POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Matematica

High order discontinuous Galerkin methods on simplicial elements for the elastodynamics equation

Relatori:
Dr. Paola F. Antonietti
Prof. Alfio Quarteroni

Correlatore:
Dr. Ilario Mazzieri

Elaborato di laurea di:
Carlo Marcati
Matr. 769937

Anno Accademico 2012-2013

Contents

List of Tables

List of Figures

Abstract

High order methods for elastic wave propagation phenomena in seismic regions have seen a great development in the last years. Specifically, the spectral element (SE) method combines the high accuracy of spectral methods with the flexibility of finite element methods (FEM). SE methods have been introduced and are currently built mainly on mashes made of tensor product elements (i.e., deformed squares and cubes). In this work we consider the application of the discontinuous spectral element method on meshes made of simplicial elements to the approximation of the elastodynamics equation. This approach provides the high accuracy of spectral methods, combined with the geometrical flexibility of simplex elements and with the computational scalability of Discontinuous Galerkin (DG) methods. We thoroughly analyze the dissipation, dispersion and stability properties of the scheme introduced, with a focus on the choice of the basis functions. Finally, we test the method on benchmark test cases.

Sommario

Negli ultimi anni, vi è stato un grande aumento sia nella richiesta che nell'avanzamento teorico dei metodi di alto ordine per l'approssimazione di fenomeni di propagazione di onde sismiche. Fra i metodi di alto ordine si distingue il metodo agli elementi spettrali, che combina l'alto ordine di accuratezza dei metodi spettrali con la flessibilità dei metodi agli elementi finiti. Le griglie computazionali su cui i metodi agli elementi spettrali sono stati introdotti (e su cui vengono usualmente implementati) sono composte da elementi formati tramite prodotto tensoriale di intervalli monodimensionali, cioè quadrati e cubi deformati. In questo lavoro si considera un metodo agli elementi spettrali discontinuo, basato su griglie composte di triangoli o tetraedri, nella sua applicazione all'approssimazione dell'equazione dell'elastodinamica. Questo permette di ottenere un metodo che offre sia l'alto ordine dei metodi agli elementi spettrali che la flessibilità geometrica e l'h-adattabilità fornita dagli elementi triangolari. Inoltre, l'uso di un metodo di Galerkin discontinuo fornisce un approccio computazionalmente scalabile e P-adattabile. Si è quindi provveduto ad analizzare in dettaglio le proprietà di dissipazione, dispersione e stabilità del metodo introdotto, considerando le possibili scelte per quanto le riguarda le funzioni di base. Lo schema numerico è stato infine provato su diversi casi di riferimento per la simulazione della propagazione di onde sismiche.

Introduction

The analysis of elastic and acoustic wave propagation phenomena has been widely studied by mathematicians and scientists since the XIX century. Elastic waves in solids have also been historically studied, though analytic solutions are available only for simple domains and settings. The approximation of the solution to the elastodynamics equation is therefore of critical importance for the analysis of the propagation of seismic waves in complex scenarios.

In recent years, seismological, geophysical and technological advances have allowed for a greater insight into physical seismological events and this has contributed to the growth of the demand for accurate and flexible numerical methods. Those, indeed, permit the comparison between the empirical observations and accurate numerical wave fields in complex domains.

In this work we consider the application of the discontinuous spectral element method on meshes made of simplicial elements to the approximation of the elastodynamic equation. Spectral element methods were introduced in the computational fluid dynamics field [Pat84, MP89] and they combine the high order accuracy of spectral methods with the flexibility and computational feasibility of finite elements methods. They are strictly related with the h-p version of the finite element method and they have been extensively used for computational geodynamics in the last two decades [KV98, KT02]. Spectral element methods have been introduced and are currently built on mashes made of tensor product elements (i.e. deformed squares and cubes), since this is the context in which the extension from one spatial dimension to d dimensions, $d = 2, 3$, is more straightforward. Spectral elements on triangles and tetrahedra have been historically less widely studied, though different formulations (modal with different bases [KS05], nodal with different interpolation nodes [HW08]) have been proposed and analyzed in the last years. Several of these formulations have been employed in geodynamical applications [MVSS06, PdlPA$^+$12], but a thorough analysis has not been carried out for the coupling of modal bases and discontinuous methods. In general, they provide the flexibility and geometrical adaptability of simplicial mesh combined with the high order of spectral methods.

Discontinuous methods were introduced for hyperbolic equations, have been extended to to the elliptic case and developed independently in both environments. The advantages of discontinuous methods lies in the fact that they allow for the accurate approximation of sharp gradients in the solution, that they can be used to develop an h-p adaptive strategy and that the computational cost can be distributed without much overhead.

Of the spectral bases on triangular elements presented, one (the Legendre-Dubiner basis [Dub91, Koo75]) can be used only in the framework of a fully discontinuous approximation, since there is no way to enforce the continuity of the space between neighboring elements. The boundary adapted basis functions [KS05, Dub91], on the other hand, are modified in order to be used in a continuous scheme. Therefore, a non-conforming scheme as in [AMQR12] is possible, and the following analysis helps to understand the properties of such a scheme.

In Chapter 1 the global setting and the elastodynamics equation are presented. We pay particular attention to the illustration of the body and surface waves, which will be extremely important in the forthcoming analysis of the numerical results. The variational formulation of the equation is presented, since it is the natural formulation for finite element methods.

The discontinuous Galerkin approximation is introduced in Chapter 2. Specifically, we consider the interior penalty method, in its symmetric, non-symmetric and incomplete variants. The introduction of a basis for the finite-dimensional approximation space leads us to the development of the semi-discrete algebraic formulation.

The various possibilities for the expansion basis are presented in Chapter 3, along with a full treatment of the details of high order methods on simplicial elements. We consider the orthonormal Legendre-Dubiner basis and a non orthogonal basis modified in order to allow for a continuous approximation. Those bases have their respective advantages and drawbacks, and the numerical experiments will be carried out for both. We also introduce, alongside with classical Gaussian quadrature rules, a more natural set of quadrature rules, whose derivation takes into account the symmetries of the triangle.

Two of the classical time stepping methods for the simulation of wave propagation phenomena are introduced in Chapter 4, while Chapter 5 deals with the practical implementation of a numerical spectral element code. In particular, the role of the reference triangle is considered, and global and local (to every element) operations are presented.

The tools listed so far are thoroughly analyzed in Chapter 6, where the dissipation, dispersion and stability properties of the method are considered. We ideally simulate a plane wave on an infinite domain, and, via an analysis of the generalized eigenvalues of the method's matrices, we get estimates on the quality of the approximation provided by the scheme. We investigate also how much restrictive are the bounds to the time step, imposed to explicit time stepping methods.

The analysis of the methods proceeds in Chapter 7, where the convergence of the numerical solution to a smooth exact solution is taken into account. The order of the method with respect to the time step and to the polynomial approximation order is computed.

In the last chapter, the method is applied to a series of test cases and benchmarks. We show its capacity to accurately model complex wave propagation phenomena as well as its performance in benchmark tests. Finally, we draw some conclusions and we outline some future perspectives.

In Appendix A a C++ library which has been developed and implements the method is presented, and its application to some of the test cases is analyzed.

Chapter 1

Seismic waves and the elastodynamics equation

The goal of the numerical study of seismic waves is to analyze their behavior and how they propagate inside bodies with different physical properties. Seismic waves, often resulting from an underground rupture or other geological events, travel at thousands of meters per second through the interior of the Earth and on the surface of the crust, often reaching great lengths before their effect fades. Here we will present the elastodynamics equation, which models the displacement induced by traveling seismic waves and we will analyze in detail some of the simple solutions to the equation, which constitute building blocks for the approximation in more complex scenarios.

1.1 The mathematical model

Let Ω be a sufficiently smooth finite region of $\mathbb{R}^d$, $d = 2, 3$. Let $\Gamma = \partial\Omega$ be its boundary, subdivided into a part where Dirichlet (i.e. displacement) conditions are prescribed, Γ_D and a part where Neumann (i.e. stress) conditions are imposed, Γ_N. We suppose that those regions are not overlapping, i.e. $\Gamma_D \cap \Gamma_N = \emptyset$, and Γ_N can be empty. Given a final time $T > 0$, the elastodynamics equation takes the form

$$\begin{cases} \rho\mathbf{u}_{tt} - \nabla\cdot\sigma(\mathbf{u}) = \mathbf{f}, & \text{in } \Omega\times[0,T], \\ \mathbf{u} = \mathbf{g}, & \text{on } \Gamma_D\times[0,T], \\ \sigma(\mathbf{u})\cdot\mathbf{n} = \mathbf{d}, & \text{on } \Gamma_N\times[0,T], \\ \mathbf{u}_t(0) = \mathbf{v_0}, & \text{in } \Omega, \\ \mathbf{u}(0) = \mathbf{u_0}, & \text{in } \Omega. \end{cases} \tag{1.1}$$

Here, σ is the stress tensor and will be defined later, ρ is the medium mass density and $\mathbf{g}$ is a boundary data (often assumed to be null in practical applications). If we define the strain tensor as the symmetrization of the gradient of the displacement

$$\varepsilon(\mathbf{u}) = \frac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^\top\right),$$

then the stress tensor for a continuous medium, considered in the regime of linear elasticity, is prescribed by the *Hooke's law*, that in components is given by

$$\sigma_{ij}(\mathbf{u}) = \mathbb{C}_{ijkl}\varepsilon_{kl}(\mathbf{u}) \tag{1.2}$$

where $\mathbb{C}$ is a fourth order tensor. Under the assumption of isotropy, relation (1.2) takes the form

$$\sigma_{ij}(\mathbf{u}) = \lambda\varepsilon_{kk}(\mathbf{u})\delta_{ij} + 2\mu\varepsilon_{ij}(\mathbf{u}) \tag{1.3}$$

where λ and μ are referred to as the first and second Lamé parameters. Those parameters define the elastic properties of the medium and the speed of the waves traveling inside it. They are constant in a homogeneous medium, while in a heterogeneous medium they can be assumed piecewise constants, i.e. $\lambda,\mu\in L^\infty(\Omega)$; the same goes for the stress tensor. Note that, if $\mu > 0$ and $\lambda \geq 0$, $\mathbb{C}$ is bounded:

$$2\mu x_{ij}x_{ij} \leq x_{ij}\mathbb{C}_{ijkl}x_{kl} \leq 2(\lambda+\mu)x_{ij}x_{ij} \quad \forall x\in\mathbb{R}^{d\times d}. \tag{1.4}$$

Finally, (1.3) can be rewritten, in a more practical manner, considering that the stress and strain tensors are symmetric,

$$\begin{pmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{pmatrix} = \begin{pmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & 2\mu \end{pmatrix} \begin{pmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{pmatrix}$$

for $d = 2$, and similarly for $d = 3$.

1.2 Seismic waves

There are two main types of seismic waves: body waves and surface waves. Specifically, body waves (which, as the name suggests, travel through the medium) are themselves divided into compressional (longitudinal, primary or P–) waves and shear (transverse, secondary or S–) waves. Longitudinal waves travel faster than shear ones, and they can travel through any type of medium, while S–waves propagation is limited to solid materials. Surface waves (again, the name is quite expressive) are of a somewhat more mixed nature, slower than both P– and S–waves and can be extremely disruptive in earthquake scenarios. Rayleigh and Love waves are surface waves.

1.2.1 Body waves

Let us consider the first equation of (1.1) inside an unbounded domain and suppose that the solution $\mathbf{u}$ is sufficiently smooth and that the force is null. Then, it can be rewritten as

$$\rho \mathbf{u}_{tt} - (\mu \Delta \mathbf{u} + (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u})) = \mathbf{0}. \tag{1.5}$$

Taking the divergence yields

$$\frac{\partial^2}{\partial t^2} (\nabla \cdot \mathbf{u}) = \frac{\lambda + 2\mu}{\rho} \Delta (\nabla \cdot \mathbf{u}),$$

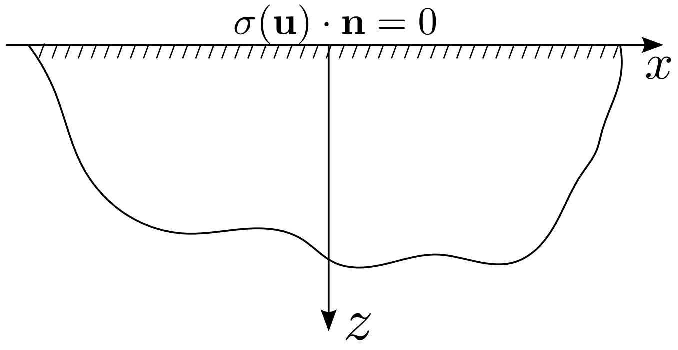


Figure 1.1: Domain for Rayleigh wave propagation

that is, a traveling deformation with speed

$$c_P = \sqrt{\frac{\lambda + 2\mu}{\rho}}. \tag{1.6}$$

Relation (1.6) is the speed of compressional waves. If we suppose $\nabla \cdot u = 0$ in (1.5), we get

$$\mathbf{u}_{tt} = \frac{\mu}{\rho} \Delta \mathbf{u}$$

which is again a traveling wave with speed

$$c_S = \sqrt{\frac{\mu}{\rho}}. \tag{1.7}$$

This is more meaningful in view of the Helmholtz decomposition theorem, since the first traveling wave can be derived as the irrotational part of $\mathbf{u}$, while the secondary wave is the divergence-free part.

1.2.2 Rayleigh waves

Let Ω be the semi-infinite domain in Figure 1.1 and let ψ and φ be respectively a divergence-free and irrotational function, with

$$\mathbf{u} = \nabla \psi + \nabla \times \boldsymbol{\varphi}, \qquad \boldsymbol{\varphi} = -\varphi \mathbf{j},$$

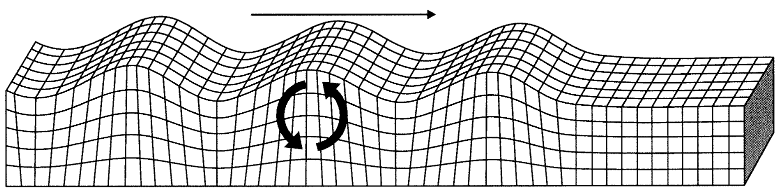


Figure 1.2: Qualitative displacement induced by a Rayleigh wave

a plane wave traveling in the positive x direction. Then, proceeding as in Section 1.2.1, we obtain

$$\begin{aligned} \psi_{tt} &= c_S^2 \Delta\psi \\ \varphi_{tt} &= c_P^2 \Delta\varphi. \end{aligned} \tag{1.8}$$

Then, we can write φ and ψ in the general form

$$\begin{aligned} \psi &= F(z)e^{i(\omega t - kx)} \\ \varphi &= G(z)e^{i(\omega t - kx)} \end{aligned}$$

so that φ and ψ are waves traveling with the velocity $c_{\text{Surf}} = \omega/k$. Substituting into (1.8) and taking into account the boundedness of the amplitude yields

$$\begin{aligned} \psi &= Ae^{-sz+i(\omega t-kx)}, \qquad s^2 = k^2 - \frac{\omega^2}{c_S^2}, \\ \varphi &= Be^{-qz+i(\omega t-kx)}, \qquad q^2 = k^2 - \frac{\omega^2}{c_P^2}. \end{aligned}$$

Imposing the Neumann boundary condition at the top edge of the domain we can obtain the following equation for the velocity of the wave:

$$k_s^6 - 8k_s^4 + (24 - 16\tilde{c}^2)k_s^2 + (16\tilde{c}^2 - 16) = 0, \qquad k_s^2 = \frac{c_{\text{Surf}}^2}{c_S^2}, \qquad \tilde{c}^2 = \frac{c_S^2}{c_P^2}. \tag{1.9}$$

An explicit solution for the traveling surface wave can therefore be derived, using the relations introduced before; the displacement induced by Rayleigh waves is reported in Figure 1.2.

1.3 The variational formulation

By multiplying equation (1.1) by a sufficiently regular test function $\mathbf{v}$ we obtain

$$(\rho \mathbf{u}_{tt}, \mathbf{v})_\Omega - (\nabla \cdot \sigma(\mathbf{u}), \mathbf{v})_\Omega = (\mathbf{f}, \mathbf{v})_\Omega$$

where $(\mathbf{u}, \mathbf{v})_\Omega = \int_\Omega \mathbf{u} \cdot \mathbf{v}$. Then, since

$$-(\nabla \cdot \sigma(\mathbf{u}), \mathbf{v})_\Omega = (\sigma(\mathbf{u}), \varepsilon(\mathbf{v}))_\Omega - (\sigma(\mathbf{u}) \cdot \mathbf{n}, \mathbf{v})_\Omega,$$

it is possible to derive the variational (or weak) formulation of the model. Let V be a suitable functional space that we will choose later: the variational formulation reads

$$\begin{aligned} &\text{find } \mathbf{u} = \mathbf{u}(t) \in V \text{ such that, } \forall t \in (0, T], \\ &\qquad \frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbf{u}(t), \mathbf{v})_\Omega + \mathcal{A}(\mathbf{u}(t), \mathbf{v}) = \mathcal{F}(\mathbf{v}), \\ &\qquad \mathbf{u}(0) = \mathbf{u}_0, \\ &\qquad \mathbf{u}_t(0) = \mathbf{v}_0, \end{aligned} \tag{1.10}$$

$\forall\, \mathbf{v} \in V$. Here, $\mathcal{A}(\cdot, \cdot) : V \times V \to \mathbb{R}$ is the bilinear form

$$\mathcal{A}(\mathbf{u}, \mathbf{v}) = (\sigma(\mathbf{u}), \varepsilon(\mathbf{v}))_\Omega$$

and $\mathcal{F} : V \to \mathbb{R}$ is a linear functional taking into account the body force and the boundary conditions (Dirichlet conditions are assumed homogeneous):

$$\mathcal{F}(\mathbf{v}) = (\mathbf{f}, \mathbf{v})_\Omega + (\mathbf{d}, \mathbf{v})_{\Gamma_N}.$$

Let then $V = \left[H^2_{\Gamma_\mathrm{D}}(\Omega)\right]^d := \{\mathbf{v} \in [H^2(\Omega)]^d : \mathbf{v}_{|\Gamma_\mathrm{D}} = \mathbf{0}\}$. From now on, d will be omitted: vectors in a space W will implicitly be assumed in W^d. Problem (1.10) admits a unique solution $\in \mathcal{C}^0(0, T; V) \cap \mathcal{C}^1(0, T; L^2(\Omega))$, provided that $\mathbf{u}_0 \in V$, $\mathbf{v}_0 \in L^2(\Omega)$ and $\mathbf{f} \in L^2(\Omega \times [0, T])$ and that the parameters ρ, λ and μ are in $L^\infty(\Omega)$.

1.3.1 Semi-discrete variational formulation

Problem (1.10) leads to a semi-discrete approximation by taking a finite-dimensional space V_h and looking for a solution $u_h(t) \in V_h$ such that, $\forall t$

$$\begin{aligned} \frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbf{u}_h(t), \mathbf{v}_h)_\Omega + \mathcal{A}(\mathbf{u}_h(t), \mathbf{v}_h) &= \mathcal{F}(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in V_h, \\ \mathbf{u}(0) &= \Pi \mathbf{u}_0, \\ \mathbf{u}_t(0) &= \Pi \mathbf{v}_0, \end{aligned} \tag{1.11}$$

where Π is a suitable projection operator in V_h. Frequently, in practice, the bilinear form $\mathcal{A}$, the linear functional $\mathcal{F}$ and the scalar product are themselves modified to adapt to practical or theoretical needs (e.g. because of numerical quadrature, discontinuous approximations or stabilizations), so that the first equation in (1.11) reads

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbf{u}_h(t), \mathbf{v}_h)_{\Omega,h} + \mathcal{A}_h(\mathbf{u}_h(t), \mathbf{v}_h) = \mathcal{F}_h(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in V_h. \tag{1.12}$$

In the following chapters, the construction of such finite-dimensional space will be investigated, together with the formulation of a suitable “discrete” bilinear form $\mathcal{A}_h$, of a linear functional $\mathcal{F}_h$ and of a discrete scalar product. Additional emphasis will be put on the identification of an efficient basis for V_h and the set of quadrature rules used for computing the integrals defined in (1.12).

Chapter 2

Discontinuous Galerkin methods

Discontinuous Galerkin (DG) methods for the solution of partial differential equations have been greatly developed and extended since they were first introduced in the seventies for the numerical approximation of hyperbolic problems [RH73] and, independently, in the context of elliptic and parabolic equations [Arn82, DD76]. Gathering ideas from both the finite elements and the finite volumes methods, they exhibit many interesting properties and they are highly scalable and flexible.

On the one hand, since many computations take place at the element basis, discontinuous Galerkin methods are prone to being implemented not only on clusters of CPUs, but also in highly parallel environments such as GPUs [KWBH09]. On the other hand, DG methods can handle naturally non-matching grids and variable approximation orders. Such flexibility has been successfully exploited in conjunction with spectral elements methods (SEM) in the context of seismic wave propagation phenomena [AMQR12].

DG methods for elliptic operators can be presented under a unified approach [ABCM02]; hereafter the analysis will be more focused on interior penalty methods [Arn82], in their symmetric, non-symmetric and incomplete variants.

2.1 Interior penalty methods

Consider a triangulation $\mathcal{T}_h$ made of d-simplices of the domain Ω such that $K_i \cap K_j = \emptyset$ if $i \neq j$, for all $K_{i,j} \in \mathcal{T}_h$. The set of all edges is

$$\mathcal{E} = \bigcup_k e_k$$

where $e_{k(i,j)} = \partial K_i \cap \partial K_j$ for two adjacent triangles K_i and K_j and an injective function k. The set of all internal edges will be denoted as $\mathcal{E}_I = \mathcal{E} \setminus \partial\Omega$. The function space on which DG methods are based relaxes the continuity requirements between the elements of the triangulation typical of classical finite elements. The functions are therefore globally required to be only in L^2. I.e., the finite-dimensional approximation space introduced in Section 1.3.1 is defined as

$$V_h = \{\mathbf{u} \in L^2(\Omega) : \mathbf{u}_{|K} \in \mathbb{P}_{N_K}(K)\}. \tag{2.1}$$

The polynomial order $N_K \geq 0$ can vary elementwise, since DG methods are naturally p-adaptable. Before proceeding to the full formulation of interior penalty methods, it is necessary to introduce the average $\{\!\{\cdot\}\!\}$ and jump $[\![\cdot]\!]$ operators for vectors and tensors

$$\begin{aligned}
[\![\mathbf{u}]\!] &= \mathbf{u}^+ \otimes \mathbf{n}^+ + \mathbf{u}^- \otimes \mathbf{n}^-, \\
[\![\sigma]\!] &= \sigma^+\mathbf{n}^+ + \sigma^-\mathbf{n}^-, \\
\{\!\{\mathbf{u}\}\!\} &= \frac{1}{2}(\mathbf{u}^+ + \mathbf{u}^-), \\
\{\!\{\sigma\}\!\} &= \frac{1}{2}(\sigma^+ + \sigma^-).
\end{aligned} \tag{2.2}$$

Those operators are defined on every $e \in \mathcal{E}_I$ and $\mathbf{u}^\pm$ are the restrictions of $\mathbf{u}$ to $\partial K^\pm$, where $K^\pm$ are two neighboring elements. $\mathbf{n}^\pm$ are the normals pointing outwards from the elements $K^\pm$. The operators are invariant to the choice of the label of the elements. If the edge lies on the boundary, one can simply define

$$\begin{aligned}
\{\!\{\mathbf{u}\}\!\} &= \mathbf{u}, & [\![\mathbf{u}]\!] &= \mathbf{u} \otimes \mathbf{n}, \\
\{\!\{\sigma(\mathbf{u})\}\!\} &= \sigma(\mathbf{u}), & [\![\sigma(\mathbf{u})]\!] &= \sigma(\mathbf{u})\mathbf{n}.
\end{aligned}$$

Writing (1.1) elementwise, integrating by parts and summing over all $K \in \mathcal{T}_h$ we obtain

$$\sum_{K\in\mathcal{T}_h} \frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbf{u},\mathbf{v})_K - (\nabla\cdot\sigma(\mathbf{u}),\mathbf{v})_K = \sum_{K\in\mathcal{T}_h} \frac{\mathrm{d}^2}{\mathrm{d}t^2}(\mathbf{u},\mathbf{v})_K + (\sigma(\mathbf{u}),\varepsilon(\mathbf{v}))_K - (\sigma(\mathbf{u})\cdot\mathbf{n},\mathbf{v})_{\partial K}.$$

Exploiting the operators defined in (2.2) and through algebraic manipulations we can write

$$-\sum_{K\in\mathcal{T}_h}(\nabla\cdot\sigma(\mathbf{u}),\mathbf{v})_K = \sum_{K\in\mathcal{T}_h}(\sigma(\mathbf{u}),\varepsilon(\mathbf{v}))_K - \sum_{e\in\mathcal{E}}(\{\!\{\sigma(\mathbf{u})\}\!\},[\![\mathbf{v}]\!])_e - (\{\!\{\mathbf{v}\}\!\},[\![\sigma(\mathbf{u})]\!])_e. \tag{2.3}$$

Since both the exact solution $\mathbf{u}$ and its stress tensor and $\sigma(\mathbf{u})$ have null jumps on the edges, we remove from the formulation the second term of the right hand side in (2.3). Furthermore, consistency is not affected by adding a term

$$-\vartheta\sum_{e\in\mathcal{E}}(\{\!\{\sigma(\mathbf{v})\}\!\},[\![\mathbf{u}]\!])_e$$

which can be used to construct a symmetric or non-symmetric method. Analogously, we add the following term to the bilinear form in order to penalize discontinuities in the solution

$$\mathcal{S}_e(\mathbf{u},\mathbf{v}) = \alpha_e(c_e([\![\mathbf{u}]\!]),[\![\mathbf{v}]\!])_e \tag{2.4}$$

where the linear function c_e and the parameter $\alpha_e \in \mathbb{R}^+$ will be specified later. The bilinear form $\mathcal{A}_h$, introduced in Section 1.3.1, in the case of interior penalty methods therefore reads

$$\mathcal{A}_h(\mathbf{u},\mathbf{v}) = \sum_{K\in\mathcal{T}_h}\mathcal{A}_K(\mathbf{u},\mathbf{v}) + \sum_{e\in\mathcal{E}}\mathcal{I}_e(\mathbf{u},\mathbf{v}) + \sum_{e\in\mathcal{E}_I}\mathcal{S}_e(\mathbf{u},\mathbf{v}), \tag{2.5}$$

where $\mathcal{A}_K$ is the restriction of $\mathcal{A}$ to the element K, i.e.,

$$\mathcal{A}_K(\mathbf{u},\mathbf{v}) = (\sigma(\mathbf{u}),\varepsilon(\mathbf{v}))_K,$$

and

$$\mathcal{I}_e(\mathbf{u},\mathbf{v}) = -\left(\{\!\{\sigma(\mathbf{u})\}\!\}, [\![\mathbf{v}]\!]\right)_e - \vartheta(\{\!\{\sigma(\mathbf{v})\}\!\}, [\![\mathbf{u}]\!])_e. \tag{2.6}$$

Setting $\vartheta = 1$ gives the *symmetric* (SIPG) methods [DD76], while $\vartheta = -1$ gives the *non-symmetric* (NIPG) method [RWG99] and $\vartheta = 0$ gives the *incomplete* (IIPG) interior penalty method. Formulation (1.12) for this class of methods thus reads: find $\mathbf{u} \in V_h$ such that

$$\frac{\mathrm{d}^2}{\mathrm{d}t^2} \sum_{K\in\mathcal{T}_h} (\mathbf{u}_h(t), \mathbf{v}_h)_K + \mathcal{A}_h(\mathbf{u}_h(t), \mathbf{v}_h) = \sum_{K\in\mathcal{T}_h} \mathcal{F}_K(\mathbf{v}_h) \quad \forall \mathbf{v}_h \in V_h. \tag{2.7}$$

2.1.1 The penalty function

The penalty function in equation (2.4) is defined over every edge e as

$$\alpha_e = \alpha \frac{N_e^2}{h_e} \tag{2.8}$$

where $\alpha \in \mathbb{R}^+$ and N_e and h_e are the degree and the length associated to the edge e. Typically they take the values

$$N_e = \max(N_{K^+}, N_{K^-}), \qquad h_e = |e|.$$

The choice of the function $c : \mathbb{R}^{d\times d} \to \mathbb{R}^{d\times d}$ in (2.4) is a little more controversial. Let us define the harmonic average on the edge e by

$$\overline{a}^A = \frac{2a^+a^-}{a^+ + a^-}.$$

Note that for dimensional reasons the function c must have a relation with the tensor $\mathbb{C}$. The easiest choice is therefore to set

$$c([\![\mathbf{u}]\!]) = \overline{\lambda + 2\mu}^A [\![\mathbf{u}]\!]. \tag{2.9}$$

Another possibility is to use the full tensor in the stability form (2.4), that is

$$c([\![\mathbf{u}]\!]) = \overline{\mathbb{C}}^A [\![\mathbf{u}]\!], \tag{2.10}$$

where the harmonic average is computed component-wise.

2.2 Continuity and coercivity

The main references for this section are [ABCM02] and [AMQR12]. In order to provide an analysis of the discontinuous method (2.7), let us define the mesh dependent norm

$$||\mathbf{u}_h||_{DG}^2 = \sum_{K\in\mathcal{T}_h} ||\mathbb{C}^{1/2}\varepsilon(\mathbf{u}_h)||_{0,K}^2 + \sum_{e\in\mathcal{E}} \alpha_e ||[\![\mathbf{u}_h]\!]||_{0,e}^2, \quad \mathbf{u}_h \in V_h. \tag{2.11}$$

The following result guarantees that problem (2.7) is well posed.

Lemma 2.2.1. *The coercivity and continuity bounds*

$$\mathcal{A}_h(\mathbf{u},\mathbf{v}) \leq M||\mathbf{u}||_{DG}^2||\mathbf{v}||_{DG}^2, \qquad \forall \mathbf{u},\mathbf{v}\in V_h \tag{2.12}$$

$$\mathcal{A}_h(\mathbf{u},\mathbf{u}) \geq \eta||\mathbf{u}||_{DG}^2, \qquad \forall \mathbf{u}\in V_h \tag{2.13}$$

hold for some constants $\eta, M > 0$, provided that $\alpha \geq \alpha_{min} > 0$, with α defined in (2.8).

Before stating the proof of Lemma 2.2.1, we recall the inequalities ([AH11], see also [WH03])

$$||\mathbf{u}||_{0,e}^2 \leq CN_K^2 h_e^{-1}||\mathbf{u}||_{0,K}^2 \qquad \left|\left|\frac{\partial \mathbf{u}}{\partial \mathbf{n}}\right|\right|_{0,e}^2 \leq CN_K^2 h_e^{-1}|\mathbf{u}|_{1,K}^2 \tag{2.14}$$

valid for a $\mathbf{u}$ of degree N_K on K, $e\in\partial K$ and for a positive constant C.

Proof. We first show (2.12). Let us analyze the terms of $\mathcal{A}_h$ separately. By Cauchy-Schwartz inequality we have

$$\sum_K (\sigma(\mathbf{u}),\varepsilon(\mathbf{v}))_K \leq C\left(\sum_K ||\mathbb{C}^{1/2}\varepsilon(\mathbf{u})||^2\right)^{1/2}\left(\sum_K ||\mathbb{C}^{1/2}\varepsilon(\mathbf{v})||^2\right)^{1/2}.$$

Next, using the second of the (2.14) we get

$$\begin{aligned}\sum_e (\{\!\{\sigma(\mathbf{u})\}\!\}, [\![\mathbf{v}]\!])_e &\leq \sum_e \frac{1}{\sqrt{\alpha_e}} ||\sigma(\mathbf{u})||_e \sqrt{\alpha_e} ||[\![\mathbf{v}]\!]||_e \\ &\leq \left(\sum_K C\alpha^{-1} ||\mathbb{C}^{1/2}\varepsilon(\mathbf{u})||_{0,K}^2\right)^{1/2} \left(\sum_e \alpha_e ||[\![\mathbf{v}]\!]||_{0,e}^2\right)^{1/2}.\end{aligned} \tag{2.15}$$

Furthermore,

$$\sum_e \alpha_e ([\![\mathbf{u}]\!], [\![\mathbf{v}]\!])_e \leq C \left(\sum_e \alpha_e ||[\![\mathbf{u}]\!]||_e^2\right)^{1/2} \left(\sum_e \alpha_e ||[\![\mathbf{v}]\!]||_e^2\right)^{1/2}.$$

The continuity (2.12) follows from the above bounds. To prove (2.13) note that using (2.15) we get

$$\begin{aligned}\mathcal{A}_h(\mathbf{u}, \mathbf{u}) &\geq ||\mathbf{u}||_{DG}^2 - 2 \left(\sum_K C\alpha^{-1} ||\mathbb{C}^{1/2}\varepsilon(\mathbf{u})||_{0,K}^2\right)^{1/2} \left(\sum_e \alpha_e ||[\![\mathbf{u}]\!]||_{0,e}^2\right)^{1/2} \\ &\geq (1 - 2C\alpha^{-1}) ||\mathbf{u}||_{DG}^2.\end{aligned}$$

The last bound follows from $ab \leq (\varepsilon/2)a^2 + (1/2\varepsilon)b^2$. If $1 - 2C\alpha^{-1} > 0$ the thesis follows. □

2.3 Algebraic formulation

For the sake of simplicity, we assume $\Omega \subset \mathbb{R}^2$ and we suppose the polynomial order N is homogeneous over the domain. We then introduce the vectorial basis $\{\boldsymbol{\psi}_i^\ell\}_{i,\ell=1,2}$ on every triangle K by setting

$$\boldsymbol{\psi}_i^1 = \begin{pmatrix} \psi_i \\ 0 \end{pmatrix}, \qquad \boldsymbol{\psi}_i^2 = \begin{pmatrix} 0 \\ \psi_i \end{pmatrix}.$$

The choice of the scalar basis ψ_i will be thoroughly discussed in Section 3.2. The number of degrees of freedom per element is denoted by $N_d = (N+1)(N+2)/2$, the number of elements by N_{el}. The total number of degrees of freedom is thus $N_t = N_{el}N_d$. If $N = N_K$ is different between the elements, one has to use $N_{d,K} =$

$(N_K+1)(N_K+2)/2$ and to sum over all K. The trial function $\mathbf{u}_h$ can therefore be expanded as

$$\mathbf{u}_h = \sum_{k=0}^{N_{el}-1} \sum_{a_k}^{a_{k+1}-1} \left(u_i^1 \boldsymbol{\psi}_i^1 + u_i^2 \boldsymbol{\psi}_i^2 \right) \tag{2.16}$$

where $a_k = kN_d + 1$. Since the variational formulation (2.7) has to hold for every function $\mathbf{v}_h$, we can take as test functions $\boldsymbol{\psi}_i, \forall i$. The vector of the unknowns

$$\mathbf{U} = \begin{pmatrix} u_1^1 \\ \vdots \\ u_{N_t}^1 \\ u_1^2 \\ \vdots \\ u_{N_t}^2 \end{pmatrix},$$

is then constructed by concatenating the expansion coefficients of the first vectorial component with those of the second one. The algebraic formulation of the semi-discrete problem (2.7) is then

$$\mathbf{M}\ddot{\mathbf{U}} + (\mathbf{A} + \mathbf{I} + \mathbf{S})\,\mathbf{U} = \mathbf{F}. \tag{2.17}$$

The matrices $\mathbf{M}$ and $\mathbf{A}$ are block diagonal while the matrices $\mathbf{I}$ and $\mathbf{S}$ have a more complex structure. Then, let us decompose any of the matrices $\mathbf{M}$, $\mathbf{A}$, $\mathbf{I}$ and $\mathbf{S}$ into d^2 blocks of the form

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}^{11} & \mathbf{C}^{12} \\ \mathbf{C}^{21} & \mathbf{C}^{22} \end{pmatrix}.$$

As we have already stated, $\mathbf{M}$ is block diagonal, i.e. $\mathbf{M}^{12} = \mathbf{M}^{21} = \mathbf{0}$ and $\mathbf{M}^{ii}$ $i = 1,2$ are composed of N_{el} dense blocks of size $N_d \times N_d$. The four blocks of $\mathbf{A}$ exhibit the same structure of $\mathbf{M}^{ii}$, so that the matrix has a main block diagonal and two smaller ones in the $\mathbf{A}^{12}$ and $\mathbf{A}^{21}$ blocks. The sparsity pattern of $\mathbf{I}$ is the same for all the sub-blocks, with every row and column related to the element K containing a N_d by N_d block in diagonal position and $\#\{e \in \partial K \cap \mathcal{E}_I\}$ non-diagonal blocks. The matrix $\mathbf{S}$ has the same pattern as $\mathbf{I}$ if choice (2.10) is made. Otherwise, if it is formulated as in (2.9), $\mathbf{S}^{ii}$, $i = 1,2$, have the same structure of

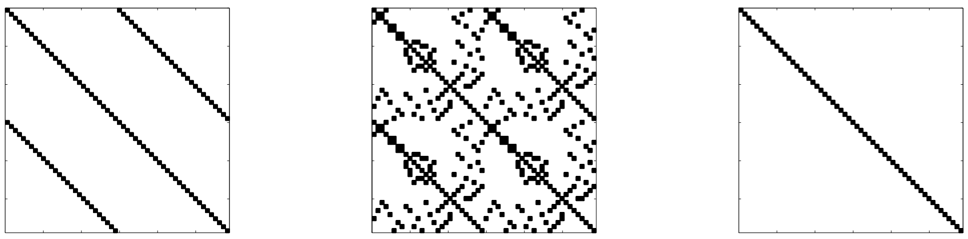


Figure 2.1: Sparsity patterns for (a) $\mathbf{A}$, (b) $\mathbf{I}$ and (c) $\mathbf{M}$

the sub-blocks of $\mathbf{I}$, but $\mathbf{S}^{12} = \mathbf{S}^{21} = \mathbf{0}$. An example of those patterns is given in Figure 2.1, where an approximation with $N = 5$ and $N_{el} = 28$ is considered.

Next, we identify by $\mathbf{A}_K$ (resp. $\mathbf{M}_K$) the sub-block of $\mathbf{A}$ (resp. $\mathbf{M}$) relative to the element K and by $\mathbf{I}_{K,J}$ and $\mathbf{S}_{K,J}$ the blocks of the inter-element and stability matrices relative to the edge of K whose neighbor is element J. Therefore,

$$\mathbf{A}_K^{lm}[i,j] = \mathcal{A}_K(\boldsymbol{\psi}_j^m, \boldsymbol{\psi}_i^l) \quad l,m = 1,2$$

and

$$\mathbf{M}_K^{l}[i,j] = (\rho\psi_j, \psi_i)_K \quad l = 1,2$$

with both i and j ranging over $a_K, \ldots, a_{K+1} - 1$. Similarly,

$$\mathbf{I}_{K,J}^{lm}[i,j] = \sum_{e \in \partial K \cap \partial J} \mathcal{I}_e(\boldsymbol{\psi}_j^m, \boldsymbol{\psi}_i^l) \quad l,m = 1,2$$

and

$$\mathbf{S}_{K,J}^{lm}[i,j] = \sum_{e \in \partial K \cap \partial J} \mathcal{S}_e(\boldsymbol{\psi}_j^m, \boldsymbol{\psi}_i^l) \quad l,m = 1,2$$

with i and j taking integer values in their appropriate element ranges, i.e. $j \in [a_J, a_{J+1})$ and $i \in [a_K, a_{K+1})$.

The properties of the matrices involved in the algebraic formulation of the method vary with the choice of the basis for the space V_h: those issues will be discussed in the next chapter. Moreover, the scalar products used throughout this chapter must be evaluated in a numerical way, and the choice of a suitable quadrature rule over the reference element is crucial.

Chapter 3

High order methods on simplices

The spectral elements method (SEM) is a high order method which employs special basis functions and quadrature rules with a high degree of exactness. They provide high order approximations to the solution of partial differential equations, while preserving an acceptable computational cost for the solution of the resulting algebraic linear systems. For a full treatment of spectral methods, see [CHQZ06] and [CHQZ07].

Spectral element methods are usually extended from one dimension to multidimensional domains using meshes made of tensor product elements (i.e. deformed squares and cubes). Those shapes, though, are not very well suited to the tessellation of complex domains, nor they are easily adaptable. To exploit the properties of triangular and tetrahedral meshes, well studied in the context of finite elements methods, the spectral element method has been extended to those meshes, and triangular spectral element methods (from now on, TSEM) have been developed.

The extension from the one dimensional SEM to the multidimensional TSEM in not as straightforward as it is in the classical SEM case and there is still ongoing debate regarding the choice of a suitable polynomial basis and of a set of quadrature points and weights. The two main approaches in the choice of a basis consider a *modal* (with Jacobi polynomials) or a *nodal* (with Lagrange-type polynomials) basis. In the former case there is a choice between an orthogonal basis, with the disadvantage that all modes are not null on the boundary of the element and that the basis is not suitable for a $\mathcal{C}^0$ approximation, and a "boundary adapted" basis,

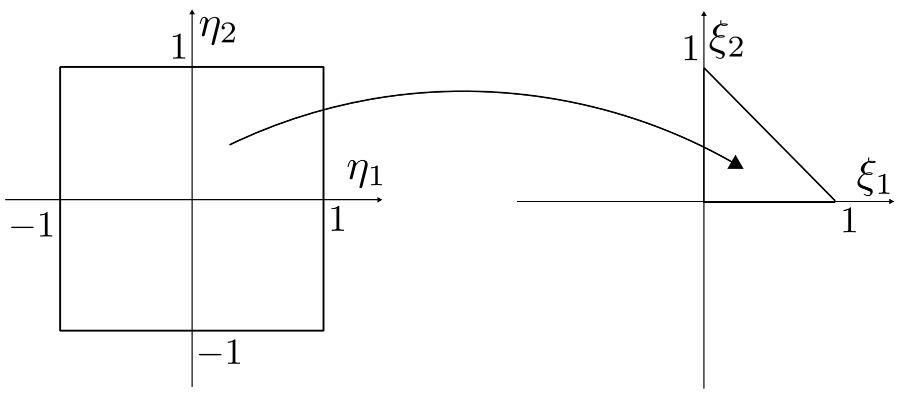


Figure 3.1: The mapping from the reference square to the reference triangle

which is not $L^2(T)$ orthogonal, but the modes can be decomposed into boundary and interior modes. In the latter case, we have a FEM-type basis and additional care must be put into the choice of a set of interpolation points: an optimal set has not yet been found and this is still an open problem. Degrees of freedom represent expansion coefficients in the modal case, while they are nodal values in the nodal case (and this can lower the computational cost of the solution post-processing).

Classical modal basis and their application to computational fluid dynamics are presented in [KS05], while an interesting alternative, though at a preliminary stage, is presented in [RMK11]; nodal basis in the frame of discontinuous Galerkin methods are discussed in [HW08].

3.1 From tensor product to simplex elements

To ease the reading we present the details for $d = 2$ elements. Let the reference square $\mathcal{Q}^2$ be defined by $\mathcal{Q}^2 = \{(\eta_1, \eta_2) : -1 \leq \eta_2, \eta_2 \leq 1\}$, and the reference triangle

$$\mathcal{T}^2 = \{(\xi_1,\, \xi_2) : \xi_1, \xi_2 \geq 0,\, \xi_1 + \xi_2 \leq 1\} \tag{3.1}$$

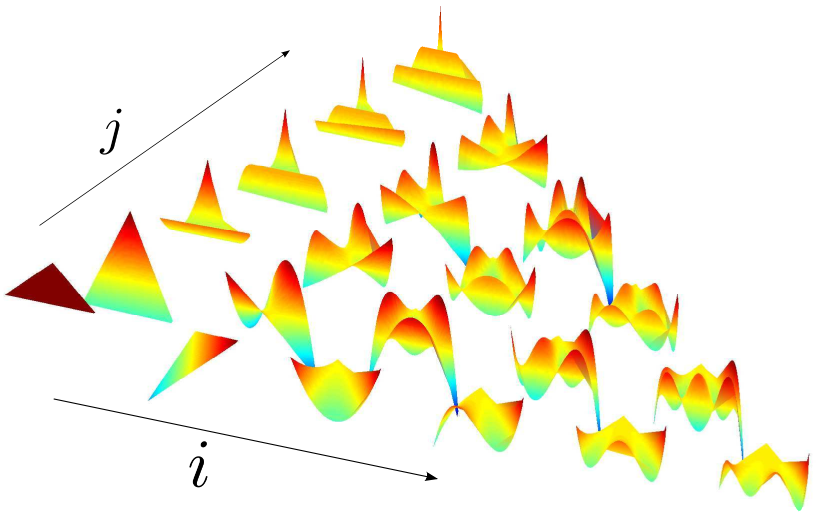


Figure 3.2: Dubiner orthonormal basis (3.3), $N = 5$.

can be derived from the square via the transformation

$$\xi_1 = \frac{(1+\eta_1)(1-\eta_2)}{4}, \qquad \xi_2 = \frac{1+\eta_2}{2}, \tag{3.2}$$

shown in Figure 3.1. It is worth noting how the inverse map, often referred to as the *Duffy transformation* and explicitly given by

$$\eta_1 = 2\frac{\xi_1}{1-\xi_2} - 1, \qquad \eta_2 = 2\xi_2 - 1,$$

is singular at the top vertex of the triangle. The map (3.2) is indeed “collapsing” the square's edge, identified by $\eta_2 = 1$, onto that single vertex.

3.2 Basis functions

3.2.1 Legendre–Dubiner basis

In the case of a modal expansion, the Jacobi polynomials $J_n^{\alpha,\beta}$ (see [Sze75]) are used to develop an orthogonal bases (as is the case with the "classical" SEM since Legendre polynomials are a subset of Jacobi polynomials) which shows a tensor product structure. The orthogonal bases $\{\psi_{ij}(\xi_1,\xi_2)\}_{ij}$ introduced by Koornwinder [Koo75] and developed by Dubiner [Dub91] takes the form

$$\psi_{ij}(x,y) = c_{ij}\varphi_i(\eta_1)\varphi_{ij}(\eta_2) \tag{3.3}$$

$$= c_{ij}\varphi_i\left(2\frac{\xi_1}{1-\xi_2}-1\right)\varphi_{ij}\left(2\xi_2-1\right), \tag{3.4}$$

where,

$$\begin{aligned}\varphi_i(x) &= J_i^{0,0}(x)\\ \varphi_{ij}(x) &= (1-x)^j J_j^{2i+1,0}(x)\\ c_{ij} &= \sqrt{2(2i+1)(i+j+1)}.\end{aligned}$$

Integrating the product $\psi_{ij}\psi_{pq}$ in the standard region $\mathcal{T}^2$, since $|\partial\xi_i/\partial\eta_j| = (1-\eta_2)/8$, we have (omitting the constants)

$$\int_{-1}^{1}\varphi_p\varphi_i \mathrm{d}\eta_1 \int_{-1}^{1}\varphi_{pq}\varphi_{ij}\left(1-\eta_2\right)\mathrm{d}\eta_2. \tag{3.5}$$

The first integral in (3.5) is null if $i \neq p$; if we suppose $i = p$ the second integral has the form

$$\int_{-1}^{1}(1-\eta_2)^{2p+1}J_q^{2p+1,0}J_j^{2p+1,0}\mathrm{d}\eta_2.$$

The above integral is not null if and only if $j = q$ and the orthogonality in the standard triangle $\mathcal{T}^2$ of the basis proposed in (3.3) follows. A representation of the basis functions up to degree 5 is given in Figure 3.2: it is evident how there is a strong asymmetry and how evaluating a function on the boundary of the element requires the evaluation of all modes. Moreover, this basis is suitable only for a discontinuous approximation, since it is not possible to "glue" continuously modes

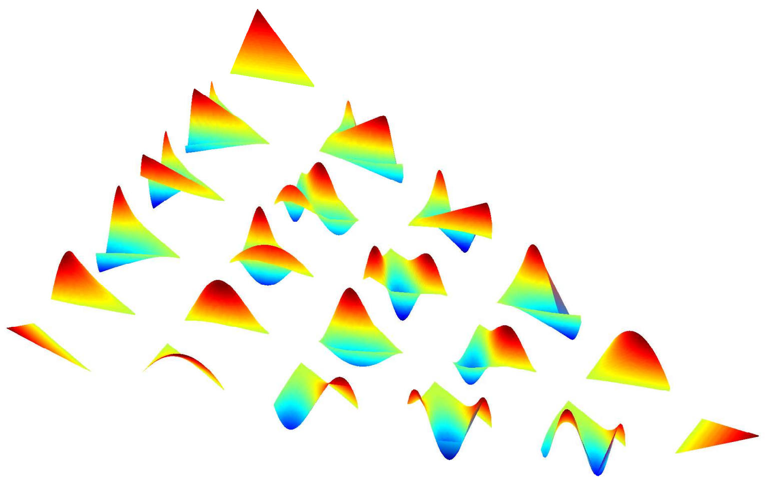


Figure 3.3: Boundary adapted bases. Vertex modes are represented at the corners and edge modes on the edges.

between diferent triangles.

3.2.2 Boundary adapted basis

The basis defined in (3.3) can be modified in order to build a $\mathcal{C}^0$ continuous expansion, while retaining some of the advantages provided by Jacobi polynomials. The modes in the modified basis, presented e.g. in [SK95, Dub91], can be divided into vertex modes (which are null on two vertex and take a unitary value on the other), boundary modes (which are null on all edges except one) and interior modes (which are null on all edges of the triangle). Given the reference triangle $\mathcal{T}^2$, if N is the polynomial degree of the basis, the vertex modes are

$$\begin{aligned}
\psi_1 &= 1 - \xi_1 - \xi_2, \\
\psi_2 &= \xi_1, \\
\psi_3 &= \xi_2,
\end{aligned}$$

the boundary modes

$$\begin{aligned}\psi_{m(i)} &= (1-\xi_1-\xi_2)\,\xi_1\,(1-\xi_2)^{i-1}\,J_{i-1}^{1,1}\left(\frac{2\xi_1}{1-\xi_2}-1\right),\\ \psi_{n(i)} &= (1-\xi_1-\xi_2)\,\xi_2 J_{i-1}^{1,1}\,(2\xi_2-1)\,,\\ \psi_{o(i)} &= \xi_1\xi_2 J_{i-1}^{1,1}\,(2\xi_2-1)\,, \qquad\qquad i=1,\dots,N-1\end{aligned}$$

and the internal modes,

$$\psi_{p(i,j)} = (1-\xi_1-\xi_2)\,\xi_1\xi_2\,(1-\xi_2)^{i-1}\,J_{i-1}^{1,1}\left(\frac{2\xi_1}{1-\xi_2}-1\right)J_{j-1}^{2i+1,1}\,(2\xi_2-1)$$
$$1\le i,j\le N-1;\; i+j\le N-1,$$

where m, n, o and p are bijections that put the basis in the desired order. A representation of the basis is shown in Figure 3.3. This basis can also be derived from a tensor product and square–to–triangle argument: see [KS05] for the details.

3.2.3 Nodal basis

The idea to develop spectral methods based on a set of interpolation nodes and on the Lagrange functions associated with them stems both from the finite element theory and from the consideration that Gauss nodes are also Fekete nodes on the square [BTW01]. Let f be a sufficiently smooth function in some space V defined in the standard region $\mathcal{T}^d$ and $\{\xi_i\}_{i=1}^N$ be a set of point. Let an interpolation operator $\mathcal{I}_N : V \to \mathbb{P}_N$ be defined such that

$$\mathcal{I}_N f(\xi_i) = f(\xi_i) \qquad \forall i=1,\dots,N.$$

Then, the Lebesgue constant $\Lambda_N = ||\mathcal{I}_N||_\infty$ is an indicator of the quality of the interpolation, since

$$||f-\mathcal{I}_N f||_\infty \le (1+\Lambda_N)\min_{p\in\mathbb{P}_N}||f-p||_\infty.$$

The choice of a good nodal points set for interpolation on simplices is a very open problem, especially in $d=3$ dimensions. See [GLMH09, Hes98, HT00, PR10, RSV12, War06]. A full description of nodal bases is beyond the scope of this work; their advantage resides mainly in the possibility to build a continuous approximation, even when dealing with meshes composed of elements of different shape.

3.3 Quadrature rules

The standard approach to quadrature in the framework of the spectral element methods on simplices is to map Gauss quadrature nodes from quadrilateral elements to triangular ones. The degree of exactness of these rules is thus preserved, but none of the symmetries of the triangle are considered and the points are cluttered near one vertex. When implementing a continuous approximation, there is therefore the need to use Gauss–Radau quadrature instead of Gauss–Lobatto ones, in order to avoid to perform differentiations at the extremal node. Other quadrature rules, which employ a smaller number of nodes, have therefore been proposed, for instance in [WX03], [TWB07] and [Dun85]. Those rules take into account the symmetries of the element, but are generally more heuristic and nodes are not explicitly defined.

3.3.1 Gauss quadrature

On meshes composed by d-parallelepipeds, Gauss quadrature rules in two or three dimensions are easily derivable from the one-dimensional case, by taking the tensor product of the nodes defined as the n zeros of the Legendre polynomial $J_n^{0,0}$. When using meshes composed by simplicial elements, Gauss nodes can be mapped to the reference element via the transformation (3.2). A representation of the quadrature nodes required for a spectral element approximation of degree 5 is given in Figure 3.4a and their sub-optimal distribution is quite evident. The main drawback of this approach resides in the fact that one has to use $(N+1)^d$, $d=2,3$, nodes for a $\mathbb{P}_N$ spectral approximation.

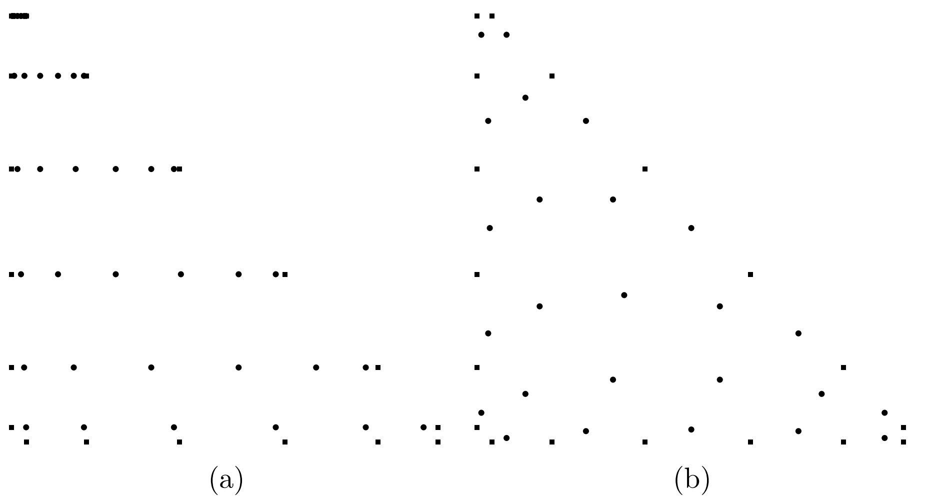


Figure 3.4: Gauss–Legendre (a) and Dunavant (b) interior quadrature nodes, for a $\mathbb{P}_5$ approximation (in both cases, edge nodes are Gauss–Legendre ones).

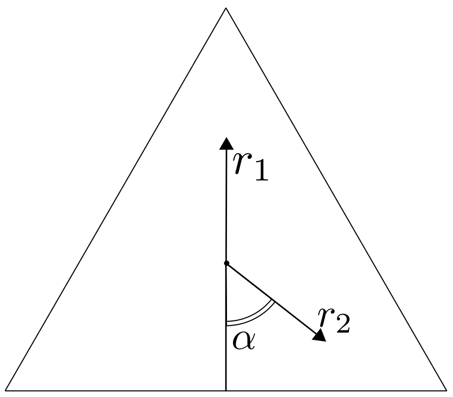


Figure 3.5: α and r for the Dunavant nodes

N	1	2	3	4	5	6	7	8	9	10
Gauss	4	9	16	25	36	49	64	81	100	121
Dunavant	3	6	12	16	25	33	42	52	70	79

Table 3.1: Number of Dunavant and Gauss quadrature nodes needed for different polynomial orders

3.3.2 Dunavant quadrature

Dunavant quadrature rules are derived in [Dun85] solving the moment equation introduced in [LJ75]. Nodes and weights are tabulated up to a polynomial degree of $p = 20$ and are subdivided into three groups: n_0 residing at the center of the equilateral triangle, n_1 residing on one height of the triangle (with unknown distance r_1 from the center) and n_2 for which one has to determine both the angle α and the distance r_2 from the center, as shown in Figure 3.5. In the last group, only the area with $0 \leq \alpha \leq \pi/3$ is considered. The total number of nodes is thus $n = n_0 + 3n_1 + 6n_2$. An example of the distribution of Dunavant quadrature nodes is shown, together with Gauss ones, in Figure 3.4b. Table 3.1 shows the different number of quadrature nodes needed in both cases for various polynomial orders. For N large, the difference is evident and using symmetric nodes can be of great use in reducing the computational load.

Chapter 4

Time discretization

In the semi-discrete formulation (2.17) a time stepping method is required to develop a fully discrete formulation of the method. In the sequel, the class of Newmark time-stepping methods will be presented, with a special focus on the Leap-Frog scheme, which is an explicit scheme widely used in the approximation of wave propagation phenomena. Since this work is focused on the spatial spectral approximation, other classes of methods will not be presented, but they are equally suited for this kind of phenomena, and some of them provide a high order of convergence. Those include the Runge-Kutta [QSS07] and the ADER-DG [PGA+13] methods. In the following, we suppose $\Omega \in \mathbb{R}^2$, and we introduce the vectors $\mathbf{U}_0$ and $\mathbf{V}_0$ which contain the expansions coefficients of $\Pi\mathbf{u}_0$ and $\Pi\mathbf{v}_0$ respectively.

4.1 The Newmark method

We shall start from formulation (2.17) and subdivide the interval $(0, T]$ in n_s intervals of (usually constant) amplitude $\delta t = T/n_s$. We introduce $t^n = n\delta t$ and denote the numerical expansion coefficients at time t^n by $\mathbf{U}^n = \mathbf{U}(t^n)$. Finally, we denote by $\mathbf{D} = \mathbf{A} + \mathbf{I} + \mathbf{S}$. The first step of the Newmark method takes the form

$$\begin{aligned}\left(\mathbf{M} + \delta t^2 \beta \mathbf{D}\right) \mathbf{U}^1 = &\left(\mathbf{M} - \delta t^2 \left(\frac{1}{2} - \beta\right) \mathbf{D}\right) \mathbf{U}^0 - \delta t \mathbf{M} \mathbf{V}_0 \\ &+ \delta t^2 \left(\beta \mathbf{F}^1 + \left(\frac{1}{2} - \beta\right) \mathbf{F}^0\right),\end{aligned} \tag{4.1}$$

and the subsequent steps are

$$\begin{aligned}\left(\mathbf{M}+\delta t^2\beta\mathbf{D}\right)\mathbf{U}^{n+1} &= \left(2\mathbf{M}-\delta t^2\left(\frac{1}{2}-2\beta+\zeta\right)\mathbf{D}\right)\mathbf{U}^n \\ &\quad -\left(\mathbf{M}+\delta t^2\left(\frac{1}{2}+\beta-\zeta\right)\right)\mathbf{U}^{n-1} \\ &\quad +\delta t^2\left(\beta\mathbf{F}^{n+1}+\left(\frac{1}{2}-2\beta+\zeta\right)\mathbf{F}^n+\left(\frac{1}{2}+\beta-\zeta\right)\mathbf{F}^{n-1}\right),\end{aligned} \tag{4.2}$$

$n \geq 1$. The parameters $\beta \geq 0$ and $\zeta \geq 1/2$ have to be chosen. For the former, the choice between $\beta \neq 0$ and $\beta = 0$ is quite relevant, since it determines the implicit or explicit nature of the method, respectively. If $\beta = 0$, every time step requires only the computation of two matrix–vector products, and the solution of a block-diagonal linear system; in the implicit case ($\beta \neq 0$), the whole matrix in the left hand side of (4.2) has to be inverted. The matrix is positive definite and its symmetry depends on the choice of the interior penalty method.

The parameter ζ impacts on the order of the method: if $\zeta = 1/2$ the method is second order accurate in time; if $\zeta \neq 1/2$ it is only first order accurate.

4.2 The Leap-Frog method

Setting $\zeta = 1/2$ and $\beta = 0$ in (4.1)-(4.2), the resulting method is the Leap-Frog method: the first step reads

$$\mathbf{M}\mathbf{U}^1 = \left(\mathbf{M}-\frac{\delta t^2}{2}\mathbf{D}\right)\mathbf{U}^0-\delta t\mathbf{M}\mathbf{V}_0+\frac{\delta t^2}{2}\mathbf{F}^0, \tag{4.3}$$

and the following ones

$$\mathbf{M}\mathbf{U}^{n+1} = \left(2\mathbf{M}-\delta t^2\mathbf{D}\right)\mathbf{U}^n-\mathbf{M}\mathbf{U}^{n-1}+\delta t^2\mathbf{F}^n, \quad n \geq 1 \tag{4.4}$$

If the basis used is (3.3), then the mass matrix is diagonal, and the method is fully explicit, with the computational cost arising only from two matrix–vector products. This product can be highly optimized, since the matrices have a sparse block structure, with the location of the dense blocks that is known when the mesh is loaded. The optimizations available for dense matrices can then be used on every block. Furthermore, every block-row results from the integration over an element: if a mesh partitioning algorithm is applied, the algorithm is trivially parallel with low computational cost. Even if the chosen basis is the boundary adapted one described in Section 3.2.2, the mass matrix is block diagonal. Moreover, the map $\mathbf{F}_K$ defined in Chapter 5 is linear, every element differs only by the Jacobian of the transformation, and the inverse can be computed only for a block. In any case, the inversion of matrix $\mathbf{M}$ is computationally cheap.

Chapter 5

Implementation issues

The bases and the quadrature rules introduced so far are, in practice, defined only on the reference element $\mathcal{T}^2$ and the integration is carried out via a mapping from the reference element to the element on the mesh. We will call such a map by $\mathbf{F}_K : \mathcal{T}^2 \to K$. We now specify better what is meant in (2.1). Let

$$\mathbb{P}_N(\mathcal{T}^2) = \mathrm{span}\{\xi_1^i \xi_2^j, 0 \leq i + j \leq N\},$$

where ξ_1 and ξ_2 are defined as in (3.1). Then,

$$\mathbb{P}_N(K) = \{v = \hat{v} \circ \mathbf{F}_K^{-1} : \hat{v} \in \mathbb{P}_N(\mathcal{T}^2)\}.$$

With the notation shown in Figure 5.1, the mapping is explicitly given by

$$\mathbf{F}_K(\xi_1, \xi_2) = \mathbf{x}_0(1 - \xi_1 - \xi_2) + \mathbf{x}_1\xi_1 + \mathbf{x}_2\xi_2 \tag{5.1}$$

and its Jacobian is therefore,

$$J_{\mathbf{F}} = (\mathbf{x}_1 - \mathbf{x}_0 \mid \mathbf{x}_2 - \mathbf{x}_0)\,. \tag{5.2}$$

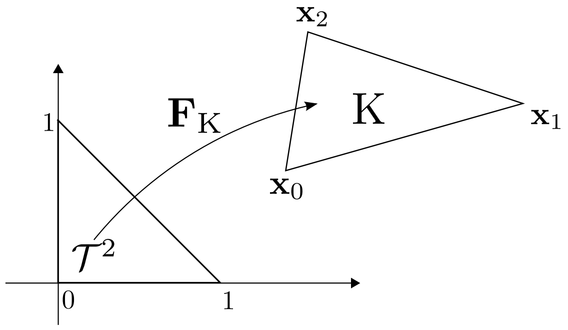


Figure 5.1: Map $\mathbf{F}_K : \mathcal{T}^2 \to K$ defined in (5.1).

5.1 Elemental operations

5.1.1 Differentiation

The evaluation of the stress–strain matrix involves the integration of the derivatives of the basis functions, which are functions defined on the reference triangle. The value of a basis function over an arbitrary triangle K is indeed given by $\psi \circ \mathbf{F}^{-1} : K \to \mathbb{R}$ (omitting the subscript K). Therefore

$$\frac{\partial(\psi \circ \mathbf{F}^{-1})}{\partial \boldsymbol{\xi}} = J_{\mathbf{F}}^{-1} \frac{\partial \psi}{\partial \boldsymbol{\xi}} \circ \mathbf{F}^{-1},$$

and

$$\int_K \left(J_{\mathbf{F}}^{-1} \frac{\partial \psi_i}{\partial \xi} \circ \mathbf{F}^{-1} \right) \cdot \left(J_{\mathbf{F}}^{-1} \frac{\partial \psi_j}{\partial \xi} \circ \mathbf{F}^{-1} \right) = \int_{\mathcal{T}^2} \left(J_{\mathbf{F}}^{-1} \frac{\partial \psi_i}{\partial \xi} \right) \cdot \left(J_{\mathbf{F}}^{-1} \frac{\partial \psi_j}{\partial \xi} \right) |J_{\mathbf{F}}|.$$

The evaluation of the scalar products and the definition of the quadrature rules is thus done on the reference element, and mapped to the actual element via (5.2).

5.1.2 Backward and forward transform

In equation (1.11) the projection operator $\Pi : V \to V_h$ has been introduced. In practice, when exploiting a modal basis, we have to determine, from a set of function evaluations $u_0(\boldsymbol{\xi}_i)$, the modal expansion coefficients $\hat{u}_i$ presented in (2.16).

Analytically, in every element K we write

$$(\mathbf{u}(0), \boldsymbol{\psi}_j^\ell)_K = (\mathbf{u}_0, \boldsymbol{\psi}_j^\ell)_K = \int_K \mathbf{u}_0 \cdot \boldsymbol{\psi}_j^\ell, \quad j = 1, \dots, N_d, \quad \ell = 1, 2.$$

Next, by expanding $\mathbf{u}$, indicating the vector of its coefficients by $\hat{\mathbf{u}}$ and summing over all elements we have

$$\left(\mathbf{M}^\ell \hat{\mathbf{u}}\right)[j] = \sum_K \int_K \mathbf{u}_0 \cdot \boldsymbol{\psi}_j^\ell, \quad j = 1, \dots, N_d, \quad \ell = 1, 2 \tag{5.3}$$

where $\mathbf{M}$ is defined as in Section 2.3, but neglecting the mass density ρ. Let then w_i and $\boldsymbol{\xi}_i$ be the weights and nodes associated to the quadrature rule employed in the triangle. Equation (5.3) thus takes the form of a collocation projection:

$$\left(\mathbf{M}^\ell \hat{\mathbf{u}}\right)[j] = \sum_K \sum_i w_i \mathbf{u}_0(\boldsymbol{\xi}_i) \cdot \boldsymbol{\psi}_j^\ell(\boldsymbol{\xi}_i), \quad j = 1, \dots, N_d, \quad \ell = 1, 2 \tag{5.4}$$

We now introduce the matrices $\mathbf{B}$ and $\mathbf{W}$ such that

$$\begin{aligned} \mathbf{B}[i, j] &= \boldsymbol{\psi}_i(\boldsymbol{\xi}_j), \\ \mathbf{W} &= \operatorname{diag}(w_i), \end{aligned} \tag{5.5}$$

and define the vector of the nodal evaluations with the same name of the function, that is $\mathbf{u}_0[i] = \mathbf{u}_0(\boldsymbol{\xi}_i)$. It is now possible to write (5.4) in its algebraic formulation, i.e.,

$$\mathbf{M}\hat{\mathbf{u}} = \mathbf{B}\mathbf{W}\mathbf{u}_0,$$

so that modal expansion coefficients are given by

$$\hat{\mathbf{u}} = \mathbf{M}^{-1}\mathbf{B}\mathbf{W}\mathbf{u}_0. \tag{5.6}$$

Note that $\mathbf{M} = \mathbf{B}\mathbf{W}\mathbf{B}^T$. This operation is the *discrete forward transform.*

The inverse operation, i.e., defining the nodal values of a function from its expansion coefficients, is important when there is the need to output the value of the numerical scheme's solution (therefore, in all practical applications). The same applies in the framework of a convergence analysis, when one has to compute the norm of the difference between the exact and approximate solutions and the values of the latter at the quadrature nodes are needed. Let us suppose we have the N_t coefficients $\hat{\mathbf{u}}$: then the values at the quadrature points in the physical space $\mathbf{u}[i] = \mathbf{u}(\boldsymbol{\xi}_i)$ can be computed as

$$\mathbf{u} = \mathbf{B}^\top \hat{\mathbf{u}}. \tag{5.7}$$

This is the *discrete backward transform*.

5.2 Global operations

When applying a discontinuous method, there are almost no global operations to be performed. The assembly of the global matrices is straightforward: every sub-block has a one-to-one correspondence with an element or an edge. When the matrices are assembled, looping over the elements equates to filling the matrices row-wise. This is important when developing a SPMD parallel implementation: in the case of explicit time stepping methods, therefore, a process has to store only the rows it has extracted, multiplying them at every time step with the right hand side vector. To reduce the communication cost a partitioning algorithm should be applied on the mesh with the goal of reducing the frontier of the mesh partition assigned to every process.

If the approximation considered is continuous (e.g. in the case of a nonconforming DG method, where the nonconforming blocks are subdivided into conforming meshes) then additional care must be put into the assembly of the triangles of the mesh. In particular, the singular vertex of transformation (3.2) has to be treated specifically, since only some combinations of adjacent triangles are allowed. See [KS05] for the details.

Chapter 6

Analysis of grid dissipation, dispersion and stability

Dissipation and dispersion errors play a key role in the quality of an approximation to the solution of wave propagation phenomena. Consider the situation shown in Figure 6.1, where a wave traveling with speed α_* is compared with its numerical approximation, traveling at speed a_h. We define the dispersion error as

$$e = \frac{\alpha_h}{\alpha_*} - 1,$$

that is, a measure of the error of the computed wave speed versus the exact one, and the dissipation error as the maximum ratio between the amplitudes

$$||f_h(x - a_h t)||_\infty / ||f(x - \alpha_* t)||_\infty.$$

In Section 6.1 the dissipation and dispersion properties of the semi-discrete algebraic scheme (2.17) are investigated, following the techniques known as *Von Neumann analysis.*

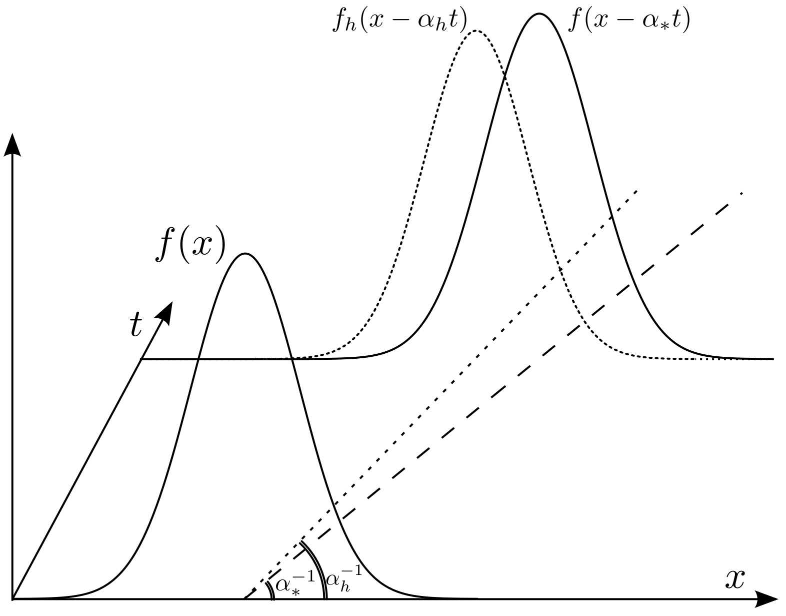


Figure 6.1: Exact and computed traveling waves. The angles are in the x–t plane.

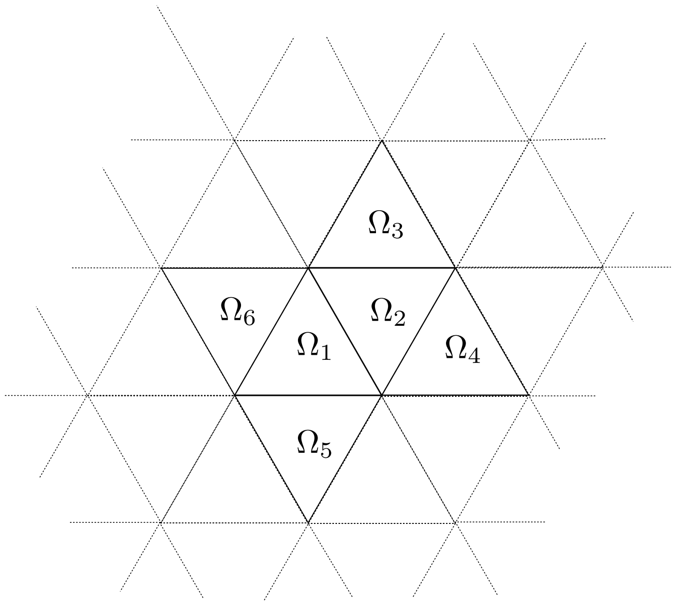


Figure 6.2: Elements numbering for dissipation and dispersion analysis

6.1 Grid dissipation and dispersion for the propagation of a plane wave

Consider the unbounded mesh shown in Figure 6.2, which can be seen as a repetition of the pattern $\Omega_c = \Omega_1 \cup \Omega_2$. In order to investigate the dissipative and dispersive properties of the numerical schemes considered, we denote by $\Omega = \bigcup_i \Omega_i$ and choose in (1.11) the test functions

$$\tilde{\boldsymbol{\psi}}_i(\mathbf{x}) = \begin{cases} \boldsymbol{\psi}_i(\mathbf{x}) & \mathbf{x} \in \Omega_c \\ 0 & \text{otherwise.} \end{cases}$$

Adopting the notation of Section 2.3 with $\mathbf{C} = \mathbf{I} + \mathbf{S}$, the matrices have the following structure

$$\begin{aligned}
\mathbf{M}^\ell &= \begin{pmatrix} \mathbf{M}_1^\ell & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}_2^\ell & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \\
\mathbf{A}^{\ell,m} &= \begin{pmatrix} \mathbf{A}_1^{\ell,m} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{\ell,m} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \\
\mathbf{C}^{\ell,m} &= \begin{pmatrix} \mathbf{C}_{1,1}^{\ell,m} & \mathbf{C}_{1,2}^{\ell,m} & \mathbf{C}_{1,3}^{\ell,m} & \mathbf{C}_{1,4}^{\ell,m} & \mathbf{C}_{1,5}^{\ell,m} & \mathbf{C}_{1,6}^{\ell,m} \\ \mathbf{C}_{2,1}^{\ell,m} & \mathbf{C}_{2,2}^{\ell,m} & \mathbf{C}_{2,3}^{\ell,m} & \mathbf{C}_{2,4}^{\ell,m} & \mathbf{C}_{2,5}^{\ell,m} & \mathbf{C}_{2,6}^{\ell,m} \end{pmatrix}, \\
\mathbf{U}^\ell &= \begin{pmatrix} \mathbf{U}_1^\ell & \mathbf{U}_2^\ell & \mathbf{U}_3^\ell & \mathbf{U}_4^\ell & \mathbf{U}_5^\ell & \mathbf{U}_6^\ell \end{pmatrix}^\top .
\end{aligned} \tag{6.1}$$

Since in general every wave could be written as a superposition of plane waves, we consider as a reference solution a traveling plane wave of unitary amplitude, i.e.,

$$\mathbf{U} = e^{i(\mathbf{k}\cdot\mathbf{x} - \omega t)}, \tag{6.2}$$

for a wave vector $\mathbf{k} = (k_x, k_y)^\top$. Therefore, if we suppose that $|e| = h$ for every edge in Figure 6.2, we easily derive

$$\begin{aligned}
\mathbf{U}_3 &= e^{-i(hk_x/2 + hk_y)}\mathbf{U}_1 = \alpha_{13}\mathbf{U}_1 & \mathbf{U}_4 &= e^{-ihk_x}\mathbf{U}_1 = \alpha_{14}\mathbf{U}_1, \\
\mathbf{U}_5 &= e^{+i(hk_x/2 + hk_y)}\mathbf{U}_2 = \alpha_{25}\mathbf{U}_2 & \mathbf{U}_6 &= e^{+ihk_x}\mathbf{U}_2 = \alpha_{26}\mathbf{U}_2.
\end{aligned} \tag{6.3}$$

Note that we are assuming that Ω_c is a pattern periodically reproducing: it is therefore crucial, in this context, that the edges in Ω_3 and Ω_4 are numbered in the same way as those in Ω_1. The same is valid for Ω_2 and the remaining triangles. This issue is not present when dealing with quadrilateral spectral elements, and stems from the fact that triangular spectral elements are not rotationally invariant. Substituting relations (6.3) into the vector in (6.1) we are then able to obtain the linear system

$$\mathbf{M}\ddot{\mathbf{U}} + \left(\tilde{\mathbf{C}} + \mathbf{A}\right)\mathbf{U} = \mathbf{0},$$

where

$$\tilde{\mathbf{C}}^{\ell,m}_{K,J} = \mathbf{C}^{\ell,m}_{K,J} + \sum_I \alpha_{KI}\mathbf{C}^{\ell,m}_{K,I}.$$

For the other two matrices $\mathbf{M}, \mathbf{A}$ only the part referring to the first two triangles is different from zero. Now, $\mathbf{U}$ has the form (6.2): taking the second derivative with respect to time we obtain the generalized eigenvalue problem

$$\left(\tilde{\mathbf{C}} + \mathbf{A}\right)\mathbf{U} = \Lambda\mathbf{M}\mathbf{U}, \tag{6.4}$$

with $\Lambda = \omega^2$. Solving problem (6.4) with the matrices generated by the numerical method, we obtain the generalized eigenvalues $\Lambda_h = \omega_h^2$, which correspond to waves traveling with speed

$$c_h = \frac{h\omega_h}{2\pi\delta}, \tag{6.5}$$

where δ is the number of points per wavelength. The pressure and shear waves described in Section 1.2.1 are plane waves traveling with speed c_P and c_S respectively: between the speeds (6.5) we select those that are the best approximations to c_P and c_S and we denote them by $c_{P,h}$ and $c_{S,h}$. We then denote by $e_S = c_{S,h}/c_S - 1$ and $e_P = c_{P,h}/c_P - 1$ the approximation errors.

6.1.1 Semi-discrete dispersion error

In what follows, the errors computed will be presented at varying $\theta = \operatorname{atan}(k_y/k_x)$, and their relationship with the order of the approximation N, with the number of quadrature nodes and with the choice of the interior penalty method (i.e. the choice of $\vartheta = -1, 1, 0$ in (2.6)). All simulations are performed with $\alpha = 2$, $\lambda = 2$,

$\mu = 1$ and $\rho = 1$. This gives a ratio between the wave speeds $c_P/c_S = 2$. In Tables 6.1, 6.2, 6.3 and 6.4 the maximum of the errors $\max_\theta |e_S(\theta)|$ and $\max_\theta |e_P(\theta)|$ are reported. It is evident how the choice between the basis (3.3) and the one in Section 3.2.2 is not relevant, since the error can be imputed to the interior penalty numerical method itself. The next results will be therefore reported only for a single basis. Moreover, the symmetric interior penalty method performs uniformly better the non-symmetric and incomplete variants. Tables 6.2 and 6.1 list the results for the stability function (2.9), while Tables 6.4 and 6.3 list the analogous ones obtained with the stability function (2.10), in this case only for the boundary adapted basis (the results are the same for the Dubiner basis). The errors have the same order of magnitude, but the error resulting from the discretization which employs (2.10) is uniformly smaller than the one obtained with (2.9).

We see how the increase in the polynomial order N yields a dramatic decrease in the dispersion error. Decreasing the sampling ratio δ (i.e., increasing the number of quadrature nodes) can also impact on the error, but with less relevance. The two errors behave qualitatively in the same way.

Figures 6.6 display the anisotropy curves of the numerical methods, i.e. the behavior of the ratio between the computed speed and the exact one. IIPG results are not reported, since they exhibit a behavior qualitatively and quantitatively similar to NIPG results. It is more evident how the stability function which includes the whole tensor has a more uniform error: it tends to overestimate the wave speed, but it doesn't exhibit the wrong behavior that the other stabilization shows. Note that, when approximating seismic waves in complex scenarios and on unstructured meshes, the quality of the approximation of waves coming from all angles is crucial. It is worth noting that the distortion of the elements in the mesh plays a role in the generation of the error: Figure 6.3 is obtained with the stability function given in (2.10) and the SIPG method, but on a mesh made of right-angled triangles. Comparing it with Figures 6.6a and 6.6b we note the impact of the distortion.

We now consider the evolution of the dispersion error with respect to the polynomial order of the basis N, which is represented in Figure 6.4. It is evident how both methods exhibit an exponential order of convergence, even though the NIPG arrives at the threshold value $\epsilon \sim 10^{-13}$ (where rounding errors become relevant) only around $N = 9$. The SIPG method, on the other hand, reaches ϵ around $N = 6$. Numerical tests not reported here show that the outcome is invariant to the choice of the stability function and of the basis among those presented.

Finally, in Figure 6.5 the dispersion error is analyzed varying the sampling ratio δ. The approximation behaves in the same way as the approximation based on quadrilateral elements, with an optimal order $\mathcal{O}(\delta^{2N})$ for the symmetric method, and a sub-optimal order $\mathcal{O}(\delta^{N+1})$ for the non-symmetric one.

6.1.2 Semi-discrete dissipation error

From (6.2) we derive the dissipation errors

$$d_S = e^{\Im \omega_{S,h} t}, \qquad d_P = e^{\Im \omega_{P,h} t}, \tag{6.6}$$

where $\omega_{S,h}$ (resp. $\omega_{P,h}$) refers to the square root of the eigenvalue of (6.4) used in the computation of $c_{S,h}$ (resp. $c_{P,h}$). A non dissipative scheme is thus one for which

$$\Im \omega_{P,h} = \Im \omega_{S,h} = 0,$$

and the closer to zero those values are, the less dissipative is the scheme. The computed values of $\Im \omega_{S,h}$ and $\Im \omega_{P,h}$ are reported in Table 6.5. The values are near machine precision and the scheme shows evidently almost irrelevant dissipation errors, with both the basis functions proposed. The dissipation error is invariant also to the choice of the stability function.

		boundary adapted basis			legendre-dubiner basis		
δ	N	SIPG	NIPG	IIPG	SIPG	NIPG	IIPG
0.5	3	2.13e-03	6.86e-03	6.05e-03	2.13e-03	6.86e-03	6.05e-03
0.25	3	2.18e-05	4.46e-04	4.19e-04	2.18e-05	4.46e-04	4.19e-04
0.29	6	3.80e-11	1.36e-07	9.06e-08	3.81e-11	1.36e-07	9.06e-08

Table 6.1: Maximum error e_S over all θ, for the stability function (2.9).

		boundary adapted basis			legendre-dubiner basis		
δ	N	SIPG	NIPG	IIPG	SIPG	NIPG	IIPG
0.5	3	4.40e-03	1.16e-02	1.28e-02	4.40e-03	1.16e-02	1.27e-02
0.25	3	2.09e-05	8.24e-04	9.32e-04	2.09e-05	8.24e-04	9.32e-04
0.29	6	2.20e-11	2.68e-07	3.13e-07	2.20e-11	2.68e-07	3.13e-07

Table 6.2: Maximum error e_P over all θ, for the stability function (2.9).

		boundary adapted basis		
δ	N	SIPG	NIPG	IIPG
0.5	3	1.21e-03	4.92e-03	3.49e-03
0.25	3	2.36e-05	2.90e-04	1.94e-04
0.29	6	2.01e-11	8.44e-08	4.85e-08

Table 6.3: Maximum error e_S over all θ, for the stability function (2.10).

		boundary adapted basis		
δ	N	SIPG	NIPG	IIPG
0.5	3	5.61e-04	6.09e-03	4.26e-03
0.25	3	1.41e-05	4.07e-04	2.87e-04
0.29	6	1.17e-11	1.22e-07	8.18e-08

Table 6.4: Maximum error e_P over all θ, for the stability function (2.10).

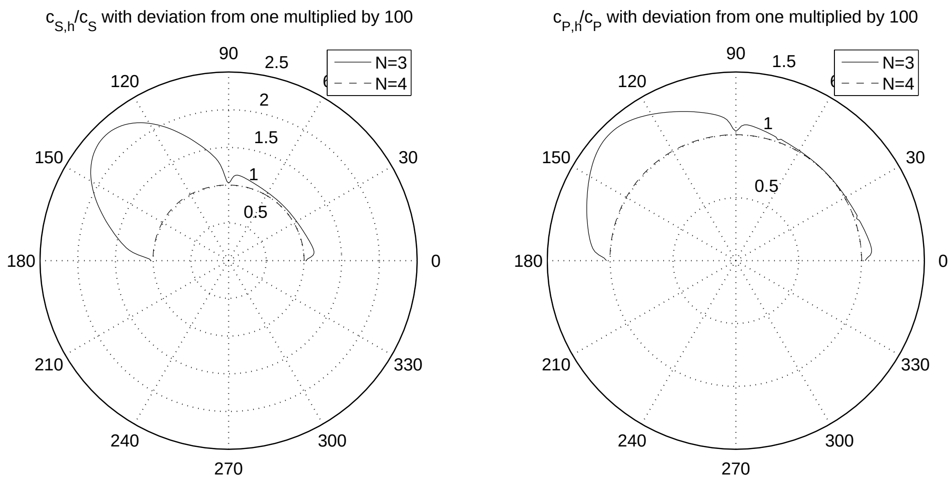
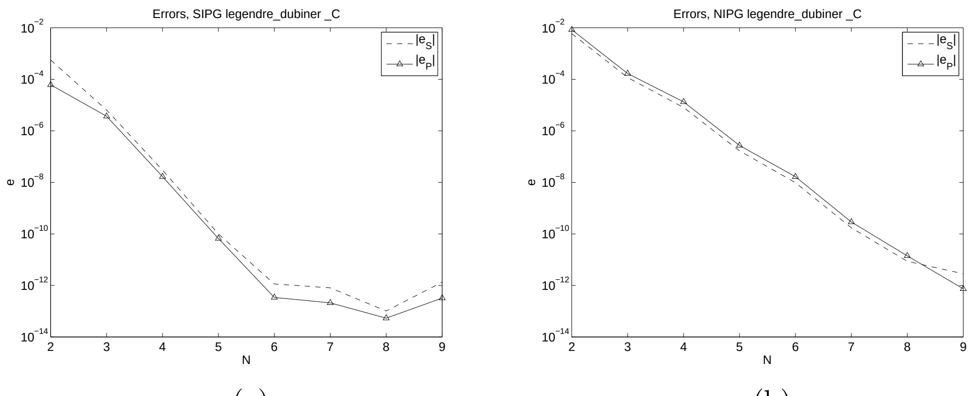
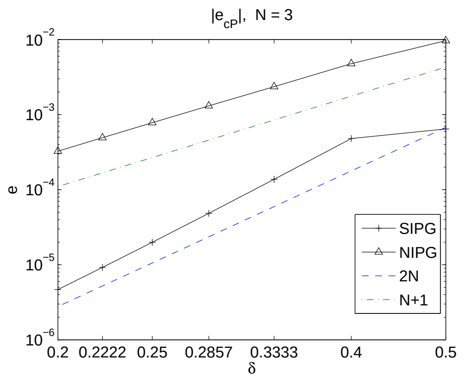


Figure 6.3: Dispersion for a mesh made of right-angled triangles. S-waves are considered on the left, P-waves on the right. Deviations from 1 are magnified by a factor of 100.

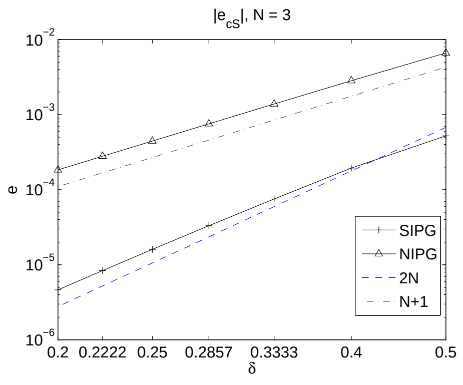


(a) (b)

Figure 6.4: Dispersion error for the semi discrete problem, with respect to the polynomial order N for the (a) SIPG and (b) NIPG methods ($\delta = 0.2$, $\theta = \pi/4$).

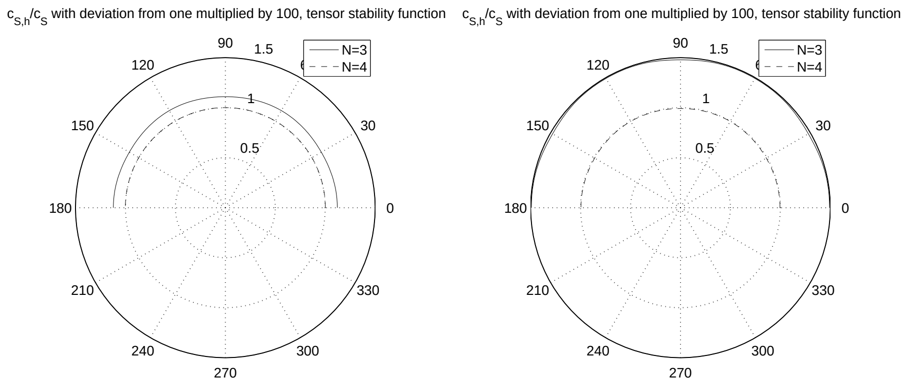


(a)

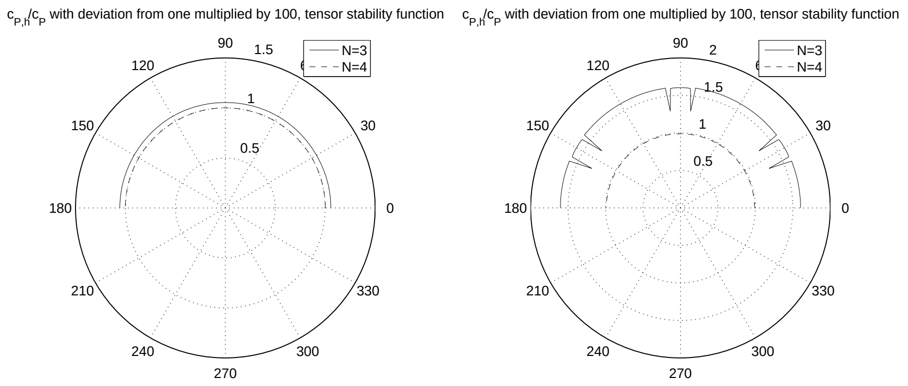


(b)

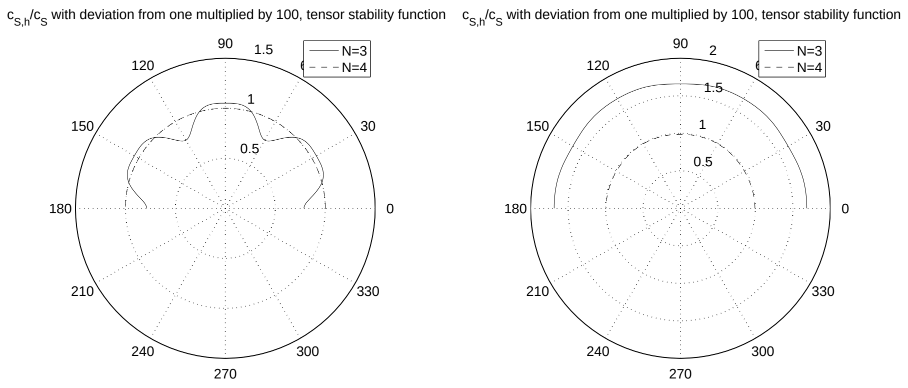
Figure 6.5: Grid dispersion versus δ, $N = 3$, for P–waves (a) and S–waves (b).



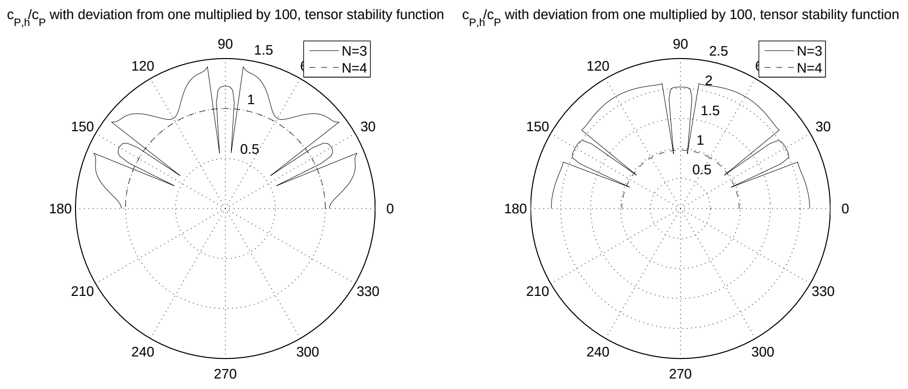
(a) Anisotropy curves $c_{S,h}/c_s$ for the SIPG (left) and NIPG (right) methods, with stability function (2.10).



(b) Anisotropy curves $c_{P,h}/c_P$ for the SIPG (left) and NIPG (right) methods, with stability function (2.10).



(c) Anisotropy curves $c_{S,h}/c_S$ for the SIPG (left) and NIPG (right) methods, with stability function (2.9).



(d) Anisotropy curves $c_{P,h}/c_P$ for the SIPG (left) and NIPG (right) methods, with stability function (2.9).

Figure 6.6: Anisotropy curves for $N = 3$ (-), 4 (- -) and $\alpha = 2$. Deviations from 1 magnified by a factor of 100.

	boundary adapted basis		legendre-dubiner basis	
N	$\Im\omega_{S,h}$	$\Im\omega_{P,h}$	$\Im\omega_{S,h}$	$\Im\omega_{P,h}$
2	-9.6715e-16	-4.2935e-17	-2.5416e-15	-1.5458e-15
3	2.0958e-15	4.7663e-16	2.3934e-15	8.9936e-15
4	-1.8285e-15	2.6355e-16	-1.0875e-14	-1.379e-14
5	-1.1778e-14	5.7042e-16	1.1702e-14	4.6197e-14
6	1.3079e-14	2.1062e-15	4.244e-14	-1.2404e-14
7	1.4445e-14	1.1461e-15	-1.1464e-13	-1.4941e-14
8	9.6095e-15	3.3993e-15	2.0431e-13	1.6433e-13
9	-1.5226e-14	4.9139e-15	-9.1985e-14	5.3414e-14

Table 6.5: Dissipation error for the semi-discrete formulation, $\delta = 0.2$, stability function (2.9), SIPG method.

6.1.3 Fully discrete dispersion error

To analyze the dispersion error for the fully discrete approximation, we start from equation (4.4), perform the same operations needed to obtain (6.4) and substitute the plane wave displacement (6.2). We thus obtain

$$\mathbf{M}\delta t^{-2}(e^{-i\tilde{\omega}_h t^{n+1}} - 2e^{-i\tilde{\omega}_h t^n} + e^{-i\tilde{\omega}_h t^{n-1}})\mathbf{U}_0 = \left(\tilde{\mathbf{C}} + \mathbf{A}\right) e^{-i\tilde{\omega} t^n}\mathbf{U}_0,$$

with $\mathbf{U}_0 = e^{\mathbf{k}\cdot\mathbf{x}}$. This means

$$\mathbf{M}\delta t^{-2}(e^{-i\tilde{\omega}_h \delta t} - 2 + e^{i\tilde{\omega}_h \delta t})\mathbf{U} = \left(\tilde{\mathbf{C}} + \mathbf{A}\right)\mathbf{U}.$$

Noting that

$$e^{-i\tilde{\omega}_h \delta t} - 2 + e^{i\tilde{\omega}_h \delta t} = 4\sin\left(\frac{\tilde{\omega}_h \delta t}{2}\right)^2,$$

we obtain the same eigenvalue problem as in (6.4), this time with

$$\sqrt{\Lambda} = \frac{2}{\delta t}\sin\left(\frac{\tilde{\omega}_h \delta t}{2}\right).$$

The computed pulsation can be then obtained by

$$\tilde{\omega}_h = \frac{2}{\delta t}\sin^{-1}\left(\frac{\delta t}{2}\sqrt{\Lambda}\right). \tag{6.7}$$

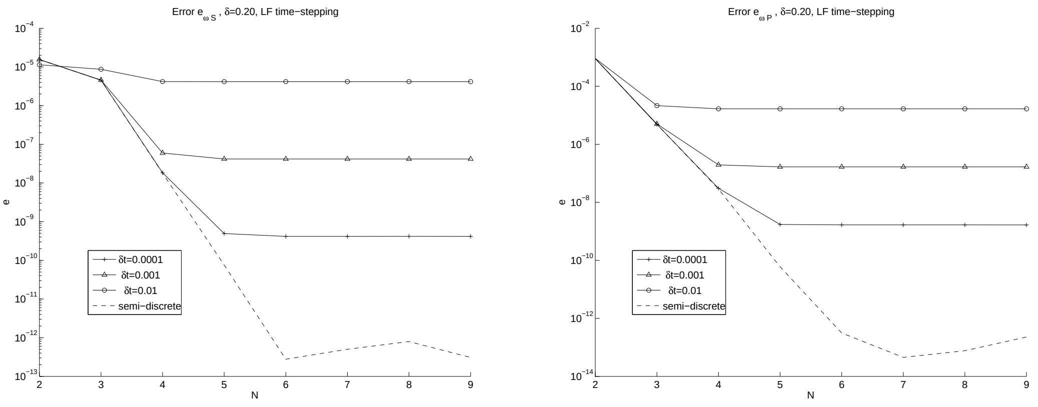


Figure 6.7: Dispersion error for the fully discrete formulation for S-waves (left) and P-waves (right). The dashed lines are the semi-discrete errors, the continuous ones are the fully discrete errors, for $\delta t = 10^{-2,-3,-4}$, $\delta = 0.2$, $\theta = \pi/2$.

For a sufficiently small δt, the right hand side in (6.7) can be expanded as a Taylor series, thus giving

$$\tilde{\omega}_h \sim \sqrt{\Lambda} + \mathcal{O}\left(\delta t^2\right).$$

We then expect the semi-discrete error to be dominant until it reaches a threshold $\sim C\delta t^2$ and the error due to the time discretization to be dominant afterwards. We note the fully discrete errors as

$$\tilde{e}_S = \frac{\tilde{\omega}_{S,h}}{\omega_S} - 1 \qquad \tilde{e}_P = \frac{\tilde{\omega}_{P,h}}{\omega_P} - 1.$$

The numerical results are presented in Figure 6.7 and confirm the expected behaviour.

6.2 Stability and CFL condition

The Leap-Frog time stepping method described in Section 4.2 is an explicit method: this means that the time step δt is restricted by the *Courant-Friedrichs-Lewy* condition. Since compressional waves move faster than the other waves, we conclude that the relation

$$\delta t \leq C_{\mathrm{CFL}} \frac{\delta x}{c_P}, \tag{6.8}$$

limits the time step, if δx represent the smallest distance between nodes. The constant C_{CFL} has to be determined. Following [DBS10] and the references therein, we consider the same setting as in Figure 6.2, though this time we take into account the effect of the mesh size h. Due to the definition of the matrices considered (i.e. considering how they scale with respect to the mesh size), we rewrite the generalized eigenvalue problem as

$$\left(\tilde{\mathbf{C}} + \mathbf{A}\right)\mathbf{U} = \Lambda'\mathbf{M}\mathbf{U},$$

with

$$\Lambda' = \left(\frac{h}{\delta t}\right)^2 \sin\left(\frac{\omega_h \delta t}{2}\right)^2,$$

where the factor 4 disappears since the computations take place on elements of edge length 2. We can then derive an estimate for the stability parameter by setting $q := c_P\delta t/h$, since

$$q \leq \frac{c_P}{\sqrt{\Lambda'}} = C_{\text{CFL}}. \tag{6.9}$$

This relation must hold for any eigenvalue; moreover, the eigenvalues depend on the angle of incidence of the plane wave, i.e. $\Lambda'_j = \Lambda'_j(\theta), \forall j$. Equation (6.9) must therefore hold for $\Lambda'_M = \max_{j,\theta} \Lambda'_j(\theta)$: by numerically estimating this quantity, we can investigate the properties of the approximation. Since an analysis of the eigenvalues of the bilinear form [AH11] shows that $\Lambda \lesssim N^4/h^2$ and since $\Lambda' = h^2\Lambda$, we expect q to scale as $\mathcal{O}(N^{-2})$. This is confirmed by the results presented in Figure 6.8, where q is plotted against the degree of the approximation N, and by the orders shown in Table 6.6, with $q \sim N^{-2}$ around $N = 7$. In Table 6.6 the computed values of q are explicitly listed. The stability function (2.9) provides less restrictive bounds than (2.10), though this comes at the price of higher dispersion, as shown in Section 6.1.1. The NIPG method is more restrictive than the SIPG one, without any gain in the accuracy in this case. Those bounds appear globally more restrictive than those obtained with spectral methods on meshes made of tensor-product elements [AMQR12]. Since the minimum distance between Gauss quadrature nodes δx decreases asymptotically as $\mathcal{O}(N_q^{-2})$ where N_q is the number

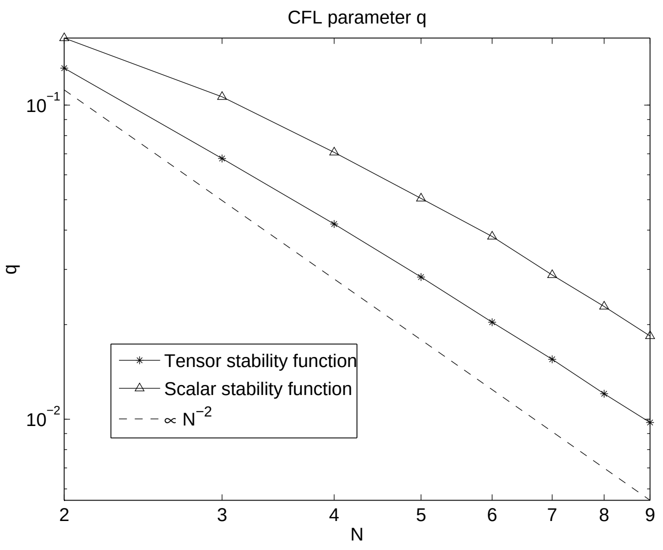


Figure 6.8: Evolution of the parameter q defined in (6.9) with respect to the degree N, for both the stability functions considered, compared to $\mathcal{O}(N^{-2})$.

of quadrature nodes, we also conclude that the modified stability parameter

$$q' = c_P \frac{\delta t}{\delta x},$$

reaches a constant value. Finally, it is worth noting that those values constitute only a necessary bound, though it has been found to be sufficient in practice [DBS10].

	SIPG				NIPG			
	Stability (2.9)		Stability (2.10)		Stability (2.9)		Stability (2.10)	
N	q	order	q	order	q	order	q	order
2	0.16		0.13		0.16		0.11	
3	0.11	-1.1	0.068	-1.6	0.095	-1.2	0.058	-1.5
4	0.071	-1.4	0.042	-1.7	0.062	-1.5	0.037	-1.6
5	0.05	-1.5	0.028	-1.7	0.044	-1.6	0.025	-1.7
6	0.038	-1.5	0.02	-1.8	0.033	-1.6	0.018	-1.8
7	0.029	-1.8	0.016	-1.8	0.025	-1.7	0.014	-1.6
8	0.023	-1.7	0.012	-1.9	0.02	-1.8	0.011	-1.9
9	0.018	-1.8	0.0098	-1.8	0.016	-1.7	0.0091	-1.6

Table 6.6: Computed values of q, defined in (6.9).

Chapter 7

Convergence analysis

In this chapter we analyze the convergence properties of the proposed method in the setting of a test case with a known and smooth exact solution. We therefore consider the domain $\Omega = (-1/2, 1/2) \times (0, 1)$ and we set $\mathbf{f}$ such that

$$\mathbf{u}_{ex} = \cos(t) \begin{pmatrix} \sin(\pi x)\sin(\pi y) \\ \cos(\pi x)\cos(\pi y) \end{pmatrix}, \tag{7.1}$$

is the exact solution to problem (1.1). We will use the discontinuous norm $||\cdot||_{\mathrm{DG}}$ on V_h defined in (2.11) and the time dependent norms

$$||u||_{L^2(0,T;V)} = \left(\int_0^T ||u(t)||_V^2 \mathrm{d}t \right)^{\frac{1}{2}},$$

where V can be $V = L^2(\Omega)$ or $V = V_h$ and $||\cdot||_{V_h} = ||\cdot||_{\mathrm{DG}}$. In practice, those norms are computed discretely and normalized over the time step. All computations are performed with the stability function (2.10), due to its best dispersion properties.

7.1 Convergence with respect to the time step

In this section the error resulting from the time integration is analyzed and compared with the theoretical results. Specifically, the errors introduced by Leap-Frog and implicit Newmark methods are considered and their numerical order of convergence is computed at a high degree of spatial approximation, to rule out any

δt	2.5e-4	5e-4	1e-3	2e-3	4e-3
$\|\|\mathbf{u}_{ex} - \mathbf{u}_h\|\|_{L^2(0,T;L^2(\Omega))}$	4.45e-10	1.68e-09	6.67e-09	2.67e-08	1.07e-07
Rate	1.9161	1.9911	1.9998	2.0007	

Table 7.1: Computed errors for the Leap-Frog scheme.

δt	0.02	0.04	0.08	0.16
$\|\|\mathbf{u}_{ex} - \mathbf{u}_h\|\|_{L^2(0,T;L^2(\Omega))}$	5.36e-06	2.15e-05	8.69e-05	6.88e-04
Rate	2.0066	2.0122	2.9857	

Table 7.2: Computed errors for the implicit Newmark scheme.

spatial discretization errors. It is also shown how the time step δt for the implicit method is not bounded by the CFL condition (6.8).

7.1.1 Leap-frog

The error $||\mathbf{u}_{ex} - \mathbf{u}_h||_{L^2(0,T;L^2(\Omega))}$ is represented in Figure 7.1, for the mesh composed of 4 triangles pictured in Figure 7.2a. The approximation employs the SIPG method, $N = 10$, $T = 2$. The magnitude of the error and the convergence rate are listed in Table 7.1: the theoretical results are confirmed and the scheme exhibits extremely low errors. Note that the time steps δt are all below the condition presented in Section 6.2, since using the computed values of the parameter q we would get $\delta t \leq 5 \cdot 10^{-3}$.

7.1.2 Implicit Newmark

The implicit Newmark method described in Section 4.1 is implemented on the finer grid, shown in Figure 7.2b. The error is shown in Figure 7.3, for the parameters $\zeta = 1/2$ and $\beta = 1/4$. The time step is larger than the previous case, and the CFL condition is not respected: the scheme shown a $\mathcal{O}(\delta t^2)$ convergence as with the leap-frog, while performing only a small fraction of the steps. Nonetheless, explicit methods are more widely used in the approximation of wave propagation phenomena, since implicit methods are unsuited to the analysis of large problems because of their computational cost.

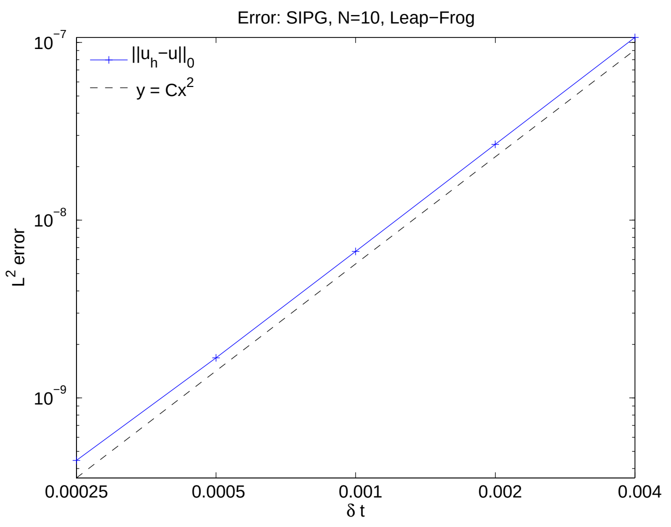


Figure 7.1: Error for the Leap-Frog time approximation and mesh.

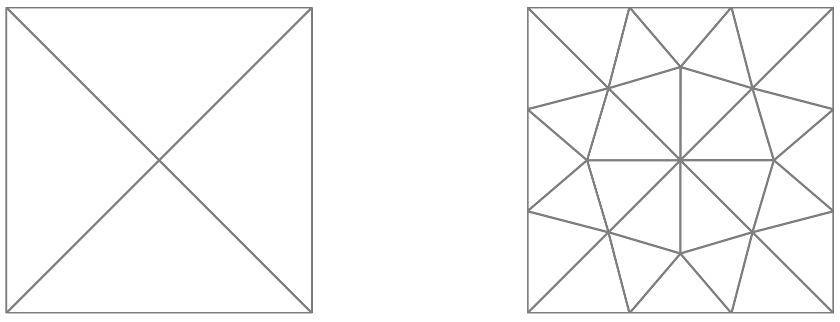


Figure 7.2: Meshes for the Leap-Frog (left) and implicit Newmark analysis (right)

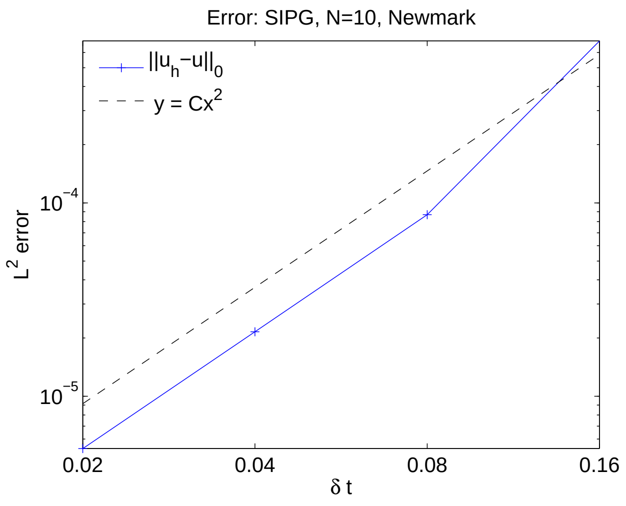


Figure 7.3: Error for the implicit Newmark scheme.

7.2 Convergence with respect to the polynomial order

In this section the relationship between the error $e_h = \mathbf{u}_h - \mathbf{u}_{ex}$ is measured in a suitable norm, and the degree of the basis N is considered. All the results presented use the Leap-Frog time stepping scheme, since the time step δt has to be low enough that the time discretization error is negligible.

In Figure 7.4 we plot the norms $L^2\left(0, T; L^2(\Omega)\right)$ and $L^2\left(0, T; V_h\right)$ of the error versus the polynomial degree, for an approximation with $T = 2$, $\delta t = 2.5 \cdot 10^{-4}$ and the mesh in Figure 7.2a, which is also tabulated in Table 7.3a. The $|| \cdot ||_{V_h}$ error is two orders of magnitude bigger than the L^2 error, but the behavior over N is qualitatively the same. In particular, it seems that there is a strong difference between even and odd polynomial orders: while the order of convergence is exponential stepping from a degree $2N - 1$ to $2N$, while it is almost null from $2N$ to $2N + 1$. The difference between even and odd orders sometimes appears in the framework of interior penalty and, more generally, discontinuous methods, and these results show that spectral elements on simplicial elements can feature this behavior. The investigation of the causes of this behavior is beyond the scope of this work. It is interesting though to consider the error presented in Figure 7.5, which is computed with $T = 1$ and a time step of $\delta t = 5 \cdot 10^{-4}$, this time on the

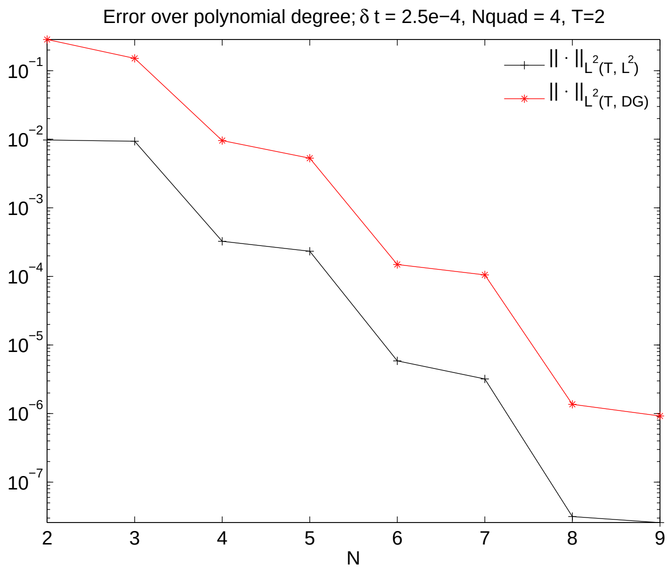


Figure 7.4: Error against degree N, for the mesh made of 4 triangles, $N = 2, \ldots, 9$, $\delta t = 2.5 \cdot 10^{-4}$.

mesh shown in Figure 7.2b. In this case the error slope is smoother than in the previous case and there is no apparent difference between the even and odd orders. Note that triangles in the mesh 7.2a are more deformed than those in 7.2b, and this could constitute a cause for the loss in accuracy.

N	$\|\|e_h\|\|_{L^2(0,T;L^2(\Omega)}$	$\|\|e_h\|\|_{L^2(0,T;V_h)}$
2	9.75e-03	2.85e-01
3	9.36e-03	1.51e-01
4	3.25e-04	9.57e-03
5	2.33e-04	5.30e-03
6	5.87e-06	1.49e-04
7	3.21e-06	1.05e-04
8	3.15e-08	1.36e-06
9	2.58e-08	9.25e-07

(a) Mesh of Figure 7.2a.

N	$\|\|e_h\|\|_{L^2(0,T;L^2(\Omega)}$	$\|\|e_h\|\|_{L^2(0,T;V_h)}$
2	0.0039	0.15
3	0.00037	0.019
4	2.4e-05	0.0016
5	1.9e-06	0.00013
6	9.3e-08	8.9e-06
7	5.1e-09	5.5e-07
8	9.7e-10	6.4e-08
9	9.6e-10	6.8e-08

(b) Mesh of Figure 7.2b.

Table 7.3: Errors for different polynomial orders and for the two meshes considered.

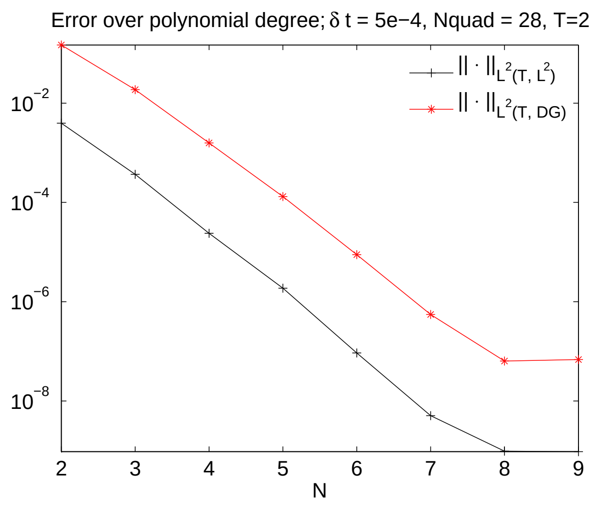


Figure 7.5: Errors against degree N, for a mesh composed of 28 triangles, $N = 2, \dots, 9$, $\delta t = 5 \cdot 10^{-4}$.

Chapter 8

Test cases

In this chapter the discontinuous approximation employing simplicial spectral elements is applied to classical benchmarks of seismic wave propagation problems. The first problem is known as Lamb's [Lam04] problem and its analysis dates back to the beginning of the XIX century. It involves the propagation of the seismic waves in an elastic half plane, with a point source located near the superior edge. The second test case studies the propagation of a surface wave and its impact on two edges, one of an homogeneous material and the other made of two different materials. The third benchmark involves a plane wave impacting on a cylinder enclosed in a different material. In those last two cases the flexibility given by triangular elements and by unstructured grids is of great relevance. The settings presented are widely similar to those listed in [MVSS06]; we used the Legendre-Dubiner basis.

8.1 Elastic half space with a pointwise force

In this test we consider a domain similar to the one represented in Figure 1.1, i.e., an ideally semi-infinite plane with null stress conditions imposed on the edge. In practice, the simulation is carried out on the domain in Figure 8.1, which is a rectangle of width 4000 meters and height 2000 meters, with a point source located at the point $\mathbf{p}_f = (1500, 1950)$, i.e., 50 meters below the free surface, and two receivers r_1 and r_2 located respectively at the points $r_1 = (2200, 2000)$ and

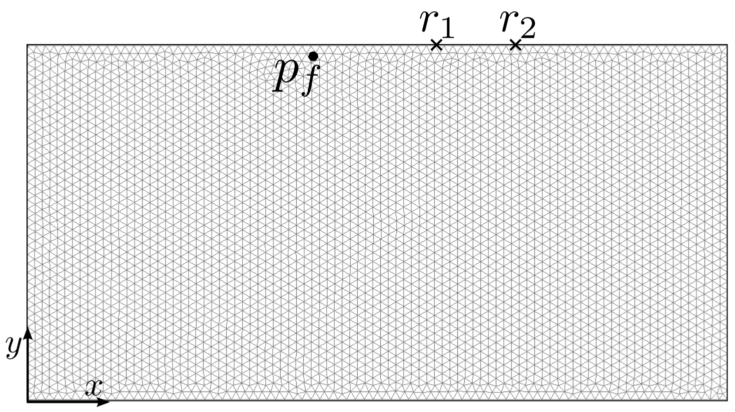


Figure 8.1: The domain for the Lamb's problem simulation.

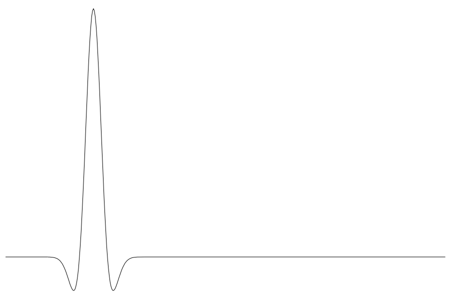


Figure 8.2: Ricker wavelet.

$r_2 = (2700, 2000)$. Null Neumann boundary conditions are imposed on every edge of the domain. The source is spatially pointwise and modulated in time as a Ricker wavelet

$$f = \left(1 - 2(f_m\pi(t - t_0))^2\right) e^{-(f_m\pi)^2(t-t_0)^2}\delta(\mathbf{x} - \mathbf{p}_f) \tag{8.1}$$

with a frequency f_m to be specified, a time delay t_0 and where $\delta(\cdot)$ is the Dirac delta. The time evolution of the function can be seen in Figure 8.2. The material considered in this simulation has the properties

c_P	3200 m/s
c_S	1847.5 m/s
ρ	2000 kg/m^3.

Figure 8.3 provides a close up of the physical phenomenon. Moments after the source activation, there is a P-wave traveling towards the bottom of the image, followed by an S-wave which is traveling almost horizontally. At the edge of the

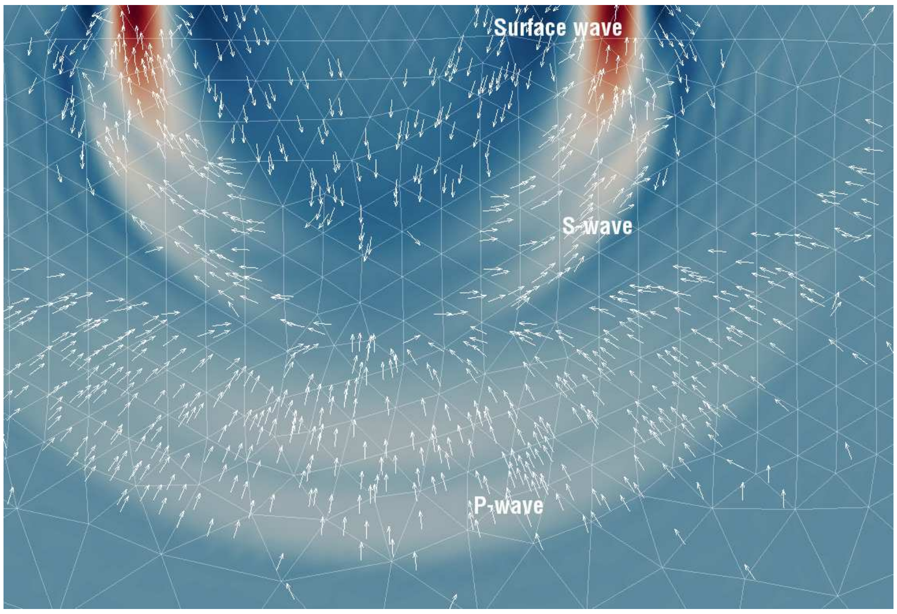


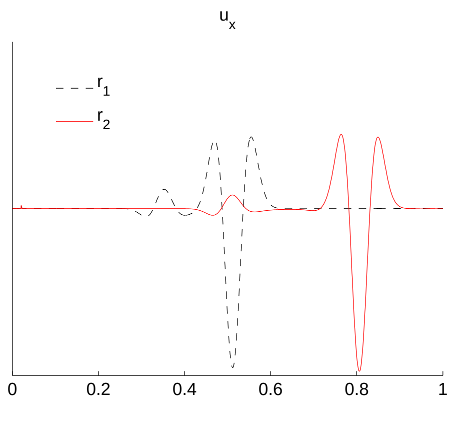
Figure 8.3: Close up of the waves propagating from the source, for a Ricker wavelet pointing upwards.

domain, the Rayleigh wave is clearly visible, and the displacement vectors have the same form as in Figure 1.2. In Figure 8.4 we show the displacements obtained from an approximation on the unstructured mesh in Figure 8.5, where the area around the surface is described more accurately. The main event recorded by the receiver is the transit of the surface wave. This simulation employs polynomials of order 4, a time step $\delta t = 2.5 \cdot 10^{-4}$, and the point source is a Ricker wavelet directed downwards with a frequency $f_m = 10$ Hz and a delay $t_0 = 0.1$. We are then able to recover the structure of the surface wave; it is interesting to compare the computed wave speeds with the analytic ones. The temporal distance Δt between the peaks at the two receivers in Figure 8.4b (computed at the available time steps) is $\Delta t = 0.294$. Considering the distance between the receivers, we get an empirical speed

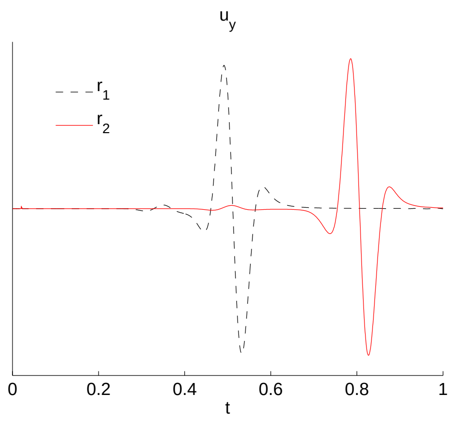
$$c_{h,\mathrm{Surf}} \simeq 1700\,\mathrm{m/s},$$

while equation (1.9) gives a speed

$$c_{\mathrm{Surf}} \simeq 1698.6\,\mathrm{m/s}.$$



(a)



(b)

Figure 8.4: Time plot of the horizontal (left) and vertical (right) displacement at receivers r_1 (- -) and r_2 (–).

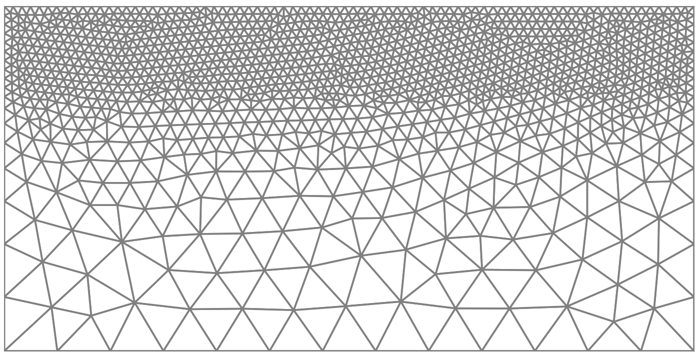


Figure 8.5: Unstructured mesh for the Lamb's problem

Material	c_P [m/s]	c_S [m/s]	ρ [kg/m^3]
I	2500	1250	2000
II	4000	2000	2200

Table 8.1: Properties of the materials for the elastic wedge problem

The error is, therefore, below the temporal resolution of the approximation.

8.2 Propagation of a surface wave in a multi–material domain

In this case, known also as the “elastic wedge problem”, we consider a square domain, which is divided by its diagonal into two different materials, with properties listed in Table 8.1. Specifically, material I occupies the bottom-right side of the square, while material II occupies the top-left one. We set a Rayleigh wave (derived as in Section 1.2.2) as the initial solution and we record what happens at the corners of the domain. Since material II has higher wave speeds, the mesh is finer in that part of the domain; the set up of the receivers and an example of the mesh are shown in Figure 8.6. Null Neumann boundary conditions are imposed on $\partial\Omega$. The recorded seismograms for an approximation with a polynomial order of 8 and time step $\delta t = 5 \cdot 10^{-5}$ are reported in Figures 8.7 and 8.8. In the homogeneous corner (Figure 8.8) the effect of the transmitted Rayleigh wave traveling downwards and of the reflected wave traveling backwards is clearly visible; the conversion to P-wave happening at that corner doesn’t leave a clear track on the seismograms. In the heterogeneous corner, as expected, the reflected wave is evident, while the transmitted one if of definitely inferior amplitude, as is expected. A closer inspection of the computed fields shows that the phenomena described in [MVSS06] are indeed resolved by the method.

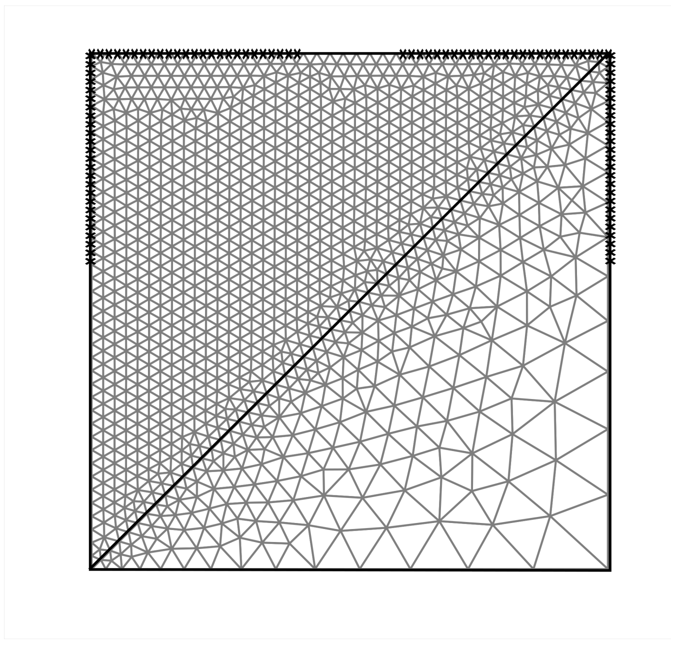


Figure 8.6: Domain, mesh and receivers for the elastic wedge problem.

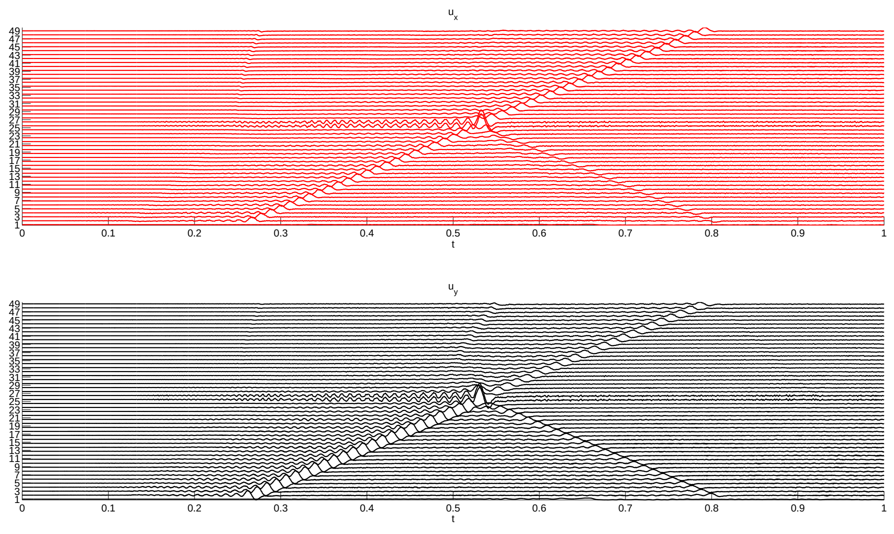


Figure 8.7: Seismograms at the top left (homogeneous) corner: horizontal (above) and vertical (below) components.

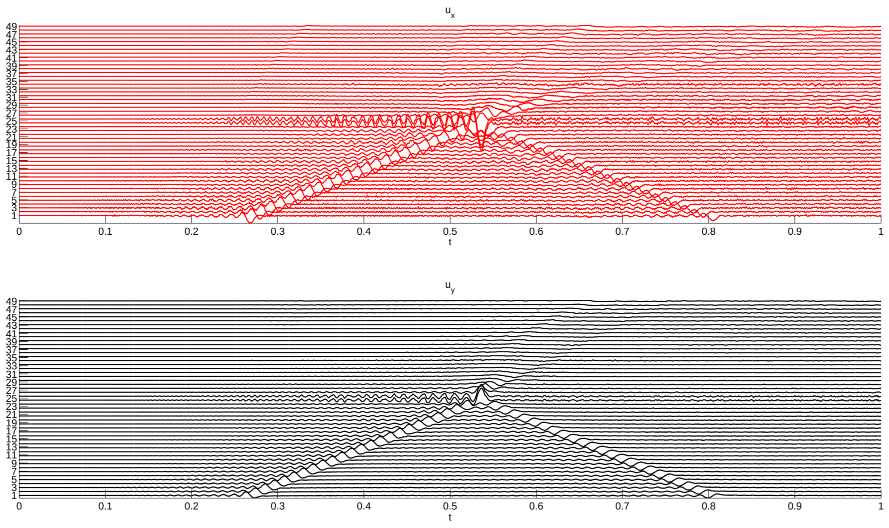


Figure 8.8: Seismograms at the top right (heterogeneous) corner: horizontal (above) and vertical (below) components.

8.3 Plane wave impacting on a cylinder enclosed in a homogenous space

In this case a plane wave impacts on a circular inclusion buried in an homogeneous space, as pictured in Figure 8.10. Its purpose is to test the behavior of the numerical method when confronted with sharp changes in the material properties and the scattering of the plane wave which occurs. The domain is a $2000\,\mathrm{m} \times 2000\,\mathrm{m}$ square, and the circle is located at the center and has a diameter of $500\,\mathrm{m}$. The properties of the two materials are listed in Table 8.2, with material I inside the cylinder and material II outside of it. The plane wave is imposed as an initial conditions and travels downwards starting from $3/4$ of the domain height. It is modulated in time with a 8 Hz central frequency Ricker wavelet and has a P polarization. Periodic boundary conditions are imposed on the left and right boundaries of the domain, in order to correctly model an ideally infinite plane wave.

All the relevant phenomena happening in this case can be seen in Figure 8.9, which represents the displacement field at time $t = 0.232$. Specifically, we see the P-wave reflected from the top of the circle traveling upwards and an S-wave, which

Material	c_P [m/s]	c_S [m/s]	ρ [kg/m^3]
I	3000	1225	2
II	2000	817	1

Table 8.2: Properties of the materials for the enclosed cylinder problem.

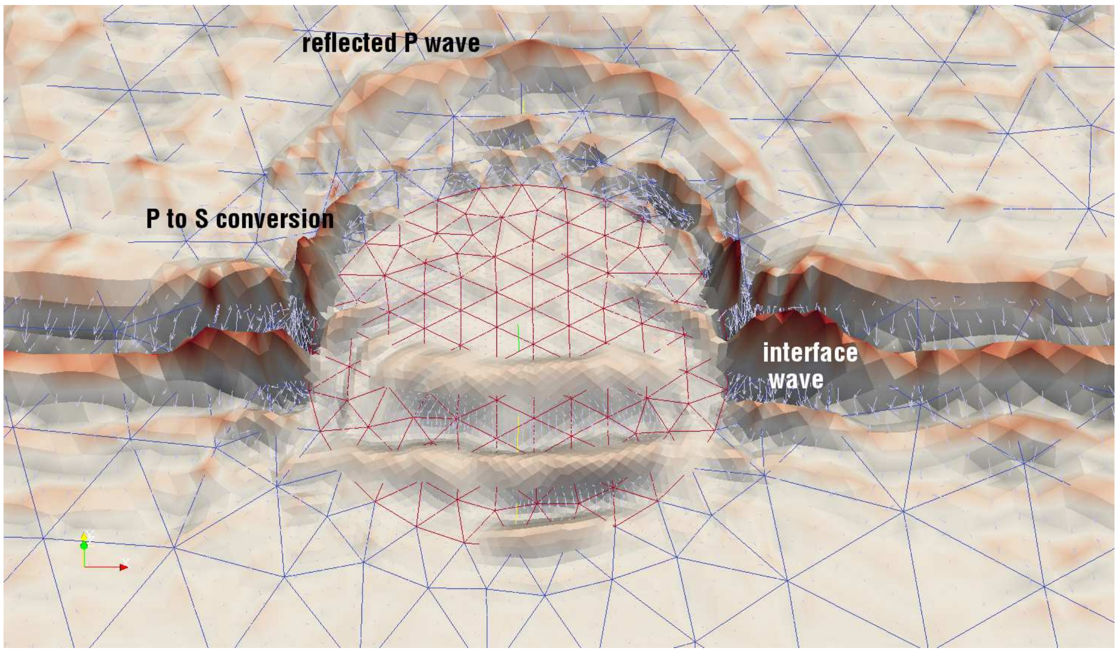


Figure 8.9: Zoom on the enclosed cylinder during the transit of the plane wave.

results from a P to S conversion on the lateral parts of the circle and is partially detached and traveling horizontally. Furthermore, an interface wave is present at the points of contact between the original plane P-wave and the inclusion. Finally, we note how the transmitted waves inside the cylinder are traveling faster than their counterparts in the outside material.

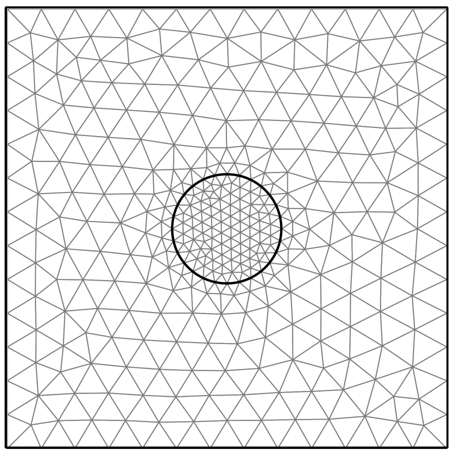


Figure 8.10: Domain for the enclosed cylinder case.

Conclusions and perspectives

Starting from the variational formulation of the elastodynamics equation, we have built a Discontinuous Galerkin spectral element method on meshes made of simplex elements. We have tested its dissipation and dispersion errors and its stability properties in the framework of the Von Neumann analysis. Then, we have tested the method on 2-dimensional benchmark test cases to show its behavior in practical scenarios.

From the numerical results obtained, we conclude that the method presented shows general approximation properties on par with those of the classical discontinuous spectral element method while benefiting from the geometrical flexibility given by simplicial meshes. Specifically, it has optimal dispersion properties and shows small dissipation, and the symmetric interior penalty method reaches a threshold for the error with moderate polynomial orders. The bounds on the CFL condition are slightly more restrictive than those obtained with the SEM on tensor product grids; other time stepping methods should be investigated to verify if this is a consistent property.

The choice of the spectral basis is a delicate issue: the orthonormal basis gives the benefit of a fully explicit method if an explicit time stepping method is considered, while on the other hand the computational cost of a fully discontinuous approximation gets high if a large number of elements is needed. The boundary adapted basis would allow for a continuous approximation in the sub-domains of Ω, but this implies the need to solve a linear system with a block-diagonal mass matrix at every time step. It is anyway worth noting that the boundary adapted basis shows stability, dispersion and dissipation properties of the same quality of both the Legendre-Dubiner basis and the Legendre basis on squares. A comparison of the computational costs of the different approaches should be carried out more extensively.

Moreover, the spectral convergence of the method with respect to the polynomial order, for smooth solutions, is a hint to the fact that the spectral basis on the triangle can have the same approximation properties as the Legendre basis on squares, in the same norms as those considered for the errors.

In any case, a great advantage of the discontinuous non-conforming approach is that it is possible to combine elements of different shapes. It would be therefore interesting to develop a scheme where simplicial elements are employed where the accuracy in the approximation of the domain geometry is important, with tensor product elements elsewhere.

Those results suggest that the development of a three dimensional version of the method would be fruitful, and would provide an interesting tool for the analysis of seismic events in complex geometries. To summarize, a future development of this work can be the construction of a three dimensional scheme, with a wider range of time discretization techniques and with the possibility to use both cubes and tetrahedra.

Appendix A

The C++ library

Alongside with the analytical and numerical study presented, a C++ library implementing the SIPG and Leap-Frog methods has been developed. In this chapter, we will describe its structure and show its usage in two of the test cases presented in Chapter 8. The code is available at `https://github.com/carlomr/tspeed.git`.

The library is designed to be extended by the addition of new bases, quadrature rules and time stepping methods. Furthermore its usage in practical applications has been made as simple as possible. The code may be divided into three layers:

1. the implementation of the geometrical entities, such as
    (a) points, edges and triangles,
    (b) the mesh;
2. the approximation space described in Chapter 3, i.e.,
    (a) shape functions,
    (b) quadrature rules,
    (c) the finite-dimensional space V_h;
3. the approximation of the equation, including
    (a) the matrices for the SIPG spatial approximation introduced in Section 2.3,

(b) the Leap-Frog time stepping method discussed in Section 4.2.

In the sequel those three parts will be presented in detail.

A.1 Geometrical entities

In this section we will present the classes `Point`, `Edge` and `Triangle` and the class `Mesh`. The first three classes are all inheriting some properties, such as the index and boundary condition, from a base class `Entity`, The `Point` class is a simple class implementing a point in a two dimensional space, with some methods to obtain the representing the point in the form of an `Eigen::Vector2d`, in order to implement algebraic operations on the point. The `Edge` class is equally simple, and consists mainly of two points constituting the extremes of the edge and of geometrical manipulation methods. It is worth mentioning that an object of the `Edge` class exposes the public function `length()` which is useful in computing (2.8). The `Triangle` class implements a mesh triangle: we show some of its relevant public functions in Listing A.1. Specifically, the map (5.1) and its inverse are implemented in the function `map` and `invmap`. The Jacobian (5.2) is also implemented, along with its inverse and determinant, in the `Jac`, `invJac` and `detJ` methods respectively. The class also encloses some connectivity information via the `neigh` and `neighedge` functions, which will list the neighbors of the triangle on every edge and the number of every edge in the neighboring triangle once the mesh is assembled.

Listing A.1: Important functions of the `Triangle` class

```cpp
class Triangle : public Entity
{
  public:
  //...
  Eigen::Matrix2d Jac()const;
  Eigen::Matrix2d invJac()const;
  double detJ()const;
  Point map(Point const & p)const;
  Point invmap(Point const & p)const;
  int const & neigh(int i)const;
```

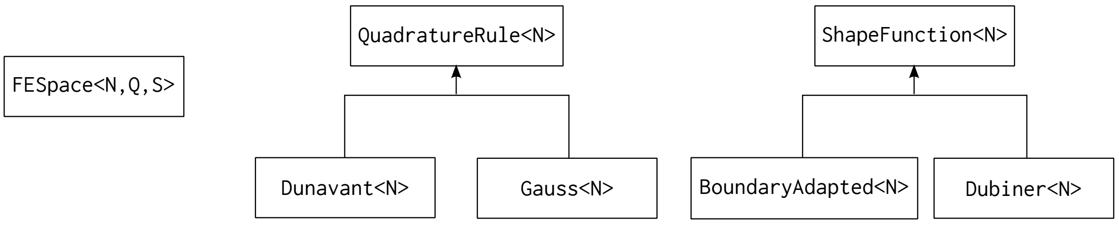


Figure A.1: Classes for the implementation of the approximation space

```
  bool intriangle(const Point & p)const;
  //...
}
```

The `Mesh` class reads a mesh generated with *Gmsh* [GR09], taking the file name as input, and builds a vector of triangles, setting their connectivity and boundary properties. An object of the `Mesh` class exposes the elements via the `elements()` method: since many operation involve cycling over all triangles, the C++11 syntax

```
Mesh Th("meshname.msh")
for(auto ie: Th.elements())
{
   //...
}
```

is used widely throughout the code.

A.2 Approximation space

This part of the code deals with the construction of a spectral element approximation space on the triangles (defined previously). The classes involved in this layer and their hierarchy are shown in Figure A.1.

A.2.1 Shape functions

The base class `ShapeFunction` and its derived classes take the degree N of the polynomial approximation $\mathbb{P}_N$ in (2.1) as a template parameter. They contain two vectors of `std::function` as private members, in which the basis functions and their gradients are stored. In practice, the functions are stored as λ-functions as is shown in Listing A.2, but the use of functors should also be possible. Using this method, the functions may be chosen in a way which is not yet linked to the choice of the quadrature points.

Listing A.2: Dubiner shape functions constructor

```
Eigen::ArrayXd jacobi_polynomial(int N, int alpha, int beta, ↩
    Eigen::ArrayXd const & z);
template<int N>
Dubiner<N>::Dubiner()
{
//...
for(int j = 0; j<=N; ++j) {
     for(int i = 0; i<=N-j; ++i) {
         double cij=std::sqrt((2.*i +1)*2.*(i+j+1)/pow(4.,i));
         this->M_phi.push_back([=](Arr const & csi,Arr const ↩
             &eta)->Arr{return cij*(pow(2.,i))*(pow((1-eta),i)) ↩
             *jacobi_polynomial(i,0,0,2*csi/(1-eta)-1) ↩
             *jacobi_polynomial(j,2*i+1,0,2*eta-1);});
         //...
}}}
```

The basis functions implemented are the Dubiner basis (3.3) and the boundary adapted basis described in Section 3.2.2. All the shape function classes expose public methods which permit their evaluation at an array of points. For the details, see the Doxygen documentation generated by `make doc` and available in the folder `doc/`.

A.2.2 Quadrature rules

The `QuadratureRule` base class stores the edge and internal quadrature nodes, along with the associated quadrature weights. The edge nodes are implemented directly in the base class, since the optimal edge nodes are the Gauss-Legendre ones. This also permits a easier future implementation of a p-adaptable scheme. All classes take the order N of the quadrature rule as a template parameter. The implementation of the internal nodes is delegated to the the derived classes. In the `Gauss` class the nodes are Gauss-Legendre nodes mapped from the reference square to the reference triangle, as shown in (3.2). The `Dunavant` class contains the nodes introduced in Section 3.3.2 and uses the functions provided in [Bur13], where the results from [Dun85] are tabulated. Note that, if N is the order of the polynomial space $\mathbb{P}_N$, the method requires a Gauss quadrature rule of order $\geq N+1$ and a Dunavant quadrature rule of order $\geq 2N$. It is furthermore strongly advised to use Dunavant rules of even order, since some of the odd order rules have negative weights.

The classes expose a public interface which allows the extraction of the nodes and weights desired. The methods are listed in detail in the Doxygen documentation.

A.2.3 Finite dimensional space

The class `FESpace` brings together the chosen shape basis and quadrature rule. Its template parameters are, indeed, the order of the approximation `N`, the quadrature rule `Q` and the shape function `S`. The default is to use the Gauss quadrature of order `N+1` and the Dubiner basis.

Listing A.3: Some methods of the public interface of the `FESpace` class

```
template<int N, typename Q = Gauss<N+1>, typename S = Dubiner<N>>
  class FESpace
{
  public:
  //...
  Eigen::Vector2d grad(unsigned int k, unsigned int i)const;
  Eigen::VectorXd b_edge(unsigned int k, unsigned int iedg)const;
  Eigen::Vector2d g_edge(unsigned int k, unsigned int i, unsigned short ↩
      int edg)const;
  typedef std::function<std::array<double,2>(double,double)> ListFun;
  Eigen::VectorXd transform(ListFun const & fun)const;
  Eigen::VectorXd loc_rhs(Geo::Triangle const & ie, ListFun const & ↩
      fun)const;
  void points_out(std::string const &fname)const;
  void field_out(std::string const &fname, Eigen::VectorXd const &uh, ↩
      unsigned int step)const;
  Eigen::MatrixXd base_mass()const
  Eigen::MatrixXd base_invmass()const
  //...
}
```

The main task of the FESpace class is to compute the values of the shape functions at the quadrature nodes (see the matrix $\mathbf{B}$ in (5.5)) and to provide an interface between the physical function $\mathbf{u}(x, y)$ and its expansion coefficients $\hat{\mathbf{u}}$ introduced in (5.3). A subset of the public methods in shown in Listing A.3, while a description of the full public interface may be found in the documentation. The first three listed functions return the values of the basis functions and of their gradients at the quadrature nodes, the other ones deal with more functional aspects. Specifically, `transform` implements the forward transform (5.6), while `field_out` implements the transform into the physical space (5.7) and writes the values of the field at the points output by `points_out`. The method `loc_rhs` is the $L^2(\texttt{ie})$ projection of a function $\mathbf{f}$, and it returns the vector $\mathbf{r}$ as output, where

$$\mathbf{r}[i] = \int_{\texttt{ie}} \mathbf{f} \cdot \boldsymbol{\psi}_i.$$

The methods `base_mass` and `base_invmass` return the plain mass matrix over the reference element

$$\widetilde{\mathbf{M}}[i, j] = \int_{\mathcal{T}^2} \psi_j \psi_i$$

and its inverse $\widetilde{\mathbf{M}}^{-1}$, respectively. This is extremely useful when dealing with non orthogonal bases, since every sub-block of the mass and inverse mass matrix may be computed by

$$\begin{aligned}\mathbf{M}_K &= \rho_K |J_{\mathbf{F_K}}| \widetilde{\mathbf{M}} \\ \mathbf{M}_K^{-1} &= \left(\rho_K |J_{\mathbf{F_K}}| \widetilde{\mathbf{M}}\right)^{-1}.\end{aligned}$$

A.3 Approximation of the equation

This part of the library deals with the implementation of the SIPG and Leap-Frog methods for the elastodynamic equation. First, we will introduce the part regarding the algebraic structure and the spatial approximation of the equation. Next, the implementation of the time stepping method is considered.

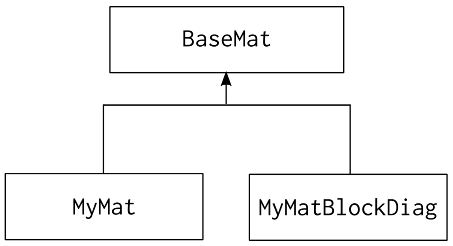


Figure A.2: Inheritance for the monodimensional matrices

A.3.1 Algebraic structure and semi-discrete approximation

In this section we will describe the details of the implementation of the matrices introduced in Section 2.3 and we will adopt the same notation used in that section. We will furthermore call *monodimensional matrix* any of the matrices $\mathbf{S}^{ij}, \mathbf{A}^{ij}, \mathbf{I}^{ij}$, for $i, j = 1, 2$ and $\mathbf{M}^i$, for $i = 1, 2$.

Matrix structure

Figure A.2 shows the inheritance structure for the classes involved in the implementation of the monodimensional matrices. Since those matrices have a block-sparse structure which may be determined from the mesh connectivity, the implementation exploits this property. The class `MyMat` uses the compressed sparse row (CSR) format, where instead of the single values, the full blocks are stored. This is well suited for the description of the monodimensional blocks of the matrices $\mathbf{S}$ and $\mathbf{I}$. The block at row J and column K thus corresponds to $\mathbf{S}^{\ell m}_{JK}$ or $\mathbf{I}^{\ell m}_{JK}$. Noting that every row can contain a maximum of 4 blocks (the diagonal one and the ones referring to the three neighbors), we see how every element is accessed in constant time. This implementation has many advantages, since the matrix structures are never modified and every local block can be computed and copied into the monodimensional matrices once, thus reducing the access to the sparse matrix. The class `MyMatBlockDiag` is the class for the monodimensional blocks of $\mathbf{M}$ and $\mathbf{A}$. It follows the same idea as `MyMat`, but, since those matrices are block-diagonal, there is no need to use a CSR format, and only the index K is used to point to the block $\mathbf{M}^{\ell}_{K}$ or $\mathbf{A}^{\ell m}_{K}$. see how classes which handle the implementation of the full matrices are the template classes `MyMatMultiDim<T>` and `MyMatMultiDimBlockDiag<T>`. Those classes are mainly a container of the monodimensional blocks of the matrices considered. Specifically, the former contains the 4 blocks $\mathbf{C}^{\ell m}$, for $\ell, m = 1, 2$. They are therefore used for $\mathbf{S}$ and $\mathbf{I}$ if `T=MyMat` and for $\mathbf{A}$ if `T=MyMatBlockDiag`. The class `MyMatMultiDimBlockDiag<T>`, with `T=MyMatBlockDiag`, is, on the other hand, designed for $\mathbf{M}$. Those classes implement some algebraic operation needed for the implementation of the time advancement method and for the access to their elements: those are detailed in the documentation.

Semi-discrete approximation

The computation of the matrices deriving from the semi-discrete approximation is (2.17) is done in the `Matrices` class. Every matrix is defined as an object of the suitable class among those defined in the previous section and is stored as a private member in the class. The core cycle of the constructor is shown in Listing A.4: every element is considered, and the local blocks of the matrices are computed and stored in the global matrices. In practice, this class is not constructed directly: it will instead be called by the `TimeAdvance` class, which is the class taking care of the actual numerical solution to the problem.

Listing A.4: Core of the constructor for the `Matrices` class

```
template <int N, typename Q, typename T >
   Matrices::Matrices(FESpace_ptr<N,Q,T> Xh, Parameters const & ↩
       P):A(Xh->mesh(), Xh->nln()),S(Xh->mesh(), ↩
       Xh->nln()),I(Xh->mesh(), Xh->nln()),M(Xh->mesh(), ↩
       Xh->nln()),invM(Xh->mesh(), Xh->nln())
{
  //...
  for(auto ie: Xh->mesh()->elements())
  {
      sigmaeps(ie, Xh, sigma, eps, P.lambda(ie.id()), P.mu(ie.id()));
      sigmaeps_edge(ie, Xh, sigma_edge, eps_edge, P.lambda(ie.id()), ↩
          P.mu(ie.id()));
      stress_loc(ie, sigma, eps, Xh);
      stability(Xh, ie, P);
      interelement(Xh, ie, sigma_edge);
      mass(Xh, ie, P.rho(ie.id()));
   }
   I.symmetrize();
}
```

A.3.2 Time stepping

The base class TimeAdvance and the derived class LeapFrog take care of the time advancement and of the solution to the fully discrete linear system. An extract of the public interface of the class is shown in Listing A.5. It consists of methods used to set some parameters in the approximation (such as δt, T, α in (2.8), $\mathbf{u}_0$ and $\mathbf{v}_0$) and of the stepping methods `first_step()` and `step()`. Two additional methods are shown: they deal with the recording of the solution at the receivers and with the output of the recorded values to a file.

The class LeapFrog implements the methods `step` (4.4) and `first_step` (4.3) introduced in Section 4.2, while the rest of the public interface is inherited. This structure was developed to allow for the expansion of the available time stepping methods. The usage of the time stepping class will be more clear from the examples presented next.

Listing A.5: Extract of the public interface of the class TimeAdvance

```
class TimeAdvance
{
public:
    template <int N, typename Q, typename S>
        TimeAdvance(FESpace_ptr<N,Q,S> Xh, Parameters const & p, Receivers ↩
            const & r);
    void first_step();
    void step();
    void set_dt(double dt){M_dt = dt;};
    void set_tmax(double tmax);
    void set_penalty(double p);
    void add_force(std::shared_ptr<Force> f);
    template<int N, typename Q, typename S>
        void set_initial_v(FESpace_ptr<N,Q,S> Xh, ↩
            std::function<std::array<double,2>(double,double)> fun);
    template<int N, typename Q, typename S>
        void set_initial_u(FESpace_ptr<N,Q,S> Xh, ↩
            std::function<std::array<double,2>(double,double)> fun);
    void eval_receivers();
    void write_receivers(std::string const & fn)const;
    //...
}
```

A.4 Examples

In this section we will consider the implementation of the Lamb's problem (Section 8.1) and of the elastic wedge problem (Section 8.2).

A.4.1 Lamb's problem implementation

The code starts with the inclusion of the TSPEED header file and with the declaration of the namespace:

```
#include"TSPEED.hpp"
using namespace Tspeed;
```

After the definition of some parameters, the mesh is loaded and some statistics are printed (number of elements, geometrical properties etc.):

```
Mesh_ptr Th(new Mesh(std::string("./Meshes/Lamb_fullyunstruct3.msh")));
Th->stats();
```

The functional space is thus created: this example employs the Dunavant quadrature rule and the boundary adapted basis. An object of the class parameters is created and the material properties are assigned.

```
FESpace_ptr<N, Dunavant<2*N>, BoundaryAdapted<N>> Xh(new ↩
    FESpace<N,Dunavant<2*N>,BoundaryAdapted<N>>(Th));

Parameters p(Th);

p.setp("lambda", 6, 6.8270e+09);
p.setp("mu", 6,6.8265125e+09);
p.setp("rho", 6, 2000 );
```

Note that the number 6 is the region number: this is defined in the `geo` file from which the mesh is generated. Indeed, in the `geo` file generating this mesh, the surface was defined by

```
Plane Surface(6) = {5} ;
```

where 5 indicates the closed line enclosing the domain. Next, the receivers are loaded and the pointwise Ricker wavelet force is created:

```
Receivers r2(Xh, std::string("Receivers/lamb.rcv"));
Force_ptr f( new PointWiseForce([](const double & t){return ↩
    std::array<double,2>{{0,-(1-2*M_PI*M_PI*(100)*(t-0.1)*(t-0.1)) * ↩
    exp(-M_PI*M_PI*(100)*(t-0.1)*(t-0.1))}};}, Geo::Point(1500,1950), Xh));
```

The time advancing object is now created, and the parameters and initial displacement and velocity are prescribed:

```
LeapFrog TA(Xh, p, r2);

TA.set_dt(dt);
TA.set_tmax(tmax);
TA.add_force(f);
TA.set_penalty(1);
TA.set_initial_u(Xh,[](double x, double y){return ↩
    std::array<double,2>{{0,0}};});
TA.set_initial_v(Xh,[](double x, double y){return ↩
    std::array<double,2>{{0,0}};});
```

Since everything is now set, the first step of the method is performed, and the displacement at the receivers is recorded

```
TA.first_step();
TA.eval_receivers();
```

and, consequently, the method advances until the final time is reached, while recording the displacement at the receivers every 2 steps.

```
int step = 0;
while(TA.is_running())
{
    t+=dt;
    TA.step();
    if(++step%2 == 0)
```

```
       TA.eval_receivers();
}
```

Finally, the recorded position are written to file (one file per receiver is output).

```
TA.write_receivers("Receivers_out/lamb");
```

A.4.2 Elastic wedge problem implementation

The code for this case is widely similar to the one presented for Lamb's problem, though in this case the functional space is defined as

```
FESpace_ptr<N> Xh(new FESpace<N>(Th));
```

which amounts to choosing the Gauss quadrature rule and the Dubiner basis. Furthermore, we show the ability of the library to deal with multiple domains: the material properties are assigned as

```
p.setp("lambda", 1, 6.25e+09);
p.setp("lambda", 2, 1.76e+10);

p.setp("mu", 1, 3.125e+09);
p.setp("mu", 2, 8.8e+09);

p.setp("rho", 1, 2000);
p.setp("rho", 2, 2200);
```

where, as before, the mesh was generated from a `geo` containing the lines

```
Plane Surface(2) = {5};
Plane Surface(1) = {6};
```

In this case the force is null and the initial Rayleigh wave has a quite complicated structure which may be found in the source file. Finally, inside the time loop we print the field every 20 steps and record the displacement at the receivers every second step.

```
while(TA.is_running())
{
    ++step;
    t+=dt;
    TA.step();
    if(step%2 == 0)
        TA.eval_receivers();
    if(step%20==0)
        Xh->field_out("Fields_out/wedge_field", TA.u(), step);
}
```

Bibliography

[ABCM02] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM J. Numer. Anal. **39** (2001/02), no. 5, 1749–1779. MR 1885715 (2002k:65183)

[AH11] P. F. Antonietti and P. Houston, *A class of domain decomposition preconditioners for hp-discontinuous Galerkin finite element methods*, J. Sci. Comput. **46** (2011), no. 1, 124–149. MR 2753254 (2012a:65318)

[AMQR12] P. F. Antonietti, I. Mazzieri, A. Quarteroni, and F. Rapetti, *Nonconforming high order approximations of the elastodynamics equation*, Comput. Methods Appl. Mech. Engrg. **209/212** (2012), 212–238. MR 2877966 (2012m:74016)

[Arn82] D. N. Arnold, *An interior penalty finite element method with discontinuous elements*, SIAM J. Numer. Anal. **19** (1982), no. 4, 742–760. MR 664882 (83f:65173)

[BTW01] L. Bos, M. A. Taylor, and B. A. Wingate, *Tensor product Gauss-Lobatto points are Fekete points for the cube*, Math. Comp. **70** (2001), no. 236, 1543–1547 (electronic). MR 1836917 (2002b:65035)

[Bur13] J. Burkardt, `http://people.sc.fsu.edu/~jburkardt/cpp_src/dunavant/dunavant.html`, 2013.

[CHQZ06] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods*, Scientific Computation, Springer-Verlag, Berlin, 2006, Fundamentals in single domains. MR 2223552 (2007c:65001)

[CHQZ07] ______, *Spectral methods*, Scientific Computation, Springer, Berlin, 2007, Evolution to complex geometries and applications to fluid dynamics. MR 2340254 (2009d:76084)

[DBS10] J. D. De Basabe and M. K. Sen, *Stability of the high-order finite elements for acoustic or elastic wave propagation with high-order time stepping*, Geophysical Journal International **181** (2010), no. 1, 577–590.

[DD76] J. Douglas, Jr. and T. Dupont, *Interior penalty procedures for elliptic and parabolic Galerkin methods*, Computing methods in applied sciences (Second Internat. Sympos., Versailles, 1975), Springer, Berlin, 1976, pp. 207–216. Lecture Notes in Phys., Vol. 58. MR 0440955 (55 #13823)

[Dub91] M. Dubiner, *Spectral methods on triangles and other domains*, Journal of Scientific Computing **6** (1991), no. 4, 345–390.

[Dun85] D. A. Dunavant, *High degree efficient symmetrical Gaussian quadrature rules for the triangle*, Internat. J. Numer. Methods Engrg. **21** (1985), no. 6, 1129–1148. MR 794241 (86h:65029)

[GLMH09] G. J. Gassner, F. Lörcher, C.-D. Munz, and J. S. Hesthaven, *Polymorphic nodal elements and their application in discontinuous Galerkin methods*, J. Comput. Phys. **228** (2009), no. 5, 1573–1590. MR 2494228 (2010b:76074)

[GR09] C. Geuzaine and J.-F. Remacle, *Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities*, Internat. J. Numer. Methods Engrg. **79** (2009), no. 11, 1309–1331. MR 2566786 (2010j:65262)

[Hes98] J. S. Hesthaven, *From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex*, Siam Journal on Numerical Analysis **35** (1998), no. 2, 655–676.

[HT00] J. S. Hesthaven and C.-H. Teng, *Stable spectral methods on tetrahedral elements*, Siam Journal on Scientific Computing **21** (2000), no. 6, 2352–2380.

[HW08] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods*, Texts in Applied Mathematics, vol. 54, Springer, New York, 2008, Algorithms, analysis, and applications. MR 2372235 (2008k:65002)

[Koo75] T. Koornwinder, *Two-variable analogues of the classical orthogonal polynomials*, Theory and application of special functions (Proc. Advanced Sem., Math. Res. Center, Univ. Wisconsin, Madison, Wis., 1975), Academic Press, New York, 1975, pp. 435–495. Math. Res. Center, Univ. Wisconsin, Publ. No. 35. MR 0402146 (53 #5967)

[KS05] G. E. Karniadakis and S. J. Sherwin, *Spectral/hp element methods for computational fluid dynamics*, second ed., Numerical Mathematics and Scientific Computation, Oxford University Press, New York, 2005. MR 2165335 (2006j:65001)

[KT02] D. Komatitsch and J. Tromp, *Spectral-element simulations of global seismic wave propagation—i. validation*, Geophysical Journal International **149** (2002), no. 2, 390–412.

[KV98] D. Komatitsch and J.-P. Vilotte, *The spectral element method: An efficient tool to simulate the seismic response of 2d and 3d geological structures*, Bulletin of the Seismological Society of America **88** (1998), no. 2, 368–392.

[KWBH09] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven, *Nodal discontinuous Galerkin methods on graphics processors*, J. Comput. Phys. **228** (2009), no. 21, 7863–7882. MR 2573336 (2011a:65320)

[Lam04] H. Lamb, *On the propagation of tremors over the surface of an elastic solid*, Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character **203** (1904), pp. 1–42 (English).

[LJ75] J. N. Lyness and D. Jespersen, *Moderate degree symmetric quadrature rules for the triangle*, J. Inst. Math. Appl. **15** (1975), 19–32. MR 0378368 (51 #14536)

[MP89] Y. Maday and A. T. Patera, *Spectral element methods for the incompressible Navier-Stokes equations*, State-of-the-art surveys on computational mechanics (A90-47176 21-64), American Society of Mechanical Engineers, 1989, pp. 71–143.

[MVSS06] E. D. Mercerat, J.-P. Vilotte, and F. J. Sánchez-Sesma, *Triangular spectral element simulation of two-dimensional elastic wave propagation using unstructured triangular grids*, Geophysical Journal International **166** (2006), no. 2, 679–698.

[Pat84] A. T. Patera, *A spectral element method for fluid dynamics: Laminar flow in a channel expansion*, Journal of Computational Physics **54** (1984), no. 3, 468 – 488.

[PdlPA$^+$12] C. Pelties, J. de la Puente, J.-P. Ampuero, G. B. Brietzke, and M. Käser, *Three-dimensional dynamic rupture simulation with a high-order discontinuous galerkin method on unstructured tetrahedral meshes*, Journal of Geophysical Research: Solid Earth **117** (2012), no. B2, n/a–n/a.

[PGA$^+$13] C. Pelties, A. Gabriel, J.-P. Ampuero, J. de la Puente, and M. Käser, *The ADER-DG method for seismic wave propagation and earthquake rupture dynamics*, EGU General Assembly Conference Abstracts, EGU General Assembly Conference Abstracts, vol. 15, April 2013, p. 1182.

[PR10] R. Pasquetti and F. Rapetti, *Spectral element methods on unstructured meshes: which interpolation points?*, Numerical Algorithms **55** (2010), no. 2-3, 349–366.

[QSS07] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical mathematics*, second ed., Texts in Applied Mathematics, vol. 37, Springer-Verlag, Berlin, 2007. MR 2265914 (2007f:65001)

[RH73] W. H. Reed and T. R. Hill, *Triangular mesh methods for the neutron transport equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[RMK11] B. N. Ryland and H. Z. Munthe-Kaas, *On multivariate Chebyshev polynomials and spectral approximations on triangles*, Spectral and High Order Methods for Partial Differential Equations, Springer, 2011, pp. 19–41.

[RSV12] F. Rapetti, A. Sommariva, and M. Vianello, *On the generation of symmetric lebesgue-like points in the triangle*, Journal of Computational and Applied Mathematics **236** (2012), no. 18, 4925–4932.

[RWG99] B. Rivière, M. F. Wheeler, and V. Girault, *Improved energy estimates for interior penalty, constrained and discontinuous Galerkin methods for elliptic problems. I*, Comput. Geosci. **3** (1999), no. 3-4, 337–360 (2000). MR 1750076 (2001d:65145)

[SK95] S. J. Sherwin and G. E. Karniadakis, *A new triangular and tetrahedral basis for high-order (hp) finite element methods*, Internat. J. Numer. Methods Engrg. **38** (1995), no. 22, 3775–3802. MR 1362003 (96h:65140)

[Sze75] G. Szegő, *Orthogonal polynomials*, fourth ed., American Mathematical Society, Providence, R.I., 1975, American Mathematical Society, Colloquium Publications, Vol. XXIII. MR 0372517 (51 #8724)

[TWB07] M. A. Taylor, B. A. Wingate, and L. P. Bos, *A cardinal function algorithm for computing multivariate quadrature points*, SIAM J. Numer. Anal. **45** (2007), no. 1, 193–205 (electronic). MR 2285850 (2008i:65046)

[War06] T. Warburton, *An explicit construction of interpolation nodes on the simplex*, Journal of Engineering Mathematics **56** (2006), no. 3, 247–262.

[WH03] T. Warburton and J. S. Hesthaven, *On the constants in hp-finite element trace inverse inequalities*, Comput. Methods Appl. Mech. Engrg. **192** (2003), no. 25, 2765–2773. MR 1986022 (2004d:65146)

[WX03] S. Wandzura and H. Xiao, *Symmetric quadrature rules on a triangle*, Comput. Math. Appl. **45** (2003), no. 12, 1829–1840. MR 1995755 (2004e:65026)