UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias de la Tierra, Energía y Ambiente

TÍTULO: Numerical Simulation of Seismic Waves in 2D using The Finite Element Method

Trabajo de integración curricular presentado como requisito para la obtención del título de Geólogo

Autor:

Troya Yunga Ariel Santiago

Tutor:

M.Sc. Pérez Roa Richard

Co-Tutor:

Dr. Manzanilla Morillo Raúl, PhD

Urcuquí, marzo 2024

Autoría

Yo, **Ariel Santiago Troya Yunga**, con cédula de identidad 1720974938, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, marzo 2024

Ariel Santiago Troya Yunga

CI: 1720974938

Autorización de publicación

Yo, **Ariel Santiago Troya Yunga**, con cédula de identidad 1720974938, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, marzo 2024

Ariel Santiago Troya Yunga

CI: 1720974938

Acknowledgment

I'm glad to thank my advisors Richard Pérez and Raúl Manzanilla, who supported me in this work with their help and their collaboration to present this project.

Ariel Santiago Troya Yunga

Resumen

En Geología, la obtención de modelos del subsuelo conduce a una mejor comprensión del mismo. Generalmente, estos estudios se realizan mediante métodos geofísicos; el más utilizado son los estudios sísmicos. Una de las técnicas para encontrar modelos de velocidad del subsuelo mediante métodos sísmicos es la Inversión de Onda Completa (FWI por sus siglas en inglés). Para utilizar este método es necesario resolver numéricamente la ecuación de la onda elástica. El método más comúnmente utilizado ha sido el de diferencias finitas, sin embargo, este método conduce a errores durante largos períodos de tiempo de simulación debido a la acumulación de errores. Por otro lado, este método implementa las condiciones de contorno en la solución de la ecuación elástica a posteriori. Para evitar estos problemas, se propone resolver numéricamente el ecuación de onda elástica usando el método de elementos finitos. Para tener un mejor enfoque, primero se resuelve el caso en 1D y se implementa en Python para comparar ambos métodos. El caso 2D se resuelve de forma explícita para el tiempo construyendo las matrices de rigidez y masa, estableciendo claramente cómo funciona el método, el cual es de elementos finitos en el espacio y diferencias finitas en el tiempo. Posteriormente, se crea un código en Python para construir las matrices anteriormente mencionadas. Se espera que este método pueda utilizarse para mejorar la simulación de ondas sísmicas como parte de un proyecto de investigación más amplio de FWI.

Palabras Clave:
Inversión completa de forma de onda, método de diferencias finitas, método de elementos finitos.

Abstract

In Geology, obtaining subsurface models leads to a better understanding of it. Generally, these surveys are carried out through geophysical methods; the most widely used is seismic surveys. One of the techniques to find subsurface velocity models through seismic methods is the Full-Wave Inversion (FWI). To use this method is necessary to solve the elastic wave equation numerically. The most common method used has been finite differences. However, this method leads to errors for long simulation periods in time due to error accumulation. On the other hand, this method does not include the boundary conditions in the elastic equation solution. To avoid these issues, it is proposed to solve numerically the elastic wave equation using the finite element method. To have a better approach, it is solved the case in 1D and implemented it in Python to compare both methods. The 2D case is solved explicitly for the time building the mass and stiffness matrices, and establishing clearly how the method works, which is Finite Elements for the space and Finite Differences for the time. Afterward, it is written a script in Python to build the matrices. It is hoped that this method could be used to improve the simulation of seismic waves as part of a larger research project of FWI.

Keywords:
Full waveform inversion, finite-difference method, finite-element method.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Full Waveform Inversion

In Geology, obtaining subsurface models leads to a better understanding of it. These subsurface models can be approached as a forward problem which consists of predicting observations whether initial conditions and model parameters are known or an inverse problem which infers initial conditions and a model from observations (Fig.1.1). Generally, models are carried out through geophysical methods; the most commonly used is seismic surveys. One of the most used techniques to find subsurface velocity models is the Full Waveform Inversion (FWI). An inverse problem seeks a model such that differences between the expected data and the observed data are minimum in the least-squares sense.

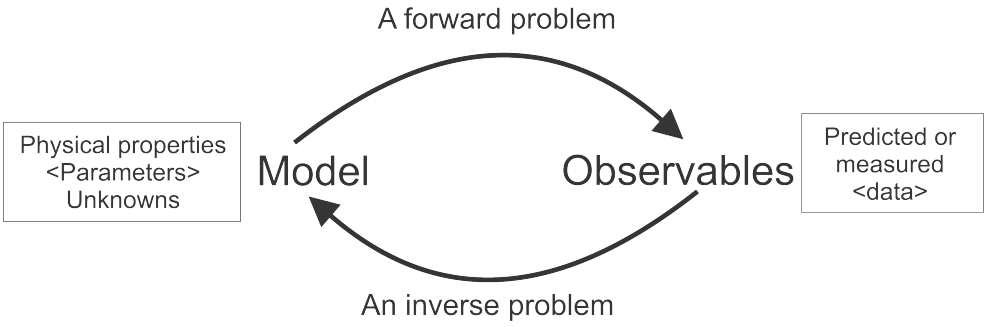

Figure 1.1: Scheme of inputs and outputs of Forward and Reverse problems

Full waveform inversion (FWI) is a high-resolution seismic imaging technique that is based on using the entire content of seismic traces for extracting physical parameters of the medium sampled by seismic waves [1]. This means finding an optimal model in which the

computed shot gathers reproduces the observed shot data. The process of finding a correct model is not trivial. Since this procedure depends on the reliability of the low-frequency content of the observed data, the proper strategy will be to determine the model iteratively, typically by successively introducing higher frequencies [8] (Fig. 1.2). Furthermore, the user needs to provide an adequate wave equation to generate synthetic wave fields and associated shot gathers in order to simulate wave propagation. Note that, due to the limited data acquisitions from the surface only and its frequency band, FWI may lead to more than one solution.

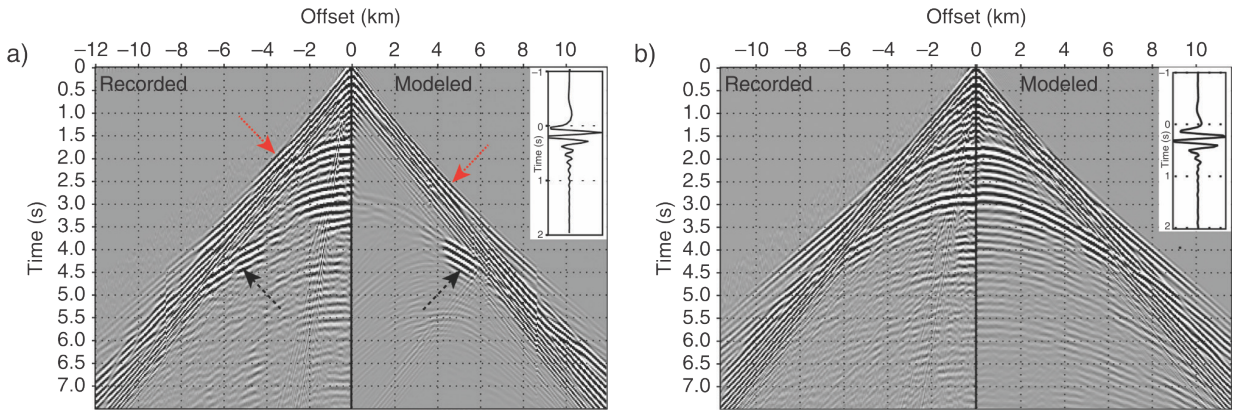

Figure 1.2: A common receiver gathers in (a) the initial model (b) the final model. Red arrows show first-arrival phases while black arrows indicate post-critical phases. The source signal is shown as an insert and differs between the initial and final models (Taken from [1]).

The objective of FWI is to minimize the least-squares misfit function (The development and deduction is taken from [1]).

$$J(m) = \frac{1}{2}||d_{calc}(m) - d_{obs}||^2, \tag{1.1}$$

where $m(x)$ is the model to be determined with $x = (x,y,z)$ spatial coordinates, the observed data $d_{obs}(s,r,t)$ at position s, receiver position r, and time t. The calculated data d_{calc} are a function of the m model and are the solution of the wave equation.

$$L(m)d_{s,x,t} = \delta(s-x)\Omega(t), \tag{1.2}$$

$$d_{calc}(s,r,t) = d(s, x=r, t), \tag{1.3}$$

where L is the wave equation operator, d the wave field, $\Omega(t)$ the seismic source wavelet,

and δ the Dirac distribution. This means that the wave field d is the solution of the wave equation for a point source located at x=s and for a seismic source wavelet.

The evaluation of *J(m)* is the first step; however, which one seeks a more suitable model. This technique turns into an iterative process. The better approach is to use a gradient-based inversion, where the model is iteratively determined. The requirements are:

- an initial model;
- computation of the objective function gradient

This procedure is related to the minimization under constraints, with the introduction of Lagrangian multipliers $\lambda(s, x, t)$. Therefore, the derivation of the gradient requires three elements:

- computation of the forward wavefield d (Eq. (1.2)) for each source,
- computation of the backward residual wave field λ, for each source. This is obtained by solving the wave equation for a source term, being the residual wave field at the receiver position $d_{calc} - d_{obs}$,
- cross-correlation between d and λ, with summation over all times, but for fixed spatial x points.

Once the gradient is computed, the new model is updated as follows:

$$m_{n+1} = m_n - \alpha \frac{\partial J}{\partial m} \tag{1.4}$$

where the gradient is $\frac{\partial J}{\partial m}$ and $\alpha > 0$ a scalar length. In practice, FWI can select an interval of data around certain windows (an implement function to avoid unwanted jumps and downs of data by smoothing the curve). For example, to only include a zone around the direct arrivals or to remove ground roll. This means that equation (1.1) is modified such:

$$J(m) = \frac{1}{2}||M(d_{calc}(m) - d_{obs})||^2 \tag{1.5}$$

where M is a mask in the data domain. It is important to be careful to avoid aliasing and leakage.

1.2 Introduction to Numerical Methods

To apply the FWI, it must be provided the correct wave equation which, in this case, is for the elastic seismic waves. This equation can be numerically solved using several methods, each one has pros and cons related to its methodology. The more commonly used methods are the spectral-element method, the finite-volume method, the finite-difference method, and the finite-element method.

The Finite-Difference Method (FDM)

The Finite Difference Method is the numerical method most often used to approximate solutions in several areas where different Partial Differential Equations (PDEs) and Ordinary Differential Equations (ODEs) appear. The procedure to apply this scheme is to replace the differential operator with a linear combination of function values over the given point surrounding the point where one wants to replace the differential operator, the most common strategy is using the Taylor series [9] to approximate the spatial and temporal derivatives by using the model values at nearby points [10] (Fig. 1.3).

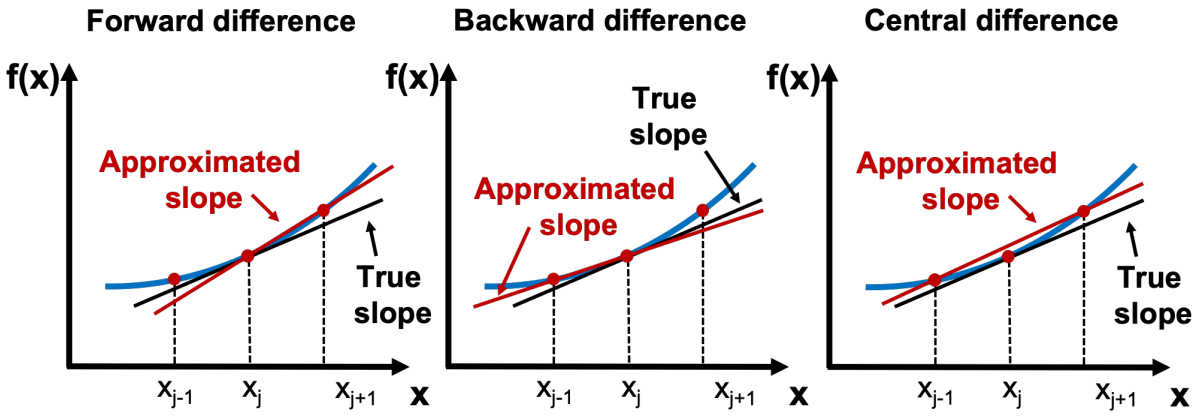

Figure 1.3: A 1D example of Finite Difference Method to approximate a function f(x) (Extracted form [2])

The Finite-Element Method (FEM)

The Finite Element Method (FEM) is a numerical method that seeks an approximated solution to the distribution of variables in the problem domain that is often difficult to obtain analytically; it is done by first dividing the problem domain into a number of small

pieces called elements, often by a simple geometry (Fig. 1.4), where the unknown variables in the FEM are simply the discrete values of the field variable at the nodes [3]. Then, the continuous solution field is replaced by a finite sum over basis functions which leads to a system of linear equations [11].

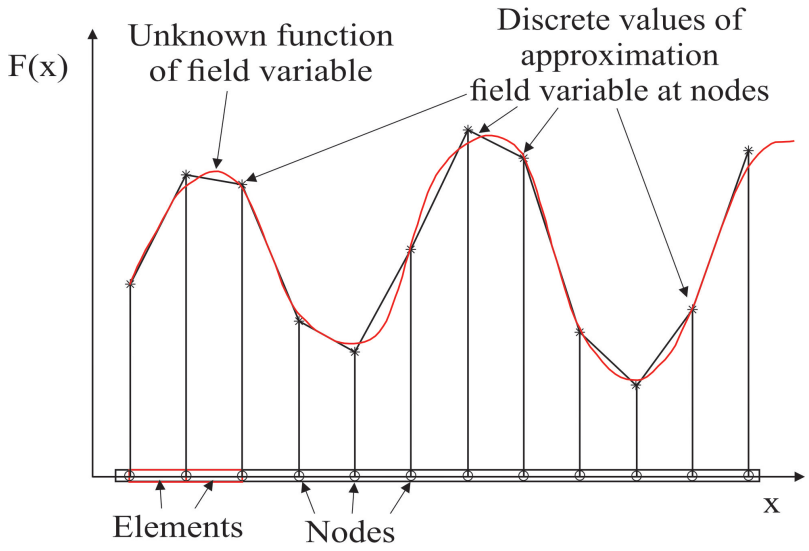

Figure 1.4: A 1D example of Finite Element Method aproximating the function F(x) using elemnents (Extracted from [3])

The Spectral-Element Method (SEM)

The Spectral Element Method (SEM) was introduced over 15 years ago in the field of computational fluid dynamics [12]. It combines some aspects of the finite element method and the pseudospectral method. In a traditional finite element method, the same points that define the element's geometry are also used for interpolating the wave field [4]. However, in SEM, the wave field is expressed using higher-degree Lagrange polynomials on Gauss-Lobatto-Legendre interpolation points, this approach ensures minimal numerical grid dispersion and anisotropy[4].

One key feature of SEM is that the mass matrix is inherently diagonal, simplifying the implementation and reducing computational costs [4]. This property allows for the use of explicit time integration schemes without the need to invert a linear system. By avoiding the inversion step, the computational efficiency is improved [4].

In summary, SEM offers a powerful approach by combining the flexibility of the finite element method and the accuracy of the pseudospectral method. This method can be

implemented for several purposes such as approximating the earth's surface using a mesh on the so-called "cubed-sphere", which is an analytical mapping from the cube to the sphere (Fig. 1.5)

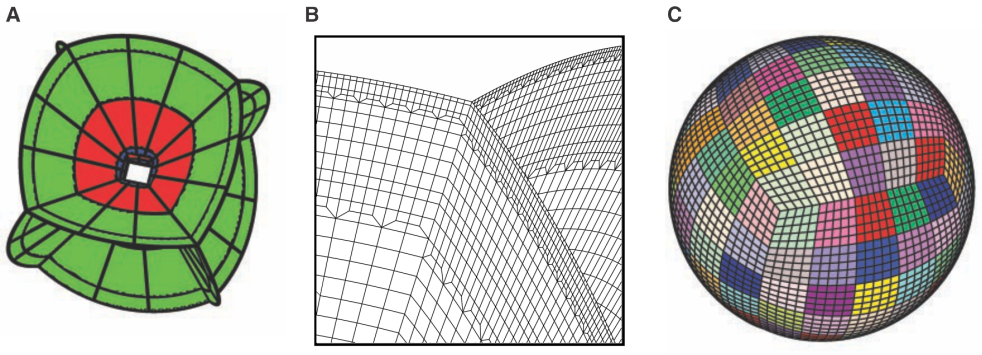

Figure 1.5: a) The six blocks constitute a cubed sphere, where the crust and mantle, outer core, and inner core are colored in green, red, and blue respectively. b) Close-up at the mesh that is doubled, this doubling ensures a relatively constant number of grid points per wavelength and reduces the computational cost. c) Global view of the mesh at the surface, illustrating that each of the six sides of the cubed sphere is divided into 25 slices (Extracted from [4]).

The Finite-Volume Method (FVM)

The finite-volume method was developed around the problem of transporting (advecting) material and conserving the integral quantity, as the first-order linear advection problem is formally equivalent to the elastic wave-propagation problem [11] (Fig. 1.6). The finite-volume method uses the same principles as the Finite Element Method such as the discretization using structured or unstructured meshes [13]. This method is locally conservative because it is based on a "balance" approach: a local balance is written on each discretization cell that is often called "control volume;" by the divergence formula, an integral formulation of the fluxes over the boundary of the control volume is then obtained [13].

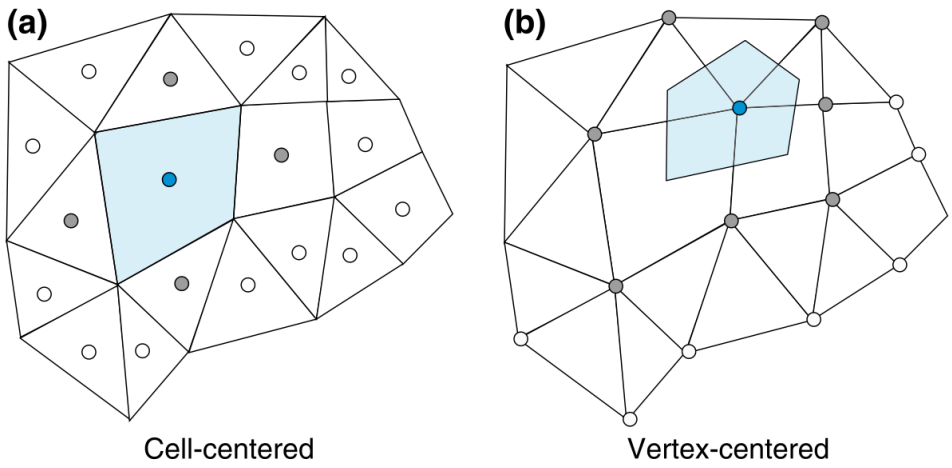

Figure 1.6: a) Cell-centered arrangement. b) Vertex-centered arrangement (Extracted from [5]).

1.3 Applications in Earth Science

Full waveform inversion is a processing technique to derive quantitative images of the subsurface from seismic measurements. This method is applied in Earth Science in different fields such as geophysical exploration, regional wave propagation, seismic tomography, simulation of ambient noise, and elastic waves in random media, among others [11]. In order to apply FWI, it is necessary to solve numerically the elastic wave function [8].

The applications of FWI have a broad spectrum which allows for performing better studies to obtain accurate results. These results are needed to improve the understanding of geological processes that has a considerable impact on the academic and economic field.

1.4 Problem statement

This work aims to approximate the elastic seismic wave equation using the finite element method. In this method, the continuous solution field is replaced by a discrete field of the finite dimension with, not necessarily orthogonal, basis functions; finite-element analysis leads to an (extensive) system of linear equations in which the associated matrices are of size N $\times$ N where N is the number of degrees of freedom [11]. In contrast, the finite-difference method replaces the partial derivatives with finite differences allowing partial differential equations, such as the wave equation, to be solved directly for (in principle) arbitrarily

heterogeneous media or spatial domain [11]. It has been observed that the finite-difference method accumulates errors in large periods of simulation whereas the finite-element method does not [3]. Furthermore, the finite-element method, compared with the finite-difference method, will provide minimum dispersion, neutral amplification, and minimum numerical anisotropy [14]. Consequently, it is pointed out that the finite-element method will reduce errors and will have an accurate approximation.

1.5 Objectives

1.5.1 General Objective

- Solve the elastic wave equation numerically using the finite element method.

1.5.2 Specific Objectives

- Deduce the equations of the finite element method for the elastic wave equation in 1D and 2D
- Computational implementation of the numerical solutions of the elastic wave equation.
- Comparison between the finite-element method and the finite-difference method.

Chapter 2

Theoretical Framework

2.1 Seismic waves in a nutshell

Seismic waves can be defined as the energy propagated through the Earth, this energy can be generated naturally (earthquakes, movement of land masses) or artificially (explosion or movement produced by an anthropic instrument) [15]. This energy is propagated through a medium and it can be described with a wave behavior, specifically, the elastic wave equation.

2.1.1 Types of seismic waves

The waves propagated through the Earth can be divided into body waves and surface waves which are different solutions for the seismic wave equation each one for different conditions. Body waves are solutions for the whole space; i.e., no boundary conditions while surface waves are a combination of body waves that are trapped in a free surface.

Body waves

Body waves travel through the Earth's body and can be divided into Primary (P waves) and Secondary (S waves). P (pressure wave, primary wave) propagates through a mechanism of uni-axial strain in the direction of propagation, the particle motion is an oscillation in the propagation direction of the wave (Fig. 2.1); while, S (shear wave, secondary wave) propagates through a shearing mechanism, the particle motion is the oscillation in any plane perpendicular to the propagation direction of the wave [16] (Fig. 2.1). Typically

in seismology, the oscillation directions are decomposed into the orthogonal components SV (Shear waves with displacement in the vertical x - z plane) and SH (Shear waves with displacement in the horizontal x - y plane) [10].

Surface waves

While body waves travel through the Earth's body, Surface waves are trapped and travel along the Earth's surface. Rayleigh wave (LR) has a retrograde-elliptical particle motion in the vertical-radial plane that is a combination of P and SV motion (Fig. 2.1), and its amplitude decays exponentially with depth below the free surface [10]. Love wave (LQ) has multiple internal reflections of horizontally polarized S waves (SH) in a near-surface medium (waveguide effect); it produces a transverse horizontal particle motion (Fig. 2.1) [10].

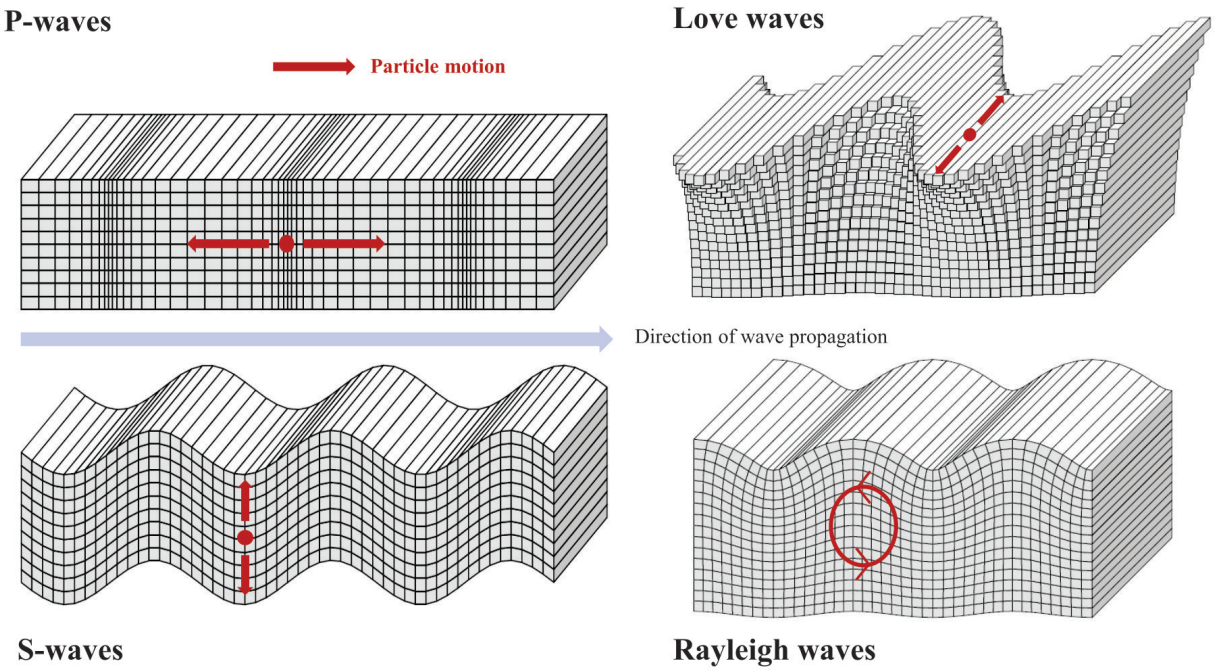

Figure 2.1: Different types of seismic waves, and particle motion: Body waves (P and S waves) on the left-hand side of the Figure, and surface waves (Love and Rayleigh waves) on the right (Extracted from [6])

2.1.2 Snell's Law

Since the Earth is formed by different rock types and lithologies that have different elastic properties, the waves will be reflected and refracted when they reach an interface between two different lithologies (Fig. 2.2). Each wave type will be propagated in different ways with different velocities through several materials. P waves travel faster than S waves since the compression involves the elastic moduli λ and the shear modulus μ. This has another consequence, the S waves can not travel through liquids since they are not able to be sheared.

$$V_P = \sqrt{\frac{\lambda + 2\mu}{\rho}}$$

$$V_S = \sqrt{\frac{\mu}{\rho}}$$

The laws for reflection and refraction are obtained using Huygens's principle which states that every point on a wavefront may be considered a source of secondary waves [17]. Also, these laws have a strong relation with Fermat's principle which states that waves travel between two points along the path that requires the least time, as compared to other nearby paths [18]. Using Fermat's principle and Huygens's principle can obtain Snell's Law which states that the ray parameter must be equal for all the incident, reflected, and refracted waves [19]. Then, the relation between angles and velocity for all waves produced by an incident wave can be described [19] as (Note that angles and velocities are stated based on the Figure 2.2a.):

$$\frac{sin(j1)}{\beta_1} = \frac{sin(i1)}{\alpha_1} = \frac{sin(i2)}{\alpha_2} = \frac{sin(j2)}{\beta_2};$$

When a refracted wave crosses the critical angle, it turns into a direct wave that travels along the interface. In this way, Snell's law is very useful to get information about the ray paths, arrival times, and the refractor's position; however, it does not provide any information about the wave amplitude.

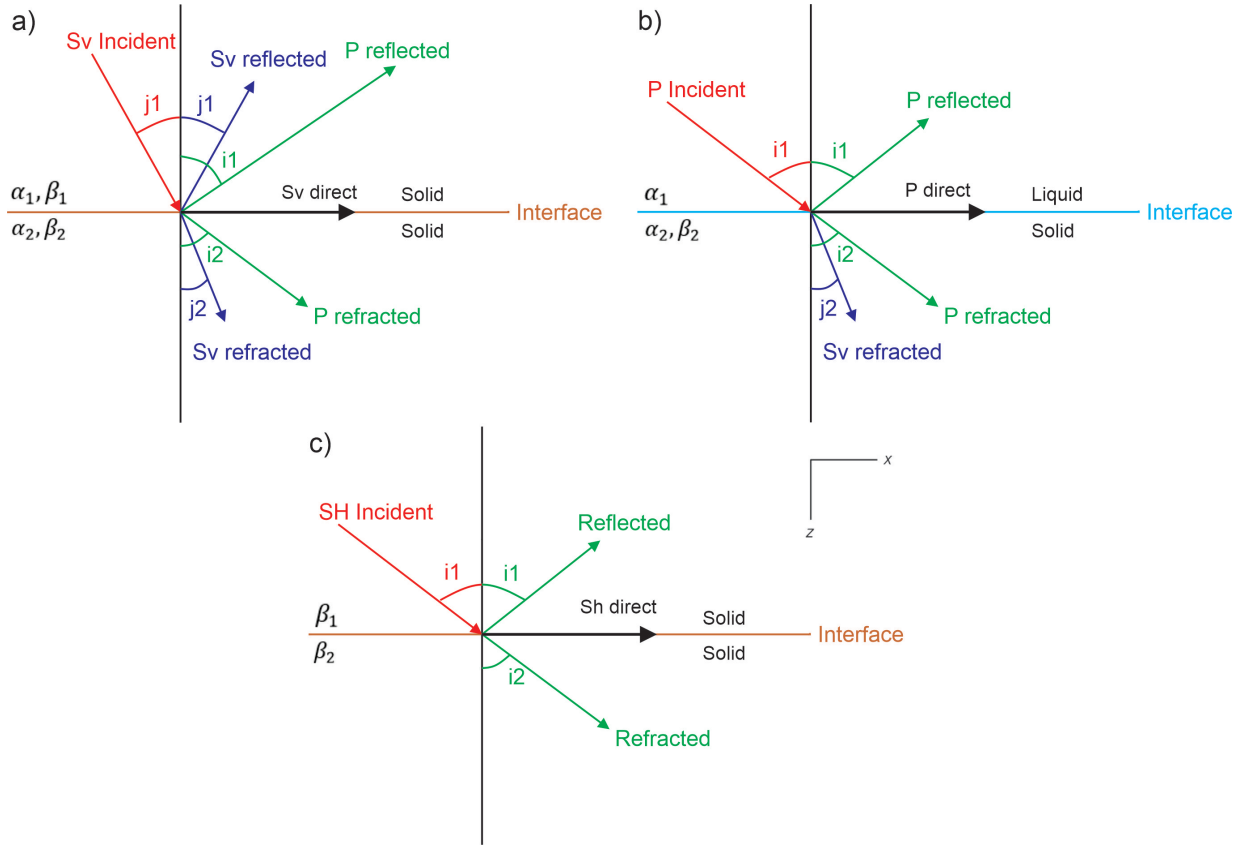

Figure 2.2: Snell's law applied for different wave types traveling through different materials where α is the P wave velocity and β is the S wave velocity (subscript just indicates the layer and its properties). a) S_v incident wave will produce S_v reflected and S_v refracted when it reaches an interface, also, there is a mode conversion that generates two additional waves P reflected and P refracted. b) P incident wave traveling through a liquid will produce a P reflected and P refracted waves and a mode conversion generating a S_v wave if the medium is solid. c) S_H wave will only produce a reflection and refraction and no mode conversion.

2.1.3 Derivation of the elastic wave equation

The elastic waves travel in three directions x (horizontal), y (profundity), and z (depth); in addition, another dimension is added which is time (t). To get the 2D elastic wave equation, the derivation was extracted from [20]. Let's place the plane coordinates:

- $u(x, z, t)$: the displacement relative to the equilibrium position, of coordinates (x, z) in the direction of the x-axis at time t.
- $w(x, z, t)$: the displacement relative to the equilibrium position, of coordinates (x, z) in the direction of the z-axis at time t.

- $\begin{pmatrix} \sum_{xx} & \sum_{xz} \\ \sum_{zx} & \sum_{zz} \end{pmatrix} \begin{pmatrix} x & z & t \end{pmatrix}$: The deformation tensor at coordinates (x, z) at time t (Fig. 2.3). The deformations are defined from the displacement derivatives using the following expressions:

$$\begin{cases} \sum_{xx} = \dfrac{\partial u}{\partial x} \\ \sum_{zz} = \dfrac{\partial w}{\partial z} \\ \sum_{xz} = \epsilon zx = \dfrac{1}{2}\left(\dfrac{\partial u}{\partial z} - \dfrac{\partial w}{\partial x}\right) \end{cases} \tag{2.1}$$

It is known that: $\tau_{xz} = \tau_{zx}$. Then, let's consider as a constraint-deformation relation the law of homogeneous isotropic, linear elastic media:

$$\begin{cases} \tau_{xx} = \lambda(\sum_{xx} + \sum_{zz}) + 2\mu \sum_{xx} \\ \tau_{zz} = \lambda(\sum_{xx} + \sum_{zz}) + 2\mu \sum_{zz} \\ \tau_{xz} = 2\mu \sum_{xz} \end{cases} \quad ; \text{Where } \lambda \text{ and } \mu \text{ are the lamé parameters.} \tag{2.2}$$

It is possible to deduce from Eq. (2.1) and Eq. (2.2) the expression of the displacement functions:

$$\begin{cases} \tau_{xx} = \lambda\left(\dfrac{\partial u}{\partial x} + \dfrac{\partial w}{\partial z}\right) + 2\mu\dfrac{\partial u}{\partial x} \\ \tau_{zz} = \lambda\left(\dfrac{\partial u}{\partial x} + \dfrac{\partial w}{\partial z}\right) + 2\mu\dfrac{\partial u}{\partial z} \\ \tau_{xz} = \mu\left(\dfrac{\partial u}{\partial z} + \dfrac{\partial w}{\partial x}\right) \end{cases} \tag{2.3}$$

In the absence of external forces, the equations can be written as:

$$\begin{cases} \rho \dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial}{\partial x}\tau_{xx} + \dfrac{\partial}{\partial z}\tau_{zx} \\ \\ \rho \dfrac{\partial^2 w}{\partial t^2} = \dfrac{\partial}{\partial x}\tau_{xz} + \dfrac{\partial}{\partial z}\tau_{zz} \end{cases} \quad ; \text{Where } \rho \text{ is the density} \tag{2.4}$$

By replacing the eq. (2.3) into the eq. (2.4), it is obtained the elastic wave equations for the x-axis and z-axis:

$$\rho \frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(\lambda\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right) + 2\mu\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial z}\left(\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right) \tag{2.5}$$

$$\rho \frac{\partial^2 w}{\partial t^2} = \frac{\partial}{\partial z}\left(\lambda\left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z}\right) + 2\mu\frac{\partial w}{\partial z}\right) + \frac{\partial}{\partial x}\left(\mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right)\right) \tag{2.6}$$

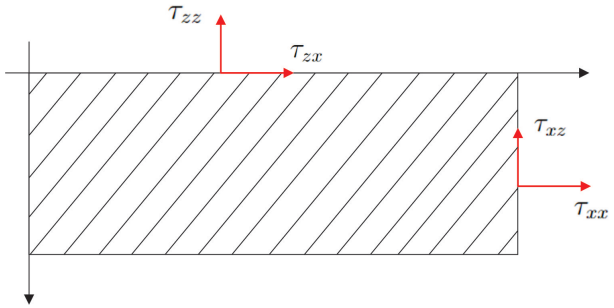

Figure 2.3: Stress efforts applied over a medium and its corresponding directions[7]

2.2 Finite Element Method (FEM) in 1D

2.2.1 Problem development

Let's consider the elastic wave equation in 1D [21], which describes the propagation of elastic waves along the x-axis with an impulse (source).

$$\rho(x)\partial_{tt}^2 u(x,t) = \partial_x(\mu(x)\partial_x u(x,t)) + f(x,t), \qquad (x,t) \in (0, x_{max}) \times (0, +\infty) \tag{2.7}$$

Where ρ is the density, μ is the shear modulus or the Lamé parameter, u is the displace-

ment field, and f is the source vector. To solve the PDE and approximate the equation, it is necessary to implement the boundary and initial conditions.

$$\text{Initial conditions} \begin{cases} u(x, t = 0) = 0 \\ \partial_t u(x, t = 0) = 0 \end{cases} \tag{2.8}$$

$$\text{Boundary conditions} \begin{cases} u(x = 0, t) = 0 \\ u(x = x_{max}, t) = 0 \end{cases} \tag{2.9}$$

The given initial conditions mean that the wave is not propagating at $t = 0$; i.e., there is no movement at the beginning. On the other hand, the boundary conditions mean that the string is fixed at the edges. Therefore, the wave reaches the end of the string and returns without any movement at the edges.

2.2.2 Analytical Solution

The following deduction for an analytical solution is extracted from [22], and can be reviewed for a detailed explanation. To obtain analytical solutions for an elastic wave equation with a source in 1D (eq. 2.7), it is usually developed by using the concept of Green's function, that is the solution for an impulse response:

$$\partial_{tt}^2 G(x, t; x_0, t_0) - c^2 \Delta G(x, t; x_0, t_0) = \delta(x - x_0)\delta(t - t_0); \text{where } c^2 = \frac{\mu}{\rho} \tag{2.10}$$

Green's functions are the solutions to the specific partial differential equations for $\delta-$functions as source terms evaluated at (x, t) and activated at (x_0, t_0), where Δ is the Laplace operator and $\delta-$function is defined by:

$$\delta(x) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases} \tag{2.11}$$

$$\int_{-\infty}^{\infty} \delta(x)dx = 1, \quad \int_{-\infty}^{\infty} f(x)\delta(x)dx = f(0) \tag{2.12}$$

Analytical solutions for eq. 2.7 are obtained using the Heaviside function, where the

result for the Green's function is:

$$\frac{1}{2c}H\left(t-\frac{|x|}{c}\right) \tag{2.13}$$

Note that Green's function is obtained after convolution with the first derivative of a Gaussian source with a frequency of $f_0 = 25Hz$. Where the first derivative of the Gaussian source is:

$$f = -2(t-t_0)(f_0^2)(e^{-f_0^2(t-t_0)^2}) \tag{2.14}$$

2.2.3 Finite Difference Method (FDM) in Homogeneous Medium

For the FDM, the elastic wave equation turns into the acoustic wave for the 1D case, especially for the case of S-waves. Rearranging the equation, it is obtained that:

$$\partial_{tt}^2 u(x,t) = c^2(\partial_{xx}^2 u(x,t)) + f; \text{where } c^2 = \frac{\mu}{\rho} \tag{2.15}$$

The FDM seeks an approximation using the Taylor series to approach the derivatives using differences of points along the wave [9]. While more points are used, the approximation is better. For this purpose, it was chosen a center difference using three points (Fig. 1.3). It is necessary to make a convention where it is specified if the step is taken in the space or time domain. Then, it is set that the superscript i refers to the time domain while the subscript l refers to the space domain. Note that, $i \pm 1$ and $l \pm 1$ mean a step forward or backward on each domain (Fig. 2.4); i.e., add or subtract Δt or Δx which are the increment in time and space respectively, where $i = 1, ..., I$ and $l = 1, ..., N$.

To approximate the equation, it is used the second order derivative of the Taylor series. For the time second derivative:

$$\partial_{tt}^2 u(x_l, t_i) \approx \frac{u_l^{i+1} - 2u_l^i + u_l^{i-1}}{\Delta t^2} \tag{2.16}$$

For the space second derivative:

$$\partial_{xx}^2 u(x_l, t_i) \approx \frac{u_{l+1}^i - 2u_l^i + u_{l-1}^i}{\Delta x^2} \tag{2.17}$$

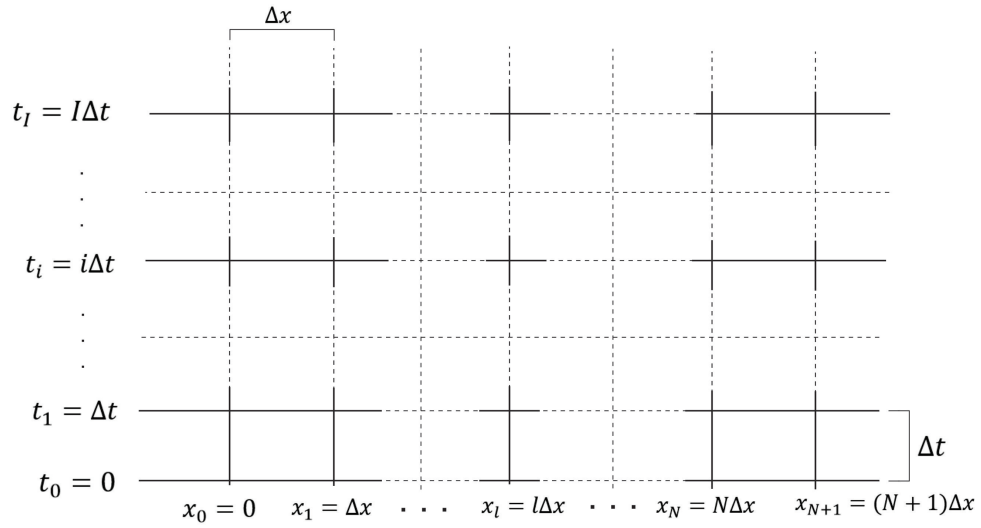

Figure 2.4: Increments of space and time discretization

Now, by replacing eq. 2.17 and eq. 2.16 into eq. 2.15, and solving for u_l^{i+1}:

$$u_l^{i+1} = c_l^2 \frac{\Delta t^2}{\Delta x^2}(u_{l+1}^i - 2u_l^i + u_{l-1}^i) + 2u_l^i - u_l^{i-1} + \Delta t^2 f_l^i \tag{2.18}$$

The eq. 2.18 is the fully explicit key for the FDM, and this equation is the final expression to be implemented in Python to simulate the acoustic wave equation.

2.2.4 Finite Element Method (FEM) in Homogeneous Medium

The FEM approximates the eq. 2.7 by discretizing the domain into elements that can be evenly spaced or not. In this case, elements have equal distances between them. This method seeks solutions for the displacement field $u(x,t)$. This approach is solved by replacing the displacement field by a finite sum over basis functions ϕ_i which verifies the boundary conditions of the problem.

$$u(x,t) \approx \bar{u}(x,t) = \sum_{i=1}^{N} u_i(t)\phi_i(x) \tag{2.19}$$

The unknowns are the coefficients $u_i(t)$. Let's formulate the weak form by multiplying the eq. 2.7 by a test function ϕ_j. Afterward, it is integrated over the physical domain (D).

$$\int_D \rho \partial_{tt}^2 u \phi_j dx + \int_D \partial_x(\mu \partial_x u)\phi_j dx = \int_D f\phi_j dx; \text{ where } j = 1, ..., N \tag{2.20}$$

Using integration by parts, the second term of the left-hand side of eq. 2.20 is replaced by:

$$\int_D \partial_x(\mu\partial_x u)\phi_j dx = [\mu\partial_x u\phi_j] - \int_D \mu\partial_x u\partial_x\phi_j dx \text{ where } j = 1, ..., N \tag{2.21}$$

The first term of the right-hand side is evaluated at the limit of the domain. Then, the antiderivative term drops out due to the null boundary conditions 2.9 which means a stress-free boundary condition. Applying the integration by parts and the finite sum; i.e., substituting the eq. 2.19 and eq. 2.21 into eq. 2.20. It is obtained that:

$$\sum_{i=1}^{N} \partial_t^2 u_i(t) \int_D \rho\phi_i\phi_j dx + \sum_{i=1}^{N} u_i(t) \int_D \mu(\partial_x\phi_i)(\partial_x\phi_j)dx = \int_D f\phi_j dx \text{ where } j = 1, ..., N \tag{2.22}$$

The eq. 2.22 is, actually, a system of equations. To simplify the notation, let's use a matrix and vector notation.

$$u \to u_i(t) \tag{2.23}$$

$$M \to M_{ij} = \int_D \rho\phi_i\phi_j dx \tag{2.24}$$

$$K \to K_{ij} = \int_D \mu(\partial_x\phi_i)(\partial_x\phi_j)dx \tag{2.25}$$

$$f \to f_i = \int_D \phi_j dx \tag{2.26}$$

Where u is the displacement field in the time domain, M is the mass matrix, K is the stiffness matrix, and f is the source vector. Now, the system can be written as:

$$\partial_t^2 uM + uK = f \tag{2.27}$$

As transpose notation:

$$M^T\partial_t^2 u = f^T - K^T u^T \tag{2.28}$$

To approximate the time field using FEM can turn into a cumbersome problem; then, it was chosen to approximate this field using FDM. Therefore, the eq. 2.28 can written as:

$$M^T \left[\frac{u(t+\Delta t) - 2u(t) + u(t-\Delta t)}{\Delta t^2} \right] = f^T - K^T u^T \tag{2.29}$$

Due to the initial conditions (eq. 2.8), it is possible to determine the displacement field at time $t + \Delta t$ by solving the eq. 2.29 for $u(t + \Delta t)$. Then, the final expression is:

$$u(t+\Delta t) = \Delta t^2 (M^T)^{-1}[f - k^T u] + 2u(t) - u(t-\Delta t) \tag{2.30}$$

To solve the equation system, we need to turn the domain into a local coordinate system for the basis functions (Fig. 2.5).

$$\begin{cases} \delta = x - x_i \\ h_i = x_{i+1} - x_i \end{cases} \tag{2.31}$$

Where h_i is the increment, and the element i is defined in the interval $x \in [x_i; x_{i+1}]$. The basis [3] functions are given by:

$$\phi_i = \begin{cases} \dfrac{\delta}{h_{i-1}} + 1; & -h_{i-1} \leq \delta \leq 0 \\ 1 - \dfrac{\delta}{h_i}; & 0 \leq \delta \leq h_i \\ 0; & \text{elsewhere} \end{cases} \tag{2.32}$$

$$\partial_\delta \phi_i = \begin{cases} \dfrac{1}{h_{i-1}}; & -h_{i-1} \leq \delta \leq 0 \\ -\dfrac{1}{h_i}; & 0 \leq \delta \leq h_i \\ 0; & \text{elsewhere} \end{cases} \tag{2.33}$$

Let's solve the Mass and Stiffness matrix with the given basis functions. It is important to clarify the solution is assuming a homogenous density and shear modulus. Consider the Mass matrix (eq. 2.24), and change the domain into the local coordinate system (eq. 2.31). The diagonal elements of the matrix, it is solved as follows:

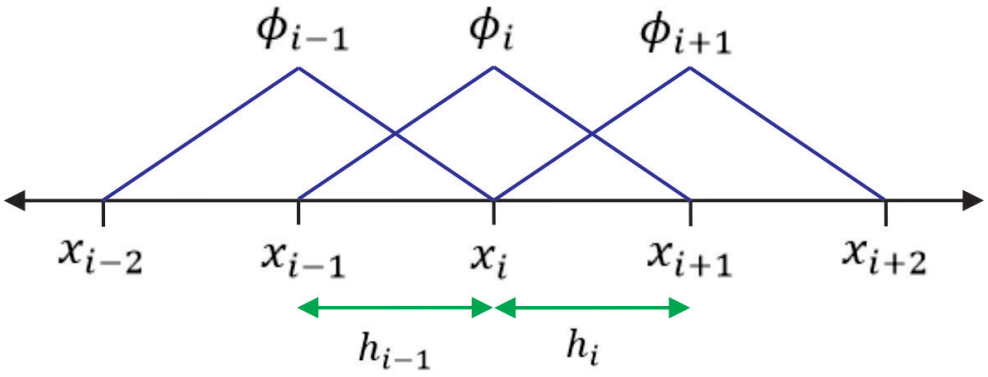

Figure 2.5: Basis functions along the elements where the increment is specified.

$$M_{ii} = \int_D \rho\phi_i\phi_i dx = \int_{D_\delta} \rho\phi_i\phi_i d\delta \tag{2.34}$$

$$M_{ii} = \rho_{i-1}\int_{-h_{i-1}}^{0}\left(\frac{\delta}{h_{i-1}}+1\right)^2 d\delta + \rho_i\int_0^{h_i}\left(1-\frac{\delta}{h_i}\right)^2 d\delta \tag{2.35}$$

$$M_{ii} = \frac{1}{3}(\rho_{i-1}h_{i-1} + \rho_i h_i) \tag{2.36}$$

For the off-diagonal elements, the basis functions overlap only in one element.

$$M_{i,i-1} = \rho_{i-1}\int_{-h_{i-1}}^{0}\left(\frac{\delta}{h_{i-1}}+1\right)\left(-\frac{\delta}{h_{i-1}}\right) d\delta \tag{2.37}$$

$$M_{i,i-1} = \frac{1}{6}\rho_{i-1}h_{i-1} \tag{2.38}$$

$$M_{i,i+1} = \rho_i\int_0^{h_i}\left(1-\frac{\delta}{h_i}\right)\left(\frac{\delta}{h_i}\right) d\delta \tag{2.39}$$

$$M_{i,i+1} = \frac{1}{6}\rho_i h_i \tag{2.40}$$

$$M = \frac{\rho h}{6} \begin{pmatrix} \ddots & \ddots & \dots & 0 & 0 \\ \ddots & 4 & 1 & \ddots & 0 \\ \vdots & 1 & 4 & 1 & \vdots \\ 0 & \ddots & 1 & 4 & \ddots \\ 0 & 0 & \dots & \ddots & \ddots \end{pmatrix} \tag{2.41}$$

For the stiffness matrix (eq. 2.25), it is solved as follows for the diagonal:

$$K_{ii} = \int_D \mu \partial_x \phi_i \partial_x \phi_i dx = \int_{D_\delta} \mu \partial_x \phi_i \partial_x \phi_i d\delta \tag{2.42}$$

$$K_{ii} = \mu_{i-1} \int_{-h_{i-1}}^{0} \left(\frac{1}{h_{i-1}} \right)^2 d\delta + \mu_i \int_0^{h_i} \left(-\frac{1}{h_i} \right)^2 d\delta \tag{2.43}$$

$$K_{ii} = \frac{\mu_{i-1}}{h_{i-1}} + \frac{\mu_i}{h_i} \tag{2.44}$$

For the elements off-diagonal:

$$K_{i,i-1} = \mu_{i-1} \int_{-h_{i-1}}^{0} \left(\frac{1}{h_{i-1}} \right) \left(\frac{-1}{h_{i-1}} \right) d\delta \tag{2.45}$$

$$K_{i,i-1} = -\frac{\mu_{i-1}}{h_{i-1}} \tag{2.46}$$

$$K_{i,i+1} = \mu_i \int_0^{h_i} \left(\frac{1}{h_i} \right) \left(-\frac{1}{h_i} \right) d\delta \tag{2.47}$$

$$K_{i,i+1} = -\frac{\mu_i}{h_i} \tag{2.48}$$

$$K = \frac{\mu}{h} \begin{pmatrix} \ddots & \ddots & \dots & 0 & 0 \\ \ddots & 2 & -1 & \ddots & 0 \\ \vdots & -1 & 2 & -1 & \vdots \\ 0 & \ddots & -1 & 2 & \ddots \\ 0 & 0 & \dots & \ddots & \ddots \end{pmatrix} \tag{2.49}$$

Note that density and shear modulus are constant through the medium, this provokes that these values can move out of the integral. Now the eq. 2.30 and eq. 2.18 can be implemented in Python to simulate each method and compare them.

2.2.5 Finite Element Method (FEM) for Two Layers in 1D

It is known that the earth's crust is not homogenous, it is composed of different rock types such as sedimentary rocks, extrusive and intrusive rocks, and metamorphic rocks [23]. FWI and the direct problem are more often used to identify anomalies related to potential reserves of minerals, oil, and gas. These two last ones are stored in sedimentary rocks that are stratified, which means a succession of several layers following a sequence (These layers possess different properties that produce different behaviors as the waves pass through). Then, the importance of simulating waves through different mediums becomes an important issue to solve. For this particular case consider two mediums with their corresponding properties; i.e., there are $\rho_1, \rho_2, \mu_1,$ and μ_2.

The two layers case follow the same reasoning for a homogeneous medium; however, it is necessary to consider some changes for the Mass and Stiffness matrices. For the mass matrix, let's consider the equations 2.35, 2.37, and 2.39 the solutions for these equations considering two layers will be:

$$M_{ii} = \frac{1}{3}(\rho_{1_{i-1}} h_{i-1} + \rho_{2_i} h_i) \tag{2.50}$$

$$M_{i,i-1} = \frac{1}{6}\rho_{1_{i-1}} h_{i-1} \tag{2.51}$$

$$M_{i,i+1} = \frac{1}{6}\rho_{2_i} h_i \tag{2.52}$$

The stiffness matrix will be changed as:

$$K_{ii} = \frac{\mu_{1_{i-1}}}{h_{i-1}} + \frac{\mu_{2_i}}{h_i} \tag{2.53}$$

$$K_{i,i-1} = -\frac{\mu_{1_{i-1}}}{h_{i-1}} \tag{2.54}$$

$$K_{i,i+1} = -\frac{\mu_{2_i}}{h_i} \tag{2.55}$$

Then, this solution can be used for more layers with different properties. It must be provided where the interfaces are located to place the respective values.

2.3 Finite Element Method (FEM) in 2D

The aim of this section is to discretize the space field of the elastic wave equation using FEM and the time domain using FDM since this combination produces a stable and accurate result [24]. The numerical approximation is obtained following [20] and [25] for the spatial and temporal discretization.

Let Ω be the rectangle of $\mathbb{R}^2$, $]0, L[\times]0, Z[$, with frontiers $\Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. Consider the equations 2.5 and 2.6. Also, let's fix the boundary conditions that are typically applied in several cases. Equations 2.56 and 2.57 specify a force, with components $\mathcal{F}_x$, $\mathcal{F}_z$, that are applied on Γ_0 (Top)

$$\tau_{xz} = \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial z} \right) = -\mathcal{F}_x \tag{2.56}$$

$$\tau_{zz} = \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} \right) + 2\mu \frac{\partial w}{\partial z} = -\mathcal{F}_z \tag{2.57}$$

The conditions of symetry respect to Γ_1 (left side) are:

$$u = 0 \tag{2.58}$$

$$\frac{\partial w}{\partial x} = 0 \tag{2.59}$$

For Γ_2 (right side) and Γ_3 (bottom)

$$u = 0 \tag{2.60}$$

$$w = 0 \tag{2.61}$$

The initial conditions are given by:

$$u(x,z,0) = u_0(x,z) \tag{2.62}$$

$$w(x,z,0) = w_0(x,z) \tag{2.63}$$

$$\frac{\partial u}{\partial t}(x,z,0) = u_1(x,z) \tag{2.64}$$

$$\frac{\partial w}{\partial t}(x,z,0) = w_1(x,z) \tag{2.65}$$

Now, let's introduce the space V:

$$\begin{cases} V = V_x \times V_z \text{ with:} \\ V_x = v \in H^1(\Omega) \text{ wich verifies eq. 2.58 and eq. 2.60} \\ V_z = v \in H^1(\Omega) \text{ wich verifies eq. 2.61} \end{cases} \tag{2.66}$$

Let $v \in V_x$; multiply the eq. 2.5 by v and integrate over Ω. After integration by parts and taking into account the boundary conditions (eq. 2.56), we obtain:

$$\int_\Omega \left(\rho \frac{\partial^2 u}{\partial t^2} v + (\lambda + 2\mu) \frac{\partial u}{\partial x} \frac{\partial v}{\partial x} + \lambda \frac{\partial w}{\partial z} \frac{\partial v}{\partial x} + \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \frac{\partial v}{\partial z} \right) d\Omega = \int_{\Gamma_0} \mathcal{F}_x v d\Gamma \tag{2.67}$$

Likewise, let $v \in V_z$; multiply the eq. 2.6 by v and integrate over Ω. After integration by parts and taking into account the boundary conditions (equations 2.57, 2.58, 2.59), we obtain:

$$\int_\Omega \left(\rho \frac{\partial^2 w}{\partial t^2} v + (\lambda + 2\mu) \frac{\partial w}{\partial z} \frac{\partial v}{\partial z} + \lambda \frac{\partial u}{\partial x} \frac{\partial v}{\partial z} + \mu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right) \frac{\partial v}{\partial x} \right) d\Omega = \int_{\Gamma_0} \mathcal{F}_z v d\Gamma \tag{2.68}$$

The Ω domain is cut by a regular mesh whose grid (fig. 2.6a) is formed of straight lines with $x = i\Delta x$ and $z = j\Delta z$, the intersection of these lines determines the node R_{ij}. Δx

and Δz are chosen as:

$$L = I\Delta x \text{ and } Z = J\Delta z; \text{ where } I, J \text{ are integers} \tag{2.69}$$

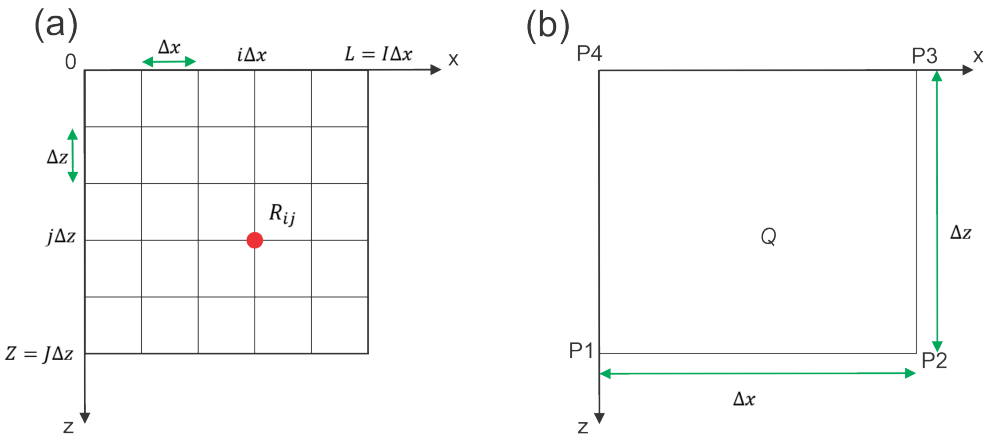

Figure 2.6: a) Grid with the increments specified. b) numbering of the vertices of a quadrilateral Q for the assembly of the sub-matrices, where each vertice corresponds to a basis function P_1, P_2, P_3, and P_4.

In that way, h is the pair of numbers $(\Delta x,\ \Delta z)$ and Q_h is the set of quadrilaterals with sides Δx, Δz formed by the grid previously defined.

It's necessary to create the restriction to each quadrilateral Q of Q_h is in polynomial of degree 1 separately in x and in z, for this let's consider the space $W_h = v_h \in C^0(\Omega)$.

It is possible to show that a basis of the vector space W_h is formed by the functions p_{ij} defined by:

$$p_{ij}(R_{i'j'}) = \begin{cases} 1 : i = i' \text{ and } j = j' & 0 \leq i, i' \leq I \\ 0 : \text{otherwise} & 0 \leq j, j' \leq J \end{cases} \tag{2.70}$$

Consequently, the functions v_h of w_h can be written:

$$v_h = \sum_{\substack{i=0,I \\ j=0,J}} v_h(R_{ij}) p_{ij} = \sum_{\substack{i=0,I \\ j=0,J}} v_{ij} p_{ij}; \text{ where } v_{ij} \text{ are numbers, } p_{ij} \text{ are functions.} \tag{2.71}$$

Now, let's define the space V_h:

$$\begin{cases} V_h = V_{hx} \times V_{hz}; \text{ with:} \\ V_{hx} = v_h \in W_h \text{ as } v_{ij} = 0 \ \forall j = 0, ..., J \text{ if } i = 0, ..., I \text{ and } v_{ij} = 0 \ \forall i = 0, ..., I \\ V_{hz} = v_h \in W_h \text{ as } v_{Ij} = 0 \ \forall j = 0, ..., J \text{ and } v_{ij} = 0 \ \forall i = 0, ..., I \end{cases} \tag{2.72}$$

The space V_h approaches the space V defined in the eq. 2.66.

It will therefore be reasonable to seek to approximate at any instant t the solution of the variational problem of the equations 2.67 and 2.68 by a function of the space V_h; this function will solve an approach problem that it is going to be defined now.

The finite element method consists in establishing an approach problem obtained by replacing in the continuous variational problem the functions (solution and test function) of V by functions of the space V_h.

It is assumed for the following deduction that $\mathcal{F}_x = \mathcal{F}_z = 0$

2.3.1 Obtaining semi-discrete equations in space

It is called $\begin{pmatrix} u_h \\ w_h \end{pmatrix}$ the solution of the approach problem obtained by writing the variational formulation of the equations 2.67 and 2.68 for any function of the basis V_h; i.e., for any vector of the form $\begin{pmatrix} p_{ij} \\ 0 \end{pmatrix}$, where p_{ij} belongs to the basis of V_{hx} or $\begin{pmatrix} 0 \\ p_{ij} \end{pmatrix}$, where p_{ij} belongs to the base of V_{hz}.

After having introduced the components $u_{i'j'}$ and $w_{i'j'}$ of the vector $\begin{pmatrix} u_h \\ w_h \end{pmatrix}$ on the vectors of the base of V_h (eq. 2.71):

$$\begin{cases} u_h = \sum\limits_{\substack{i'=0,I \\ j'=0,J}} u_{i'j'} p_{i'j'} \\ w_h = \sum\limits_{\substack{i'=0,I \\ j'=0,J}} w_{i'j'} p_{i'j'} \end{cases} \tag{2.73}$$

The approach problem consists in solving the following system of differential equations:

$$\begin{cases} \sum\limits_{\substack{i'=0,I \\ j'=0,J}} \dfrac{d^2 u_{i'j'}}{dt^2} \int_\Omega \rho p_{ij} p_{i'j'} \\ + \sum\limits_{\substack{i'=0,I \\ j'=0,J}} u_{i'j'} \int_\Omega (\lambda + 2\mu) \left(\dfrac{\partial p_{ij}}{\partial x} \dfrac{\partial p_{i'j'}}{\partial x} + \mu \dfrac{\partial p_{ij}}{\partial z} \dfrac{\partial p_{i'j'}}{\partial z} \right) \\ + \sum\limits_{\substack{i'=0,I \\ j'=0,J}} w_{i'j'} \int_\Omega \lambda \left(\dfrac{\partial p_{ij}}{\partial x} \dfrac{\partial p_{i'j'}}{\partial z} + \mu \dfrac{\partial p_{ij}}{\partial z} \dfrac{\partial p_{i'j'}}{\partial x} \right) = 0 \end{cases} \quad \begin{cases} \forall i = 1, I \\ \forall j = 0, J \end{cases} \tag{2.74}$$

$$\begin{cases} \sum\limits_{\substack{i'=0,I \\ j'=0,J}} \dfrac{d^2 w_{i'j'}}{dt^2} \int_\Omega \rho p_{ij} p_{i'j'} \\ + \sum\limits_{\substack{i'=0,I \\ j'=0,J}} w_{i'j'} \int_\Omega (\lambda + 2\mu) \left(\dfrac{\partial p_{ij}}{\partial z} \dfrac{\partial p_{i'j'}}{\partial z} + \mu \dfrac{\partial p_{ij}}{\partial x} \dfrac{\partial p_{i'j'}}{\partial x} \right) \\ + \sum\limits_{\substack{i'=0,I \\ j'=0,J}} u_{i'j'} \int_\Omega \lambda \left(\dfrac{\partial p_{ij}}{\partial z} \dfrac{\partial p_{i'j'}}{\partial x} + \mu \dfrac{\partial p_{ij}}{\partial x} \dfrac{\partial p_{i'j'}}{\partial z} \right) = 0 \end{cases} \quad \begin{cases} \forall i = 1, I \\ \forall j = 0, J \end{cases} \tag{2.75}$$

Having adopted a renumbering, depending on a single index, of the nodes R_{ij} and having posed:

$$\begin{cases} U(t) = \begin{bmatrix} u_{ij}(t) \\ w_{ij}(t) \end{bmatrix} \text{ Vector of unknown discrete functions} \\ \text{or } j = 0, ..., J \text{ and } i = 1, ..., I \text{ for } u \text{ and } 0, ..., I \text{ for } W \end{cases} \tag{2.76}$$

System 2.74 and 2.75 can be written:

$$M \frac{d^2 U}{dt^2} + KU = 0 \tag{2.77}$$

The matrices M and K are called mass matrix and stiffness matrix, respectively.

The matrices M and K are obtained by explaining in 2.74 and 2.75 the scalar products in L^2 of the functions for the finite element basis and their derivatives in space. There is an explanation of these scalar products by using the method of contributions: one calculates initially (assembly of under matrices of mass and stiffness) the contributions of each quadrilateral in the scalar products; then for a given node (i.e., having fixed the i and the j appearing in 2.74 and 2.75). We sum (assembly of the mass and stiffness matrices)

the contributions of each quadrilateral to obtain the different terms of the line, associated with 2.74 or 2.75 and with the node R_{ij}, of the matrices M and K. Taking into account the support of the basis functions, the only non-zero contributions are those corresponding to the scalar products of the basis function (or of its derivatives) associated with a node R_{ij} with a basis function (or with one of its derivatives) associated with the node $R_{i'j'}$, vertex of a quadrilateral also having R_{ij} as a vertex. Consequently, for the assembly of the submatrices, it is necessary to perform the scalar products L^2 only of the basis functions (or of their derivatives) associated with the vertices of the same quadrilateral of Q_h.

Assembly of sub-matrices.

Given a quadrilateral of Q_h, we number its vertices as shown in Figure 2.6b and we note p_i the function of the form (restriction to quadrilateral Q of the basis function associated with vertex i) associated with vertex i.

We will calculate scalar products of the type:

$$\int_Q p_i p_j \tag{2.78}$$

Analogous scalar products are obtained by replacing ρ with λ or μ and the shape functions by their derivatives.

Note that the coefficients ρ, λ, and μ are assumed to be constant on each quadrilateral.

Under these conditions, it is easy to obtain the values of the scalar products of type 2.78 from the L^2 scalar products of the shape functions or their derivatives.

These scalar products are summarized in Table 2.1, and the calculations can be seen in Appendix .1.

Assembly of the matrices R and K

Assuming that the coefficients ρ, λ, and μ are constant in the domains C_j, which are layers:

$$\begin{cases} C_j = \{(x,z) \in \mathbb{R}^2 \text{ as } 0 \leq x \leq L \text{ and} \\ j\Delta z < z < (j+1)\Delta z,\ j \in \mathbb{N}\} \end{cases} \tag{2.79}$$

Moreover, in order to simplify the presentation, it will only establish the rows of the

matrices R and K corresponding to functions of bases p_{ij} with R_{ij} node interior to omega (i.e, $R_{ij}d\Gamma$).

Let M_{ij} be the interior node and v be the basis function associated with it. We will use local numbering attached to R_{ij} in which R_{ij} will have the number 0 and it is numbered 1, 2, 3, 4, 5, 6, 7, and 8 the neighboring nodes of R_{ij} (fig. 2.7).

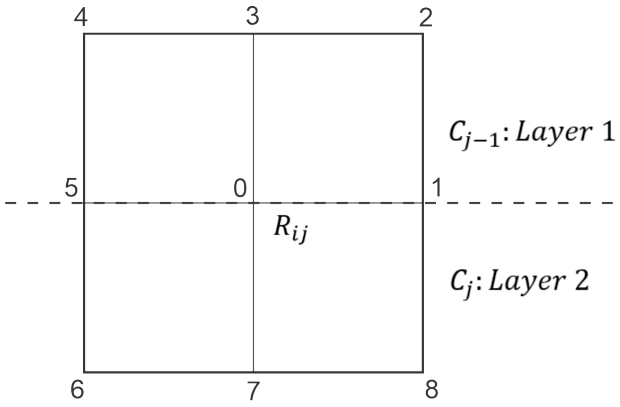

Figure 2.7: Local numbering of neighboring nodes of R_{ij} used for the assembly of matrices M and K.

It is adopted a local numbering for the layers and the coefficients: layer 1 is layer cj-1 and layer 2 is layer Cj, and the coefficients in layers 1 and 2 are called respectively ρ_1, λ_1, μ_1 and ρ_2, λ_2, μ_2.

By noting respectively u_k and w_k the approximations of the unknown functions u and w at node number k (in the local frame linked to node R_{ij}), appearing in eq. 2.74 and obtain the equation (in u) semi-discrete in space associated with node R_{ij}.

$$
\begin{cases}
\frac{\Delta x \Delta z}{9}[2(\rho_1+\rho_2)\frac{\partial^2 u_0}{dt^2}+\frac{(\rho_1+\rho_2)}{2}\left(\frac{\partial^2 u_1}{dt^2}+\frac{\partial^2 u_5}{dt^2}\right)+\rho_1\frac{\partial^2 u_3}{dt^2}+\rho_2\frac{\partial^2 u_7}{dt^2} \\
+\frac{1}{4}\rho_1\left(\frac{\partial^2 u_2}{dt^2}+\frac{\partial^2 u_4}{dt^2}\right)+\frac{1}{4}\rho_2\left(\frac{\partial^2 u_8}{dt^2}+\frac{\partial^2 u_6}{dt^2}\right)] \\
+\frac{\Delta z}{\Delta x}(\lambda+2\mu)_1\left[\frac{2}{3}u_0-\frac{1}{6}(u_2+u_4)+\frac{1}{3}u_3-\frac{1}{3}(u_1+u_5)\right] \\
+\frac{\Delta z}{\Delta x}(\lambda+2\mu)_2\left[\frac{2}{3}u_0-\frac{1}{6}(u_6+u_8)+\frac{1}{3}u_7-\frac{1}{3}(u_1+u_5)\right] \\
+\frac{\Delta x}{\Delta z}\mu_1\left[\frac{2}{3}u_0-\frac{1}{6}(u_2+u_4)-\frac{2}{3}u_3+\frac{1}{6}(u_1+u_5)\right] \\
+\frac{\Delta x}{\Delta z}\mu_2\left[\frac{2}{3}u_0-\frac{1}{6}(u_6+u_8)-\frac{2}{3}u_7+\frac{1}{6}(u_1+u_5)\right] \\
+\frac{\lambda_1}{4}(-w_2+w_4-w_5+w_1)+\frac{\lambda_2}{4}(-w_6+w_8-w_1+w_5) \\
+\frac{\mu_1}{4}(-w_2+w_4-w_1+w_5)+\frac{\mu_2}{4}(-w_6+w_8-w_5+w_1)=0
\end{cases}
\tag{2.80}
$$

Similarly, by explaining eq. 2.75, it is obtained for any interior node:

$$
\begin{cases}
\frac{\Delta x \Delta z}{9}[2(\rho_1+\rho_2)\frac{\partial^2 w_0}{dt^2}+\frac{(\rho_1+\rho_2)}{2}\left(\frac{\partial^2 w_1}{dt^2}+\frac{\partial^2 w_5}{dt^2}\right)+\rho_1\frac{\partial^2 w_3}{dt^2}+\rho_2\frac{\partial^2 w_7}{dt^2} \\
+\frac{1}{4}\rho_1\left(\frac{\partial^2 w_2}{dt^2}+\frac{\partial^2 w_4}{dt^2}\right)+\frac{1}{4}\rho_2\left(\frac{\partial^2 w_8}{dt^2}+\frac{\partial^2 w_6}{dt^2}\right)] \\
+\frac{\Delta x}{\Delta z}(\lambda+2\mu)_1\left[\frac{2}{3}w_0-\frac{1}{6}(w_2+w_4)-\frac{2}{3}w_3+\frac{1}{6}(w_1+w_5)\right] \\
+\frac{\Delta x}{\Delta z}(\lambda+2\mu)_2\left[\frac{2}{3}w_0-\frac{1}{6}(w_6+w_8)-\frac{2}{3}w_7+\frac{1}{6}(w_1+w_5)\right] \\
+\frac{\Delta z}{\Delta x}\mu_1\left[\frac{2}{3}w_0-\frac{1}{6}(w_2+w_4)+\frac{1}{3}w_3-\frac{1}{3}(w_1+w_5)\right] \\
+\frac{\Delta z}{\Delta x}\mu_2\left[\frac{2}{3}w_0-\frac{1}{6}(w_6+w_8)+\frac{1}{3}w_7-\frac{1}{3}(w_1+w_5)\right] \\
+\frac{\lambda_1}{4}(-u_2+u_4-u_1+w_5)+\frac{\lambda_2}{4}(-u_6+u_8-u_5+u_1) \\
+\frac{\mu_1}{4}(-u_2+u_4-u_5+u_1)+\frac{\mu_2}{4}(-u_6+u_8-u_1+u_5)=0
\end{cases}
\tag{2.81}
$$

The semi-discretization in space by finite elements, therefore, leads to the system of differential equations 2.80 and 2.81 to be written for any node inside Ω to which must be added the equations 2.74 written for $R_{ij} \in \Gamma_0$ and 2.75 for $R_{ij} \in \Gamma_0 \cup \Gamma_1$.

Note: Schema for Γ_0 boundary nodes can be obtained from equations 2.80 and 2.81

with $\rho_1 = \lambda_1 = \mu_1 = 0$.

2.3.2 Time approximation scheme

The system of differential equations 2.77 is solved in an approximate way using an "explicit" numerical scheme with finite differences.

Let's set the time step Δt and approach the vector U, solution of 2.77 at time $n\Delta t$ by a vector U^n.

$$\begin{cases} \dfrac{\partial^2 U}{dt^2}(n\Delta t) \text{ for } \dfrac{1}{\Delta t^2}(U^{n+1} - 2U^n + U^{n-1}) \text{ and} \\ u(n\Delta t) \text{ for } U^n \end{cases} \tag{2.82}$$

Then, the following schema is obtained:

$$\frac{1}{\Delta t^2}(MU^{n+1} - 2MU^n + MU^{n-1}) + KU^n = 0 \tag{2.83}$$

which allows the successive calculation of U^n starting from U^1 and U^0 which one obtains using the initial conditions 2.62, 2.63, 2.64, 2.65.

The diagram will be computer-explicit only if the matrix R is diagonal which is the case of the approximation in space with condensation of the mass matrix.

	p_1	$\frac{\partial p_1}{\partial x}$	$\frac{\partial p_1}{\partial z}$	p_2	$\frac{\partial p_2}{\partial x}$	$\frac{\partial p_2}{\partial z}$	p_3	$\frac{\partial p_3}{\partial x}$	$\frac{\partial p_3}{\partial z}$	p_4	$\frac{\partial p_4}{\partial x}$	$\frac{\partial p_4}{\partial z}$
p_1	$\frac{\Delta x \Delta z}{9}$			$\frac{\Delta x \Delta z}{18}$			$\frac{\Delta x \Delta z}{36}$			$\frac{\Delta x \Delta z}{18}$		
$\frac{\partial p_1}{\partial x}$		$\frac{\Delta z}{3\Delta x}$	$\frac{1}{4}$		$-\frac{\Delta z}{3\Delta x}$	$\frac{1}{4}$		$-\frac{\Delta z}{6\Delta x}$	$-\frac{1}{4}$		$\frac{\Delta z}{6\Delta x}$	$-\frac{1}{4}$
$\frac{\partial p_1}{\partial z}$		$\frac{1}{4}$	$\frac{\Delta x}{3\Delta z}$		$-\frac{1}{4}$	$\frac{\Delta x}{6\Delta z}$		$-\frac{1}{4}$	$-\frac{\Delta x}{6\Delta z}$		$\frac{1}{4}$	$-\frac{\Delta x}{3\Delta z}$
p_2	$\frac{\Delta x \Delta z}{18}$			$\frac{\Delta x \Delta z}{9}$			$\frac{\Delta x \Delta z}{18}$			$\frac{\Delta x \Delta z}{36}$		
$\frac{\partial p_2}{\partial x}$		$-\frac{\Delta z}{3\Delta x}$	$-\frac{1}{4}$		$\frac{\Delta z}{3\Delta x}$	$-\frac{1}{4}$		$\frac{\Delta z}{6\Delta x}$	$\frac{1}{4}$		$-\frac{\Delta z}{6\Delta x}$	$\frac{1}{4}$
$\frac{\partial p_2}{\partial z}$		$\frac{1}{4}$	$\frac{\Delta x}{6\Delta z}$		$-\frac{1}{4}$	$\frac{\Delta x}{3\Delta z}$		$-\frac{1}{4}$	$-\frac{\Delta x}{3\Delta z}$		$\frac{1}{4}$	$-\frac{\Delta x}{6\Delta z}$
p_3	$\frac{\Delta x \Delta z}{36}$			$\frac{\Delta x \Delta z}{18}$			$\frac{\Delta x \Delta z}{9}$			$\frac{\Delta x \Delta z}{18}$		
$\frac{\partial p_3}{\partial x}$		$-\frac{\Delta z}{6\Delta x}$	$-\frac{1}{4}$		$\frac{\Delta z}{6\Delta x}$	$-\frac{1}{4}$		$\frac{\Delta z}{3\Delta x}$	$\frac{1}{4}$		$-\frac{\Delta z}{3\Delta x}$	$\frac{1}{4}$
$\frac{\partial p_3}{\partial z}$		$-\frac{1}{4}$	$-\frac{\Delta x}{6\Delta z}$		$\frac{1}{4}$	$-\frac{\Delta x}{3\Delta z}$		$\frac{1}{4}$	$\frac{\Delta x}{3\Delta z}$		$-\frac{1}{4}$	$\frac{\Delta x}{6\Delta z}$
p_4	$\frac{\Delta x \Delta z}{18}$			$\frac{\Delta x \Delta z}{36}$			$\frac{\Delta x \Delta z}{18}$			$\frac{\Delta x \Delta z}{9}$		
$\frac{\partial p_4}{\partial x}$		$\frac{\Delta z}{6\Delta x}$	$\frac{1}{4}$		$-\frac{\Delta z}{6\Delta x}$	$\frac{1}{4}$		$-\frac{\Delta z}{3\Delta x}$	$-\frac{1}{4}$		$\frac{\Delta z}{3\Delta x}$	$-\frac{1}{4}$
$\frac{\partial p_4}{\partial z}$		$-\frac{1}{4}$	$-\frac{\Delta x}{3\Delta z}$		$\frac{1}{4}$	$-\frac{\Delta x}{6\Delta z}$		$\frac{1}{4}$	$\frac{\Delta x}{6\Delta z}$		$-\frac{1}{4}$	$\frac{\Delta x}{3\Delta z}$

Table 2.1: Scalar Products

Chapter 3

Results and Discussion

3.1 Finite Element Method (FEM) in 1D

The implementation of FDM and FEM in Python was performed using a Gaussian source (eq. 2.14), which showed less dispersion through time than other sources. The used frequency was $f_0 = 25\text{Hz}$ (Fig. 3.1).

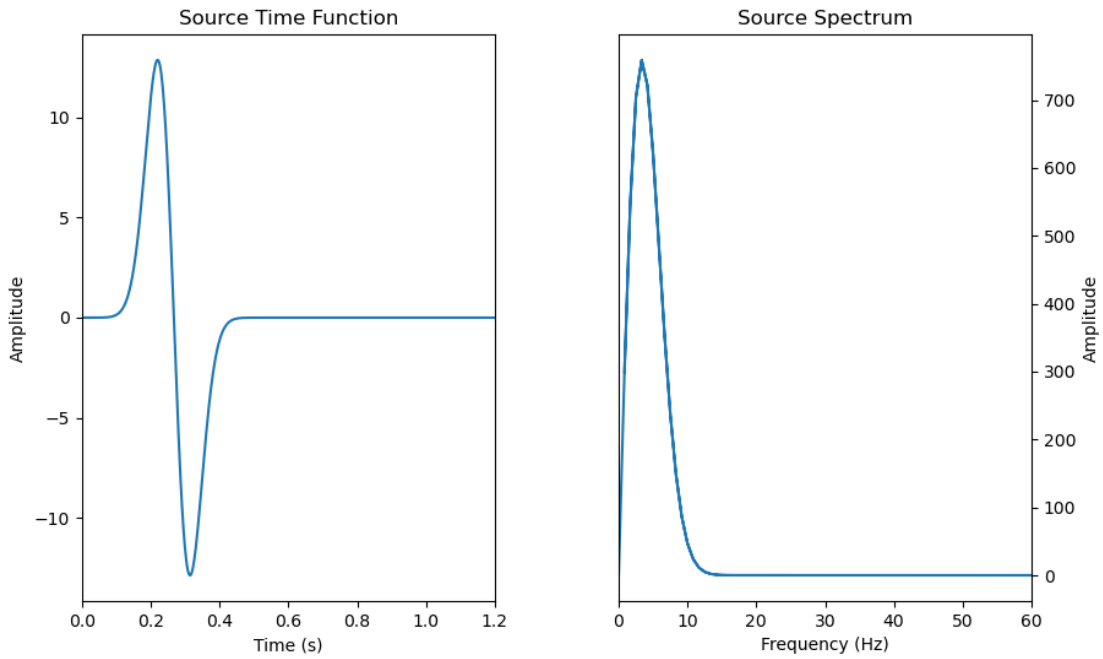

Figure 3.1: Left: Gaussian function through time, this shape presents a stable solution with low dispersion in short periods of simulation. Right: The source spectrum that shows the frequency used for the wave propagation.

3.1.1 Homogeneous Media

To find a stable increment in time and space, it was used the Courant-Friedrich-Lewy Criterion (CFL Criterion) [26], and it was found that the value for this case should be within this range $0.6 \leq c\frac{dt}{dx} \leq 1$. To accomplish this criterion it was used an s-wave velocity of $vs = 375$ m/s, a space increment of $dx = 0.65$ m, and a time increment of $dt = 0.00010$ s.

With this configuration, the simulation is stable and has a minimum dispersion at the beginning. However, for large periods of simulation, the dispersion increased significantly. To test this, it was used 1 000 000 time steps (Fig. 3.2).

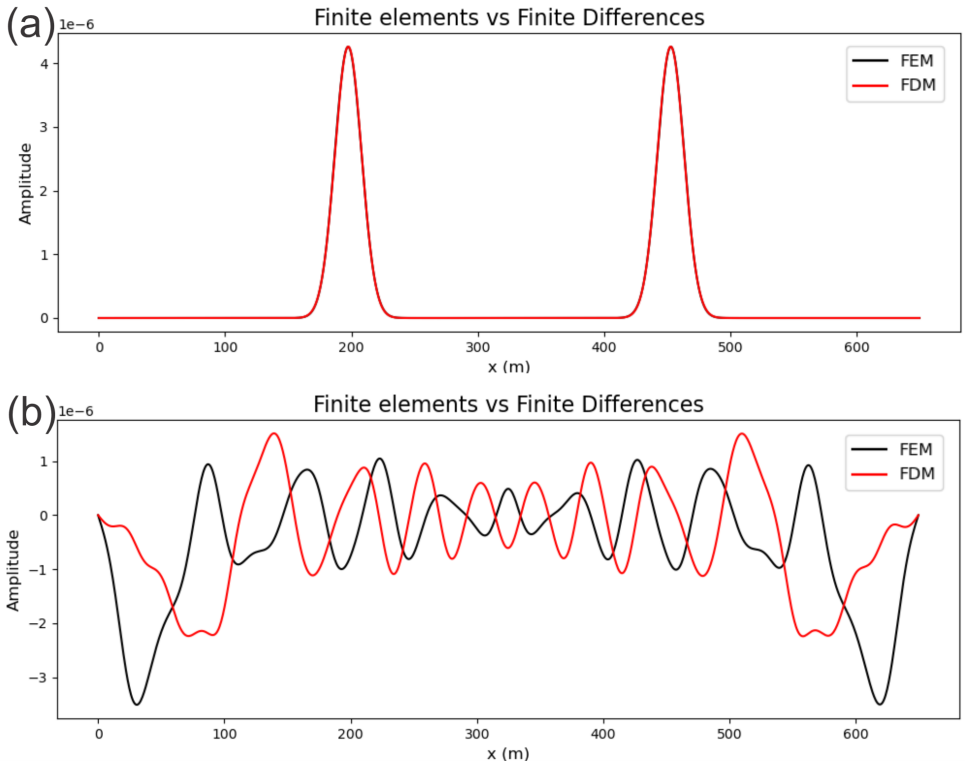

Figure 3.2: a) FDM and FEM at the beginning of the simulation. It shows almost 0 dispersion and a good match between both methods. B) FDM and FEM after 1 000 000 time steps, it is shown a gap and delay between the two signals due to the different way how boundary conditions are introduced for each method.

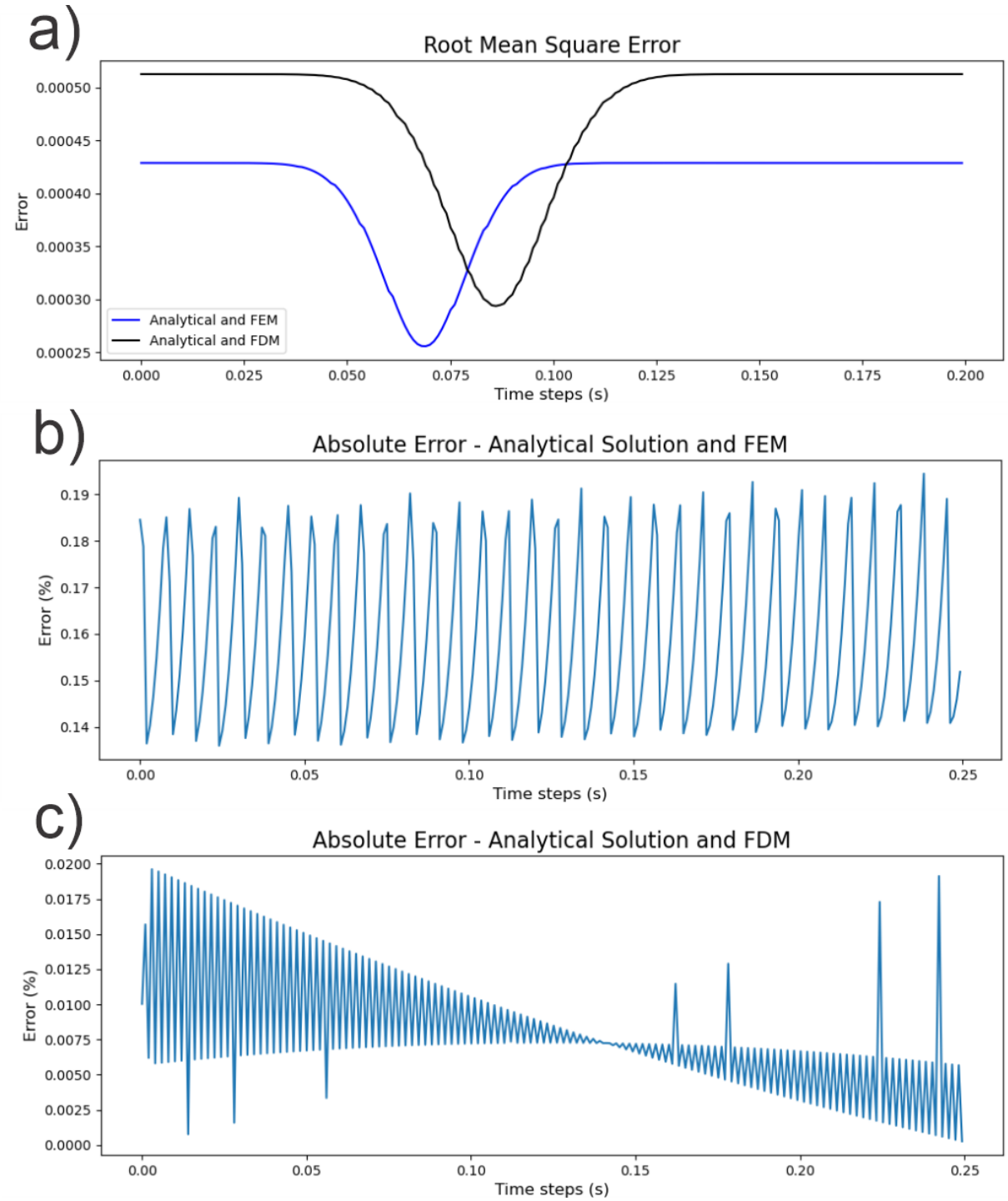

Figure 3.3: a) Root Mean Square Error (RMSE) between analytical solution and FEM-FDM, showing a better approximation of FEM since the error does not deviate in the same magnitude from the analytical solution as FDM does. b) Absolute error in percentage of the FEM with respect to the maximum amplitude showing an interval where the error varies from 0.14 % to 0.19 %. c) Absolute error of the FDM showing a decreasing tendency from a maximum value of 0.0019 % to a minimum value of 0.00026 %.

To calculate the compilation time, the code was run using 24000 time-steps. The result was that the FDM takes less time to compile the code (see Table 3.1).

Time (s)		
FEM and FDM	**FDM**	**FEM**
500.8159170150	279.3736398220	305.5113596916

Table 3.1: Compilation times

The comparison between both methods is validated since it has been proved that the Finite Difference Method is stable and converges to analytical solutions for the acoustic wave case; to reach this aim, it was applied the Dirichlet and Neuman analysis for stability and convergence [7]. In addition, it was seen that FDM is stable for values of the CFL criterion ≤ 1. It means that FDM is a reliable approach for approximating the acoustic wave equation. By comparing FDM and FEM it was observed that FEM eventually converges to the FDM (Fig. 3.4), since the error starts to converge and its curve flattens in large periods of compilations. The error calculated is the root mean square error:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_i - u_i)^2} \tag{3.1}$$

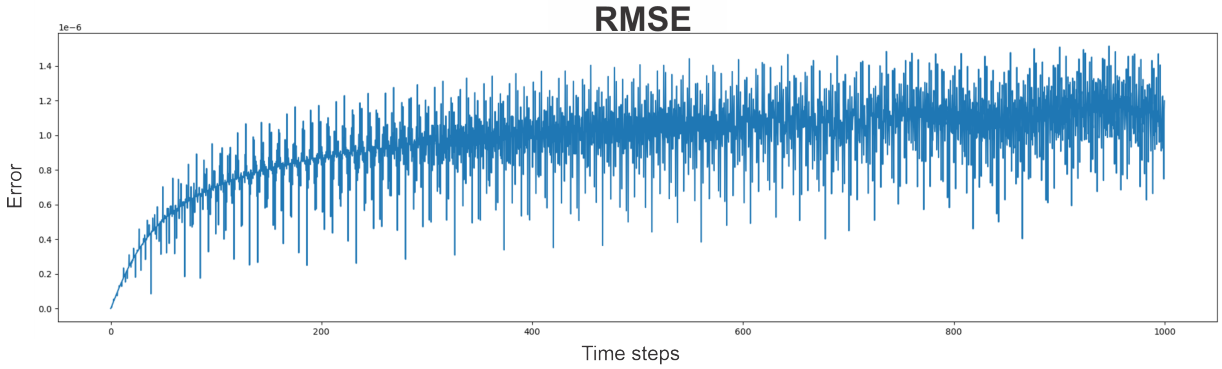

Figure 3.4: Root Mean square error between FEM and FDM. The image shows how the error curve converges; i.e., the curve starts to flatten. However, it is clear the perturbations provoked by the gap and delay produced by the boundary conditions

Where p_i is the value for FDM and u_i is the value for FEM. However, there is an important observation for the error calculation. There are disruptions and inconsistencies between both methods at the edges, since the boundary conditions are implemented in different manners this produces a gap between FDM and FEM that in large periods of

simulation will create a delay between both signals. To get a better understanding of the methods a comparison between numerical methods and analytical solutions was performed using the root mean square error (eq. 3.1) and the absolute error in percentage:

$$Error(\%) = \left| \frac{X_{Analytical} - X_{Numerical}}{X_{Analytical}} \right| * 100 \tag{3.2}$$

Where X is the amplitude value of the wavelength for each case. For the root mean square error p_i must be replaced by the analytical solution vector, and u_i must be the FDM and FEM vectors for each case. In general, it was observed the FEM approaches better the analytical solution since its RMSE is less than FDM; i.e., the FEM does not deviate from the values so far from the analytical solution as FDM does. Nevertheless, both methods have a stable error that could decrease but after that will be kept constant and stable (Fig. 3.3a). In contrast, the absolute error shows that for FEM the error is bounded between 0.14 % and 0.19 % which means that the amplitude is slightly different than expected (Fig. 3.3b); while for FDM the error shows a decreasing tendency with a maximum value of 0.019 % and a minimum value of 0.00026 %, which means a better approach through time to the analytical solution (Fig. 3.3c). Then, the amplitude of the FDM tends to converge to the analytical solution while the amplitude of FEM keeps constant within an interval through time.

3.1.2 Two Layers

The two-layer case was performed to simulate the wave behavior when it reaches an interface. This interface divides the medium creating two layers with different properties that are the common case; i.e., a heterogenous medium.

For layer 1 the properties are: $Vs_1 = 350\ m/s$ and $\rho_1 = 300\ kg/m^3$, for layer 2 the properties are: $Vs_2 = 375\ m/s$ and $\rho_2 = 315\ kg/m^3$.

It has been observed that the wave for both methods has a stable behavior, and both waves have the same shape without any changes at the beginning of the simulation (fig. 3.5a). However, after the waves reach the interface, both methods have different amplitudes. The transmitted wave has a higher amplitude for FDM and the reflected wave has a different polarity. The FEM has a positive polarity while the FDM has a negative polarity

(fig. 3.5b).

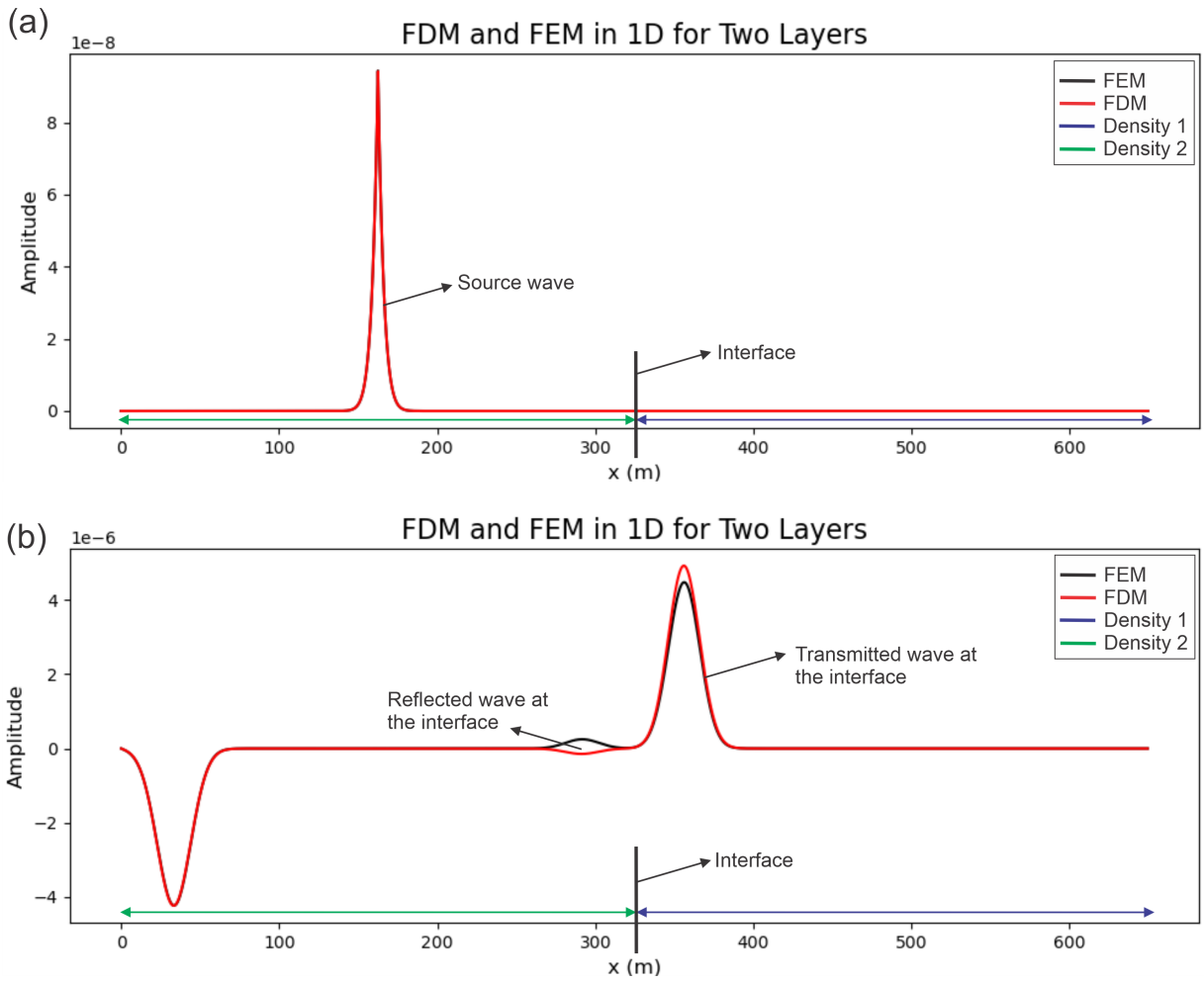

Figure 3.5: a)First wave after the activation of the source. b) Reflected and transmitted waves after the source wave reaches the interface between the two densities

In the same way for the homogeneous medium, the RMSE shows some deviation between both methods due to the boundary conditions. The methods have different behavior at the edges of the simulation. Also, the behavior changes at the interface (fig. 3.5b) where the reflected wave polarity is different. However, the RMSE shows a convergence tendency; even, it starts to decrease over time (Fig. 3.6). Nevertheless, there are some discrepancies due to the delay and gaps between both signals due to the boundary conditions and the interface.

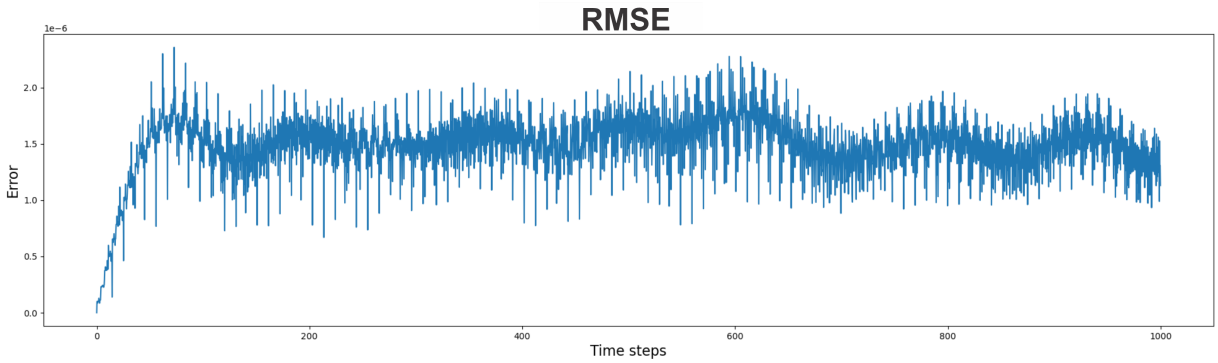

Figure 3.6: RMSE between FDM and FEM, it is showing a convergence tendency with a decrease in the error magnitude which means the methods start to have less deviation between them over time.

3.2 Finite Element Method (FEM) road to 2D

In order to implement the FEM for 2D using the procedure developed in Section 2, it is necessary to assemble explicitly the Mass and Stiffness matrices. Let's start with the numerical approximation:

$$u_h = \sum_{\substack{i=0,I \\ j=0,J}} u_{ij}(t)P_{ij}(x,z);\ w_h = \sum_{\substack{i=0,I \\ j=0,J}} w_{ij}(t)P_{ij}(x,z) \tag{3.3}$$

Where u_{ij} and w_{ij} are dependent on time and P_{ij} are the basis functions. The numerical approximations for the elastic wave equation to approach the displacement u_h and w_h are:

$$\begin{cases} \int_\Omega \rho \sum\limits_{\substack{i=0,I \\ j=0,J}} u''_{ij}P_{ij}P_{i'j'} + \int_\Omega (\lambda+2\mu)\partial x(\sum\limits_{\substack{i=0,I \\ j=0,J}} u_{ij}P_{ij})\partial x P_{i'j'} \\ + \int_\Omega \lambda \partial z(\sum\limits_{\substack{i=0,I \\ j=0,J}} w_{ij}P_{ij})\partial z P_{i'j'} + \int_\Omega \mu \left[\partial z(\sum\limits_{\substack{i=0,I \\ j=0,J}} u_{ij}P_{ij}) + \partial x(\sum\limits_{\substack{i=0,I \\ j=0,J}} w_{ij}P_{ij}) \right] \partial z P_{i'j'} = 0 \end{cases} \tag{3.4}$$

$$\begin{cases} \int_\Omega \rho \sum\limits_{\substack{i=0,I \\ j=0,J}} w''_{ij} P_{ij} P_{i'j'} + \int_\Omega (\lambda + 2\mu) \partial z (\sum\limits_{\substack{i=0,I \\ j=0,J}} w_{ij} P_{ij}) \partial x P_{i'j'} \\ + \int_\Omega \lambda \partial x (\sum\limits_{\substack{i=0,I \\ j=0,J}} u_{ij} P_{ij}) \partial z P_{i'j'} + \int_\Omega \mu \left[\partial z (\sum\limits_{\substack{i=0,I \\ j=0,J}} u_{ij} P_{ij}) + \partial x (\sum\limits_{\substack{i=0,I \\ j=0,J}} w_{ij} P_{ij}) \right] \partial x P_{i'j'} = 0 \end{cases} \tag{3.5}$$

Mass matrix (M)

The mass matrix will be calculated using the following expression:

$$M = \int_\Omega \rho \sum_{\substack{i=0,I \\ j=0,J}} P_{ij} P_{i'j'} \tag{3.6}$$

Stiffness matrix (K)

The Global Stiffness matrix (K) is composed of the sum of three matrices, each one corresponding to the partial derivatives of the basis function with respect to each component $K_{xx}, K_{zz},$ and $K_{xz} = K_{zx}$, where:

$$\begin{cases} K_{xx} = \int_\Omega \Upsilon \partial x(P_{ij}) \partial x P_{i'j'} \\ K_{zz} = \int_\Omega \Upsilon \partial z(P_{ij}) \partial z P_{i'j'} \quad ; \ \Upsilon \text{ is a Lamé parameter or a combination of them} \\ K_{xz} = \int_\Omega \Upsilon \partial x(P_{ij}) \partial z P_{i'j'} \end{cases} \tag{3.7}$$

Matrix notation

Then, using the eq. 3.6 and eq. 3.7, the system of partial differential equations eq. 3.4 and eq. 3.5 can be written as:

$$\begin{cases} M u''_{ij} + [(\lambda + 2\mu) K_{xx} + \mu K_{zz}] u_{ij} + [\lambda k_{zz} + \mu K_{xz}] w_{ij} = 0 \\ M w''_{ij} + [\lambda K_{xz} + \mu K_{xx}] u_{ij} + [(\lambda + 2\mu) K_{zz} + \mu K_{xx}] w_{ij} = 0 \end{cases} ; \text{respectively} \tag{3.8}$$

The Lamé parameters are not into the integrals of each matrix just to clarify what parameters should be integrated for each combination of the partial Stiffness matrices; i.e., they must be included for each matrix solution. Following the eq. 2.76, the system of differential equations 3.7 can be expressed as the eq. 2.77. It is clear that the system is coupled; i.e., both systems must be solved together as follows:

$$\begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} \begin{pmatrix} u''_{ij} \\ w''_{ij} \end{pmatrix} + \begin{pmatrix} (\lambda+2\mu)K_{xx} + \mu K_{zz} & \lambda K_{zz} + \mu K_{xz} \\ \lambda K_{xz} + \mu K_{xx} & (\lambda+2\mu)K_{zz} + \mu K_{xx} \end{pmatrix} \begin{pmatrix} u_{ij} \\ w_{ij} \end{pmatrix} = 0 \qquad (3.9)$$

Before the assembly of the matrices let's introduce another index k which will enumerate the elements from $k = 1, ..., K$. The number of elements will be $K = (I-2) * (J-2)$ since the border elements are not included in making the calculations due to the boundary conditions; i.e., their values are zero. This means the elements are only the internal points (Fig: 3.7) which implies that:

$$\begin{cases} I' = (I-2) * \Delta x \\ J' = (J-2) * \Delta z \end{cases} \qquad (3.10)$$

This denotes that $K = I' * J'$ and the relation between the indexes (i, j) are related with k as follows:

$$\begin{cases} (i,j) \to k = (i-1) * J' + j \\ k \to i = \left\lfloor \dfrac{k}{J} \right\rfloor + 1; \; j = k - I' * (i-1) \end{cases} ; \; i = 1, ..., I'; \; j = 1, ..., J' \qquad (3.11)$$

To explain better how the matrices are filled, let's clarify the size. The size of the matrices is $K * K$ since it is necessary to calculate the interaction of each element with the whole mesh (the other elements). Therefore, these combinations create matrices of size $K * K$, where the diagonal is filled with the interactions of each element with itself. Furthermore, it is needed to fill the diagonal and the upper values due to the matrices are symmetric; then, the bottom values will be filled by symmetry with the upper values

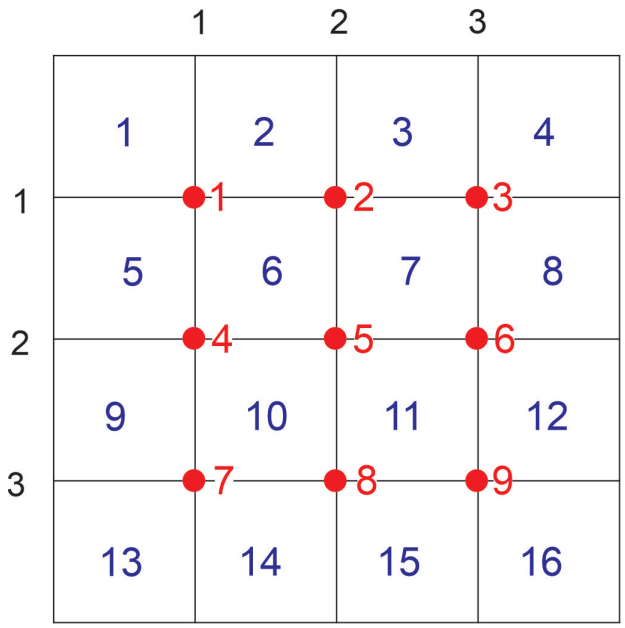

Figure 3.7: An example of $5 * 5$ grid points and $3 * 3$ elements. Numeration of elements using the index k (red) and the numbering for each quadrilateral Q_h (blue). The numbering outside the mesh represents the indexes for rows and columns of each element

already calculated. The spaces of the matrices with values different from zero will be those ones where the fixed element interacts with the neighboring elements; i.e., the positions $(k, k+1)$, $(k, k+J'-1)$, $(k, k+J')$, and $(k, k+J'+1)$

3.2.1 1-Layer homogeneous medium

The one-layer isotropic medium is a good approach for understanding how the method works and seeing how the matrices are built.

Explicit assembly of the mass matrix

As it was established, the elements of the matrices must be filled depending on the interaction of the elements of the mesh with itself and the neighboring elements. The combination with the non-neighboring elements will be zero since there is no interaction. The solution is obtained using the reference quadrilateral in Fig. 2.6b, and the scalar products in $L2$ are already calculated in Table 2.1.

Then, let's consider:

$$M = \int_{\Omega} \rho \sum_{\substack{i=0,I \\ j=0,J}} P_{ij} P_{i'j'}$$

Changing into the local coordinate system and considering an isotropic 1-layer medium, we have that

$$M_{k,k'} = \rho \int_{\hat{Q}} P_{ij} P_{i'j'}; \text{ where } \hat{Q} \text{ is the set of quadrilaterals involved.} \tag{3.12}$$

- **Diagonal**

  The elements of the diagonal are $M_{k,k'}$ where $k' = k$. Then, solving the eq. 3.12, for this case, we have that:

$$M_{k,k'} = \rho \left[\int_{Q_k} P_{ij} P_{i'j'} + \int_{Q_{k+i}} P_{ij} P_{i'j'} + \int_{Q_{k+J'+i}} P_{ij} P_{i'j'} + \int_{Q_{k+J'+i+1}} P_{ij} P_{i'j'} \right]$$

  Turn each integral into the coordinates for the corresponding quadrilateral reference (Fig. 2.6), and we have that:

$$M_{k,k'} = \rho \left[\int_{Q_k} P_2 P_2 + \int_{Q_{k+i}} P_1 P_1 + \int_{Q_{k+J'+i}} P_3 P_3 + \int_{Q_{k+J'+i+1}} P_4 P_4 \right]$$

$$M_{k,k'} = \rho \frac{4}{9} \Delta x \Delta z \tag{3.13}$$

- **Right side**

  The solution for the interactions with the right side element is $M_{k,k'}$, where $k' = k+1$ and $(i', j') = (i, j+1)$. Then, solving the eq. 3.12, for this case, we have that:

$$M_{k,k'} = \rho \left[\int_{Q_{k+i}} P_{ij} P_{i'j'} + \int_{Q_{k+J'+i+1}} P_{ij} P_{i'j'} \right]$$

  Turn each integral into the coordinates for the corresponding quadrilateral of reference (Fig. 2.6):

$$M_{k,k'} = \rho\left[\int_{Q_{k+i}} P_1P_2 + \int_{Q_{k+J'+i+1}} P_4P_3\right]$$

$$M_{k,k'} = \rho\frac{1}{9}\Delta x\Delta z \tag{3.14}$$

- **Down left**

  Considering the eq: 3.12, let's solve $M_{k,k'}$, where $k' = k + J' - 1$, and $(i', j') = (i + 1, j - 1)$.

$$M_{k,k'} = \rho\left[\int_{Q_{k+J'+i}} P_{ij}P_{i'j'}\right]$$

$$M_{k,k'} = \rho\left[\int_{Q_{k+J'+i}} P_3P_1\right]$$

$$M_{k,k'} = \rho\frac{1}{36}\Delta x\Delta z \tag{3.15}$$

- **Down**

  Considering the eq: 3.12, let's solve $M_{k,k'}$, where $k' = k + J'$, and $(i', j') = (i + 1, j)$.

$$M_{k,k'} = \rho\left[\int_{Q_{k+J'+i}} P_{ij}P_{i'j'} + \int_{Q_{k+J'+i+1}} P_{ij}P_{i'j'}\right]$$

$$M_{k,k'} = \rho\left[\int_{Q_{k+J'+i}} P_3P_2 + \int_{Q_{k+J'+i+1}} P_4P_1\right]$$

$$M_{k,k'} = \rho\frac{1}{9}\Delta x\Delta z \tag{3.16}$$

- **Down right**

  Considering the eq: 3.12, let's solve $M_{k,k'}$, where $k' = k + J' + 1$, and $(i', j') = (i + 1, j + 1)$.

$$M_{k,k'} = \rho\left[\int_{Q_{k+J'+i+1}} P_{ij}P_{i'j'}\right]$$

$$M_{k,k'} = \rho\left[\int_{Q_{k+J'+i+1}} P_4P_2\right]$$

$$M_{k,k'} = \rho\frac{1}{36}\Delta x\Delta z \tag{3.17}$$

With these solutions, it is clear that the shape of the matrices will vary depending on the number of elements. To illustrate how looks like the shape, let's consider Figure 3.7. This mesh will create a matrix size of $9X9$ as follows:

$$M = \rho\frac{\Delta x\Delta z}{9}\begin{pmatrix} 4 & 1 & 0 & 1 & 1/4 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 1/4 & 1 & 1/4 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & 1/4 & 1 & 0 & 0 & 0 \\ 1 & 1/4 & 0 & 4 & 1 & 0 & 1 & 1/4 & 0 \\ 1/4 & 1 & 1/4 & 1 & 4 & 1 & 1/4 & 1 & 1/4 \\ 0 & 1/4 & 1 & 0 & 1 & 4 & 0 & 1/4 & 1 \\ 0 & 0 & 0 & 1 & 1/4 & 0 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1/4 & 1 & 1/4 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & 1/4 & 1 & 0 & 1 & 4 \end{pmatrix} \tag{3.18}$$

Explicit assembly of the Stiffness Matrix

In order to build the Global Stiffness matrix, it is necessary to first build the partial stiffness matrices. The procedure to follow is the same as the one taken for building the Mass matrix, it just changes the evaluation of the integrals with the corresponding partial derivatives. Then, the results can be summarized in Table 3.2, in this table the indexes correspond to $(k,k') = (k,k+1)$, $(k,k'') = (k,k+J'-1)$, $(k,k''') = (k,k+J')$, and $(k,k^4) = (k,k+J'+1)$

Solving for Figure 3.7, the partial stiffness matrices have the shape:

Indexes	$K_{xx} = \int_\Omega \Upsilon \partial x(P_{ij}) \partial x P_{i'j'}$	$K_{zz} = \int_\Omega \Upsilon \partial z(P_{ij}) \partial z P_{i'j'}$	$K_{xz} = \int_\Omega \Upsilon \partial x(P_{ij}) \partial z P_{i'j'}$
(k,k)	$K_{xx}^{(k,k)} = \Upsilon \frac{4\Delta z}{3\Delta x}$	$K_{zz}^{(k,k)} = \Upsilon \frac{4\Delta x}{3\Delta z}$	$K_{xz}^{(k,k)} = 0$
(k,k')	$K_{xx}^{(k,k')} = -\Upsilon \frac{2\Delta z}{3\Delta x}$	$K_{zz}^{(k,k')} = \Upsilon \frac{\Delta x}{3\Delta z}$	$K_{xz}^{(k,k')} = 0$
(k,k'')	$K_{xx}^{(k,k'')} = -\Upsilon \frac{\Delta z}{6\Delta x}$	$K_{zz}^{(k,k'')} = -\Upsilon \frac{\Delta x}{6\Delta z}$	$K_{xz}^{(k,k'')} = -\Upsilon \frac{1}{4}$
(k,k''')	$K_{xx}^{(k,k''')} = \Upsilon \frac{\Delta z}{3\Delta x}$	$K_{zz}^{(k,k''')} = -\Upsilon \frac{2\Delta x}{3\Delta z}$	$K_{xz}^{(k,k''')} =$
(k,k^4)	$K_{xx}^{(k,k^4)} = -\Upsilon \frac{\Delta z}{3\Delta x}$	$K_{zz}^{(k,k^4)} = -\Upsilon \frac{\Delta x}{6\Delta z}$	$K_{xz}^{(k,k^4)} = \Upsilon \frac{1}{4}$

Table 3.2: Partial Stiffness Matrices

$$K_{xx} = \Upsilon \frac{\Delta z}{3\Delta x} \begin{pmatrix} 4 & -2 & 0 & 1 & -1/2 & 0 & 0 & 0 & 0 \\ -2 & 4 & -2 & -1/2 & 1 & -1/2 & 0 & 0 & 0 \\ 0 & -2 & 4 & 0 & -1/2 & 1 & 0 & 0 & 0 \\ 1 & -1/2 & 0 & 4 & -2 & 0 & 1 & -1/2 & 0 \\ -1/2 & 1 & -1/2 & -2 & 4 & -2 & -1/2 & 1 & -1/2 \\ 0 & -1/2 & 1 & 0 & -2 & 4 & 0 & -1/2 & 1 \\ 0 & 0 & 0 & 1 & -1/2 & 0 & 4 & -2 & 0 \\ 0 & 0 & 0 & -1/2 & 1 & -1/2 & -2 & 4 & -2 \\ 0 & 0 & 0 & 0 & -1/2 & 1 & 0 & -2 & 4 \end{pmatrix} \quad (3.19)$$

$$K_{zz} = \Upsilon \frac{\Delta x}{3\Delta z} \begin{pmatrix} 4 & 1 & 0 & -2 & -1/2 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & -1/2 & -2 & -1/2 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & -1/2 & -2 & 0 & 0 & 0 \\ -2 & -1/2 & 0 & 4 & 1 & 0 & -2 & -1/2 & 0 \\ -1/2 & -2 & -1/2 & 1 & 4 & 1 & -1/2 & -2 & -1/2 \\ 0 & -1/2 & -2 & 0 & 1 & 4 & 0 & -1/2 & -2 \\ 0 & 0 & 0 & -2 & -1/2 & 0 & 4 & 1 & 0 \\ 0 & 0 & 0 & -1/2 & -2 & -1/2 & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & -1/2 & -2 & 0 & 1 & 4 \end{pmatrix} \tag{3.20}$$

$$K_{xz} = \Upsilon \frac{1}{4} \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{3.21}$$

Once the matrices are built, it is possible to continue with the implementation in Python to build the matrices and start the simulation.

Let's consider the eq: 2.83. The first step is to introduce a source for each displacement component, the source used is the same for the 1D case. Now, let's solve the eq. 2.83 for U^{n+1}:

$$U^{n+1} = M^{-1}[\Delta t^2 \mathcal{F} - KU^n] + 2U^n - U^{n-1} \tag{3.22}$$

The implementation of the numerical approximation starts by incorporating the proper libraries, setting the data to build the matrices and values to be integrated into the matrices (Fig. 3.8); once the data is set, the matrices are built and filled using a for loop (Fig.

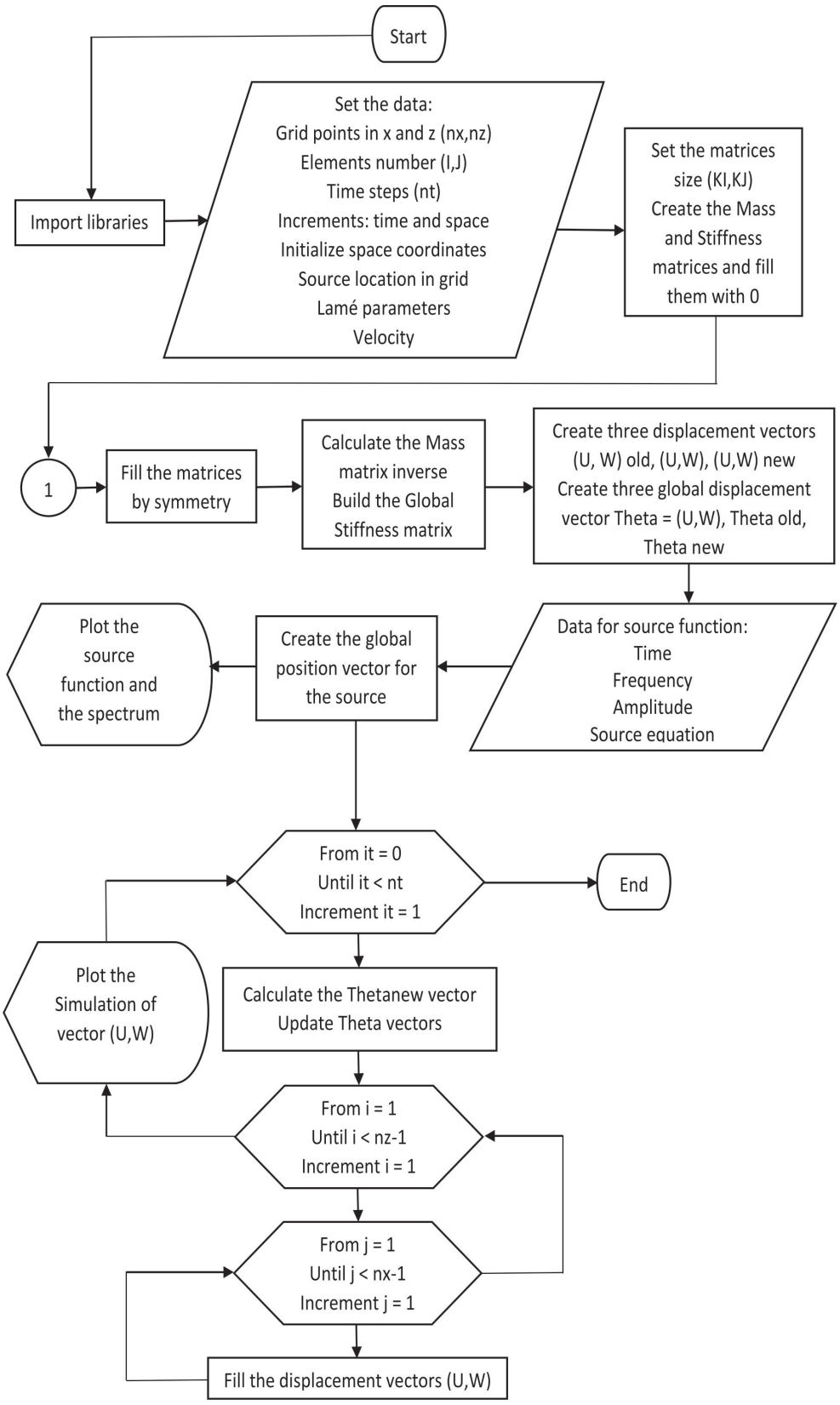

Figure 3.8: Flux diagram of how the code works to simulate the numerical approximation done using the Finite element method. Circle 1 is how to build the matrices, it can be checked in Fig 3.9.

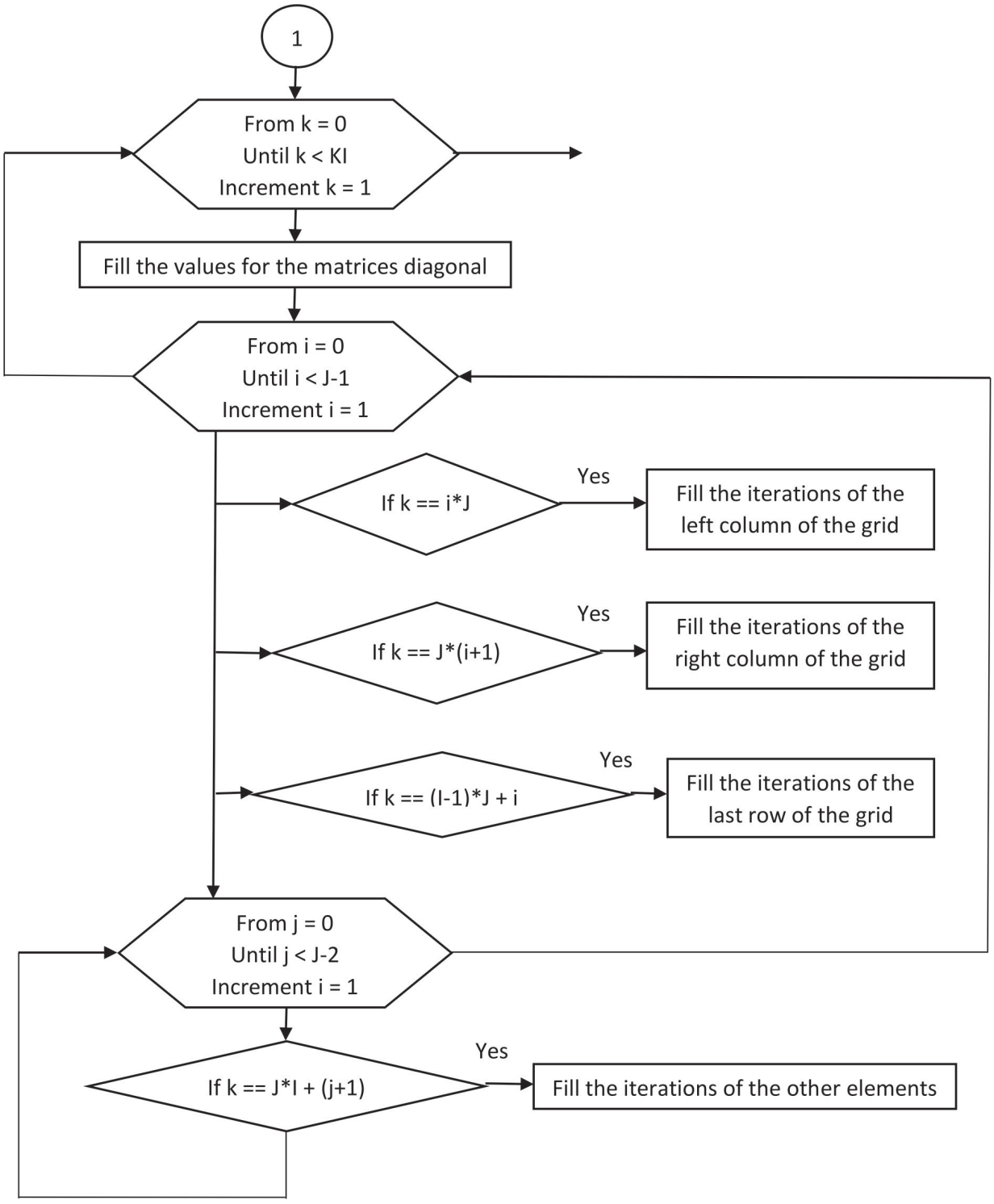

Figure 3.9: Flux diagram for building the Mass and Stiffness matrices. This flux diagram is appliable for 1 to 2 layers since the procedure is the same, it is just necessary to fill each place of the matrices with the proper Lamé parameter

3.9) and complemented by symmetry. Afterward, the source function is initialized and incorporated into the equation obtained form the application of FEM to the elastic wave equation. A code built in for python can be seen in Appendix .2

3.2.2 Two Layers

The formulation to build the 2 layered matrices is the same for the 1 layered. However, when we are solving the integrals, it is necessary to consider the lame parameters for each quadrilateral in order to solve the integral (Fig: 3.7).

Then the solutions for each position of the Mass and stiffness matrices are the following (The Lamé parameter's subscripts are for the quadrilateral values):

- **Position** $(k, k') = (k, k)$

$$M_{k,k'} = \frac{\Delta x \Delta z}{9} \left[\rho_k + \rho_{k+i} + \rho_{k+J'+i} + \rho_{k+J'+i+1}\right] \tag{3.23}$$

$$K_{xx}^{k,k'} = \frac{\Delta z}{3\Delta x} \left[\Upsilon_k + \Upsilon_{k+i} + \Upsilon_{k+J'+i} + \Upsilon_{k+J'+i+1}\right] \tag{3.24}$$

$$K_{zz}^{k,k'} = \frac{\Delta x}{3\Delta z} \left[\Upsilon_k + \Upsilon_{k+i} + \Upsilon_{k+J'+i} + \Upsilon_{k+J'+i+1}\right] \tag{3.25}$$

$$K_{xz}^{k,k'} = -\frac{1}{4}\Upsilon_k + \frac{1}{4}\Upsilon_{k+i} + \frac{1}{4}\Upsilon_{k+J'+i} - \frac{1}{4}\Upsilon_{k+J'+i+1} \tag{3.26}$$

- **Position** $(k, k') = (k, k+1)$

$$M_{k,k'} = \frac{\Delta x \Delta z}{18} \left[\rho_{k+i} + \rho_{k+J'+i+1}\right] \tag{3.27}$$

$$K_{xx}^{k,k'} = -\frac{\Delta z}{3\Delta x} \left[\ \Upsilon_{k+i} + \Upsilon_{k+J'+i+1}\right] \tag{3.28}$$

$$K_{zz}^{k,k'} = \frac{\Delta x}{6\Delta z} \left[\Upsilon_{k+i} + \Upsilon_{k+J'+i+1}\right] \tag{3.29}$$

$$K_{xz}^{k,k'} = \frac{1}{4}\Upsilon_{k+i} - \frac{1}{4}\Upsilon_{k+J'+i+1} \tag{3.30}$$

- **Position** $(k, k') = (k, k + J' - 1)$

$$M_{k,k'} = \frac{\Delta x \Delta z}{36}\rho_{k+J'+i} \tag{3.31}$$

$$K_{xx}^{k,k'} = -\frac{\Delta z}{6\Delta x}\Upsilon_{k+J'+i} \tag{3.32}$$

$$K_{zz}^{k,k'} = -\frac{\Delta x}{6\Delta z}\Upsilon_{k+J'+i} \tag{3.33}$$

$$K_{xz}^{k,k'} = -\frac{1}{4}\Upsilon_{k+J'+i} \tag{3.34}$$

- **Position** $(k, k') = (k, k + J')$

$$M_{k,k'} = \frac{\Delta x \Delta z}{18}\left[\rho_{k+J'+i} + \rho_{k+J'+i+1}\right] \tag{3.35}$$

$$K_{xx}^{k,k'} = \frac{\Delta z}{6\Delta x}\left[\ \Upsilon_{k+J'+i} + \Upsilon_{k+J'+i+1}\right] \tag{3.36}$$

$$K_{zz}^{k,k'} = -\frac{\Delta x}{3\Delta z}\left[\Upsilon_{k+J'+i} + \Upsilon_{k+J'+i+1}\right] \tag{3.37}$$

$$K_{xz}^{k,k'} = -\frac{1}{4}\Upsilon_{k+J'+i} + \frac{1}{4}\Upsilon_{k+J'+i+1} \tag{3.38}$$

- **Position** $(k, k') = (k, k + J' + 1)$

$$M_{k,k'} = \frac{\Delta x \Delta z}{36}\rho_{k+J'+i+1} \tag{3.39}$$

$$K_{xx}^{k,k'} = -\frac{\Delta z}{3\Delta x}\Upsilon_{k+J'+i+1} \tag{3.40}$$

$$K_{zz}^{k,k'} = -\frac{\Delta x}{6\Delta z}\Upsilon_{k+J'+i+1} \tag{3.41}$$

$$K_{xz}^{k,k'} = \frac{1}{4}\Upsilon_{k+J'+i+1} \tag{3.42}$$

Then, the mass and stiffness matrices can be filled using the equations from eq. 3.23 and 3.42. These equations are for a general case where each quadrilateral has its own Lamé parameter value. Therefore, to build a two-layer mesh, it is necessary to set the interface and the corresponding values for each layer.

The implementation is the same for the case of 1 layer case, the only difference is how the matrices are built. The procedure to simulate is the same for the 1 layered case; however, when the matrices are filled with the proper values, it is necessary to establish the correct Lamé parameters for each place that is well explained in the script in Appendix .3 that is valid for Python. In addition, we must create additional partial Stiffness matrices according to the Lamé parameters preceding these matrices following the eq. 3.9. Therefore, the Global Stiffness matrix is built.

Once the matrices are implemented the simulation can be started. To avoid dispersion, it is necessary to consider the proper values for whole parameters, for this it is important to take into consideration the stability condition for finite elements in 2D [20]:

$$\sqrt{V_p^2 + V_s^2}\frac{\Delta t}{\Delta x} \leq \frac{1}{\sqrt{3}} \tag{3.43}$$

With this stability condition, we are ensuring to avoid numerical dispersion.

The more efficient way to implement the numerical approximation is using matrices since all calculations are done at once; however, there are some inconveniences. Computer memory plays an important role since the matrices grow significantly when the number of grid points increases too. Also, it is important to consider the source type and how it is going to be implemented in the simulation in order to obtain accurate results.

Moreover, the implementation looks to get better results than finite differences. Nev-

ertheless, the procedure to apply FEM to a partial differential equation, in this case, the elastic wave equation, results harder than using FDM due to the approximation using basis functions and the integration over the local coordinate system. Furthermore, there are some advantages to this approximation. The implementation of boundary conditions in order to solve the PDE allows to setting and simulation of a proper behavior of the seismic waves through a medium. Also, the elastic properties can be attached to each element, this has significant consequences since it is possible to approach the real medium using a more realistic distribution of elastic properties through the subsurface to represent lithology changes, faults, and geological features.

Chapter 4

Conclusions

FEM can be treated as a good approach for the elastic wave equation in 1D since it converges to the FDM. However, it presents some disadvantages such as it needs more time to compile and the procedure to discretize and express the equation using the basis functions can turn into a complicated work for more dimensions or more structured equations. Furthermore, both methods vary from each other due to the way the boundary and initial conditions are implemented. However, both methods show a reliable approach to the analytical solution where the FEM presents less deviation to this solution while FDM shows a bigger discrepancy from the analytical solution. Although, the FDM approaches better to real amplitude and shows a convergence tendency to analytical solutions while FEM is bounded showing a discrepancy to the exact solution.

In the case of the 2D, FEM presents a numerical procedure that can seem harder than FDM; however, the results present accurate approaches to the elastic wave equation since it is possible the set the proper and desired boundary conditions and the elastic properties can be arranged according to the geological setting that will be described. The assembly of the matrices allows to perform an efficient way to implement the numerical solution in coding; however, it presents some disadvantages like the computer memory, which occurs because the size's matrices increase rapidly as the grid points increase too. Moreover, with this building, it is possible to simulate the behavior of the seismic waves with a proper implementation of the source and the boundary conditions which lead to future comparisons with other methods such as FDM. Finally, this study contributes to the FWI

since it gives the mathematical procedure for the direct problem of approaching the PDE of the elastic wave. This gives a better understanding of how the inverse problem can be treated.

Bibliography

[1] J. Virieux, A. Asnaashari, R. Brossier, L. Métivier, A. Ribodetti, and W. Zhou, *6. An introduction to full waveform inversion*, 2017, pp. R1–1–R1–40. [Online]. Available: https://library.seg.org/doi/abs/10.1190/1.9781560803027.entry6

[2] Q. Kong, T. Siauw, and A. Bayen, *Python Programming and Numerical Methods: A Guide for Engineers and Scientists*. Elsevier Science, 2020. [Online]. Available: https://books.google.com.ec/books?id=cZ4LEAAAQBAJ

[3] G.-R. Liu and S. S. Quek, *The finite element method: a practical course*. Butterworth-Heinemann, 2013.

[4] D. Komatitsch, J. Ritsema, and J. Tromp, "The spectral-element method, beowulf computing, and global seismology," *Science*, vol. 298, no. 5599, pp. 1737–1742, 2002.

[5] F. Moukalled, L. Mangani, M. Darwish, F. Moukalled, L. Mangani, and M. Darwish, *The finite volume method*. Springer, 2016.

[6] R. Hohensinn, "Detection of hazardous ground movements with instantaneous velocity estimates by gnss," Ph.D. dissertation, 06 2019.

[7] K. N. Habib, "Convergence analysis for wave equation by explicit finite difference equation with drichlet and neumann boundary condition," *Matrix*, vol. 2, no. 2, p. 9, 2021.

[8] H. Chauris, *Chapter 5 Full waveform inversion*. EDP Sciences, 2021, pp. 123–146. [Online]. Available: https://doi.org/10.1051/978-2-7598-2351-2.c007

[9] I. R. Khan and R. Ohba, "Closed-form expressions for the finite difference approximations of first and higher derivatives based on taylor series," *Journal of Computational and Applied Mathematics*, vol. 107, no. 2, pp. 179–193, 1999.

[10] P. M. Shearer, *Introduction to seismology*. Cambridge university press, 2019.

[11] H. Igel, *Computational seismology: a practical introduction*. Oxford University Press, 2017.

[12] A. T. Patera, "A spectral element method for fluid dynamics: laminar flow in a channel expansion," *Journal of computational Physics*, vol. 54, no. 3, pp. 468–488, 1984.

[13] R. Eymard, T. Gallouët, and R. Herbin, "Finite volume methods," in *Solution of Equation in n (Part 3), Techniques of Scientific Computing (Part 3)*, ser. Handbook of Numerical Analysis. Elsevier, 2000, vol. 7, pp. 713–1018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570865900070058

[14] K. J. Marfurt, "Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations," *GEOPHYSICS*, vol. 49, no. 5, pp. 533–549, 1984. [Online]. Available: https://doi.org/10.1190/1.1441689

[15] T. E. o. E. Britannica, *Seismic wave*. Encyclopedia Britannica., 2023. [Online]. Available: https://www.britannica.com/science/seismic-wave

[16] A. Ben-Menahem and S. Singh, *Seismic Waves and Sources*. Springer New York, 2012. [Online]. Available: https://books.google.com.ec/books?id=L0PTBwAAQBAJ

[17] M. Norton and J. Pan, "Noise — noise radiated from elementary sources," in *Encyclopedia of Vibration*, S. Braun, Ed. Oxford: Elsevier, 2001, pp. 877–887. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B0122270851001454

[18] H. Farkas, K. Kály-Kullai, and S. Sieniutycz, "Chapter 17 - the fermat principle and chemical waves," in *Variational and Extremum Principles in Macroscopic Systems*, S. Sieniutycz and H. Farkas, Eds. Oxford: Elsevier, 2005, pp.

355–373. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080444888500205

[19] R. J. Schechter, "Snell's law: Optimum pathway analysis," *Survey of Ophthalmology*, vol. 21, no. 6, pp. 464–466, 1977. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0039625777800027

[20] A. Bamberger, G. Chavent, and P. Lailly, "Étude de schémas numériques de l'élastodynamique linéaire," 01 1980.

[21] R. Aster, "The seismic wave equation," *Seismological Society of America*, p. 18, 2011.

[22] S. Rienstra, "Hirschberg: An introduction to acoustics," *Eindhoven, Netherlands, Eindhoven University of Technology*, 2016.

[23] A. Yaroshevsky, "Abundances of chemical elements in the earth's crust," *Geochemistry International*, vol. 44, pp. 48–55, 2006.

[24] M. Frehner, S. M. Schmalholz, E. H. Saenger, and H. Steeb, "Comparison of finite difference and finite element methods for simulating two-dimensional scattering of elastic waves," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1, pp. 112–121, 2008, recent Advances in Computational Geodynamics: Theory, Numerics and Applications. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031920108001647

[25] D.-J. Min, C. Shin, R. G. Pratt, and H. S. Yoo, "Weighted-Averaging Finite-Element Method for 2D Elastic Wave Equations in the Frequency Domain," *Bulletin of the Seismological Society of America*, vol. 93, no. 2, pp. 904–921, 04 2003. [Online]. Available: https://doi.org/10.1785/0120020066

[26] H. Zhou, Y. Liu, and J. Wang, "Acoustic finite-difference modeling beyond conventional courant-friedrichs-lewy stability limit: Approach based on variable-length temporal and spatial operators," *Earthquake Science*, vol. 34, no. 2, pp. 123–136, 2021.

Appendices

.1 Appendix 1.

In this section, we are showing the calculations to obtain the results for the scalar products in $L2$, which are shown in Table 2.1.

Let's consider the quadrilateral in Figure 1

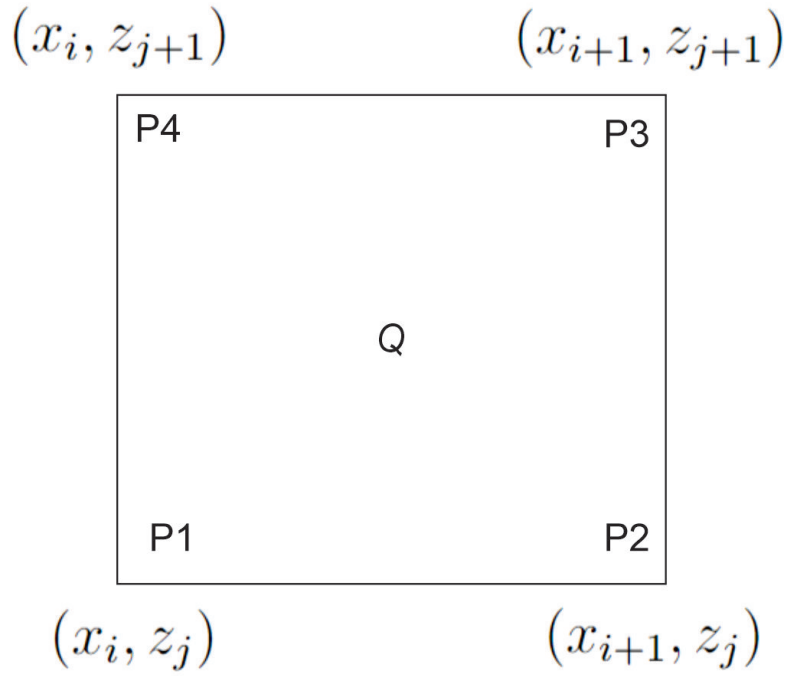

Figure 1: Reference Quadrilateral Q to express the basis functions P_1, P_2, P_3, and P_4 for the reference coordinate system.

The basis functions can be expressed as:

$$\begin{cases} P_1(x,z) = \dfrac{(x-x_{i+1})(z-z_{j+1})}{(x_i-x_{i+1})(z_j-z_{j+1})} = \dfrac{(x-x_{i+1})(z-z_{j+1})}{\Delta x \Delta z} \\ P_2(x,z) = \dfrac{(x-x_i)(z-z_{j+1})}{(x_{i+1}-x_i)(z_j-z_{j+1})} = \dfrac{(x-x_i)(z_{j+1}-z)}{\Delta x \Delta z} \\ P_3(x,z) = \dfrac{(x-x_i)(z-z_j)}{(x_{i+1}-x_i)(z_{j+1}-z_j)} = \dfrac{(x-x_i)(z-z_j)}{\Delta x \Delta z} \\ P_4(x,z) = \dfrac{(x-x_{i+1})(z-z_j)}{(x_i-x_{i+1})(z_{j+1}-z_j)} = \dfrac{(x_{i+1}-x)(z-z_j)}{\Delta x \Delta z} \end{cases} \tag{1}$$

Let's calculate the partial derivatives of the basis functions.

Partial derivative respect to x:

$$\begin{cases} \dfrac{\partial P_1(x,z)}{\partial x} = \dfrac{z - z_{j+1}}{\Delta x \Delta z} \\ \dfrac{\partial P_2(x,z)}{\partial x} = \dfrac{z_{j+1} - z}{\Delta x \Delta z} \\ \dfrac{\partial P_3(x,z)}{\partial x} = \dfrac{z - z_j}{\Delta x \Delta z} \\ \dfrac{\partial P_4(x,z)}{\partial x} = \dfrac{z_j - z}{\Delta x \Delta z} \end{cases} \tag{2}$$

Partial derivative respect to z:

$$\begin{cases} \dfrac{\partial P_1(x,z)}{\partial z} = \dfrac{x - x_{i+1}}{\Delta x \Delta z} \\ \dfrac{\partial P_2(x,z)}{\partial z} = \dfrac{x_i - x}{\Delta x \Delta z} \\ \dfrac{\partial P_3(x,z)}{\partial z} = \dfrac{x - x_i}{\Delta x \Delta z} \\ \dfrac{\partial P_4(x,z)}{\partial z} = \dfrac{x_{i+1} - x}{\Delta x \Delta z} \end{cases} \tag{3}$$

Since the function is explicitly expressed, let's calculate each scalar product shown in Table 2.1. The procedure is the same for all combinations, for some integrals it is going to be used antiderivatives, and for others integration by parts. This is going to be explained clearly in 1 and 2, respectively; therefore, the rest is to apply the same.

1. $\displaystyle\int_Q P_1 P_1 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})^2 (z - z_{j+1})^2 dx dz$

   Using the antiderivatives of each integral, we obtain:

   $$= \left[\frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3} \frac{1}{3} (x - x_{i+1})^3 (z - z_{j+1})^3 \right]_{x_i, z_j}^{x_{i+1}, z_{j+1}} = \frac{\Delta x^3 \Delta z^3}{9 \Delta x^2 \Delta z^2} = \boxed{\frac{\Delta x \Delta z}{9}}$$

2. $\displaystyle\int_Q P_1 P_2 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})(z - z_{j+1})(x - x_i)(z_{j+1} - z) dx dz$

   $$= \frac{1}{\Delta x^2 \Delta z^2} \int_{x_i}^{x_{i+1}} (x - x_{i+1})(x - x_i) dx \int_{z_j}^{z_{j+1}} (z - z_{j+1})(z_{j+1} - z) dz$$

   Using integration by parts$\int uv' = uv - \int vu'$ for the x variable and antiderivates for z variable, we obtain that:

$$= \frac{1}{\Delta x^2 \Delta z^2} \left[(x - x_{i+1}) \frac{1}{2} (x - x_i)^2 \Big|_{x_i}^{x_{i+1}} - \frac{1}{2} \int_{x_i}^{x_{i+1}} (x - x_i)^2 dx \right] \int_{z_j}^{z_{j+1}} - (z - z_{j+1})^2 dz$$

$$= \frac{1}{\Delta x^2 \Delta z^2} \left[\frac{1}{2} \left[\frac{1}{3} (x - x_i)^3 \right]_{x_i}^{x_{i+1}} \right] \frac{1}{3} (z - z_{j+1})^3 \Big|_{z_j}^{z_{j+1}} = \frac{\Delta x^3 \Delta z^3}{18 \Delta x^2 \Delta z^2} = \boxed{\frac{\Delta x \Delta z}{18}}$$

3. $\int_Q P_1 P_3 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})(z - z_{j+1})(x - x_i)(z - z_j) dx dz$

Then, using integration by parts for both variables, we obtain:

$$= \frac{1}{\Delta x^2 \Delta z^2} \left[-\frac{1}{2} \frac{1}{3} (x - x_i)^3 \right]_{x_i}^{x_{i+1}} \left[-\frac{1}{2} \frac{1}{3} (z - z_j)^3 \right]_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x \Delta z}{36}}$$

4. $\int_Q P_1 P_4 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})(z - z_{j+1})(x_{i+1} - x)(z - z_j) dx dz$

Using antiderivatives for x and integration by parts for Z:

$$= \frac{1}{\Delta x^2 \Delta z^2} \left(-\frac{1}{3} \right) (x - x_{i+1})^3 \Big|_{x_i}^{x_{i+1}} \left(-\frac{1}{2} \right) \frac{1}{3} (z - z_j)^3 \Big|_{z_j}^{z_{j+1}} = \frac{\Delta x^3 \Delta z^3}{18 \Delta x^2 \Delta z^2} = \boxed{\frac{\Delta x \Delta z}{18}}$$

5. $\int_Q P_2 P_2 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_i)^2 (z_{j+1} - z)^2 dx dz$

Using the antiderivatives:

$$= \frac{1}{\Delta x^2 \Delta z^2} \left[\frac{1}{3} (x - x_i)^3 \right]_{x_i}^{x_{i+1}} \left[\frac{1}{3} (z - z_{j+1})^3 \right]_{z_j}^{z_{j+1}} = \frac{\Delta x^3 \Delta z^3}{9 \Delta x^2 \Delta z^2} = \boxed{\frac{\Delta x \Delta z}{9}}$$

6. $\int_Q P_2 P_3 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_i)(z_{j+1} - z)(x - x_i)(z - z_j) dx dz$

Using antiderivative for x and integration by parts for z:

$$= \frac{1}{\Delta x^2 \Delta z^2} \left(-\frac{1}{3} \right) (x - x_i)^3 \Big|_{x_i}^{x_{i+1}} \left(-\frac{1}{2} \right) \left(\frac{1}{3} \right) (z - z_j)^3 \Big|_{z_j}^{z_{j+1}} = \frac{\Delta x^3 \Delta z^3}{18 \Delta x^2 \Delta z^2} = \boxed{\frac{\Delta x \Delta z}{18}}$$

7. $\int_Q P_2 P_4 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_i)(z_{j+1} - z)(x_{i+1} - x)(z - z_j) dx dz$

Using integration for both variables:

$$= \frac{1}{\Delta x^2 \Delta z^2}\left(\frac{1}{3}\right)\left(\frac{1}{3}\right)\left(\frac{1}{4}\right)(x - x_{i+1})^3\Big|_{x_i}^{x_{i+1}}(z - z_{j+1})^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x \Delta z}{36}}$$

8. $\int_Q P_3 P_3 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x - x_i)^2(z - z_j)^2 dx dz$

Using the antiderivatives for both variables:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{9}(x - x_i)^3\Big|_{x_i}^{x_{i+1}}(z - z_j)^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x \Delta z}{9}}$$

9. $\int_Q P_3 P_4 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x - x_i)(z - z_j)(x_{i+1} - x)(z - z_j) dx dz$

Using the integration by parts for x, and antiderivative for z:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{2}\frac{1}{3}\frac{1}{3}(x - x_{i+1})^3\Big|_{x_i}^{x_{i+1}}(z - z_j)^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x \Delta z}{18}}$$

10. $\int_Q P_3 P_4 dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_{i+1} - x)^2(z - z_j)^2 dx dz$

Using antiderivatives for both variables:

$$\frac{1}{\Delta x^2 \Delta z^2}\frac{1}{9}(x - x_{i+1})^3\Big|_{x_i}^{x_{i+1}}(z - z_j)^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x \Delta z}{9}}$$

11. $\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_1}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})^2 dx dz$

Using the antiderivatives for both variables, we get:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{3}(z - z_{j+1})^3\Big|_{z_j}^{z_{j+1}} x\Big|_{x_i}^{x_{i+1}} = \frac{\Delta x \Delta z^3}{3\Delta x^2 \Delta z^2} = \boxed{\frac{\Delta z}{3\Delta x}}$$

12. $\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_1}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})(x - x_{i+1}) dx dz$

Using the antiderivatives:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_{j+1})^2 \Big|_{z_j}^{z_{j+1}} (x - x_{i+1})^2 \Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

13. $$\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_2}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})(z_{j+1} - z) dx dz$$

Using the antiderivatives for both variables, we get:

$$= -\frac{1}{\Delta x^2 \Delta z^2}\frac{1}{3}(z - z_{j+1})^3 \Big|_{z_j}^{z_{j+1}} x \Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{\Delta z}{3\Delta x}}$$

14. $$\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_2}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})(x_i - x) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2}(z - z_{j+1})^2 \Big|_{z_j}^{z_{j+1}} (x - x_i)^2 \Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

15. $$\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_3}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})(z - z_j) dx dz$$

Applying the integration by parts:

$$= -\frac{1}{\Delta x^2 \Delta z^2}\frac{1}{2}\frac{1}{3}(z - z_j)^3 \Big|_{z_j}^{z_{j+1}} x \Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{\Delta z}{6\Delta x}}$$

16. $$\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_3}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})(x - x_i) dx dz$$

Using the antiderivative:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_{j+1})^2 \Big|_{z_j}^{z_{j+1}} (x - x_i)^2 \Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

17. $$\int_Q \frac{\partial P_1}{\partial x}\frac{\partial P_4}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_{j+1})(z_j - z) dx dz$$

Applying the integration by parts:

$$= \frac{1}{\Delta x^2 \Delta z^2} \frac{1}{2} \frac{1}{3} (z - z_j)^3 \Big|_{z_j}^{z_{j+1}} x \Big|_{x_i}^{x_{i+1}} = \boxed{\frac{\Delta z}{6\Delta x}}$$

18. $$\int_Q \frac{\partial P_1}{\partial x} \frac{\partial P_4}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (z - z_{j+1})(x_{i+1} - x) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2} (z - z_{j+1})^2 \Big|_{z_j}^{z_{j+1}} (x - x_{i+1})^2 \Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

19. $$\int_Q \frac{\partial P_1}{\partial z} \frac{\partial P_1}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})^2 dx dz$$

Using the antiderivatives:

$$= \frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3} (x - x_{i+1})^3 \Big|_{x_i}^{x_{i+1}} z \Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x}{3\Delta z}}$$

20. $$\int_Q \frac{\partial P_1}{\partial z} \frac{\partial P_2}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})(z_{j+1} - z) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2} (z - z_{j+1})^2 \Big|_{z_j}^{z_{j+1}} (x - x_{i+1})^2 \Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

21. $$\int_Q \frac{\partial P_1}{\partial z} \frac{\partial P_2}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})(x_i - x) dx dz$$

Applying the integration by parts:

$$= \frac{1}{\Delta x^2 \Delta z^2} \frac{1}{6} (x - x_i)^3 \Big|_{x_i}^{x_{i+1}} z \Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x}{6\Delta z}}$$

22. $$\int_Q \frac{\partial P_1}{\partial z} \frac{\partial P_3}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_{i+1})(z - z_j) dx dz$$

Using the antiderivatives:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_j)^2\Big|_{z_j}^{z_{j+1}}(x - x_{i+1})^2\Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

23. $$\int_Q \frac{\partial P_1}{\partial z}\frac{\partial P_3}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x - x_{i+1})(z - z_j)dxdz$$

Applying the antiderivatives:

$$= -\frac{1}{\Delta x^2 \Delta z^2}\frac{1}{6}(x - x_i)^3\Big|_{x_i}^{x_{i+1}} z\Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta x}{6\Delta z}}$$

24. $$\int_Q \frac{\partial P_1}{\partial z}\frac{\partial P_4}{\partial x}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x - x_{i+1})(z_j - z)dxdz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2}(z - z_j)^2\Big|_{z_j}^{z_{j+1}}(x - x_{i+1})^2\Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

25. $$\int_Q \frac{\partial P_1}{\partial z}\frac{\partial P_4}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x - x_{i+1})(x_{i+1} - x)dxdz$$

Using the antiderivatives:

$$= -\frac{1}{\Delta x^2 \Delta z^2}\frac{1}{3}(x - x_{i+1})^3\Big|_{x_i}^{x_{i+1}} z\Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta x}{3\Delta z}}$$

26. $$\int_Q \frac{\partial P_2}{\partial x}\frac{\partial P_2}{\partial x}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_{j+1} - z)^2 dxdz$$

Using the antiderivatives:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{3}x\Big|_{x_i}^{x_{i+1}}(z - z_{j+1})^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta z}{3\Delta x}}$$

27. $$\int_Q \frac{\partial P_2}{\partial x}\frac{\partial P_2}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_{j+1} - z)(x_i - x)dxdz$$

Using the antiderivatives:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_{j+1})^2\Big|_{z_j}^{z_{j+1}}(x - x_i)^2\Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

28. $$\int_Q \frac{\partial P_2}{\partial x}\frac{\partial P_3}{\partial x}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_{j+1} - z)(z - z_j)dxdz$$

Applying integration by parts:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{6}x\Big|_{x_i}^{x_{i+1}}(z - z_j)^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta z}{6\Delta x}}$$

29. $$\int_Q \frac{\partial P_2}{\partial x}\frac{\partial P_3}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_{j+1} - z)(x - x_i)dxdz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2}(z - z_{j+1})^2\Big|_{z_j}^{z_{j+1}}(x - x_i)^2\Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

30. $$\int_Q \frac{\partial P_2}{\partial x}\frac{\partial P_4}{\partial x}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_{j+1} - z)(z_j - z)dxdz$$

Applying integration by parts:

$$= -\frac{1}{\Delta x^2 \Delta z^2}\frac{1}{6}x\Big|_{x_i}^{x_{i+1}}(z - z_j)^3\Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta z}{6\Delta x}}$$

31. $$\int_Q \frac{\partial P_2}{\partial x}\frac{\partial P_4}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_{j+1} - z)(x_{i+1} - x)dxdz$$

Using the antiderivatives:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_{j+1})^2\Big|_{z_j}^{z_{j+1}}(x - x_{i+1})^2\Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

32. $$\int_Q \frac{\partial P_2}{\partial z}\frac{\partial P_2}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_i - x)dxdz$$

Using the antiderivatives:

$$= \frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3}(x - x_i)^3 \Big|_{x_i}^{x_{i+1}} z \Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta x}{3\Delta z}}$$

33. $$\int_Q \frac{\partial P_2}{\partial z} \frac{\partial P_3}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_i - x)(z - z_j) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2}(z - z_j)^2 \Big|_{z_j}^{z_{j+1}} (x - x_i)^2 \Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

34. $$\int_Q \frac{\partial P_2}{\partial z} \frac{\partial P_3}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_i - x)(x - x_i) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3}(x - x_i)^3 \Big|_{x_i}^{x_{i+1}} z \Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta x}{3\Delta z}}$$

35. $$\int_Q \frac{\partial P_2}{\partial z} \frac{\partial P_4}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_i - x)(z_j - z) dx dz$$

Using the antiderivatives:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_j)^2 \Big|_{z_j}^{z_{j+1}} (x - x_i)^2 \Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

36. $$\int_Q \frac{\partial P_2}{\partial z} \frac{\partial P_4}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_i - x)(x_{i+1} - x) dx dz$$

Applying integration by parts:

$$= -\frac{1}{\Delta x^2 \Delta z^2} \frac{1}{6}(x - x_i)^3 \Big|_{x_i}^{x_{i+1}} z \Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta x}{6\Delta z}}$$

37. $$\int_Q \frac{\partial P_3}{\partial x} \frac{\partial P_3}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z - z_j)^2 dx dz$$

Using the antiderivatives:

$$= \frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3} x \Big|_{x_i}^{x_{i+1}} (z - z_j)^3 \Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta z}{3\Delta x}}$$

38. $$\int_Q \frac{\partial P_3}{\partial x} \frac{\partial P_3}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (z - z_j)(x - x_i) dx dz$$

Using the antiderivatives:

$$= \frac{1}{4\Delta x^2 \Delta z^2} (z - z_j)^2 \Big|_{z_j}^{z_{j+1}} (x - x_i)^2 \Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

39. $$\int_Q \frac{\partial P_3}{\partial x} \frac{\partial P_4}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (z - z_j)(z_j - z) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3} x \Big|_{x_i}^{x_{i+1}} (z - z_j)^3 \Big|_{z_j}^{z_{j+1}} = \boxed{-\frac{\Delta z}{3\Delta x}}$$

40. $$\int_Q \frac{\partial P_3}{\partial x} \frac{\partial P_4}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (z - z_j)(x_{i+1} - x) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2} (z - z_j)^2 \Big|_{z_j}^{z_{j+1}} (x - x_{i+1})^2 \Big|_{x_i}^{x_{i+1}} = \boxed{\frac{1}{4}}$$

41. $$\int_Q \frac{\partial P_3}{\partial z} \frac{\partial P_3}{\partial z} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_i)^2 dx dz$$

Using the antiderivatives:

$$= \frac{1}{\Delta x^2 \Delta z^2} \frac{1}{3} (x - x_i)^3 \Big|_{x_i}^{x_{i+1}} z \Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x}{3\Delta z}}$$

42. $$\int_Q \frac{\partial P_3}{\partial z} \frac{\partial P_4}{\partial x} dQ = \int_{x_i}^{x_{i+1}} \int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2} (x - x_i)(z_j - z) dx dz$$

Using the antiderivatives:

$$= -\frac{1}{4\Delta x^2 \Delta z^2}(z - z_j)^2\Big|_{z_j}^{z_{j+1}}(x - x_i)^2\Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

43. $$\int_Q \frac{\partial P_3}{\partial z}\frac{\partial P_4}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x - x_i)(x_{i+1} - x)dxdz$$

Applying the integration by parts:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{6}(x - x_i)^3\Big|_{x_i}^{x_{i+1}} z\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x}{6\Delta z}}$$

44. $$\int_Q \frac{\partial P_4}{\partial x}\frac{\partial P_4}{\partial x}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_j - z)^2 dxdz$$

Applying the antiderivative:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{3}x\Big|_{x_i}^{x_{i+1}}(z - z_j)^3\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta z}{3\Delta x}}$$

45. $$\int_Q \frac{\partial P_4}{\partial x}\frac{\partial P_4}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(z_j - z)(x_{i+1} - x)dxdz$$

Applying the antiderivative:

$$= \frac{1}{4\Delta x^2 \Delta z^2}(z - z_j)^2\Big|_{z_j}^{z_{j+1}}(x - x_{i+1})^2\Big|_{x_i}^{x_{i+1}} = \boxed{-\frac{1}{4}}$$

46. $$\int_Q \frac{\partial P_4}{\partial z}\frac{\partial P_4}{\partial z}dQ = \int_{x_i}^{x_{i+1}}\int_{z_j}^{z_{j+1}} \frac{1}{\Delta x^2 \Delta z^2}(x_{i+1} - x)^2 dxdz$$

Applying the antiderivative:

$$= \frac{1}{\Delta x^2 \Delta z^2}\frac{1}{3}(x - x_{i+1})\Big|_{x_i}^{x_{i+1}} z\Big|_{z_j}^{z_{j+1}} = \boxed{\frac{\Delta x}{3\Delta z}}$$

Since the products are commutative, the other products are equal to the already calculated ones.

.2 Appendix 2.

The code to create the matrices and implement the eq. 3.22, is presented:

Import libraries

import numpy as np

import matplotlib.pyplot as plt

from scipy import signal

from tqdm import tqdm

get_ipython().run_line_magic('matplotlib', 'notebook')

Show Plot in The Notebook

import matplotlib

matplotlib.use("nbagg")

Sub-plot Configuration

from matplotlib import gridspec

from mpl_toolkits.axes_grid1 import make_axes_locatable

Ignore Warning Messages

import warnings

warnings.filterwarnings("ignore")

Set the grid

Grid points

nx # number of grid points in the x direction

nz # number of grid points in the x direction

I = nx-2 # number of grid points in x-direction -2 (since the first and last values are 0)

J = nz-2 # number of grid points in z-direction -2

nt # maximum number of time steps

Increments

dt = 0.00060 # Time increment

xmax # Maximum length

hx = xmax/(nx-2) # calculate space increment

x = np.arange(0, nx)*hx # initialize space coordinates

x = np.transpose(x)

dx = np.diff(x)[0] # Element sizes [m]

zmax

hz = xmax/(nz-2) # calculate space increment

z = np.arange(0, nz)*hz # initialize space coordinates

z = np.transpose(z)

dz = np.diff(x)[0]

Position configuration

xi = 10 # coordinates of the element where the src is initialized

xz = 25 # i.e position i,j of the element

idisp = 10 # display frequency

isx = (xi-1)*J+xz-1 # source location in grid in the x-direction (k value)

isz = (xi-1)*J+xz-1 # source location in grid in the z-direction (k value)

Mass and Stiffness matrices and displacement vectors

Setting lame parameters

rho1 # Density kg/m3

vs1 # s-velocity m/s

vp1 # s-velocity m/s

mu1 = rho1 * (vs1 * vs1) # mu

lam1 = (vp1*vp1)*(rho1)-2*mu1 # lambda

Size of the matrices

KI = I*J

KJ = I*J

Create the matrices and fill them with 0

M = np.zeros((KI,KJ), dtype=float) # Mass matrix

Kxx = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the x,x partial derivatives

Kzz = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the z,z partial derivatives

```
Kxz = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the x,z partial derivatives
# Fill the matrices
for k in range(KI):
    # Filling the values for diagonal
    M[k,k] = 4
    Kxx[k,k] = 4
    Kzz[k,k] = 4
    Kxz[k,k] = 0
    for i in range(J-1):
        if k == J*i: # fill the iterations of the left column of the grid
            # Mass Matrix
            M[k,k+1] = 1 #Right
            M[k,k+J] = 1 #Down
            M[k,k+J+1] = 1/4 #Dri
            # Rxx Matrix
            Kxx[k,k+1] = -2 #Right
            Kxx[k,k+J] = 1 #Down
            Kxx[k,k+J+1] = -1/2 #Dri
            # Rzz Matrix
            Kzz[k,k+1] = 1 #Right
            Kzz[k,k+J] = -2 #Down
            Kzz[k,k+J+1] = -1/2 #Dri
            # Rxz Matrix
            Kxz[k,k+1] = 0 #Right
            Kxz[k,k+J] = 0 #Down
            Kxz[k,k+J+1] = 1 #Dri
        if k == J*(i+1)-1: # fill the iterations of the right column of the grid
            # Mass Matrix
            M[k,k+J] = 1 #Down
            M[k,k+J-1] = 1/4 #Dle
            # Rxx Matrix
```

```
        Kxx[k,k+J] = 1 #Down
        Kxx[k,k+J-1] = -1/2 #Dle
        # Rzz Matrix
        Kzz[k,k+J] = -2 #Down
        Kzz[k,k+J-1] = -1/2 #Dle
        # Rxz Matrix
        Kxz[k,k+J] = 0 #Down
        Kxz[k,k+J-1] = -1 #Dle
    if k == (I-1)*(J)+i: # fill the iterations of the last row of the grid
        # Mass Matrix
        M[k,k+1] = 1 #Right
        # Rxx Matrix
        Kxx[k,k+1] = -2 #Right
        # Rzz Matrix
        Kzz[k,k+1] = 1 #Right
        # Rxz Matrix
        Kxz[k,k+1] = 0 #Right
    for j in range(J-2): # fill the iterations of the other elements
        if k == J*i + (j+1):
            # Mass matrix
            M[k,k+1] = 1 #Right
            M[k,k+J] = 1 #Down
            M[k,k+J+1] = 1/4 #Dri
            M[k,k+J-1] = 1/4 #Dle
            # Rxx matrix
            Kxx[k,k+1] = -2 #Right
            Kxx[k,k+J] = 1 #Down
            Kxx[k,k+J+1] = -1/2 #Dri
            Kxx[k,k+J-1] = -1/2 #Dle
            # Rzz matrix
            Kzz[k,k+1] = 1 #Right
```

```
            Kzz[k,k+J] = -2 #Down
            Kzz[k,k+J+1] = -1/2 #Dri
            Kzz[k,k+J-1] = -1/2 #Dle
            # Rxz matrix
            Kxz[k,k+1] = 0 #Right
            Kxz[k,k+J] = 0 #Down
            Kxz[k,k+J+1] = 1 #Dri
            Kxz[k,k+J-1] = -1 #Dle
# The previous filling was for the upper diagonal of the matrices, this step is to fill the bottom diagonal since the matrices are symmetric; then, is just necessary to fill the bottom with the upper values.
for i in range(KI):
    for j in range(KJ):
        M[j,i] = M[i,j]
        Kxx[j,i] = Kxx[i,j]
        Kzz[j,i] = Kzz[i,j]
        Kxz[j,i] = Kxz[i,j]
# Multiply the matrices with the constants
M = rho1*((dx*dz)/9)*M
Kxx = (dz/(3*dx))*Kxx
Kzz = (dx/(3*dz))*Kzz
Kxz = (1/4)*Kxz
# Let's assembly the following system
#(M 0)(u") + ((lam+2mu)*Kxx + mu*Kzz = K1u lam*Kzz + mu*Kxz = K1w)(u) = src
#(0 M)(w") + (lam*Kxz + mu*Kxx = K2u (lam+2mu)*Kzz + mu*Kxx = K2w)(w) = src
# Let's create the matrices K1u, K1w, K2u, and K2w
K1u = (lam1+2*mu1)*Kxx + mu1*Kzz
K1w = lam1*Kzz + mu1*Kxz
K2u = lam1*Kxz + mu1*Kxx
K2w = (lam1+2*mu1)*Kzz + mu1*Kxx
```

```
# For visualization and simplicity
#(M 0)(u") + (K1u K1w)(u) = src
#(0 M)(w") + (K2u K2w)(w) = src
Minv = np.linalg.inv(M)
# Create the global mass and stiffness matrices
Mg = np.zeros((KI*2,KJ*2))
Mginv = np.zeros((KI*2,KJ*2))
Kg = np.zeros((KI*2,KJ*2))
# Fill the global mass and stiffness matrices
for i in range(len(Mg)):
    for j in range(len(Mg)):
        if (i < (len(Mg)/2)) and (j < (len(Mg)/2)):
            Mg[i,j] = M[i,j]
            Mginv[i,j] = Minv[i,j]
            Kg[i,j] = K1u[i,j]
        if (i < (len(Mg)/2)) and (j >= (len(Mg)/2)):
            Kg[i,j] = K1w[i,j-len(M)]
        if (i >= (len(Mg)/2)) and (j < (len(Mg)/2)):
            Kg[i,j] = K2u[i-len(M),j]
        if (i >= (len(Mg)/2)) and (j >= (len(Mg)/2)):
            Mg[i,j] = M[i-len(M),j-len(M)]
            Mginv[i,j] = Minv[i-len(M),j-len(M)]
            Kg[i,j] = K2w[i-len(M),j-len(M)]
# Create the displacement vectors U and W
U = np.zeros((nx,nz), dtype=float)
W = np.zeros((nx,nz), dtype=float)
# Create the global displacement vector theta = (u,w)
theta = np.zeros(I*J*2)
thetaold = np.zeros(I*J*2)
thetanew = np.zeros(I*J*2)
```

Source function

```
# Plot Source Time Function
Ts=0.5
time = np.arange(0,(Ts-dt),dt) # Time vector
# Source signal - Ricker-wavelet
f0= 25 # Center frequency Ricker-wavelet
q0= 1 # Maximum amplitude Ricker-Wavelet
t0= 1.5/f0
tau=np.pi*f0*(time-t0);
src=q0*(1-2*tau**2)*np.exp(-tau**2);
fx = np.zeros(I*J); fx[isx:isx+1] = fx[isx:isx+1] + 1.
fz = np.zeros(I*J); fz[isz:isz+1] = fz[isz:isz+1] + 1.
# Create the global position vector for the source
F = np.zeros(I*J*2)
for i in range(len(F)):
    if i < len(F)/2:
        F[i] = fx[i]
    if i >= len(F)/2:
        F[i] = fz[i-int(len(F)/2)]
# Plot Position Configuration
plt.ion()
fig1 = plt.figure(figsize=(12, 6))
gs1 = gridspec.GridSpec(1, 2, width_ratios=[1, 1], hspace=0.3, wspace=0.3)
# Plot Source Time Function
ax1 = plt.subplot(gs1[0])
ax1.plot(time, src) # plot source time function
ax1.set_title('Source Time Function')
ax1.set_xlim(time[0], time[-1])
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Amplitude')
```

```
# Plot Source Spectrum
ax2 = plt.subplot(gs1[1])
spec = np.fft.fft(src) # source time function in the frequency domain
freq = np.fft.fftfreq(spec.size, d = dt / 4.) # time domain to frequency domain
ax2.plot(np.abs(freq), np.abs(spec)) # plot frequency and amplitude
ax2.set_xlim(0, 250) # only display frequency from 0 to 250 Hz
ax2.set_title('Source Spectrum')
ax2.set_xlabel('Frequency (Hz)')
ax2.yaxis.tick_right()
ax2.yaxis.set_label_position("right")
plt.show()
```

Plot the simulation

```
plt.ion()
fig1 = plt.figure(figsize=(9,9))
f1 = fig1.add_subplot(2, 2, 1)
f2 = fig1.add_subplot(2, 2, 2)
f1.set_title('X')
f2.set_title('Z')
xp = f1.imshow(U, interpolation='nearest', animated=True, vmin=-2.5e-11, vmax=2.5e-
11, cmap=plt.cm.RdBu,aspect = 'equal')
zp = f2.imshow(W, interpolation='nearest', animated=True, vmin=-2.5e-11, vmax=2.5e-
11, cmap=plt.cm.RdBu,aspect = 'equal')
plt.show()
```

Data actualization

```
# Calculating the displacement vector at each time step
for it in range(nt):
    thetanew = (dt**2) * Minv @ (F*src[it] - Kg @ theta) + 2*theta - thetaold
```

```
    thetaold, theta = theta, thetanew
    for i in range(1, nz - 1):
        for j in range(1, nx - 1):
            # Filling the displacement elements
            U[i,j] = theta[(i-1)*J + j - 1]
            W[i,j] = theta[(i-1)*J + j - 1 + int(len(F)/2)]
    # Plot every time step (nt)
    if (it % idisp) == 0:
        f1.set_title("Time Step (nt) = %i" % (it))
        xp.set_data(U)
        zp.set_data(W)
        plt.gcf().canvas.draw()
```

.3 Appendix 3.

Code to build matrices for 2 layered medium for 2D

```
    # Create and fill the Lamé parameter vector with values for each position
rho(nx*nz)
lam(nx*nz)
mu(nx*nz)
lams2mu(nx*nz)


    # Create the matrices and fill them with 0
M = np.zeros((KI,KJ), dtype=float) # Mass matrix
Kxx1 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the x,x partial derivatives
Kxx2 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the x,x partial derivatives
Kzz1 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the z,z partial derivatives
Kzz2 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the z,z partial derivatives
Kzz3 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the z,z partial derivatives
Kxz1 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the x,z partial derivatives
```

```
Kxz2 = np.zeros((KI,KJ), dtype=float) # stiffness matrix with the x,z partial derivatives


    for k in range(KI):
    for i in range(J-1):
        # Filling the values for diagonal
        M[k,k] = 1/9 * dx*dz * (rho[k] + rho[k+i] + rho[k+J+i] + rho[k+J+i+1] )
        Kxx1[k,k] = 1/3 * dz/dx * (lams2mu[k] + lams2mu[k+i] + lams2mu[k+J+i] +
lams2mu[k+J+i+1] )
        Kxx2[k,k] = 1/3 * dz/dx * (mu[k] + mu[k+i] + mu[k+J+i] + mu[k+J+i+1] )
        Kzz1[k,k] = 1/3 * dx/dz * (mu[k] + mu[k+i] + mu[k+J+i] + mu[k+J+i+1] )
        Kzz2[k,k] = 1/3 * dx/dz * (lam[k] + lam[k+i] + lam[k+J+i] + lam[k+J+i+1] )
        Kzz3[k,k] = 1/3 * dx/dz * (lams2mu[k] + lams2mu[k+i] + lams2mu[k+J+i] +
lams2mu[k+J+i+1] )
        Kxz1[k,k] = - 1/4 * mu[k] + 1/4 * mu[k+i] + 1/4 * mu[k+J+i] - 1/4 * mu[k+J+i+1]
        Kxz2[k,k] = - 1/4 * lam[k] + 1/4 * lam[k+i] + 1/4 * lam[k+J+i] - 1/4 * lam[k+J+i+1]
        if k == J*i: # fill the iterations of the left column of the grid
            # Mass Matrix
            M[k,k+1] = 1/18 * dx*dz * (rho[k+i] + rho[k+J+i+1]) #Right
            M[k,k+J] = 1/18 * dx*dz * (rho[k+J+i] + rho[k+J+i+1]) #Down
            M[k,k+J+1] = 1/36 * dx*dz * rho[k+J+i+1] #Dri
            # Rxx Matrix
            Kxx1[k,k+1] = -1/3 * dz/dx * (lams2mu[k+i] + lams2mu[k+J+i+1]) #Right
            Kxx1[k,k+J] = 1/6 * dz/dx * (lams2mu[k+J+i] + lams2mu[k+J+i+1]) #Down
            Kxx1[k,k+J+1] = - 1/3 * dz/dx * lams2mu[k+J+i+1] #Dri
            Kxx2[k,k+1] = -1/3 * dz/dx * (mu[k+i] + mu[k+J+i+1]) #Right
            Kxx2[k,k+J] = 1/6 * dz/dx * (mu[k+J+i] + mu[k+J+i+1]) #Down
            Kxx2[k,k+J+1] = - 1/3 * dz/dx * mu[k+J+i+1] #Dri
            # Rzz Matrix
            Kzz1[k,k+1] = 1/6 * dx/dz * (mu[k+i] + mu[k+J+i+1]) #Right
            Kzz1[k,k+J] = - 1/3 * dx/dz * (mu[k+J+i] + mu[k+J+i+1]) #Down
            Kzz1[k,k+J+1] = -1/6 * dx/dz * mu[k+J+i+1] #Dri
```

```
        Kzz2[k,k+1] = 1/6 * dx/dz * (lam[k+i] + lam[k+J+i+1]) #Right
        Kzz2[k,k+J] = - 1/3 * dx/dz * (lam[k+J+i] + lam[k+J+i+1]) #Down
        Kzz2[k,k+J+1] = -1/6 * dx/dz * lam[k+J+i+1] #Dri
        Kzz3[k,k+1] = 1/6 * dx/dz * (lamsmu[k+i] + lamsmu[k+J+i+1]) #Right
        Kzz3[k,k+J] = - 1/3 * dx/dz * (lamsmu[k+J+i] + lamsmu[k+J+i+1]) #Down
        Kzz3[k,k+J+1] = -1/6 * dx/dz * lamsmu[k+J+i+1] #Dri
        # Rxz Matrix
        Kxz1[k,k+1] = 1/4 * mu[k+i] - 1/4 * mu[k+J+i+1] #Right
        Kxz1[k,k+J] = - 1/4 * mu[k+J+i] + 1/4 * mu[k+J+i+1] #Down
        Kxz1[k,k+J+1] = 1/4 * mu[k+J+i+1] #Dri
        Kxz2[k,k+1] = 1/4 * lam[k+i] - 1/4 * lam[k+J+i+1] #Right
        Kxz2[k,k+J] = - 1/4 * lam[k+J+i] + 1/4 * lam[k+J+i+1] #Down
        Kxz2[k,k+J+1] = 1/4 * lam[k+J+i+1] #Dri
    if k == J*(i+1)-1: # fill the iterations of the right column of the grid
        # Mass Matrix
        M[k,k+J] = 1/18 * dx*dz * (rho[k+J+i] + rho[k+J+i+1]) #Down
        M[k,k+J-1] = 1/36 * dx*dz * rho[k+J+i] #Dle
        # Rxx Matrix
        Kxx1[k,k+J] = 1/6 * dz/dx * (lams2mu[k+J+i] + lams2mu[k+J+i+1]) #Down
        Kxx1[k,k+J-1] = - 1/6 * dz/dx * lams2mu[k+J+i] #Dle
        Kxx2[k,k+J] = 1/6 * dz/dx * (mu[k+J+i] + mu[k+J+i+1]) #Down
        Kxx2[k,k+J-1] = - 1/6 * dz/dx * mu[k+J+i] #Dle
        # Rzz Matrix
        Kzz1[k,k+J] = - 1/3 * dx/dz * (mu[k+J+i] + mu[k+J+i+1]) #Down
        Kzz1[k,k+J-1] = - 1/6 * dx/dz * mu[k+J+i] #Dle
        Kzz2[k,k+J] = - 1/3 * dx/dz * (lam[k+J+i] + lam[k+J+i+1]) #Down
        Kzz2[k,k+J-1] = - 1/6 * dx/dz * lam[k+J+i] #Dle
        Kzz3[k,k+J] = - 1/3 * dx/dz * (lamsmu[k+J+i] + lamsmu[k+J+i+1]) #Down
        Kzz3[k,k+J-1] = - 1/6 * dx/dz * lamsmu[k+J+i] #Dle
        # Rxz Matrix
        Kxz1[k,k+J] = - 1/4 * mu[k+J+i] + 1/4 * mu[k+J+i+1] #Down
```

```
        Kxz1[k,k+J-1] = - 1/4 * mu[k+J+i] #Dle
        Kxz2[k,k+J] = - 1/4 * lam[k+J+i] + 1/4 * lam[k+J+i+1] #Down
        Kxz2[k,k+J-1] = - 1/4 * lam[k+J+i] #Dle
    if k == (I-1)*(J)+i: # fill the iterations of the last row of the grid
        # Mass Matrix
        M[k,k+1] = 1/18 * dx*dz * (rho[k+i] + rho[k+J+i+1]) #Right
        # Rxx Matrix
        Kxx1[k,k+1] = -1/3 * dz/dx * (lams2mu[k+i] + lams2mu[k+J+i+1]) #Right
        Kxx2[k,k+1] = -1/3 * dz/dx * (mu[k+i] + mu[k+J+i+1]) #Right
        # Rzz Matrix
        Kzz1[k,k+1] = 1/6 * dx/dz * (mu[k+i] + mu[k+J+i+1]) #Right
        Kzz2[k,k+1] = 1/6 * dx/dz * (lam[k+i] + lam[k+J+i+1]) #Right
        Kzz3[k,k+1] = 1/6 * dx/dz * (lamsmu[k+i] + lamsmu[k+J+i+1]) #Right
        # Rxz Matrix
        Kxz1[k,k+1] = 1/4 * mu[k+i] - 1/4 * mu[k+J+i+1] #Right
        Kxz2[k,k+1] = 1/4 * lam[k+i] - 1/4 * lam[k+J+i+1] #Right
    for j in range(J-2): # fill the iterations of the other elements
        if k == J*i + (j+1):
            # Mass matrix
            M[k,k+1] = 1/18 * dx*dz * (rho[k+i] + rho[k+J+i+1]) #Right
            M[k,k+J] = 1/18 * dx*dz * (rho[k+J+i] + rho[k+J+i+1]) #Down
            M[k,k+J+1] = 1/36 * dx*dz * rho[k+J+i+1] #Dri
            M[k,k+J-1] = 1/36 * dx*dz * rho[k+J+i] #Dle
            # Rxx matrix
            Kxx1[k,k+1] = -1/3 * dz/dx * (lams2mu[k+i] + lams2mu[k+J+i+1]) #Right
            Kxx1[k,k+J] = 1/6 * dz/dx * (lams2mu[k+J+i] + lams2mu[k+J+i+1]) #Down
            Kxx1[k,k+J+1] = - 1/3 * dz/dx * lams2mu[k+J+i+1] #Dri
            Kxx1[k,k+J-1] = - 1/6 * dz/dx * lams2mu[k+J+i] #Dle
            Kxx2[k,k+1] = -1/3 * dz/dx * (mu[k+i] + mu[k+J+i+1]) #Right
            Kxx2[k,k+J] = 1/6 * dz/dx * (mu[k+J+i] + mu[k+J+i+1]) #Down
            Kxx2[k,k+J+1] = - 1/3 * dz/dx * mu[k+J+i+1] #Dri
```

Kxx2[k,k+J-1] = - 1/6 * dz/dx * mu[k+J+i] #Dle

Rzz matrix

Kzz1[k,k+1] = 1/6 * dx/dz * (mu[k+i] + mu[k+J+i+1]) #Right

Kzz1[k,k+J] = - 1/3 * dx/dz * (mu[k+J+i] + mu[k+J+i+1]) #Down

Kzz1[k,k+J+1] = -1/6 * dx/dz * mu[k+J+i+1] #Dri

Kzz1[k,k+J-1] = - 1/6 * dx/dz * mu[k+J+i] #Dle

Kzz2[k,k+1] = 1/6 * dx/dz * (lam[k+i] + lam[k+J+i+1]) #Right

Kzz2[k,k+J] = - 1/3 * dx/dz * (lam[k+J+i] + lam[k+J+i+1]) #Down

Kzz2[k,k+J+1] = -1/6 * dx/dz * lam[k+J+i+1] #Dri

Kzz2[k,k+J-1] = - 1/6 * dx/dz * lam[k+J+i] #Dle

Kzz3[k,k+1] = 1/6 * dx/dz * (lamsmu[k+i] + lamsmu[k+J+i+1]) #Right

Kzz3[k,k+J] = - 1/3 * dx/dz * (lamsmu[k+J+i] + lamsmu[k+J+i+1]) #Down

Kzz3[k,k+J+1] = -1/6 * dx/dz * lamsmu[k+J+i+1] #Dri

Kzz3[k,k+J-1] = - 1/6 * dx/dz * lamsmu[k+J+i] #Dle

Rxz matrix

Kxz1[k,k+1] = 1/4 * mu[k+i] - 1/4 * mu[k+J+i+1] #Right

Kxz1[k,k+J] = - 1/4 * mu[k+J+i] + 1/4 * mu[k+J+i+1] #Down

Kxz1[k,k+J+1] = 1/4 * mu[k+J+i+1] #Dri

Kxz1[k,k+J-1] = - 1/4 * mu[k+J+i] #Dle

Kxz2[k,k+1] = 1/4 * lam[k+i] - 1/4 * lam[k+J+i+1] #Right

Kxz2[k,k+J] = - 1/4 *lam[k+J+i] + 1/4 * lam[k+J+i+1] #Down

Kxz2[k,k+J+1] = 1/4 * lam[k+J+i+1] #Dri

Kxz2[k,k+J-1] = - 1/4 * lam[k+J+i] #Dle