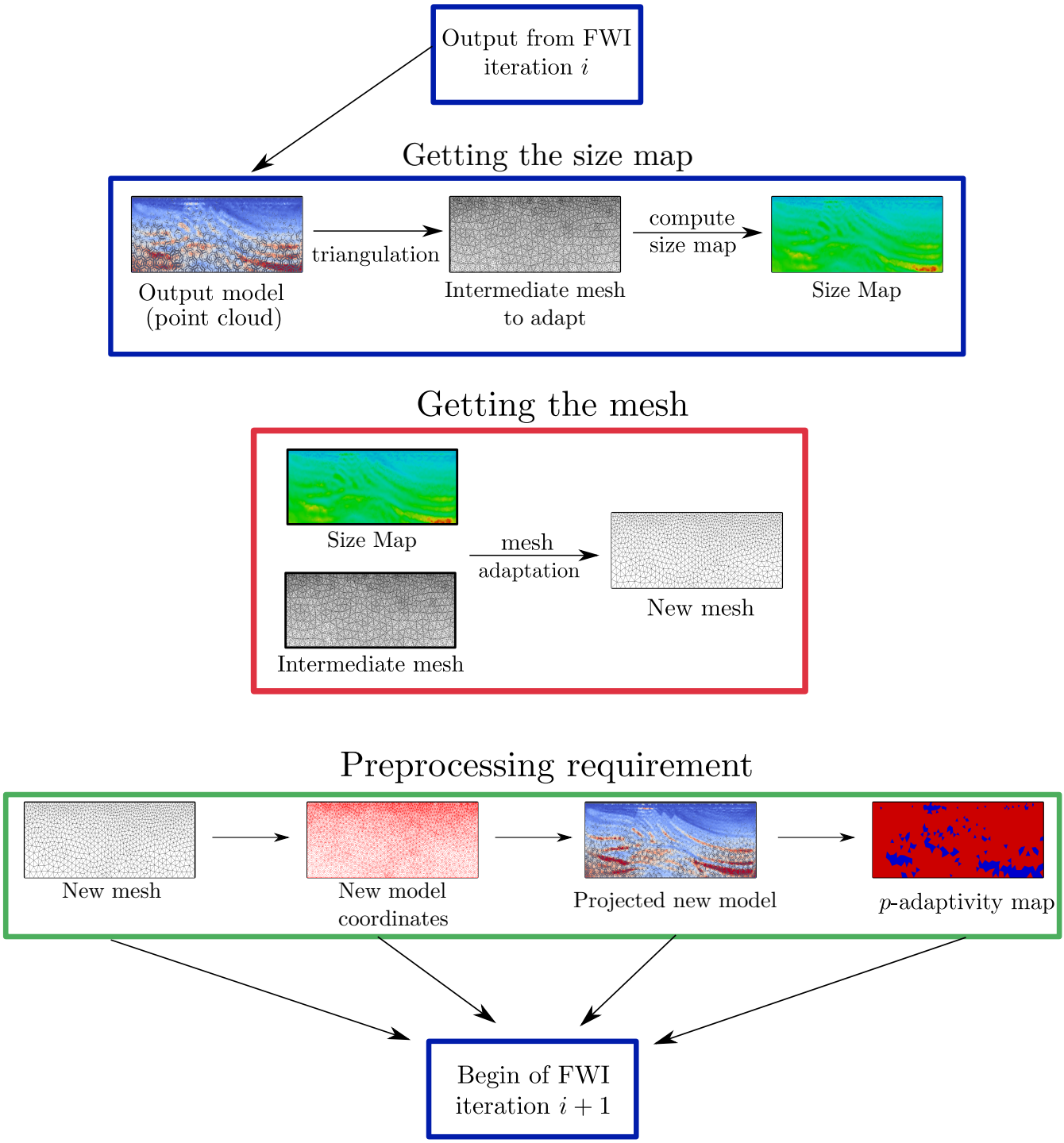


Figure 5.24: Flowchart explaining the steps to be followed to automate the remeshing process between two iterations of the FWI.

defined as follows:

$$\Delta t^K \leq C \frac{r^K}{c^K N^K}.$$

This formula illustrates the difficulty induced by the presence of small elements. Indeed, if r^K is very small, the time step expressed at the scale of the element tends to be small too, more particularly if the speed is high and the order of approximation is also large. Then, the time step associated to the element K is small and it penalizes the overall computational cost. It is a typical situation where re-meshing is efficient by allowing to increase locally the mesh size (r^K) while applying p-adaptivity.

One important feature in the inversion process is the multiscale reconstruction. Indeed, to avoid to fall in a local minimum of the cost function to be minimized, Bunks et al. [1995] recommended reconstructing the medium from low to high frequencies. Reconstructing first large scales with low frequencies enables to deal with simplified problem where there are less local minima. Such a method is performed by filtering the forward source and the observed data with the frequency of interest. We remind that the default isotropic size map formula we defined in (5.12), is a function of the frequency ($\lambda(P) = c(P)/f_{max}$). It is then possible to use a coarser mesh at low frequencies.

Industrial context

At the beginning of the thesis, there was no FWI code. It has been developed from an existing code solving the forward problem. Hence, it has inherited a certain rigidity that imposes the same mesh throughout the run. Such an approach means using a mesh constructed to satisfy the highest frequency for all the steps of the multiscale reconstruction. This is a serious disadvantage resulting in unwelcome additional costs at low frequency.

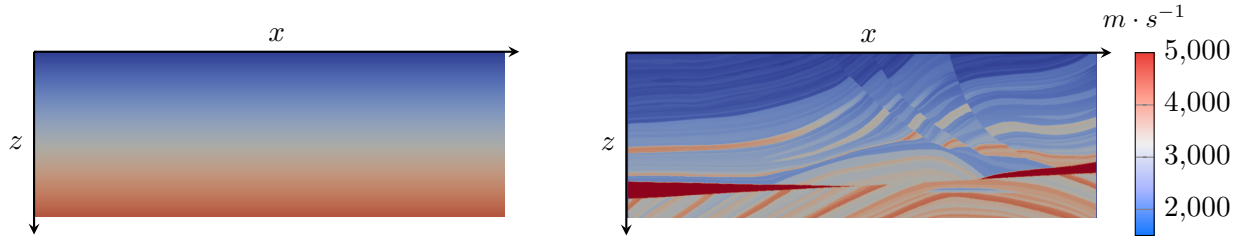
Having a mesh adapted to the frequency component of the simulations is clearly promising in terms of efficiency. This feature is particularly interesting when doing FWI, where it is important to address simulations in a wide band of frequencies including absolutely low frequencies. Indeed, it has been shown [Pratt et al., 1996] that the low frequency content is required. This will drastically improve the computational time when reconstructing low frequencies. Furthermore, between two frequencies, the model parameters and more precisely the wavespeed model, will be taken into account in order to optimize the *hp*-adaptivity.

As a proof of concept of the re-meshing, we decide, in what follows, to re-mesh only at the end of all frequency band in the multiscale reconstruction process. We will present, in the next subsection, comparisons of reconstruction with and without the mesh adaptation in a multiscale reconstruction of the Marmousi wavespeed model.

5.3.2 Mesh adaptation applied on Marmousi reconstruction

In this subsection, we will provide a comparison of the Marmousi reconstruction with and without using the mesh adaptation. We aim to reconstruct the wavespeed model Marmousi starting with a linear initial guess where the wavespeed goes from 1500m.s^{-1} to 4500m.s^{-1} (see Figure 5.25). We will consider a constant density, $\rho = 1000$kg.m^{-3}, all over the domain.

The observed data have been generated by using 20 sources given as a first order Ricker pressure perturbation ($t_{peak} = 0.2$s, $f_{peak} = 10$Hz). Those sources are localized at a constant depth of 50m and are evenly disposed on the X-axis from 550m to 8150m with 400m between each other. Concerning the receivers, we are using 183 receivers at 100m depth positioned



(a) Blind wavespeed model (m.s^{-1}) (b) Target Marmousi wavespeed model (m.s^{-1})

Figure 5.25: Comparison between blind model and target wavespeed model.

from 50m to 9150m on the X-axis with 50m in between. For the reconstruction, we choose to perform 20 iterations at each scale. One scale corresponds to a low pass frequency we apply on the direct and adjoint source. We select five frequency bands, which are displayed in Table 5.10.

	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz	Total
Number of iterations	20	20	20	20	20	100

Table 5.10: Number of iterations for each frequency band.

With such a configuration, we will perform four times the mesh adaptation respectively after the iterations 20, 40, 60 and 80. Without mesh adaptation, we are constrained to use the same mesh during the 100 iterations.

P4 reconstruction with and without adapted meshes

In this experiment we aim to compare the reconstruction of Marmousi wavespeed model, with and without the mesh adaptation tools we have developed. To perform this comparison, we use P4 elements. We will set up the mesh adaptation scheme to adapt meshes for P4 simulations. For the sake of simplicity, we will ignore the p-adaptivity map computed at the end of the workflow pictured in Figure 5.24 and we will force all the elements to be of order 4.

The initial mesh is determined by a blind linear wavespeed model, the polynomial approximation order and the value of the considered maximal frequency. In the classical reconstruction (without the mesh adaptation), we have a mesh that contains 7218 elements. This mesh has been constructed in order to have an accurate representation of a 15Hz signal propagation. However, concerning the experiment with mesh adaptation, the initial mesh contains only 277 elements for the first frequency band since the maximal frequency treated is 2Hz.

We display in Table 5.11 the number of elements in the mesh that has been used at each frequency during the FWI process. Since we are using classical piecewise constant model per element, we have the same amount of parameters to recover as there are elements in the mesh.

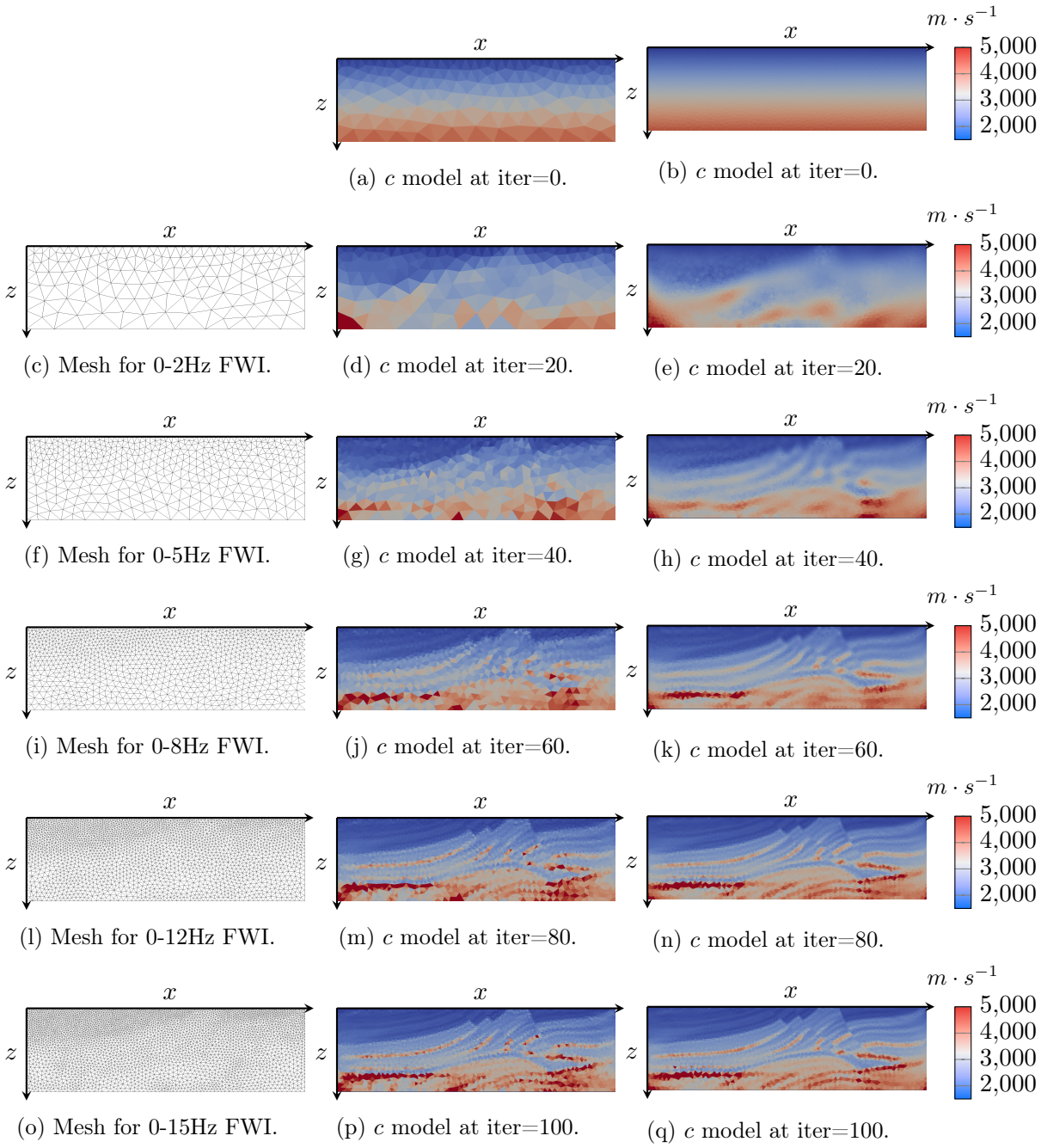
	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz
Number of elements (with mesh adaptation)	277	779	2090	4859	7065
Number of elements (without mesh adaptation)	7218	7218	7218	7218	7218

Table 5.11: Comparison of the number of elements in the mesh for each frequency band for both strategies.

We can observe in Table 5.11 that the number of elements increases with the frequency,

as expected. Adapting the mesh to the frequency leads to have, most of the time, fewer elements except for the last frequency band where there is a similar amount of cells since both configurations are built in order to have accurate propagation of 15Hz signals.

We represent in Figure 5.26 the evolution of the wavespeed model for both strategies. We also display the mesh obtained by using mesh adaptation in the left column.



(a) c model at iter=0. (b) c model at iter=0.

(c) Mesh for 0-2Hz FWI. (d) c model at iter=20. (e) c model at iter=20.

(f) Mesh for 0-5Hz FWI. (g) c model at iter=40. (h) c model at iter=40.

(i) Mesh for 0-8Hz FWI. (j) c model at iter=60. (k) c model at iter=60.

(l) Mesh for 0-12Hz FWI. (m) c model at iter=80. (n) c model at iter=80.

(o) Mesh for 0-15Hz FWI. (p) c model at iter=100. (q) c model at iter=100.

Figure 5.26: Comparison of Marmousi multiscale reconstruction with and without mesh adaptation. (left column = current mesh using the mesh adaptation / central column = with mesh adaptation / right column = without mesh adaptation).

We observe that both methodologies allow to reconstruct the Marmousi model. It is even possible to recover the deepest structures of the model considered. The mesh adaptation strategy offers an interesting computational time saving in comparison with the classical configuration. We display in Table 5.12 the CPU time obtained at each frequency band. Without mesh adaptation, we only know the overall computational time. We can easily assume that each frequency band has the same load.

Remark. We do not consider the time spent in the mesh adaption workflow since it is negligible in comparison with the time required by FWI workflow. For this case, it goes from 5s to 5min depending on we are at low or high frequencies.

It is clear that we are saving a lot of time when reconstructing low frequencies. Concerning the total computational time of those runs, we obtained an overall speed-up factor of 3, which is an impressive speed-up for a similar final result. We can also study the speed-up of the method for each frequency band. In this case, at the first frequency band considered, for instance, we are able to operate 20 iterations of the FWI 144 times faster.

	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz	Total
CPU time with mesh adaptation (h)	3	11	60	230	408	712
CPU time without mesh adaptation (h)	∼432	∼432	∼432	∼432	∼432	2164
CPU time ratio (No mesh adapt/mesh adapt)	144.0	39.3	7.2	1.9	1.1	3.0

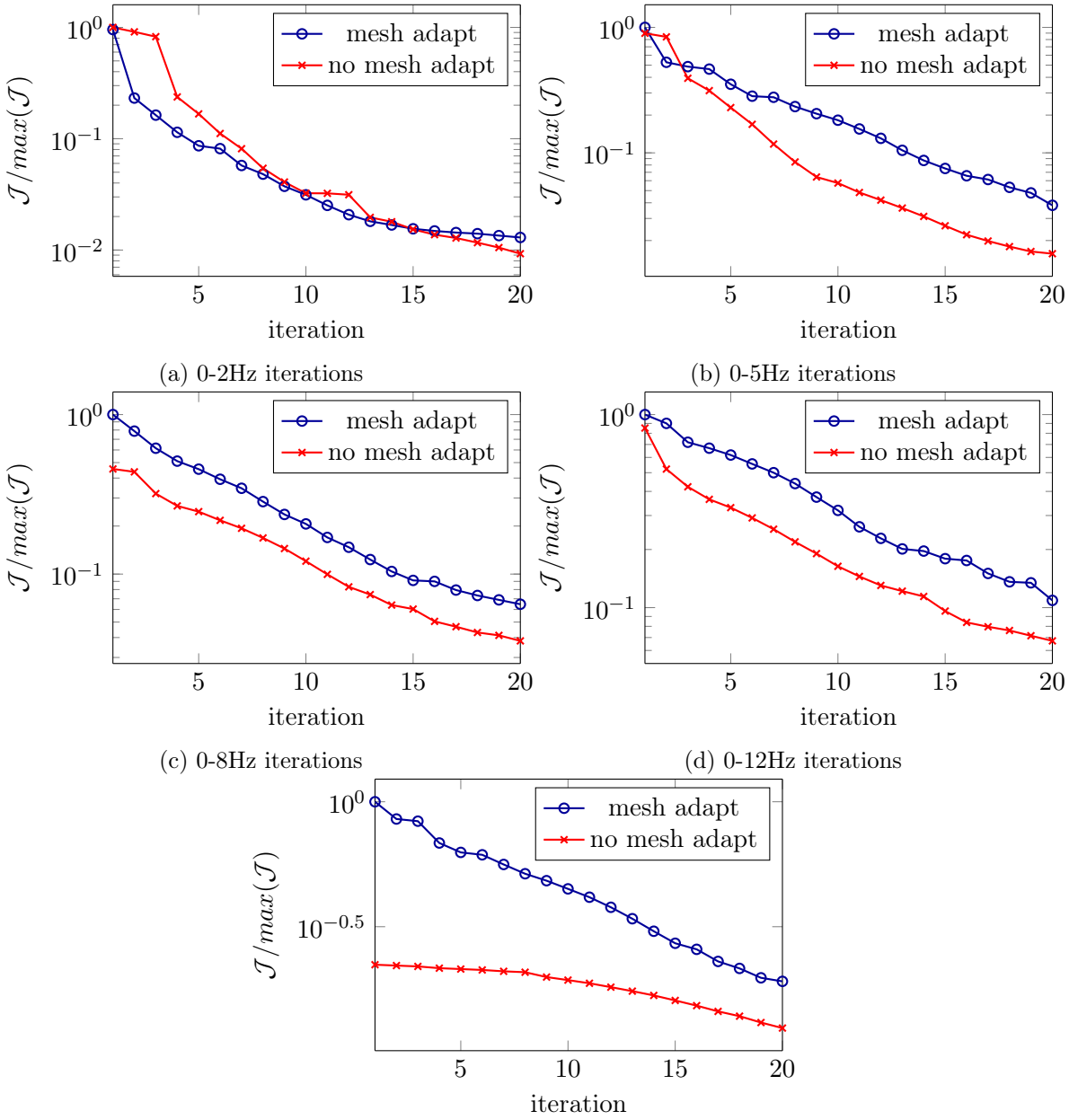
Table 5.12: CPU time comparison between FWI with and without mesh adaptation.

Remark. Note that we are using the CPU time to compare the different methods. The CPU time represents the sum of the contributions of all the processors. We may use the elapsed time but the comparison will not be relevant. For instance, at the lowest frequency, the number of elements is relatively different in the two configurations (277 against 7218) so that we adapt the parallelism. For instance, we are using respectively 20/40/60/80/120 cores for the 5 frequency bands using mesh adaptation against 120 cores during all the old workflow. The method we developed also proposes a great flexibility on the choice of the parallelism, since we are able to change the number of cores in the course of the iterations. Furthermore, a lower CPU time means that the code developed has a lower energy consumption.

For further comparison, we also display the behavior of the cost function for both strategies at each frequency band in Figure 5.27.

We can see at low frequencies in Figure 5.27a and Figure 5.27b that the behavior of the cost function is similar despite the parameterization that is obviously different. At higher frequencies, in Figure 5.27c and Figure 5.27d, both curves have the same trend even if the advantage is still in favor of the strategy that does not use the mesh adaptation. This is clearly visible on the last frequency reconstruction in Figure 5.27e.

This behavior can be explained by the coarse discretization, which can lead to undersample the required number of parameters that describe the medium, hence hindering the reconstruction of the model. In fact, we choose P4 elements based on the observation that it is computationally more efficient to use high order elements than a very fine mesh. Thus, not only we lower calculation costs, but we also reduce the number of parameters to be found during the inversion. For P4 elements, we determined the size of the element h with respect



(a) 0-2Hz iterations

(b) 0-5Hz iterations

(c) 0-8Hz iterations

(d) 0-12Hz iterations

(e) 0-15Hz iterations

Figure 5.27: Evolution of the cost function $\mathcal{J}$ for the different frequency bands during the minimization algorithm, with and without mesh adaptation.

to (5.12), leading to have elements of size $h = \lambda/1.70$, where λ is the local wavelength. Such element is quite large knowing that the FWI is theoretically supposed to retrieve events of size $\lambda/3$ or $\lambda/2$ [Virieux and Operto]. With the mesh adaptation, the low frequency reconstruction is probably undersampled.

To overcome this problem, we can increase the density of the physical parameters. We have then two different ways to proceed. We can either use h-adaptivity to lower the size of the element and then p-adaptivity to adapt as good as possible the polynomial order approximation, or we can use Weight Adjusted Discontinuous Galerkin (WADG) and define the physical model located at quadrature points to consider a variable model per element.

Since our mesh adaptation methodology has been developed to be compatible with WADG formulation, we aim, in what follows, to perform the same experiment with a much denser parameterization using WADG method. Furthermore, we saw in the section dedicated to this technique that its computational cost is not negligible (see page 100). But the use of a coarse mesh as the one we are using at low frequency would reduce the computational load of WADG method while keeping a good model approximation on each element.

P4 + WADG reconstruction with and without adapted meshes

In this experiment, we performed reconstructions using P4-Q9 elements. P4-Q9 means that the wavefield is approached using a polynomial approximation of order 4 and the model is defined at WADG quadrature points related to a quadrature of order 9 on each element. The use of WADG increases the number of unknowns and enables us to overcome the lack of parameters due to large P4 elements. In Table 5.13, we summed up the number of elements and the number of parameters that we aim to retrieve in the inverse problem. In this experiment, the mesh contains a similar number of elements as in the previous experiment. The difference is in the number of parameters that define the physical model. Here since we are in a 2D configuration with a quadrature of order 9, each element contains 19 parameters. For this problem, we have therefore at each frequency band about 19 times more parameters to reconstruct.

	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz
Number of elements (with mesh adaptation)	277	754	1958	4465	6905
Number of parameters (with mesh adaptation)	5263	14326	37202	54340	131195
Number of elements (without mesh adaptation)	7218	7218	7218	7218	7218
Number of parameters (without mesh adaptation)	137142	137142	137142	137142	137142

Table 5.13: Number of elements in the mesh for each frequency band.

With such a parameterization, we observe in Figure 5.28 that we are able to reconstruct the wavespeed Marmousi model in both configurations. As before, an interesting speed-up is offered by the mesh adaptation. We saw, with the previous experiment, that the mesh adaptation offers a FWI reconstruction that can be performed 3.0 times faster. With WADG method, since the computational cost per element is higher, we observe an enhanced speed-up. For this experiment, we compute the 100 FWI iterations 3.75 times faster, which is a really interesting time saving for the FWI. We summed up the computational time at each frequency band in Table 5.14.

	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz	Total
CPU time with mesh adaptation (h)	5	22	73	334	541	975
CPU time without mesh adaptation (h)	∼730	∼730	∼730	∼730	∼730	3654
CPU time ratio (No mesh adapt/ mesh adapt)	146.0	33.2	10.0	2.2	1.3	3.75

Table 5.14: CPU time comparison between FWI with and without mesh adaptation.

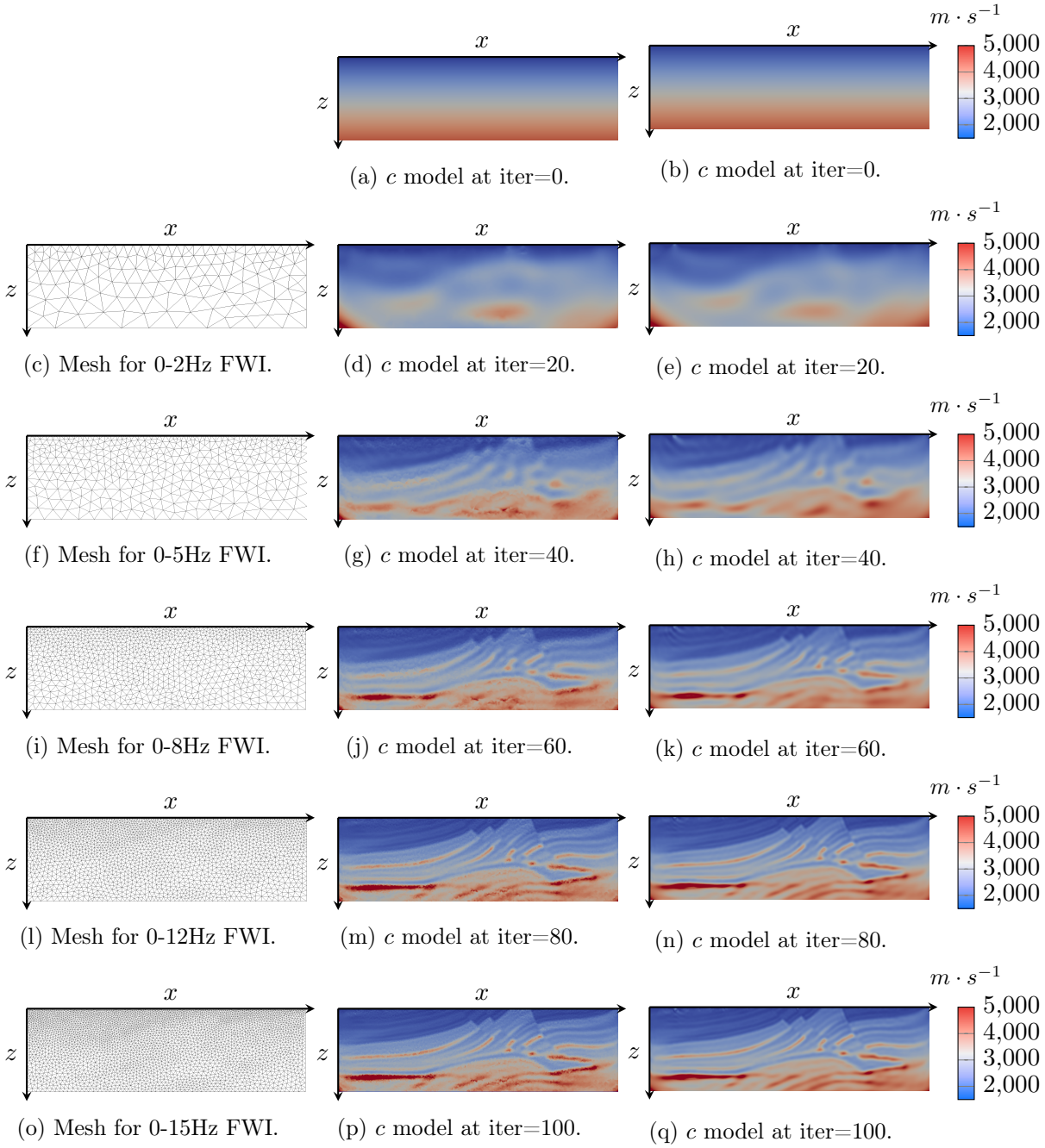
Table 5.14 highlights the efficiency provided by the mesh adaptation. It seems that mesh adaptation is even more efficient while using WADG formulation. Given that WADG discretization increases the computational time compared to constant model per element (see results from Subsection 2.4 at page 100), it is even more important to optimize the number of elements with respect to the frequency and the physical parameters. For instance, regarding the ongoing experiment, using WADG method without any mesh adaptation requires a total computational time of 3654h while 2164h were required with the standard DG method for the same amount of FWI iterations. Those figures represent a global computational time multiplied by 1.7 while we have a factor of 1.4 when using mesh adaptation. This meshing strategy makes the WADG method more affordable in terms of computational time. In addition, it makes sense to use this technique when using coarse elements as we did in this experiment.

In Figure 5.28 we displayed at iteration 20, 40, 60, 80 and 100 the evolution of the model we have reconstructed. Both strategies result in a high definition reconstruction that is close to the target model. Furthermore, at each iteration, we can observe clear similarities between the two reconstructions performed with and without adapting the mesh. For instance, the final models we obtained in Figure 5.28p and in Figure 5.28q are almost identical.

Remark. In pictures obtained with mesh adaptation reconstruction, we can observe small scale textures while the image from the reconstruction keeping the same mesh all along the optimization is smoother. These artifacts come from the projection of the model from the old mesh to the new one. For now, we are using a simplistic projection that consists in taking the value of the nearest neighbor. We have the perspective to enhance this projection but due to time constraints, we have not yet completed this project.

We display in Figure 5.29 the behavior of the cost function, we aim to minimize, for all frequency band as a function of the FWI iterations. We observe that the behavior is similar for both strategies. In contrast with the previous experiment, the trend is in favor of the configuration applying the mesh adaptation. Most of the time, the cost function in the reconstruction performed with mesh adaptation is lower and the slope gives better perspectives of evolution more precisely at high frequency where the curve obtained with the method without mesh adaptation decreases slightly (see Figure 5.29e).

We have successfully included the mesh adaptation in the FWI workflow. We have shown, through two experiments for reconstructing the Marmousi wavespeed model, that we are able to retrieve the targeted model with a comparable accuracy and with an impressive speed-up factor on the overall inversion. For the two experiments performed in this subsection, we have been able to obtain a speed-up factor between 3.0 and 3.75, depending on whether we are using WADG parameterization or not. We also show that WADG technique particularly makes sense for these reconstructions, since we are using very coarse meshes at low frequency.



(a) c model at iter=0. (b) c model at iter=0.

(c) Mesh for 0-2Hz FWI. (d) c model at iter=20. (e) c model at iter=20.

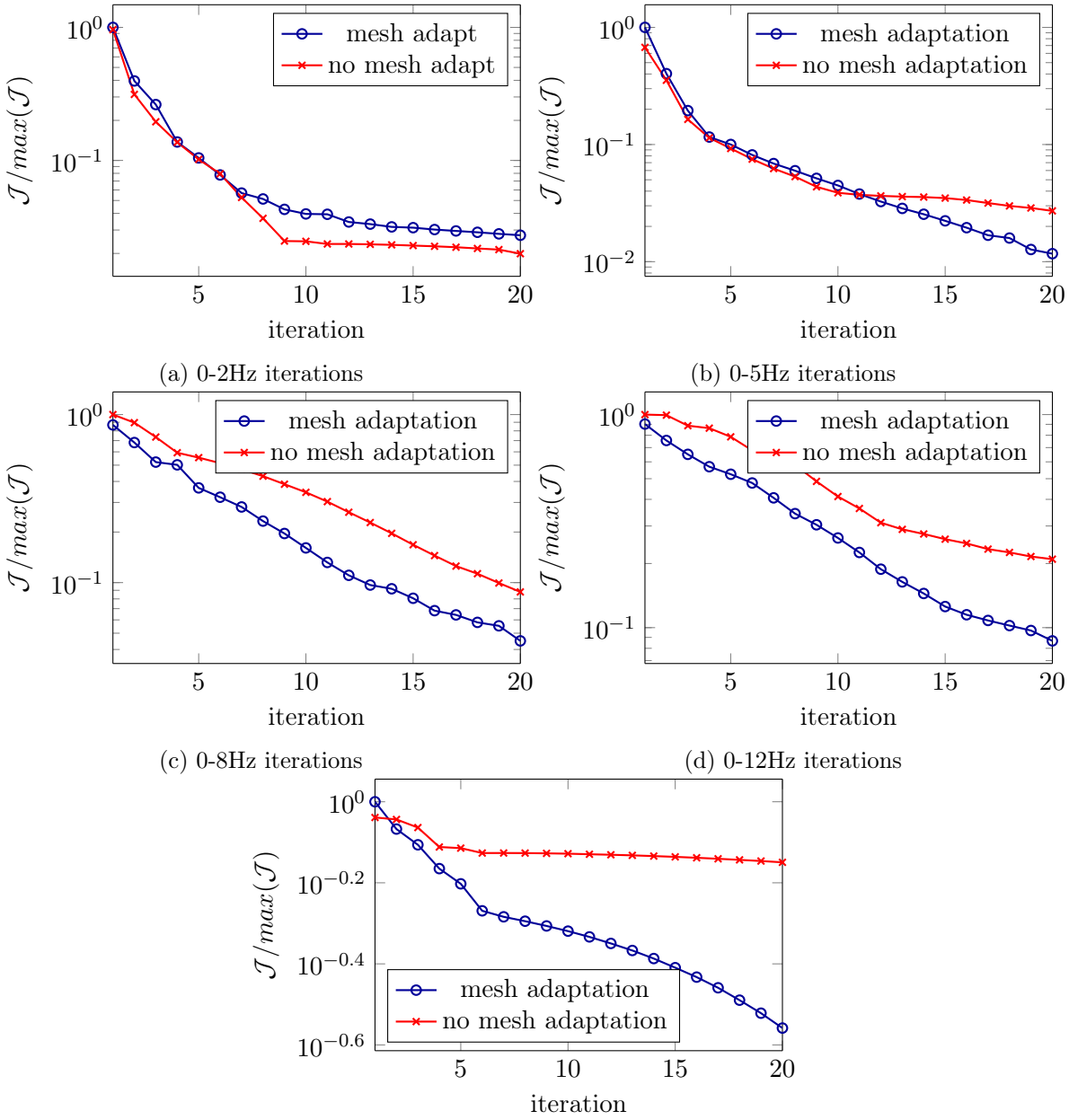
(f) Mesh for 0-5Hz FWI. (g) c model at iter=40. (h) c model at iter=40.

(i) Mesh for 0-8Hz FWI. (j) c model at iter=60. (k) c model at iter=60.

(l) Mesh for 0-12Hz FWI. (m) c model at iter=80. (n) c model at iter=80.

(o) Mesh for 0-15Hz FWI. (p) c model at iter=100. (q) c model at iter=100.

Figure 5.28: Comparison of Marmousi multiscale reconstruction with and without mesh adaptation using WADG method (central column = with mesh adaptation / right column = without mesh adaptation).



(a) 0-2Hz iterations

(b) 0-5Hz iterations

(c) 0-8Hz iterations

(d) 0-12Hz iterations

(e) 0-15Hz iterations

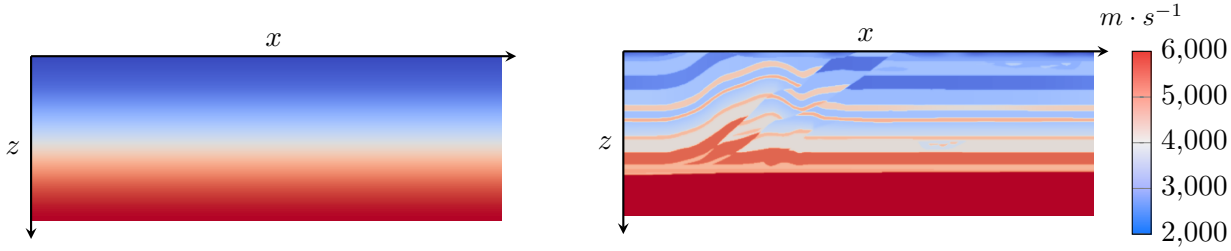
Figure 5.29: Evolution of the cost function $\mathcal{J}$ for the different frequency bands with and without mesh adaptation.

In this situation, having a constant parameter per element leads to undersample the physical model compared to the capability of the FWI to retrieve events of size approximately about $\lambda/3$ or $\lambda/2$ [Virieux and Operto]. These first reconstructions provided the opportunity to test the overall framework including the mesh adaptation that has been automated. Thanks to the size map formula we introduced sooner in this chapter, the mesh is abstracted to the user's eye. Depending on the polynomial approximation, the current physical model and the frequency of the source we are injecting, the mesh is defined according to the user needs. This abstraction of the mesh is even more important during the inversion process, where the intermediate reconstructed physical parameters are not necessarily known by the user and do not necessarily represent the reality.

The above presented workflow is also really flexible. In the experiments shown in this subsection, we restrict ourselves to generate meshes according to a fixed polynomial order approximation (P4). We also defined in advance whether we use WADG formulation or not. In what follows, we aim to make a demonstration of the flexibility of the tools we developed by performing a reconstruction using several configurations in terms of polynomial approximation and model approximation.

5.3.3 Flexibility of the workflow applied to Overthrust 2D model

In this subsection, we intend to fully exploit the flexibility of the mesh adaptation workflow we developed. For this experiment, we propose to reconstruct the 2D section of the Overthrust wavespeed model from a linear initial approximation of the wavespeed model. We pictured the initial and target models in Figure 5.30.



(a) Blind wavespeed model ($m.s^{-1}$)

(b) Target Overthrust wavespeed model ($m.s^{-1}$)

Figure 5.30: Comparison between blind model and target wavespeed model.

For this reconstruction, we propose to use smart configurations in between each re-meshing. The workflow we developed enables to select the polynomial order approximation and the quadrature order each time we adapt the mesh. On the one hand, this experiment will enable to demonstrate the versatility brought by the method. On the other hand, it will give the opportunity to discuss on a strategy to fasten the reconstruction with a smart use of the h-adaptivity, p-adaptivity and WADG parameterization of the model.

For this experiment, we will use 30 sources located at 50m depth with 600m between each other, on the X-axis, from 1000m to 18000m. Concerning the receivers, we are using 391 receivers located at 100m depth from 250m to 19800m, in the X-axis, with 50m in between. We focus on the wavespeed model reconstruction, thus we consider the density as a constant parameter ($\rho = 1000\text{kg.m}^{-3}$).

For the reconstruction, we propose the following schedule that we motivate with the following computational arguments:

- **[0-2Hz] 20 iterations using P4 Q9 elements:** for the low frequencies, we aim to

perform these iterations as fast as possible. We then propose to use high polynomial orders to get as few elements as possible. WADG here compensates the lack of information when the mesh comprises large elements in which the parameters are supposed to be constant.

- **[0-5Hz] 20 iterations using P3 Q7 elements:** we aim to decrease the cost of WADG discretization by reducing the polynomial order and thus the quadrature order. With P3 elements, WADG formulation requires applying a quadrature formula of order 7.
- **[0-8Hz] 20 iterations using P2 Q5 elements:** we continue to decrease the polynomial order approximation from 3 to 2 in order to reduce the cost of the model application that can be important using WADG. With P2 elements, WADG formulation requires applying a quadrature formula of order 5.
- **[0-12Hz] 20 iterations using P2 Q1 elements:** For this frequency band, we give up WADG method and use the classical DG formulation set in a model with elementwise constant parameters. The polynomial order is 2 and the quadrature formula is of order 1. We use P2 elements since we determined previously the ratio $\lambda/h \approx 3.5$. In that way, we can have a parameterization that covers an area of size smaller than $\lambda/2$ and $\lambda/3$.
- **[0-15Hz] 20 iterations using P5 Q11 elements:** each iteration will be very time-consuming. For this reason, we aim to use the most efficient configuration as possible. For these final iterations, we aim to exploit Bernstein-Bézier polynomial basis efficiency with high polynomial order elements. We showed, in the chapter dedicated to the forward problem, that Bernstein-Bézier polynomials is getting more efficient in comparison with nodal polynomials for polynomial order $N \geq 5$ in 2D (see Table 2.5 in page 89). But high order polynomials go with large elements. Hence, we go back to WADG formulation. By this way, this step will compensate the lack of resolution inherited from the previous step where the model was misrepresented with piecewise constant parameters.

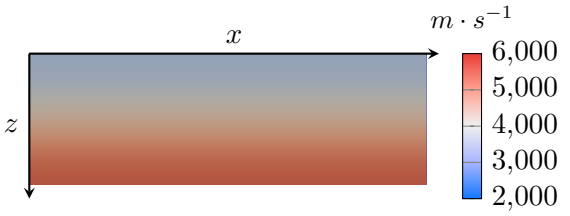
Remark. It is worth noting that, until now, we defined a size map according to a restrictive accuracy threshold of 1% on the experimental setup developed in 5.2.2. Such criterion is probably too strong for FWI where the uncertainty on the observed data is way more important than 1%. To fasten the last frequency band reconstruction, we use $n_{ppw} = 5$ instead of 9. In such a case, WADG method is even more important since the elements are approximately of size $h \approx 1.4\lambda$.

The strategy we propose here, is probably not optimal in terms of time consumption, but it proves the capability and the flexibility of the mesh adaption to switch easily from one discretization to another.

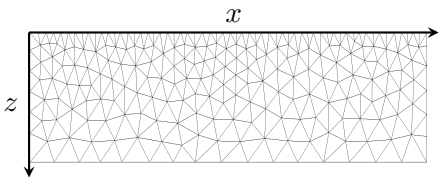
We display in Figure 5.31 the evolution of the wavespeed model through the FWI iterations. We see in this picture the capability of our FWI workflow to reconstruct the 2D Overthrust wavespeed model while changing the polynomial and model approximations at each frequency band.

We display in Table 5.15 the number of elements for each configuration with the associated number of parameters defining the wavespeed model. We also sum-up in this table all the details of the discretization we have chosen. The last configuration is very coarse but has the most important number of parameters. In that case, the space discretization seems to remain sufficiently precise in light of the wavespeed model that keeps being improved.

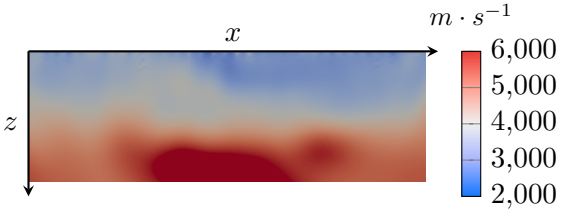
We exhibit in Table 5.16 the computational time of the reconstruction for the different configurations. As expected, the low frequency iterations are solved faster.



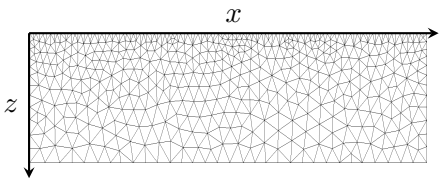
(a) c model at iter=0.



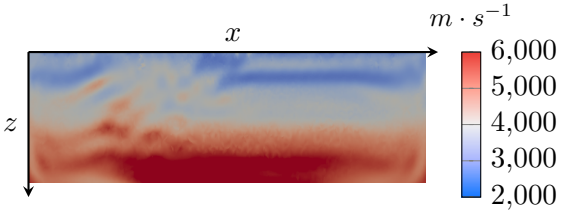
(b) Mesh for 0-2Hz FWI.



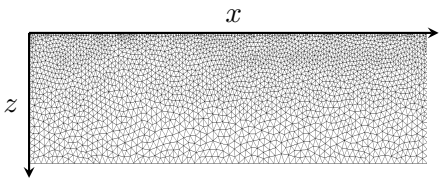
(c) c model at iter=20.



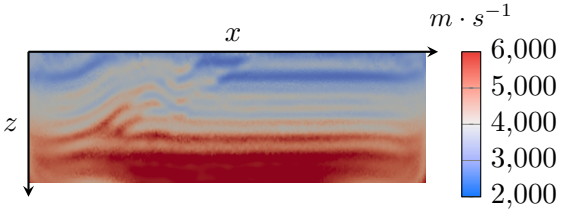
(d) Mesh for 0-5Hz FWI.



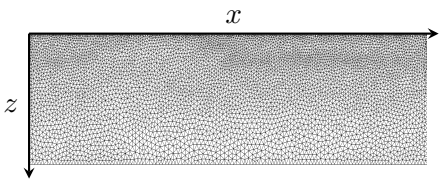
(e) c model at iter=40.



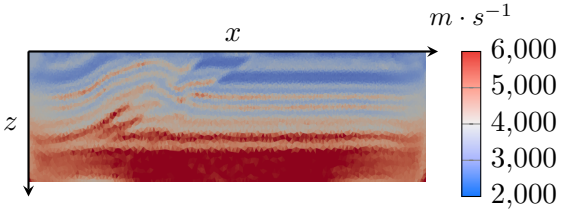
(f) Mesh for 0-8Hz FWI.



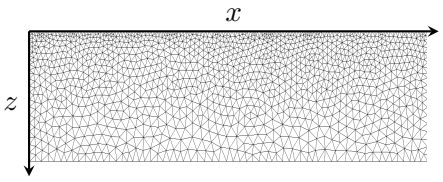
(g) c model at iter=60.



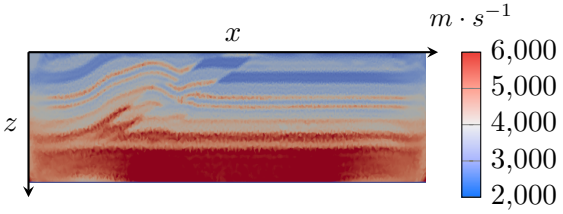
(h) Mesh for 0-12Hz FWI.



(i) c model at iter=80.



(j) Mesh for 0-15Hz FWI.



(k) c model at iter=100.

Figure 5.31: Comparison of Overthrust multiscale reconstruction with several polynomial and WADG approximations.

	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz
Number of elements (N_e)	391	907	5144	11252	1825
Number of parameters	7429	13605	36008	11252	51100
Polynomial order (N)	4	3	2	2	5
WADG quadrature order (N_q)	9	7	5	1	11
Polynomial basis	Nodal	Nodal	Nodal	Nodal	Bernstein-Bézier

Table 5.15: Summary of the discretization chosen for each frequency band.

However, we can observe that we spent less time in the 20 iterations associated to the 0-15Hz reconstruction than the ones for the 0-12Hz frequency band. The run took an average of 13h CPU time per iteration for the 15Hz reconstruction compared to 13.8h for 12Hz. Those figures tend to prove that it is probably computationally more efficient to use high polynomial orders with coarse elements in WADG formulation instead of a fine mesh with low polynomial order and constant model per element. Furthermore, we have to keep in mind that the last configuration benefits from the efficiency of Bernstein-Bézier polynomial basis.

	0-2Hz	0-5Hz	0-8Hz	0-12Hz	0-15Hz	Total
CPU time (h)	18	33	134	276	260	721
Ratio over the total CPU time	2.5%	4.6%	18.6%	38.3%	36.0%	100%

Table 5.16: CPU time for the different frequency bands considered for Overthrust 2D wavespeed model reconstruction.

In this subsection, we were able to highlight the flexibility of the method by switching between several polynomial approximation orders and model quadrature orders in the FWI process. We also try to relax the criterion on the size of the element, we defined for accurate forward simulations, as it is too strict for the FWI. To fasten the last 20 iterations, we relax the criterion for P5 elements by changing $n_{ppw} = 9$ to $n_{ppw} = 5$. Regarding the graph at Figure 5.10 in page 189, this change makes the accuracy criterion move from 1% to 2%, which is still severe but compatible with having much coarser elements of size $h \approx 1.4\lambda$ instead of $h \approx 0.8\lambda$.

The framework we presented enables the user to customize very precisely each step of the FWI process. We also show that it is possible to relax the space discretization without deteriorating the model reconstructed leading to enlarge the space discretization and fasten the reconstruction. On this example, we proposed one strategy that combines most of the numerical tools addressed in the thesis, that is to say:

- *hp*-adaptivity of DGm;
- nodal and Bernstein-Bézier polynomial basis;
- WADG model approximation,

5.4 Conclusion and perspectives

In this chapter, we first introduced classical *h*-adaptation that is usually employed during PDE simulations for memory and computational time savings. Inspired by what is done for direct problem simulations, we proposed an implementation that is compatible with 2D DG

time domain FWI workflow. This resulted in adding one extra step in the FWI workflow in order to perform the *hp*-adaptivity when required (see Figure 5.22).

h-adaptivity is basically defined in two steps:

- definition of a size map (metric);
- adaptation of the mesh according to the metric.

To include these two key features in the FWI process, we adopted and adjusted existing pieces of software contributing to mesh adaptation in the industrial framework in which we are developing. We used:

- Mshmet [AdaptTools - Mshmet], for size map computation;
- Mmg [Mmg Software], for the mesh adaptation.

First of all, we defined and validated, thanks to numerical experiments, a heuristic isotropic size map formula that yields a guideline to adapt the mesh according to the geophysical model represented as the wavespeed model. This size map has been proved to be efficient and accurate for both direct and inverse problems.

Since we were able to define the size map explicitly, we have included the spatial gradient of the model in the size map formula in order to take into account the strong interfaces of the geophysical model. We then proceeded to an isotropic and anisotropic mesh refinement at these tough areas. Such refinements improve the accuracy of the simulation by introducing a better parameterization using *h*-adaptivity of DG solver where the model undergoes strong contrasts. On the example treated in this chapter, we showed that anisotropic refinement seems to be more accurate but the time step should be reduced to guarantee the stability of the explicit time scheme. In fact, we have observed that an anisotropic refinement roughly doubles the calculation time. We showed that, with only two parameters (a refinement factor r and a threshold ϵ), the user is able to adjust the interface refinement as desired. Unfortunately, such refinement requires the user's hand. For the moment, we do not provide any algorithm that is performing a well adapted refinement whatever the model parameters in input is. After performing benchmarks and tests on the size map, we have included the entire *h*-adaptivity in the FWI course. The benchmarks we performed were crucial to automate the workflow in the industrial framework. To achieve a proof of concept, we have compared reconstructions of Marmousi wavespeed model with and without *h*-adaptivity. For this test, we chose to apply *h*-adaptation at each frequency band of the multiscale reconstruction. The comparison of the results obtained with and without *h*-adaptivity highlighted clearly the asset of reconstructing faster the model at low frequencies. This comparison has been performed by considering piecewise constant model per element but also with parameters varying inside each element thanks to WADG technology. WADG method seems to be justified when the elements are coarse, which is the case when considering high polynomial order approximation. In such situation, WADG technology helps to avoid undersampling the parameterization that could compromise the inverse problem resolution.

Finally, in a last experiment, we performed a reconstruction of a 2D section of the Overthrust wavespeed model by combining all the technologies treated in the thesis. We have shown that the new workflow, we developed, gives a high degree of flexibility by adapting at any iterations of the FWI the *hp*-adaptivity, the model parameterization, the choice of the polynomial basis and also the parallelism. There are so many possible configurations that the question arises as to which one is more efficient in terms of computational time.

Perspectives concerning the mesh adaptation

The work presented in this chapter raises numerous perspectives concerning the future of the mesh adaptation workflow we developed in an industrial framework. One of these perspectives is the definition of an absolute criterion, or algorithm, that enables an autonomous interface detection regardless the model considered. This refinement would improve the model approximation while using piecewise constant models. We aim to use h-adaptivity, to increase the density of parameters where the model occurs variations and then use p-adaptivity to adapt the polynomial approximation with respect to the size of the cell.

We defined in this chapter an appropriate formulation of a default isotropic size map according to wavespeed model in order to have accurate direct problem simulations. As it has been discussed during this chapter, such an accuracy is probably too strict for solving an inverse problem regarding the uncertainty on the data. We have the ambition to determine an equivalent criterion that will be a trade-off between the computational time and the accuracy of the forward and backward simulation in order to get an efficient FWI workflow.

And last but not least, we aim to adapt the size map formula we defined in 2D to 3D problems. Here the crucial point is to obtain a formula of the 3D size map, as the rest of the workflow remains unchanged. The extension of the existing tools that enable the h-adaptivity is straightforward once the proper metrics is defined. The 2D results we presented in this chapter proves the efficiency of FWI workflow enhanced with mesh adaptation. The application of such a method in 3D are really promising in terms of computational time savings.

Conclusion

Full Waveform Inversion (FWI) is a high-definition imaging method, based upon the minimization of a cost function that is carried out by applying the adjoint method. The cost function is defined as the difference of the real data (observables) and numerical data. It has been widely used in the frequency domain because its implementation uses only a few frequency values and the Lagrangian formulation of the inverse problem is simplified. It has demonstrated a high level of efficiency in seismic imaging, in particular by using a formulation of the problem in the Laplace domain (see [Faucher, 2017] and the references therein). Seismic imaging of realistic regions by solving frequency wave problems actually corresponds to discrete representations involving more than $n = 500^3$ degrees of freedom, several hundreds of wavelengths per dimension and many thousands of right-hand sides. The system is currently intractable with direct linear solvers due to the size of the resulting linear system. Arithmetic factorization complexity indeed grows usually like $O(n^2)$ and memory consumption like $\mathrm{O}(n^{4/3})$ where n, the size of the system, increases with the cube of the frequency or even more in order to ensure stability of the discretization. An iterative solver would certainly be more suitable, but then the question arises of the simultaneous management of the right-hand sides.

In collaboration with the company Total, we were interested in the FWI written completely in the time domain and compatible with the computing environment deployed in the industrial platform of our partner. This platform is equipped with a DG (Discontinuous Galerkin) high order solver and several time domain schemes. In the framework of this thesis, we developed from scratch a time domain inversion code whose computational performances have been optimized by exploiting all the interesting properties of the DG method. We have considered the acoustic wave equation formulated as a first order system. In the purpose of improving the existing solver, we have first considered the Bernstein-Bézier polynomial basis [Chan and Warburton, 2016]. We conducted a performance analysis based on the comparison between standard DG methods and DG methods with Bernstein-Bézier polynomial basis. This study, made on CPU architecture, led us to conclude that the Bernstein-Bézier basis is interesting for high order polynomials (5 in dimension 2, 3 in dimension 3) and that and that the nodal bases should be privileged for low orders.

Most solvers assume that the propagation medium is discretized by a mesh comprised of elements in which the physical parameters are constant. This hypothesis is not completely satisfactory from the perspective of solving an inverse problem. Indeed, the solution of the inverse problem is the propagation medium itself and it is obtained through the convergence of an iterative minimization method. The set of parameters is thus updated at each iteration, which suggests the possibility that some parameters may change in value within an element during the inversion procedure. We have studied this issue and shown that without taking it into consideration, updating the model can create numerical artifacts. We have therefore integrated the WADG (Weight Adjusted Discontinuous Galerkin) formulation proposed by Chan and Warburton [2016] into the inversion method. WADG formulation is able to take into ac-

count variable parameters within an element thanks to a quadrature formula. We have shown that this formulation eliminates numerical artifacts possibly caused by a misrepresentation of the velocity model.

One important question is the discrete formulation of the minimization problem. There are two options called OtD (Optimize then Discretize) and DtO (Discretize then Optimize) and several authors [Bonnans, 2006, Chavent, 2010, Kern, 2016] have shown that DtO is the best approach because it consists in using an exact gradient for the discrete problem. With OtD, the minimization is implemented with an approximate gradient which can therefore be calculated with errors and these errors can significantly hamper the efficiency of local optimization routines [Gunzburger, 2002], in particular when using the so-called L-BFGS algorithm yet reputed to be among the most efficient. We have thus constructed the DtO and OtD discrete gradients, and we have shown that since they are based upon DG discretizations, DtO gradient turns out to be difficult to implement, due to constraints of programming environment. Moreover, automatic differentiation in order to implement DtO methodology could increase significantly the computational costs, which we already anticipate to be very high. Hence, we opted for the implementation of OtD method, the DtO one being reserved for future development.

We have then tested and validated the optimization code developed on 2D and 3D cases, the objective being first to reconstruct physical parameters constant per element. We have also performed reconstructions of physical parameters varying inside the elements thanks to WADG method [Chan et al., 2017]. As a recent technology in Total's code, it lacked an associated visualization tool. Hence, we have proposed one which has been integrated into the industrial code. We have then shown that WADG technique clearly helps to avoid an under-sampling of the parameterization that could compromise the inverse problem resolution. For more complex reconstructions, i.e. without prior information about the medium to be reconstructed, we have developed a multiscale algorithm that consists in reconstructing the model by increasing frequency bands [Bunks et al., 1995, Boonyasiriwat et al., 2009]. Concerning the 3D case, the results are mixed and this is essentially because we were forced to relax several constraints in order to limit calculation costs. In particular, we have clearly identified the need of having a discretization method that adapts to the wavespeed model but also to the current frequency band used for the reconstruction. This point led us to consider adaptive meshing (h and p-adaptivity) integration into FWI iterations.

As the solution of a minimization iterative process, the model is updated at each iteration until convergence, and we have first implemented h-adaptivity inside the FWI workflow. We inspired ourselves from what is currently done in mesh adaptation for memory and computational time savings when solving PDEs. This work is preliminary and is meant to be a proof of concept in 2D. For that purpose, we have developed a routine which constructs a size map or metrics related to the model and another routine doing the mesh adaptation according to the metrics. We have used Mshmet [AdaptTools - Mshmet], for size map computation and Mmg [Mmg Software], for the mesh adaptation. We have implemented isotropic and anisotropic size maps. Both turn out to be efficient tools for geophysical applications where the model may undergo strong contrasts. Unfortunately, if anisotropic map provides better results, it increases the computational time as it requires halving the time step. We have then compared reconstructions of Marmousi model with and without h-adaptivity. The comparison of the results obtained with and without h-adaptivity highlighted clearly the asset of reconstructing faster the model at low frequencies, hence demonstrating the interest of h-adaptivity. This comparison has been performed by considering piecewise constant model per element but also with parameters varying inside each element thanks to WADG technology. The latter turns out to be interesting when the elements are coarse, as they should be with high polynomial

order approximation.

The perspectives for the work carried out in this thesis are numerous, and we propose to distinguish two major axes.

On the one hand, we can consider everything that is related to the inverse problem itself. Indeed, the next logical step is to treat more complex physics by adapting all the development done for the acoustic solver to the existing elastic and elasto-acoustic sovler in the industrial code. However, considering a more complex physics makes it inevitably more difficult to solve the minimization problem under constraint considered. As seen in the results presented in Chapter 3 and in the theory presented in Chapter 1, the problem is ill-posed and does not necessarily converge to the expected result. To improve the ability of the optimization to find out the global minimum and avoid local ones (cycle-skipping), it would be appropriate to improve the convexity of the cost function. To do so, we plan to add different regularization methods such as Tikhonov or Total Variation which attenuate the strong discontinuity of the reconstructed model but improve the convexity of the regularized cost function [Lopez, 2014]. We also wish to define other cost functions than the classical L2 error function used during the thesis, in particular functions coming from the optimal transport theory which seems, according to the recent literature, to give impressive results for geophysical problems [Engquist and Froese, 2013, Métivier et al., 2016b, Yang et al., 2017].

On the other hand, many improvements are possible to accelerate the direct problem improving de facto the opportunity of iterating the optimization loop more times. To speed up the direct solver we can, as we did in 2D, in the chapter 5, determine a set of criteria to fix the *hp*-adaptivity efficiently. It is also possible to deal with limited aperture, that consists in computing each shot on a sub-mesh which allows to reduce the cost of the direct problem by limiting the domain of calculation. To do this, the mesh adaptation tools but also the solver within the inversion problem must be improved to take into account this additional functionality. In this thesis, we also attempted to reduce the computational cost by using properties of DGs (principally the use of Bernstein-Bézier bases and the use of WADGs). However, these technologies are much more relevant on a GPU architecture than a CPU one [Chan and Warburton, 2016, Chan et al., 2017]. Adding GPU parallelism would reduce the computational costs of the direct DG solver and has already proved its efficiency for imaging applications [Shin et al., 2014]. Finally, as illustrated several times in this manuscript, DGm are particularly efficient for solving PDEs in complex medium and geometries. Coupling the DGm and the SEM as in [Citrain, 2019] would allow to use the DGm to achieve a high accuracy near complex geological structures and to maintain the SEM on more homogeneous or of less interesting areas such as water layers, interior of salt domes or very deep zones.

All of these achievements are possible ways to improve the inversion program, developed during this thesis, for industrial driven applications.

Bibliography

[1] Sigsbee Models. http://www.reproducibility.org/RSF/book/data/sigsbee/paper_html/node2.html. Cited on pages 89 and 169.

[2] AdaptTools - Mshmet. Frey Pascal. ISCD toolbox. Cited on pages 181, 223, and 226.

[3] M. Afanasiev, C. Boehm, M. van Driel, L. Krischer, M. Rietmann, D. A. May, M. G. Knepley, and A. Fichtner. Modular and flexible spectral-element waveform modelling in two and three dimensions. Geophysical Journal International, 216(3):1675–1692, 2019. Cited on page 20.

[4] M. Ainsworth. Pyramid Algorithms for Bernstein–Bézier Finite Elements of High, Nonuniform Order in Any Dimension. SIAM Journal on Scientific Computing, 36(2):A543–A569, Jan. 2014. ISSN 1064-8275. doi: 10.1137/130914048. Cited on page 79.

[5] M. Ainsworth, P. Monk, and W. Muniz. Dispersive and Dissipative Properties of Discontinuous Galerkin Finite Element Methods for the Second-Order Wave Equation. Journal of Scientific Computing, 27(1-3):5–40, June 2006. ISSN 0885-7474, 1573-7691. doi: 10.1007/s10915-005-9044-x. Cited on page 61.

[6] M. Ainsworth, G. Andriamaro, and O. Davydov. Bernstein–Bézier finite elements of arbitrary order and optimal assembly procedures. SIAM J. Scientific Computing, 33(6):3087–3109, Jan. 2011. ISSN 1064-8275. doi: 10.1137/11082539X. Cited on page 76.

[7] F. Alauzet and P. Frey. Estimateur d'erreur géométrique et métriques anisotropes pour l'adaptation de maillage. Partie I : aspects théoriques. report, INRIA, 2003. Cited on page 200.

[8] R. M. Alford, K. R. Kelly, and D. M. Boore. Accuracy of finite-difference modeling of the acoustic wave equation. Geophysics, 39(6):834–842, Dec. 1974. ISSN 0016-8033. doi: 10.1190/1.1440470. Cited on page 187.

[9] G. Allaire. Approximation numérique et optimisation.Une introduction à la modélisation mathématique et à la simulation numérique. June 2019. Cited on page 34.

[10] A. Almomin and B. Biondi. Tomographic full waveform inversion: Practical and computationally feasible approach. In SEG Technical Program Expanded Abstracts 2012, pages 1–5. Society of Exploration Geophysicists, 2012. Cited on page 21.

[11] F. Aminzadeh, N. Burkhard, L. Nicoletis, F. Rocca, and K. Wyatt. SEG/EAEG 3-D modeling project: 2nd update. The Leading Edge, 13(9):949–952, Sept. 1994. ISSN 1070-485X. doi: 10.1190/1.1437054. Cited on page 168.

[12] L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. Pacific Journal of mathematics, 16(1):1–3, 1966. Cited on page 41.

[13] 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, 39(5): 1749–1779, Jan. 2002. ISSN 0036-1429. doi: 10.1137/S0036142901384162. Cited on page 84.

[14] U. Ayachit. The ParaView Guide: A Parallel Visualization Application. Kitware, Inc., Clifton Park, NY, USA, 2015. ISBN 1-930934-30-0. Cited on page 159.

[15] G. J. M. Baeten. Theoretical and Practical Aspects of the Vibroseis Method. PhD thesis, s.n.], S.l., 1989. Cited on page 29.

[16] C. Baldassari. Modélisation et simulation numérique pour la migration terrestre par équation d'ondes. PhD thesis, Université de Pau et des Pays de l'Adour, Dec. 2009. Cited on pages 47 and 66.

[17] A. Bamberger, G. Chavent, C. Hemon, and P. Lailly. Inversion of normal incidence seismograms. Geophysics, 47(5):757–770, 1982. Cited on page 19.

[18] A. Bamberger et al. Une application de la théorie du controle à un probleme inverse de sismique. 1977. Cited on page 19.

[19] W. Bangerth, R. Hartmann, and G. Kanschat. Deal. II—a general-purpose object-oriented finite element library. ACM Transactions on Mathematical Software (TOMS), 33(4):24–es, 2007. Cited on page 20.

[20] H. Barucq. Etude Asymptotique Du Système de Maxwell Avec Conditions Aux Limites Absorbantes. These de doctorat, Bordeaux 1, Jan. 1993. Cited on page 115.

[21] H. Barucq, M. Duruflé, and M. N'Diaye. High-order locally A-stable implicit schemes for linear ODEs. page 37. Cited on pages 67 and 203.

[22] P. T. Bauman and R. H. Stogner. GRINS: A multiphysics framework based on the libmesh finite element library. SIAM Journal on Scientific Computing, 38(5):S78–S100, 2016. Cited on page 20.

[23] A. Bayliss and E. Turkel. Radiation boundary conditions for wave-like equations. Communications on Pure and Applied Mathematics, 33(6):707–725, Nov. 1980. ISSN 00103640, 10970312. doi: 10.1002/cpa.3160330603. Cited on page 50.

[24] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: Basic analysis and examples. East-West Journal of Numerical Mathematics, 4, Jan. 1996. Cited on page 179.

[25] J. B. Bednar. A brief history of seismic migration. Geophysics, 70(3):3MJ–20MJ, 2005. Cited on page 19.

[26] J. B. Bednar, C. Shin, and S. Pyun. Comparison of waveform inversion, part 2: Phase approach. Geophysical Prospecting, 55(4):465–475, 2007. ISSN 1365-2478. doi: 10.1111/j.1365-2478.2007.00618.x. Cited on page 30.

[27] P.-E. Bernard, N. Chevaugeon, V. Legat, E. Deleersnijder, and J.-F. Remacle. High-order h-adaptive discontinuous Galerkin methods for ocean modelling. Ocean Dynamics, 57(2): 109–121, Mar. 2007. ISSN 1616-7341, 1616-7228. doi: 10.1007/s10236-006-0093-y. Cited on page 192.

[28] S. Bernštein. Démonstration du théoreme de Weierstrass fondée sur le calcul des probabilités. Comm. Soc. Math. Kharkov, 13:1–2, 1912. Cited on page 76.

[29] B. Biondi and A. Almomin. Tomographic full waveform inversion (TFWI) by combining full waveform inversion with wave-equation migration velocity anaylisis. In SEG Technical Program Expanded Abstracts 2012, pages 1–5. Society of Exploration Geophysicists, 2012. Cited on page 21.

[30] R. Biswas, K. D. Devine, and J. E. Flaherty. Parallel, adaptive finite element methods for conservation laws. Applied Numerical Mathematics, 14(1):255–283, Apr. 1994. ISSN 0168-9274. doi: 10.1016/0168-9274(94)90029-9. Cited on page 147.

[31] J. F. Bonnans, editor. Numerical Optimization: Theoretical and Practical Aspects. Universitext. Springer, Berlin ; New York, 2nd ed edition, 2006. ISBN 978-3-540-35445-1. Cited on pages 38, 39, 41, 42, 46, 133, 140, and 226.

[32] C. Boonyasiriwat, P. Valasek, P. Routh, W. Cao, G. T. Schuster, and B. Macy. An efficient multiscale method for time-domain waveform tomography. Geophysics, 74(6):WCC59–WCC68, Nov. 2009. ISSN 0016-8033. doi: 10.1190/1.3151869. Cited on pages 165, 166, 177, and 226.

[33] E. Bozdağ, J. Trampert, and J. Tromp. Misfit functions for full waveform inversion based on instantaneous phase and envelope measurements. Geophysical Journal International, 185 (2):845–870, 2011. Cited on page 22.

[34] F. H. Branin. Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations. IBM Journal of Research and Development, 16(5):504–522, Sept. 1972. ISSN 0018-8646. doi: 10.1147/rd.165.0504. Cited on page 43.

[35] R. Brossier, S. Operto, and J. Virieux. Which data residual norm for robust elastic frequency-domain full waveform inversion? Geophysics, 75(3):R37–R46, Apr. 2010. ISSN 0016-8033. doi: 10.1190/1.3379323. Cited on pages 30 and 152.

[36] R. Brossier, B. Pajot, L. Combe, S. Operto, L. Métivier, and J. Virieux. Time and frequency-domain FWI implementations based on time solver-analysis of computational complexities. In 76th EAGE Conference and Exhibition 2014, volume 2014, pages 1–5. European Association of Geoscientists & Engineers, 2014. Cited on pages 21 and 144.

[37] K. P. Bube and R. T. Langan. Hybrid 1/2 minimization with applications to tomography. Geophysics, 62(4):1183–1195, July 1997. ISSN 0016-8033. doi: 10.1190/1.1444219. Cited on page 30.

[38] C. Bunks, F. M. Saleck, S. Zaleski, and G. Chavent. Multiscale seismic waveform inversion. Geophysics, 60(5):1457–1473, Sept. 1995. ISSN 0016-8033. doi: 10.1190/1.1443880. Cited on pages 21, 37, 165, 177, 210, and 226.

[39] J.-P. Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. Journal of Computational Physics, 114(2):185–200, Oct. 1994. ISSN 0021-9991. doi: 10.1006/jcph.1994.1159. Cited on page 51.

[40] Y. Capdeville, E. Chaljub, and J. P. Montagner. Coupling the spectral element method with a modal solution for elastic wave propagation in global earth models. Geophysical Journal International, 152(1):34–67, 2003. Cited on page 20.

[41] D. H. Carlson, A. Long, W. Söllner, H. Tabti, R. Tenghamn, and N. Lunde. Increased resolution and penetration from a towed dual-sensor streamer. First Break, 25(12), Dec. 2007. ISSN 0263-5046, 1365-2397. doi: 0.3997/1365-2397.25.12.27722. Cited on page 29.

[42] M. Carpenter and C. Kennedy. Fourth-order 2N-storage runge-kutta schemes. Nasa reports TM, 109112,, July 1994. Cited on page 68.

[43] M. J. Castro-Díaz, F. Hecht, B. Mohammadi, and O. Pironneau. Anisotropic unstructured mesh adaption for flow simulations. International Journal for Numerical Methods in Fluids, 25(4):475–491, 1997. ISSN 1097-0363. doi: 10.1002/(SICI)1097-0363(19970830)25:4<475::AID-FLD575>3.0.CO;2-6. Cited on page 179.

[44] E. Chaljub, Y. Capdeville, and J.-P. Vilotte. Solving elastodynamics in a fluid–solid heterogeneous sphere: A parallel spectral element approximation on non-conforming grids. Journal of Computational Physics, 187(2):457–491, 2003. Cited on page 20.

[45] J. Chan and T. Warburton. GPU-accelerated Bernstein-Bezier discontinuous Galerkin methods for wave problems. arXiv:1512.06025 [math], Aug. 2016. Cited on pages 22, 76, 82, 83, 87, 156, 177, 225, and 227.

[46] J. Chan, R. J. Hewett, and T. Warburton. Weight-adjusted discontinuous Galerkin methods: Wave propagation in heterogeneous media. arXiv:1608.01944 [math], Jan. 2017. Cited on pages 22, 93, 94, 96, 100, 158, 159, 177, 226, and 227.

[47] G. Chavent. Nonlinear Least Squares for Inverse Problems: Theoretical Foundations and Step-by-Step Guide for Applications. Scientific Computation. Springer Netherlands, first edition, 2010. ISBN 978-90-481-2784-9 978-90-481-2785-6. Cited on pages 32, 34, 46, 105, 140, and 226.

[48] G. Chen, S. Chen, and R.-S. Wu. Full waveform inversion in time domain using time-damping filters. In SEG Technical Program Expanded Abstracts 2015, pages 1451–1455. Society of Exploration Geophysicists, 2015. Cited on page 21.

[49] J. Chen, Y. Chen, H. Wu, and D. Yang. The quadratic Wasserstein metric for earthquake location. Journal of Computational Physics, 373:188–209, Nov. 2018. ISSN 00219991. doi: 10.1016/j.jcp.2018.06.066. Cited on page 31.

[50] B. Chi, L. Dong, and Y. Liu. Full waveform inversion method using envelope objective function without low frequency data. Journal of Applied Geophysics, 109:36–46, 2014. Cited on page 22.

[51] A. Citrain. Hybrid Finite Element Methods for Seismic Wave Simulation: Coupling of Discontinuous Galerkin and Spectral Element Discretizations. PhD thesis, 2019. Cited on pages 51, 70, and 227.

[52] R. G. Clapp. Reverse time migration: Saving the boundaries. Stanford Exploration Project, 137, 2008. Cited on page 144.

[53] B. Cockburn. Devising discontinuous Galerkin methods for non-linear hyperbolic conservation laws. Journal of Computational and Applied Mathematics, 128(1):187–204, Mar. 2001. ISSN 0377-0427. doi: 10.1016/S0377-0427(00)00512-4. Cited on page 52.

[54] B. Cockburn. Discontinuous Galerkin methods. ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik, 83(11):731–754, 2003. ISSN 1521-4001. doi: 10.1002/zamm.200310088. Cited on page 52.

[55] B. Cockburn, G. Karniadakis, and S. (Eds. Discontinuous Galerkin Methods: Theory, Computation and Application, volume 11. Dec. 2000. doi: 10.1007/978-3-642-59721-3. Cited on page 52.

[56] R. Cools. Monomial cubature rules since "Stroud": A compilation — part 2. Journal of Computational and Applied Mathematics, 112(1-2):21–27, Nov. 1999. ISSN 03770427. doi: 10.1016/S0377-0427(99)00229-0. Cited on page 96.

[57] P. Cupillard, E. Delavaud, G. Burgos, G. Festa, J.-P. Vilotte, Y. Capdeville, and J.-P. Montagner. RegSEM: A versatile code based on the spectral element method to compute seismic wave propagation at the regional scale. Geophysical Journal International, 188(3): 1203–1220, 2012. Cited on page 20.

[58] F. A. Dahlen and J. Tromp. Theoretical Global Seismology. Princeton University Press, Oct. 1998. ISBN 978-0-691-00124-1. Cited on page 48.

[59] Y.-H. Dai and Y. Yuan. A nonlinear conjugate gradient method with a strong global convergence property. SIAM Journal on optimization, 10(1):177–182, 1999. Cited on page 39.

[60] R. Dautray and J.-L. Lions. Mathematical Analysis and Numerical Methods for Science and Technology: Volume 3 Spectral Theory and Applications. Springer Science & Business Media, 2012. Cited on pages 55 and 115.

[61] L. Davis. Handbook of genetic algorithms. 1991. Cited on page 25.

[62] P. J. Davis. Interpolation and Approximation. Courier Corporation, Jan. 1975. ISBN 978-0-486-62495-2. Cited on page 65.

[63] J. de la Puente, M. Käser, M. Dumbser, and H. Igel. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes-IV. Anisotropy. Geophysical Journal International, 169(3):1210–1228, 2007. Cited on page 20.

[64] J. de la Puente, M. Ferrer, M. Hanzich, J. E. Castillo, and J. M. Cela. Mimetic seismic wave modeling including topography on deformed staggered grids. Geophysics, 79(3):T125–T141, 2014. Cited on page 20.

[65] A. Dedner, R. Klöfkorn, M. Nolte, and M. Ohlberger. A generic interface for parallel and adaptive discretization schemes: Abstraction principles and the DUNE-FEM module. Computing, 90(3-4):165–196, 2010. Cited on page 20.

[66] E. Deriaz. Stability Conditions for the Numerical Solution of Convection-Dominated Problems with Skew-Symmetric Discretizations. SIAM Journal on Numerical Analysis, 50(3):1058–1085, Jan. 2012. ISSN 0036-1429, 1095-7170. doi: 10.1137/100808472. Cited on page 69.

[67] J. Diaz. Approches Analytiques et Numeriques de Problemes de Transmission En Propagation d'ondes En Regime Transitoire. Application Au Couplage Fluide-Structure et Aux Me}thodes de Couches Parfaitement Adaptees. PhD thesis, ENSTA ParisTech, Feb. 2005. Cited on page 51.

[68] J. Diaz and M. J. Grote. Multi-level explicit local time-stepping methods for second-order wave equations. Computer Methods in Applied Mechanics and Engineering, 291:240–265, July 2015. ISSN 0045-7825. doi: 10.1016/j.cma.2015.03.027. Cited on pages 66 and 203.

[69] V. Dolejší, G. May, and A. Rangarajan. A continuous hp-mesh model for adaptive discontinuous Galerkin schemes. Applied Numerical Mathematics, 124:1–21, Feb. 2018. ISSN 0168-9274. doi: 10.1016/j.apnum.2017.09.015. Cited on pages 185 and 186.

[70] M. Dumbser. Arbitrary High Order Schemes for the Solution of Hyperbolic Conservation Laws in Complex Domains. Shaker, 2005. Cited on page 47.

[71] M. Dumbser, M. Käser, and J. De La Puente. Arbitrary high-order finite volume schemes for seismic wave propagation on unstructured meshes in 2D and 3D. Geophysical Journal International, 171(2):665–694, 2007. Cited on page 20.

[72] Elasticus. Diaz Julien. Magique 3D. Cited on page 188.

[73] B. Engquist and B. D. Froese. Application of the Wasserstein metric to seismic signals. arXiv:1311.4581 [math-ph], Nov. 2013. Cited on pages 31 and 227.

[74] B. Engquist and A. Majda. Absorbing boundary conditions for numerical simulation of waves. Proceedings of the National Academy of Sciences, 74(5):1765–1766, May 1977. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.74.5.1765. Cited on pages 50 and 51.

[75] E. Faccioli, F. Maggio, A. Quarteroni, and A. Tagliani. Spectral-domain decomposition methods for the solution of acoustic and elastic wave equations. Geophysics, 61(4):1160–1174, 1996. Cited on page 20.

[76] E. Faccioli, F. Maggio, R. Paolucci, and A. Quarteroni. 2D and 3D elastic wave propagation by a pseudo-spectral domain decomposition method. Journal of seismology, 1(3):237–251, 1997. Cited on page 20.

[77] R. T. Farouki. The Bernstein polynomial basis: A centennial retrospective. Computer Aided Geometric Design, 29(6):379–419, Aug. 2012. ISSN 0167-8396. doi: 10.1016/j.cagd.2012.03.001. Cited on page 76.

[78] F. Faucher. Contributions à l'imagerie Sismique Par Inversion Des Formes d'onde Pour Les Équations d'onde Harmoniques : Estimation de Stabilité, Analyse de Convergence, Expériences Numériques Avec Algorithmes d'optimisation à Grande Échelle. Thesis, Pau, Nov. 2017. Cited on pages 21, 31, 37, 128, 143, 170, 171, and 225.

[79] A. Ferroni, P. F. Antonietti, I. Mazzieri, and A. Quarteroni. Dispersion-dissipation analysis of 3-D continuous and discontinuous spectral element methods for the elastodynamics equation. Geophysical Journal International, 211(3):1554–1574, 2017. Cited on page 20.

[80] A. Fichtner. Full Seismic Waveform Modelling and Inversion. Springer Science & Business Media, Nov. 2010. ISBN 978-3-642-15807-0. Cited on page 190.

[81] A. Fichtner and H. Igel. Efficient numerical surface wave propagation through the optimization of discrete crustal models—a technique based on non-linear dispersion curve matching (DCM). Geophysical Journal International, 173(2):519–533, 2008. Cited on page 20.

[82] A. Fichtner and J. Trampert. Hessian kernels of seismic data functionals based upon adjoint techniques. Geophysical Journal International, 185(2):775–798, May 2011. ISSN 0956-540X. doi: 10.1111/j.1365-246X.2011.04966.x. Cited on page 25.

[83] A. Fichtner, B. L. Kennett, H. Igel, and H.-P. Bunge. Full seismic waveform tomography for upper-mantle structure in the Australasian region using adjoint methods. Geophysical Journal International, 179(3):1703–1725, 2009. Cited on page 21.

[84] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. The computer journal, 7(2):149–154, 1964. Cited on page 39.

[85] B. Fornberg. The pseudospectral method: Accurate representation of interfaces in elastic wave calculations. Geophysics, 53(5):625–637, 1988. Cited on page 20.

[86] S. French and B. Romanowicz. Whole-mantle radially anisotropic shear velocity structure from spectral-element waveform tomography. Geophysical Journal International, 199(3): 1303–1327, 2014. Cited on page 21.

[87] P. J. Frey and F. Alauzet. Anisotropic mesh adaptation for CFD computations. Computer Methods in Applied Mechanics and Engineering, 194(48):5068–5082, Nov. 2005. ISSN 0045-7825. doi: 10.1016/j.cma.2004.11.025. Cited on pages 179, 183, and 184.

[88] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In European Parallel Virtual Machine/Message Passing Interface Users' Group Meeting, pages 97–104. Springer, 2004. Cited on page 146.

[89] Gar6more2D. Diaz Julien. Magique 3D. Cited on pages 71 and 188.

[90] E. Gauci, A. Belme, A. Carabias, A. Loseille, F. Alauzet, and A. Dervieux. A Priori error-based mesh adaptation in CFD. Methods and Applications of Analysis, 26(2):195–216, 2019. ISSN 10732772, 19450001. doi: 10.4310/MAA.2019.v26.n2.a6. Cited on page 179.

[91] O. Gauthier, J. Virieux, and A. Tarantola. Two-dimensional nonlinear inversion of seismic waveforms: Numerical results. Geophysics, 51(7):1387–1403, July 1986. ISSN 0016-8033. doi: 10.1190/1.1442188. Cited on page 165.

[92] D. Givoli, T. Hagstrom, and I. Patlashenko. Finite element formulation with high-order absorbing boundary conditions for time-dependent waves. Computer Methods in Applied Mechanics and Engineering, 195(29):3666–3690, June 2006. ISSN 0045-7825. doi: 10.1016/j.cma.2005.01.021. Cited on page 51.

[93] N. Gödel, S. Schomann, T. Warburton, and M. Clemens. GPU Accelerated Adams–Bashforth Multirate Discontinuous Galerkin FEM Simulation of High-Frequency Electromagnetic Fields. IEEE Transactions on Magnetics, 46(8):2735–2738, Aug. 2010. ISSN 1941-0069. doi: 10.1109/TMAG.2010.2043655. Cited on pages 66, 69, and 203.

[94] A. Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. Optimization Methods and Software, 1(1):35–54, Jan. 1992. ISSN 1055-6788. doi: 10.1080/10556789208805505. Cited on page 144.

[95] A. Griewank and A. Walther. Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation. SIAM, 2008. Cited on page 127.

[96] M. J. Grote and D. Schötzau. Optimal Error Estimates for the Fully Discrete Interior Penalty DG Method for the Wave Equation. Journal of Scientific Computing, 40(1):257–272, July 2009. ISSN 1573-7691. doi: 10.1007/s10915-008-9247-z. Cited on pages 185 and 186.

[97] M. D. Gunzburger. Perspectives in Flow Control and Optimization. SIAM, 2002. Cited on pages 133, 140, and 226.

[98] K. Guo and J. Chan. Bernstein-Bezier weight-adjusted discontinuous Galerkin methods for wave propagation in heterogeneous media. arXiv:1808.08645 [math], Mar. 2019. Cited on pages 95 and 103.

[99] J. Hadamard. Lectures on Cauchy's Problem in Linear Partial Differential Equations. New Haven Yale University Press, 1923. Cited on page 24.

[100] A.-R. Hedar. Global optimization test problems. Disponible Online en: http://wwwoptima. amp. i. kyoto-u. ac. jp/member/student/hedar/Hedar_files/TestGO. htm, 2007. Cited on page 43.

[101] M. R. Hestenes, E. Stiefel, et al. Methods of conjugate gradients for solving linear systems. Journal of research of the National Bureau of Standards, 49(6):409–436, 1952. Cited on page 39.

[102] J. S. Hesthaven and T. Warburton. Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications. Springer Science & Business Media, Dec. 2007. ISBN 978-0-387-72067-8. Cited on pages 52, 58, 61, 65, 70, 71, 76, 156, 177, and 190.

[103] S. Hestholm. Three-dimensional finite difference viscoelastic wave modelling including surface topography. Geophysical Journal International, 139(3):852–878, 1999. Cited on page 20.

[104] R. L. Higdon. Absorbing boundary conditions for difference approximations to the multi-dimensional wave equation. Mathematics of Computation, 47(176):437–459, 1986. ISSN 0025-5718, 1088-6842. doi: 10.1090/S0025-5718-1986-0856696-4. Cited on page 51.

[105] Hou10ni. Diaz Julien. Magique 3D. Cited on page 71.

[106] H. Igel. Computational Seismology: A Practical Introduction. Oxford University Press, 2017. ISBN 978-0-19-871740-9. Cited on page 190.

[107] H. Igel, P. Mora, and B. Riollet. Anisotropic wave propagation through finite-difference grids. Geophysics, 60(4):1203–1216, 1995. Cited on pages 20 and 21.

[108] M. Jamil and X.-S. Yang. A Literature Survey of Benchmark Functions For Global Optimization Problems. International Journal of Mathematical Modelling and Numerical Optimisation, 4(2):150, 2013. ISSN 2040-3607, 2040-3615. doi: 10.1504/IJMMNO.2013.055204. Cited on page 43.

[109] J.C. Butcher. Numerical Methods for Ordinary Differential Equations. John Wiley & Sons, Ltd, first edition, 2016. doi: 10.1002/9781119121534. Cited on page 67.

[110] C. Johnson, U. Navert, and J. Pitkaranta. Finite element methods for linear hyperbolic problems. Computer Methods in Applied Mechanics and Engineering, 45:285–312, Sept. 1984. ISSN 0045-7825. doi: 10.1016/0045-7825(84)90158-0. Cited on page 52.

[111] G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. 1997. Cited on page 146.

[112] M. Käser, P. M. Mai, and M. Dumbser. Accurate calculation of fault-rupture models using the high-order discontinuous Galerkin method on tetrahedral meshes. Bulletin of the Seismological Society of America, 97(5):1570–1586, 2007. Cited on page 20.

[113] M. Kern. Problèmes inverses : Aspects numériques. Sept. 2002. Cited on pages 32 and 34.

[114] M. Kern. Numerical Methods for Inverse Problems. Wiley-ISTE, first edition, 2016. ISBN 1-84821-818-4 978-1-84821-818-5 978-1-119-13694-1 1-119-13694-6 978-1-119-13695-8 1-119-13695-4. Cited on pages 36, 140, and 226.

[115] R. G. Keys. Absorbing boundary conditions for acoustic media. Geophysics, 50(6):892–902, June 1985. ISSN 0016-8033, 1942-2156. doi: 10.1190/1.1441969. Cited on page 50.

[116] P. A. Khoury and G. Chavent. Global line search strategies for nonlinear least squares problems based on curvature and projected curvature. Inverse Problems in Science and Engineering, 14(5):495–509, July 2006. ISSN 1741-5977. doi: 10.1080/17415970600573684. Cited on page 42.

[117] R. C. Kirby and K. T. Thinh. Fast simplicial quadrature-based finite element operators using Bernstein polynomials. Numerische Mathematik, 121(2):261–279, June 2012. ISSN 0029-599X, 0945-3245. doi: 10.1007/s00211-011-0431-y. Cited on page 76.

[118] D. Komatitsch and J. Tromp. Introduction to the spectral element method for three-dimensional seismic wave propagation. Geophysical journal international, 139(3):806–822, 1999. Cited on page 20.

[119] D. Komatitsch and J. Tromp. Spectral-element simulations of global seismic wave propagation—I. Validation. Geophysical Journal International, 149(2):390–412, May 2002. ISSN 0956-540X. doi: 10.1046/j.1365-246X.2002.01653.x. Cited on page 206.

[120] C. Koutschan, C. Lehrenfeld, and J. Schoeberl. Computer Algebra Meets Finite Elements: An Efficient Implementation for Maxwell's Equations. In Computing Research Repository - CORR, volume 1, pages 105–121. Jan. 2012. ISBN 978-3-7091-0793-5. doi: 10.1007/978-3-7091-0794-2_6. Cited on page 93.

[121] G. Kunert. Toward anisotropic mesh construction and error estimation in the finite element method. Numerical Methods for Partial Differential Equations, 18(5):625–648, 2002. ISSN 1098-2426. doi: 10.1002/num.10023. Cited on page 182.

[122] P. Lailly. As a sequence of before stack migrations. In Conference on Inverse Scattering–Theory and Application, volume 11, page 206. Siam, 1983. Cited on pages 19, 26, and 46.

[123] V. Lekić and B. Romanowicz. Inferring upper-mantle structure by full waveform tomography with the spectral element method. Geophysical Journal International, 185(2):799–831, 2011. Cited on page 21.

[124] P. Lesaint and P. A. Raviart. On a Finite Element Method for Solving the Neutron Transport Equation. Publications mathématiques et informatique de Rennes, (S4):1–40, 1974. Cited on page 52.

[125] A. Logg, K.-A. Mardal, and G. Wells. Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book, volume 84. Springer Science & Business Media, 2012. Cited on page 20.

[126] C. C. Lopez. Speed up and Regularization Techniques for Seismic Full Waveform Inversion. PhD thesis, Université Nice Sophia Antipolis, Apr. 2014. Cited on pages 37 and 227.

[127] A. Loseille and R. Feuillet. Vizir: High-order mesh and solution visualization using OpenGL 4.0 graphic pipeline. In 2018 AIAA Aerospace Sciences Meeting. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2018-1174. Cited on page 160.

[128] J. Luo and R.-S. Wu. Envelope inversion–some application issues. In SEG Technical Program Expanded Abstracts 2013, pages 1042–1047. Society of Exploration Geophysicists, 2013. Cited on page 21.

[129] K. J. Marfurt. Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations. Geophysics, 49(5):533–549, 1984. Cited on page 20.

[130] E. D. Mercerat and N. Glinsky. A nodal high-order discontinuous Galerkin method for elastic wave propagation in arbitrary heterogeneous media. Geophysical Journal International, 201 (2):1101–1118, Mar. 2015. ISSN 1365-246X. doi: 10.1093/gji/ggv029. Cited on pages 93 and 94.

[131] L. Métivier, R. Brossier, J. Virieux, and S. Operto. Full Waveform Inversion and the Truncated Newton Method. SIAM Journal on Scientific Computing, 35(2):B401–B437, Jan. 2013. ISSN 1064-8275. doi: 10.1137/120877854. Cited on pages 25 and 40.

[132] L. Métivier, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux. Measuring the misfit between seismograms using an optimal transport distance: Application to full waveform inversion. Geophysical Journal International, 205(1):345–377, Apr. 2016a. ISSN 0956-540X, 1365-246X. doi: 10.1093/gji/ggw014. Cited on page 31.

[133] L. Métivier, R. Brossier, Q. Mérigot, E. Oudet, and J. Virieux. An optimal transport approach for seismic tomography: Application to 3D full waveform inversion. Inverse Problems, 32 (11):115008, Nov. 2016b. ISSN 0266-5611, 1361-6420. doi: 10.1088/0266-5611/32/11/115008. Cited on pages 31 and 227.

[134] Mmg Software. Dopagny Charles, Dobrzynsci Cécile, Frey Pascal, Froehly Algiane. Cited on pages 162, 181, 223, and 226.

[135] P. Moczo, J. Kristek, and M. Gális. The Finite-Difference Modelling of Earthquake Motions: Waves and Ruptures. Cambridge University Press, 2014. Cited on page 20.

[136] M. N'Diaye. Étude et Développement de Méthodes Numériques d'ordre Élevé Pour La Résolution Des Équations Différentielles Ordinaires (EDO) : Applications à La Résolution Des Équations d'ondes Acoustiques et Électromagnétiques. These de doctorat, Pau, Dec. 2017. Cited on pages 67, 68, and 203.

[137] T. Nissen-Meyer, M. van Driel, S. C. Stähler, K. Hosseini, S. Hempel, L. Auer, A. Colombi, and A. Fournier. AxiSEM: Broadband 3-D seismic wavefields in axisymmetric media. Solid Earth, 5(1):425–445, June 2014. ISSN 1869-9529. doi: 10.5194/se-5-425-2014. Cited on pages 20 and 206.

[138] J. Nocedal. Updating quasi-Newton matrices with limited storage. Mathematics of computation, 35(151):773–782, 1980. Cited on pages 40 and 46.

[139] J. Nocedal and S. Wright. Numerical Optimization. Springer Science & Business Media, Dec. 2006. ISBN 978-0-387-40065-5. Cited on pages 38, 40, 41, and 151.

[140] A. Nuttall. Some windows with very good sidelobe behavior. IEEE Transactions on Acoustics, Speech, and Signal Processing, 29(1):84–91, 1981. Cited on page 166.

[141] T. Ohminato and B. A. Chouet. A free-surface boundary condition for including 3D topography in the finite-difference method. Bulletin of the Seismological Society of America, 87 (2):494–515, 1997. Cited on page 20.

[142] S. Operto, J. Virieux, P. Amestoy, J.-Y. L'Excellent, L. Giraud, and H. B. H. Ali. 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study. Geophysics, 72(5):SM195–SM211, 2007. Cited on page 144.

[143] S. Operto, R. Brossier, L. Combe, L. Métivier, A. Ribodetti, and J. Virieux. Computationally efficient three-dimensional acoustic finite-difference frequency-domain seismic modeling in vertical transversely isotropic media with sparse direct solver. Geophysics, 79(5):T257–T275, 2014. Cited on pages 21 and 143.

[144] D. Peter, D. Komatitsch, Y. Luo, R. Martin, N. Le Goff, E. Casarotti, P. Le Loher, F. Magnoni, Q. Liu, C. Blitz, et al. Forward and adjoint simulations of seismic wave propagation on fully unstructured hexahedral meshes. Geophysical Journal International, 186(2):721–739, 2011. Cited on page 20.

[145] N. A. Petersson and B. Sjögreen. Wave propagation in anisotropic elastic materials and curvilinear coordinates using a summation-by-parts finite difference method. Journal of Computational Physics, 299:820–841, 2015. Cited on page 20.

[146] P. V. Petrov and G. A. Newman. Three-dimensional inverse modelling of damped elastic wave propagation in the Fourier domain. Geophysical Journal International, 198(3):1599–1617, July 2014. ISSN 1365-246X. doi: 10.1093/gji/ggu222. Cited on page 171.

[147] M. Pocock and S. Walker. Iterative solution of multiple right hand side matrix equations for frequency domain monostatic radar cross section calculations. Applied Computational Electromagnetics Society Journal, 13:4–13, 1998. Cited on pages 21 and 143.

[148] E. Polak and G. Ribiere. Note sur la convergence de méthodes de directions conjuguées. ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 3(R1):35–43, 1969. Cited on page 39.

[149] R. G. Pratt, Z.-M. Song, P. Williamson, and M. Warner. Two-dimensional velocity models from wide-angle seismic data by wavefield inversion. Geophysical Journal International, 124 (2):323–340, Feb. 1996. ISSN 0956-540X. doi: 10.1111/j.1365-246X.1996.tb07023.x. Cited on page 210.

[150] R. G. Pratt, C. Shin, and G. J. Hick. Gauss–Newton and full Newton methods in frequency–space seismic waveform inversion. Geophysical Journal International, 133(2):341–362, May 1998. ISSN 0956-540X. doi: 10.1046/j.1365-246X.1998.00498.x. Cited on page 37.

[151] S. Pyun, C. Shin, and J. B. Bednar. Comparison of waveform inversion, part 3: Amplitude approach. Geophysical Prospecting, 55(4):477–485, May 2007. ISSN 1365-2478. doi: 10.1111/j.1365-2478.2007.00619.x. Cited on page 30.

[152] W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479; CONF-730414-2, Los Alamos Scientific Lab., N.Mex. (USA), Oct. 1973. Cited on page 52.

[153] J.-F. Remacle, N. Chevaugeon, É. Marchandise, and C. Geuzaine. Efficient visualization of high-order finite elements. International Journal for Numerical Methods in Engineering, 69 (4):750–771, 2007. ISSN 1097-0207. doi: 10.1002/nme.1787. Cited on page 159.

[154] M. Rietmann, D. Peter, O. Schenk, B. Uçar, and M. Grote. Load-Balanced Local Time Stepping for Large-Scale Wave Propagation. In 2015 IEEE International Parallel and Distributed Processing Symposium, pages 925–935, May 2015. doi: 10.1109/IPDPS.2015.10. Cited on page 66.

[155] J. O. Robertsson. A numerical free-surface condition for elastic/viscoelastic finite-difference modeling in the presence of topography. Geophysics, 61(6):1921–1934, 1996. Cited on page 20.

[156] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. The Computer Journal, 3(3):175–184, Jan. 1960. ISSN 0010-4620. doi: 10.1093/comjnl/3.3.175. Cited on page 43.

[157] E. Schall, D. Leservoisier, A. Dervieux, and B. Koobus. Mesh adaptation as a tool for certified computational aerodynamics. International Journal for Numerical Methods in Fluids, 45:179–196, May 2004. doi: 10.1002/fld.642. Cited on page 179.

[158] A. Sei and W. W. Symes. Gradient calculation of the traveltime cost function without ray tracing. In SEG Technical Program Expanded Abstracts 1994, SEG Technical Program Expanded Abstracts, pages 1351–1354. Society of Exploration Geophysicists, Jan. 1994. doi: 10.1190/1.1822780. Cited on page 134.

[159] G. Seriani and E. Priolo. Spectral element method for acoustic wave simulation in heterogeneous media. Finite elements in analysis and design, 16(3-4):337–348, 1994. Cited on page 20.

[160] C. Shin and D.-J. Min. Waveform inversion using a logarithmic wavefield. Geophysics, 71(3):R31–R42, May 2006. ISSN 0016-8033. doi: 10.1190/1.2194523. Cited on page 31.

[161] C. Shin, S. Pyun, and J. B. Bednar. Comparison of waveform inversion, part 1: Conventional wavefield vs logarithmic wavefield. Geophysical Prospecting, 55(4):449–464, May 2007. ISSN 1365-2478. doi: 10.1111/j.1365-2478.2007.00617.x. Cited on page 30.

[162] C. Shin, N.-H. Koo, Y. H. Cha, and K.-P. Park. Sequentially ordered single-frequency 2-D acoustic waveform inversion in the Laplace–Fourier domain. Geophysical Journal International, 181(2):935–950, May 2010. ISSN 0956-540X. doi: 10.1111/j.1365-246X.2010.04540.x. Cited on page 171.

[163] J. Shin, W. Ha, H. Jun, D.-J. Min, and C. Shin. 3D Laplace-domain full waveform inversion using a single GPU card. Computers & Geosciences, 67:1–13, June 2014. ISSN 0098-3004. doi: 10.1016/j.cageo.2014.02.006. Cited on page 227.

[164] S. Shoja, S. Abolhassani, and N. Amini. A comparison between time-domain and frequency-domain full waveform inversion. In 80th EAGE Conference and Exhibition 2018, volume 2018, pages 1–3. European Association of Geoscientists & Engineers, 2018. Cited on page 143.

[165] J. Shragge. Solving the 3D acoustic wave equation on generalized structured meshes: A finite-difference time-domain approach. Geophysics, 79:P1–P16, Nov. 2014. doi: 10.1190/geo2014-0172.1. Cited on page 53.

[166] J. Shragge. Acoustic wave propagation in tilted transversely isotropic media: Incorporating topographyTopographic 3D TTI propagation. Geophysics, 81(5):C265–C278, 2016. Cited on page 20.

[167] H. Si. TetGen, a delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw., 41(2), Feb. 2015. ISSN 0098-3500. doi: 10.1145/2629697. Cited on page 162.

[168] S. Simutė, H. Steptoe, L. Cobden, A. Gokhberg, and A. Fichtner. Full-waveform inversion of the Japanese Islands region. Journal of Geophysical Research: Solid Earth, 121(5): 3722–3741, 2016. Cited on page 21.

[169] S. Singh, A. I. Kanli, and S. Mukhopadhyay. Full waveform inversion in time and frequency domain of velocity modeling in seismic imaging: FWISIMAT a Matlab code. Earth Sciences Research Journal, 22(4):291–300, 2018. Cited on pages 21 and 143.

[170] L. Sirgue and R. G. Pratt. Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies. Geophysics, 69(1):231–248, 2004. Cited on page 165.

[171] L. Sirgue, J. T. Etgen, U. Albertin, and S. Brandsberg-Dahl. System and method for 3D frequency domain waveform inversion based on 3D time-domain forward modeling. May 2010. Cited on pages 21 and 144.

[172] L. Sirgue, B. Denel, and F. Gao. Integrating 3D full waveform inversion into depth imaging projects. In SEG Technical Program Expanded Abstracts 2011, pages 2354–2358. Society of Exploration Geophysicists, 2011. Cited on page 21.

[173] B. Smith, P. Bjorstad, and W. Gropp. Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge university press, 2004. Cited on page 146.

[174] A. Szenicer, K. Leng, and T. Nissen-Meyer. A complexity-driven framework for waveform tomography with discrete adjoints. Geophysical Journal International, 223(1):1247–1264, May 2020. ISSN 1365-246X. doi: 10.1093/gji/ggaa349. Cited on page 140.

[175] J. Tago, V. M. Cruz-Atienza, J. Virieux, V. Etienne, and F. J. Sánchez-Sesma. A 3D hp-adaptive discontinuous Galerkin method for modeling earthquake dynamics. Journal of Geophysical Research: Solid Earth, 117(B9), 2012. Cited on page 20.

[176] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise. Benchmark Functions for the CEC'2010 Special Session and Competition on Large-Scale Global Optimization. page 24. Cited on page 43.

[177] A. Tarantola. Inversion of seismic reflection data in the acoustic approximation. Geophysics, 49(8):1259–1266, Aug. 1984. ISSN 0016-8033, 1942-2156. doi: 10.1190/1.1441754. Cited on pages 19, 26, and 46.

[178] A. Tarantola. Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation. Elsevier, Amsterdam, 1987. Cited on pages 31 and 37.

[179] S. Thrastarson, M. van Driel, L. Krischer, C. Boehm, M. Afanasiev, D.-P. van Herwaarden, and A. Fichtner. Accelerating numerical wave propagation by wavefield adapted meshes. Part II: Full-waveform inversion. Geophysical Journal International, 221(3):1591–1604, June 2020. ISSN 0956-540X, 1365-246X. doi: 10.1093/gji/ggaa065. Cited on page 179.

[180] F. Tröltzsch. Optimal Control of Partial Differential Equations: Theory, Methods, and Applications, volume 112. American Mathematical Soc., 2010. Cited on page 127.

[181] E. Turkel and A. Yefet. Absorbing PML boundary layers for wave-like equations. Applied Numerical Mathematics, 27:533–557, Aug. 1998. doi: 10.1016/S0168-9274(98)00026-9. Cited on page 51.

[182] M. van Driel, C. Boehm, L. Krischer, and M. Afanasiev. Accelerating numerical wave propagation using wavefield adapted meshes. Part I: Forward and adjoint modelling. Geophysical Journal International, 221(3):1580–1590, June 2020. ISSN 0956-540X. doi: 10.1093/gji/ggaa058. Cited on page 206.

[183] P. J. M. van Laarhoven and E. H. L. Aarts. Simulated annealing. In P. J. M. van Laarhoven and E. H. L. Aarts, editors, Simulated Annealing: Theory and Applications, Mathematics and Its Applications, pages 7–15. Springer Netherlands, Dordrecht, 1987. ISBN 978-94-015-7744-1. doi: 10.1007/978-94-015-7744-1_2. Cited on page 25.

[184] T. Van Leeuwen and F. J. Herrmann. Mitigating local minima in full-waveform inversion by expanding the search space. Geophysical Journal International, 195(1):661–667, 2013. Cited on page 21.

[185] 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. page 139. Cited on page 61.

[186] J. Virieux. SH-wave propagation in heterogeneous media: Velocity-stress finite-difference method. Geophysics, 49(11):1933–1942, 1984. Cited on page 20.

[187] J. Virieux. P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method. Geophysics, 51(4):889–901, 1986. Cited on page 20.

[188] J. Virieux and S. Operto. An overview of full-waveform inversion in exploration geophysics. https://www.researchgate.net/publication/228078264_An_overview_of_full-waveform_inversion_in_exploration_geophysics. Cited on pages 21, 152, 215, and 219.

[189] Z. Wang. GPU-Accelerated Discontinuous Galerkin Methods on Hybrid Meshes: Applications in Seismic Imaging. May 2017. Cited on page 140.

[190] M. Warner, I. Stekl, A. Umpleby, J. Morgan, C. Pain, and Y. Wang. 3D wavefield tomography: Problems, opportunities and future directions. In 70th EAGE Conference and Exhibition-Workshops and Fieldtrips, pages cp–41. European Association of Geoscientists & Engineers, 2008. Cited on page 144.

[191] M. Warner, A. Ratcliffe, T. Nangoo, J. Morgan, A. Umpleby, N. Shah, V. Vinje, I. Štekl, L. Guasch, C. Win, et al. Anisotropic 3D full-waveform inversion. Geophysics, 78(2): R59–R80, 2013. Cited on page 21.

[192] L. C. Wilcox, G. Stadler, C. Burstedde, and O. Ghattas. A high-order discontinuous Galerkin method for wave propagation through coupled elastic–acoustic media. Journal of Computational Physics, 229(24):9373–9396, Dec. 2010. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.09.008. Cited on page 20.

[193] L. C. Wilcox, G. Stadler, T. Bui-Thanh, and O. Ghattas. Discretely exact derivatives for hyperbolic PDE-constrained optimization problems discretized by the discontinuous Galerkin method. arXiv:1311.6900 [math], Nov. 2013. Cited on pages 135 and 140.

[194] R.-S. Wu, J. Luo, and B. Wu. Seismic envelope inversion and modulation signal model. Geophysics, 79(3):WA13–WA24, 2014. Cited on page 21.

[195] Y. Yang. Analysis and Application of Optimal Transport For Challenging Seismic Inverse Problems. arXiv:1902.01226 [math], Feb. 2019. Cited on pages 171 and 177.

[196] Y. Yang, B. Engquist, J. Sun, and B. F. Hamfeldt. Application of optimal transport and the quadratic Wasserstein metric to full-waveform inversion. Geophysics, 83(1):R43–R62, Oct. 2017. ISSN 0016-8033. doi: 10.1190/geo2016-0663.1. Cited on pages 31, 165, and 227.

[197] Y. Yunan. Optimal transport for seismic inverse problems. page 184, 2018. Cited on page 31.

[198] H.-W. Zhou, H. Hu, Z. Zou, Y. Wo, and O. Youn. Reverse time migration: A prospect of seismic imaging methodology. Earth-science reviews, 179:207–227, 2018. Cited on page 19.

[199] H. Zhu, E. Bozdağ, and J. Tromp. Seismic structure of the European upper mantle based on adjoint tomography. Geophysical Journal International, 201(1):18–52, 2015. Cited on page 21.

Abstract

Inversion par forme d'ondes complète en domaine temporel utilisant des méthodes de Galerkine Discontinues avancées.

Dans ce projet, nous avons développé des outils de reconstruction du sous-sol pour l'imagerie sismique ainsi que la caractérisation de réservoirs dans un contexte industriel. Pour ce faire, nous utilisons la méthode d'inversion par forme d'ondes complète (Full Waveform Inversion, FWI). Cette reconstruction emploie les données issues de perturbations sismiques qui génèrent des ondes dont le comportement est influencé par le milieu dans lequel elles se propagent. Dans le cadre de cette thèse, on considère des ondes acoustiques dont la simulation est mise en œuvre par des méthodes de Galerkine Discontinues. Ces dernières reposent sur une discrétisation en espace très flexible permettant d'approcher des modèles et des géométries complexes. Elles sont aussi parfaitement adaptées à une mise en œuvre dans un environnement de Calcul Haute-Performance. En effet cette technique permet une forte scalabilité grâce au faible coût des communications induites par le calcul des flux caractéristiques des éléments finis discontinus. Ici, l'équation d'onde est résolue en domaine temporel afin d'outrepasser les limitations en mémoire rencontrées en domaine fréquentiel pour la reconstruction de milieux industriels 3D de grandes échelles.

Pour reconstruire de manière quantitative le modèle physique étudié, nous avons formulé le problème inverse comme un problème de minimisation résolu par la méthode de l'état adjoint. Cette méthode permet d'obtenir le gradient de la fonction coût par rapport aux paramètres physiques au prix de deux simulations ; celle du problème direct et celle du problème rétro-propagé aussi appelé problème adjoint. L'état adjoint est solution du problème adjoint continu discrétisé ("*Optimiser Puis Discrétiser*"). Ce choix est justifié par une comparaison 1D avec la stratégie qui consiste à "*Discrétiser puis Optimiser*" complété par une étude algébrique en dimension supérieure. Le gradient ainsi calculé s'inclut dans une procédure d'optimisation développée et intégrée au code industriel fourni par le partenaire industriel, Total.

Le propagateur joue un rôle central dans la résolution du problème inverse. En effet, cette dernière met en jeu une méthode itérative dont chaque itération implique des résolutions successives du problème direct. Il est alors important de tirer parti au mieux de la discrétisation de Galerkine Discontinue. Dans cette thèse, nous avons notamment étudié le choix de la base polynomiale d'approximation (Legendre ou Bernstein-Bézier) ainsi que le choix de la paramétrisation qui peut être constante par élément ou variable grâce à l'utilisation de la méthode de Galerkine Discontinue à Pondération Ajustée (Weight Adjusted Discontinuous Galerkin, WADG). Cette dernière stratégie offre l'occasion d'élargir les cellules du maillage sans perdre d'information sur le modèle et permet donc une utilisation plus poussée de la *hp*-adaptivité qu'on proposera d'exploiter pleinement grâce à un maillage adaptatif s'ajustant au modèle qui évolue avec les itérations du problème inverse.

Abstract

Time-Domain Full Waveform Inversion using advanced Discontinuous Galerkin methods.

In this project, we developed tools for the reconstruction of subsurface media for seismic imaging and reservoir characterization in an industrial context. For that purpose, we used the Full Waveform Inversion (FWI) method. It is a reconstruction technique using data taken from seismic disturbances and whose behavior reflects the properties of the environment in which they propagate. In the framework of this thesis, we consider acoustic waves which are simulated thanks to Discontinuous Galerkin methods. These methods offer a very flexible discretization in space allowing to approach complex models and geometries. Discontinuous Galerkin methods are characterized by the use of fluxes in between each cell. Those fluxes contribute to have low communication costs which are highly recommended for High Performance Computing. Here, the wave equation is solved in time domain to overcome the memory limitations encountered in frequency domain for the reconstruction of large-scale 3D industrial media.

To reconstruct quantitatively the physical model under study, we wrote the inverse problem as a minimization problem solved by adjoint state method. This method makes it possible to obtain the gradient of the cost function with respect to the physical parameters for the cost of two simulations; the direct problem and the backward problem also called adjoint problem. The adjoint state will be the solution of the discretized continuous adjoint problem ("*Optimize Then Discretize*"). This choice is justified by a 1D comparison with the strategy which consists in "*Discretize then Optimize*" completed by an algebraic study in superior dimension. The gradient thus calculated, is a key in the optimization procedure developed and integrated in the industrial environment provided by the industrial partner, Total.

The propagator is a keystone in solving the inverse problem. Indeed, it is repeated successively and represents the majority of the computation time of the optimization process. It is therefore important to control the discretization by the Discontinuous Galerkin method as well as possible. In particular, in this thesis, we have considered the idea of using different polynomial bases of approximation (Legendre or Bernstein-Bézier) as well as the choice of the parameterization, which can either be constant per element or variable thanks to the use of the Weight Adjusted Discontinuous Galerkin (WADG) method. This last strategy offers the opportunity to enlarge the mesh cells without losing information on the model and thus allows a more advanced use of the *hp*-adaptivity that we propose to fully exploit thanks to an adaptive mesh that is adjusted to the model meant to evolve with the iterations of the inverse problem.