3D finite-difference time-domain modeling of acoustic wave propagation based on domain decompostion

UMR Géosciences Azur
CNRS-IRD-UNSA-OCA
Villefranche-sur-mer

Supervised by: Dr. Stéphane Operto

Jade Rachele S. Garcia

July 21, 2009

Abstract

This report mainly focuses on three-dimensional acoustic wave propagation modeling in the time domain through finite difference method which comprises the forward modeling part of the 3D acoustic Full Waveform Inversion (FWI) method. An explicit, time-domain, finite-difference (FD) numerical scheme is used to solve the system for both pressure and particle velocity wavefields. Dependent variables are stored on staggered spatial and temporal grids, and centered FD operators possess 2nd-order and 4th-order time/space accuracy. The algorithm is designed to execute on parallel computational platforms by utilizing a classical spatial domain-decomposition strategy.

Contents

1 Introduction

1.1 Seismic Exploration

Seismology is commonly thought of as just the study of earthquakes, as the Greek etymology of the word may suggest. However, the scope of seismology extends beyond natural geological vibrations as it also involves the study of the propagation of elastic waves from controlled sources. One of the applications of this other part of seismology which is commonly utilized in the industrial world is seismic exploration − the search for subsurface deposits of crude oil, natural gas and minerals.

The main work in seismic exploration is to form a model of the subsurface. The basic idea governing seismic imaging is quite simple. As the Earth's crust is composed of different layers, each with its own properties, energy in the form of seismic waves traveling underground interacts differently with each of these layers. The seismic waves emitted by a source that artificially creates vibrations on the surface, will travel through the earth, but will also be reflected back towards the surface by the different underground layers. It is this reflection that allows for the use of seismic data processing to discover the properties of underground geology. The travel times (usually measured in milliseconds) of the returned seismic energy are recorded by a network of receivers and are then integrated with existing information. These aid geoscientists in estimating the structure and stratification of subsurface formations, and facilitate the location of prospective drilling targets.

1.2 Full Waveform Inversion

One of the various seismic data processing methods is Full Waveform Inversion. What makes Full Waveform Inversion a challenging yet worthwhile endeavor is that it aims to utilize the entire information from the collected seismic data to produce a high resolution model of the subsurface. As a quantitative data fitting procedure, it is composed of two main ingredients − the forward problem and the inverse problem − both of which involve numerical methods[1].

The forward problem consists of numerically computing for the solution of the equations of motion that model seismic wave propagation for an *a priori* model. There are several numerical methods used for modeling seismic wave propagation, such as finite element methods, finite difference methods and finite volume methods, and two domains, the time domain and the frequency domain, where the wave equation can be solved.

Using the matrix notations to denote the partial differential operators of the wave equation, the wave equation in the time domain is generally expressed as:

$$\mathbf{M}(\mathbf{x})\frac{d^2\mathbf{u}(\mathbf{x},t)}{dt^2} = \mathbf{A}(\mathbf{x})\mathbf{u}(\mathbf{x},t) + \mathbf{s}(\mathbf{x},t) \tag{1}$$

where $\mathbf{M}$ is the mass matrix and $\mathbf{A}$ is the stiffness matrix, $\mathbf{x}$ is the spacial coordinate, t is the time, $\mathbf{u}$ is the seismic wavefield and $\mathbf{s}$ is the source term.

In the frequency domain, the wave equation is transformed into a system of linear equations whose right hand side is the source term and whose solution is the seismic wavefield. This system is compactly expressed as

$$\mathbf{B}(\mathbf{x},\omega)\mathbf{u}(\mathbf{x},\omega) = \mathbf{s}(\mathbf{x},\omega) \tag{2}$$

where $\mathbf{B}$ is the impedance matrix and ω is the frequency term.

In full waveform inversion, it is necessary to perform simulations for a large number of sources. This need leads to the tendency of frequency domain modeling to be a better choice since this simply translates into changing the $\mathbf{s}$ term in (2) although some compressive sensing techniques based on phase encoding are currently being developed to mitigate this computational burden.

The properties of the subsurface to be quantified are embedded into the coefficients of the matrices **M, A** or **B**. The relationship between the data and the parameters is nonlinear. This relationship can be compactly expressed by defining a nonlinear operator $\mathbf{G}$ such that

$$\mathbf{u} = \mathbf{G}(\mathbf{m}) \tag{3}$$

either in the time domain or in the frequency domain.

On the other hand, the inverse problem consists of finding the model parameters which describes the physical properties of the subsurface that will result to a good match between the model and the recorded data. Waveform fitting is the underlying mechanism for this process of physical parameter reconstruction. This problem is often recast as a local optimization problem that aims to decrease the misfit between the model and the recorded data or seismograms.

The misfit vector $\Delta\mathbf{d} = \mathbf{d}_{obs} - \mathbf{d}_{cal}(\mathbf{m})$ is defined as the difference at the receiver positions between the recorded seismic data $\mathbf{d}_{obs}$ and the modeled one $\mathbf{d}_{cal}(\mathbf{m})$, for each source-receiver pair of the seismic survey. The model $\mathbf{m}$ characterizes, through some physical parameters, the subsurface which is discretized over the computational domain. $\mathcal{C}(\mathbf{m})$ is the norm of this misfit vector referred to as the misfit function or objective function.

The FWI is essentially a local optimisation therefore the minimum of the misfit function $\mathcal{C}(\mathbf{m})$ is searched in the vicinity of the starting model $\mathbf{m}_0$. Providing an accurate initial model is necessary for the fitting of the waveform to converge to the velocity structure.

2 Scope and Aim of the Work

The focus of this paper is on the forward problem part of the Full Waveform Inversion method.

A representative example in three dimensions which shows the order of magnitude of problems common in seismic exploration is the following example of hydrocarbon exploration whose specifications are the following:

Target (oil and gas exploration): 20 km x 20 km x 8 km

Maximum frequency: 20 Hz
Minimum P and S wave velocities: 1.5 km/s and 0.7 km/s

In this paper, an acoustic approximation using finite difference discretization is implemented for reasons to be elaborated in the succeeding sections.

For this example, the finite difference grid spacing is usually taken to be h=1500/20/4 18 m (refer to the formula in (**3.3.3**)). Using this grid spacing, the dimensions of the finite difference grid for the above example is 1110 x 1110 x 445 grid points.

The above example shows how massive the scale of the problems being dealt with in seismic exploration is. Building a model by full waveform inversion requires at least to solve three times per inversion iteration the forward problem (seismic wave modeling) for each source of the acquisition survey. This is a clear motivation for the main aim of this paper which is to implement and validate the parallel implementation of the 3D finite-difference time-domain code for acoustic wave modeling developed by Florent Sourbier of Géosciences Azur.

Although it has been mentioned that frequency domain modeling is preferable over its counterpart, this method becomes too memory demanding for the three dimensional case as the complexity of the matrix explodes. On the other hand, the time domain method is simple, relatively computationally efficient, robust and easier to implement even in the three dimensional case. A strategy that balances the pros and cons of these two methods is to perform 3D acoustic full-waveform inversion in the frequency domain by building a forward modeling engine in the time domain wherein the monochromatic wavefield are generated in the time loop by discrete Fourier transform. This also motivates part of the main aim of this paper, specifically the implementation in the time domain.

In the acoustic case, the earth is considered as a fluid for simplicity's sake. However, a more realistic approximation uses the elastodynamic equation instead of the acoustic one. Therefore, an aim of this paper is to design an acoustic code with judicious stencil that will be easily extended to the 3D elastic case.

The 3D elastic code will then be used as a forward modeling engine to perform the elastic full-waveform inversion in three dimensions. Currently, V. Etienne of Geosciences Azur has developed a forward modeling engine using the Discontinuous Galerkin finite element method. Developing a 3D elastic code using finite difference method can then be cross-validated with that aforementioned method. The elastic code can also be used to model other kinds of application such as seismic hazards assessment.

3 The Forward Problem: Seismic Wave Propagation Modeling

3.1 The Acoustic Wave Equation: The Earth as a Fluid

The acoustic wave equation describes sound waves in a liquid or gas. Since fluids exhibit fewer restraints to deformation, the restoring force responsible for wave propagation is simply due to a pressure change. Therefore, an acoustic wave is essentially a pressure change. A local pressure change causes immediate fluid to compress which in turn causes additional pressure changes. This leads to the propagation of an acoustic wave.

Let

$$\begin{aligned}\rho &= \text{mass per unit volume of the fluid}\\ V_x &= \text{velocity flow of fluid in the } x\text{-direction}\\ V_y &= \text{velocity flow of fluid in the } y\text{-direction}\\ V_z &= \text{velocity flow of fluid in the } z\text{-direction}\end{aligned}$$

Newton's law of momentum conservation says that a small volume within a fluid will accelerate if there is an applied force. This force may be external or internal; the internal one arises from pressure differences at opposite sides of the small volume. Newton's law states that

$$\text{mass} \;\times\; \text{acceleration} \;=\; \text{force} \;=\; \text{pressure gradient}(\cdot\text{volume})$$

$$\rho\frac{\partial V_x}{\partial t} = \frac{\partial P}{\partial x} + \mathrm{f}_x \tag{4}$$

$$\rho\frac{\partial V_y}{\partial t} = \frac{\partial P}{\partial y} + \mathrm{f}_y \tag{5}$$

$$\rho\frac{\partial V_z}{\partial t} = \frac{\partial P}{\partial z} + \mathrm{f}_z \tag{6}$$

where f_x, f_y and f_z are forces per unit volume.

The second physical process is energy storage by compression and volume change. If the velocity vector V_x at $x + \Delta x$ exceeds that at x, then the flow is said to be diverging. In other words, the small volume between x and $x + \Delta x$ is expanding. This expansion must lead to a pressure drop. The amount of the pressure drop is in proportion to a property of the fluid called its incompressibility κ.

$$\text{pressure drop} \;=\; (\text{incompressibility}) \;\times\; (\text{divergence of velocity})$$

In three dimensions, it is

$$\frac{\partial P}{\partial t} = \kappa\left(\frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}\right) \tag{7}$$

To arrive at the second-order wave equation from the equations above, divide the first set of equations by ρ and take the x-derivative, y-derivative and z-derivative of the respective equations. Let $1/\rho = b$, for buoyancy which is the inverse of density.

$$\frac{\partial V_x}{\partial t} = b\frac{\partial P}{\partial x} + f_x \tag{8}$$
$$\frac{\partial V_y}{\partial t} = b\frac{\partial P}{\partial y} + f_y \tag{9}$$
$$\frac{\partial V_z}{\partial t} = b\frac{\partial P}{\partial z} + f_z \tag{10}$$

where f_x,f_y,f_z are the accelarations due to the external forces in the respective directions.

Second, differentiate (7) with respect to time. Fortunately, in the solid-earth sciences, the medium does not change during experiments, meaning the incompressibility κ is a constant function of time:

$$\frac{\partial^2 P}{\partial t^2} = \kappa\left(\frac{\partial}{\partial t}\frac{\partial}{\partial x}V_x + \frac{\partial}{\partial t}\frac{\partial}{\partial y}V_y + \frac{\partial}{\partial t}\frac{\partial}{\partial z}V_z\right) \tag{11}$$

Inserting (8),(9) and (10) into (11), the three-dimensional scalar wave equation appears.

$$\frac{\partial^2 P}{\partial t^2} = \kappa \cdot \left[b \cdot \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) P + \nabla \cdot f\right] \tag{12}$$

However, for the numerical implementation in this report, the velocity-stress formulation with added source terms f_x , f_y and f_z is preferred:

$$\begin{array}{l} \frac{\partial P}{t} = \kappa\left(\frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z}\right) \\ \frac{\partial V_x}{\partial t} = b\frac{\partial P}{\partial x} + f_x \\ \frac{\partial V_y}{\partial t} = b\frac{\partial P}{\partial y} + f_y \\ \frac{\partial V_z}{\partial t} = b\frac{\partial P}{\partial z} + f_z \end{array} \tag{13}$$

where P is the pressure, V_x,V_y and V_z are the particle velocities in the x, y and z directions respectively, b is the buoyancy and f_x, f_y and f_z are external forces in the x, y and z direction respectively.

Initial conditions

The medium is supposed to be in equilibrium at time $t = 0$, meaning the stress and the velocity are set to zero everywhere in the medium.

Boundary conditions

Internal interfaces are not treated by explicit boundary conditions since they are naturally represented by changes of density and velocity of the medium for the heterogeneous case. Only six explicit boundary conditions have to be considered, namely, the six surface edges of the finite-sized three-dimensional grid. Depending on the problem, different boundary conditions can be used on these surfaces: absorbing boundary conditions (for simulating an infinite medium), stress-free conditions (also known as the

Neumann condition or free-surface condition), or zero-velocity conditions equivalent to zero-displacement conditions(the Dirichlet condition or rigid-surface condition).

3.2 Numerical Solution of the Wave Equation

For the acoustic wave equation, analytical solutions do exist. However, these analytical solutions only exist for special or simple cases like the homogeneous case. Therefore, for complex or sufficiently realistic models of the Earth's interior, it is necessary to resort to numerical methods. In numerical methods, the original differential equations are transformed into a system of algebraic equations that are easier to encode and implement on a computer. Since computer memory is finite, the continuous functions in the differential equation have to be represented by a finite set of numbers. The representation of the solution into finite set of numbers and the approximation of derivatives is dependent on the numerical method adapted to solve the problem.

There are a handful of numerical methods that can be used for modeling wave propagation, wherein each type of method also has many different variants. Finite element methods, finite difference methods, finite volume methods are among the most common class of methods. For seismic modeling, the choice of a suitable numerical method for modeling seismic wave propagation and seismic motion depends on several factors such as the capacity of the numerical method to allow for heterogeneity of the medium, the scale and dimensions of the domain, realistic attenuation, free-surface topography and sufficient frequency range[2]. It is also quite important for the numerical method to be computationally efficient in terms of computer memory and processing time. Practically speaking, there is no single method which would be the best with respect to all the requirements stated above since each method has its own pros and cons.

The following is a short discussion of the pros and cons of two prevailing methods in wave propagation modeling − Finite Difference and Finite Element method.

3.2.1 Finite Difference Method versus Finite Element Method

In the finite difference method each function is represented by its values at grid points. The space-time distribution of grid points may be, in principle, arbitrary but it significantly affects properties of the resulting numerical approximation. In the finite difference method usually no assumption is made about functional values in between the grid points. A derivative of a function is approximated using a finite difference formula which makes use of function values at a specified set of grid points.

In finite element method, an unknown function is usually represented by a linear combination of a finite number of continuous shape functions. Consequently, a finite set of coefficients of the expansion functions describe an approximate solution. The computational domain is covered by a mesh of elements whose nodal points form a grid of discrete points.

Finite difference is a classic method that is highly appreciated by geophysicist. Finite difference schemes are very popular since they provide the complete wavefield responses and are relatively simple and easy to implement in the computer codes. However, they are close to unsuitable for complex domain because of the staircase approximation induced by its use of a structured mesh. On the otherhand, finite element method is naturally suitable for such unstructured meshes. Therefore, even though finite element method is harder to implement, it is still prefered by others due to its capacity to adapt more naturally to the modeling of complex medium with large anisotropy, discontinuities, sharp boundaries, to name a few[3].

In this paper, the method implemented to numerically solve the 3D acoustic wave is the finite difference method due to its simplicity, speed and robustness.

3.2.2 Time Domain Modeling versus Frequency Domain Modeling

Time-domain methods for the wave equation modelling have reached a level of maturity where they can routinely be used on an industrial scale for two-dimensional problems. Realistic three dimensional simulations are still costly but feasible on present-day hardware.

In the frequency domain, the wave equation reduces to a system of linear equations, the right-hand side of which is the source and the solution is the seismic wavefield. Here, the LU decomposition required to solve a large sparse-matrix is costly. However, if many shots need to be treated, which is the case in Full Waveform Inversion, results of the matrix LU-decomposition can be reused and the method may still be competitive. This is an advantage over the time-domain variant since the time complexity of time-domain modeling linearly increases with the number of shots[4]. On the other hand, this increase in time complexity can be remedied by a coarse-grained parallel implementation by distributing the sources over processors. This strategy is quite plausible because time domain modeling is not too demanding with respect to memory.

For the two dimensional case, the frequency domain method still can still compete with the time-domain method. However in three dimensions, the complexity explodes and an iterative method appears to be necessary. The problem of finding an efficient solver for the three-dimensional case does not seem to justify their use. However, if these codes are to be used for processing purposes, in particular for migration and inversion, the balance changes. Inverse methods can be implemented to use the lowest data frequencies first, thus mitigating the notorious nonlinearities in the seismic inverse problem[5]. In that case, an additional order of magnitude can be gained by using only a limited number of frequencies[6].

In this report, time-domain modeling is used since the wave equation is three dimensional. Specifically, an explicit scheme is used for its straightforwardness in terms of implementation.

3.3 The Finite Difference Discretization of the 3D Acoustic Wave Equation

3.3.1 Staggered Grid Stencil

A major step forward in the FD modeling of seismic wave propagation in heterogeneous media was done by Virieux (1984, 1986), who used the idea of the staggered grid (Madariaga, 1976). Although Virieux did not say explicitly how he determined material grid parameters in his heterogeneous second-order SH and P-SV velocity-stress schemes, his numerical results were sufficiently accurate at the time. Moreover, the accuracy of the staggered-grid schemes did not suffer from large values of Poisson's ratio, which was the case of all displacement schemes on conventional grids[7].

Using a staggered grid is a simple way to avoid odd-even decoupling between the pressure and velocity. Odd-even decoupling is a discretization error that can occur on collocated grids and which leads to checkerboard patterns in the solutions.

In order to decrease the error of finite-difference schemes, the velocity vector components are determined in nodes (i, j, k, t) while calculations of the stress tensor or pressure components are shifted forward by a half-step of discretization in both space and time to the point $(i + 1/2, j + 1/2, k + 1/2, t + 1/2)$.

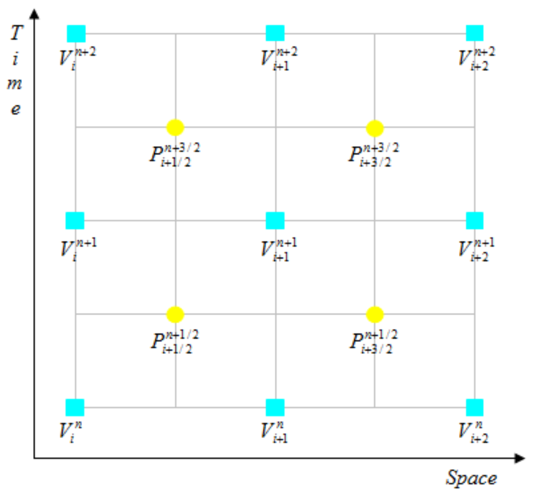
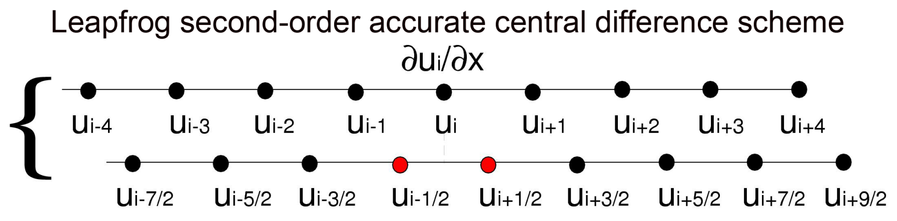


Figure 1: Staggered Grid in 1D

3.3.2 Discretized 3D Acoustic Wave Equation

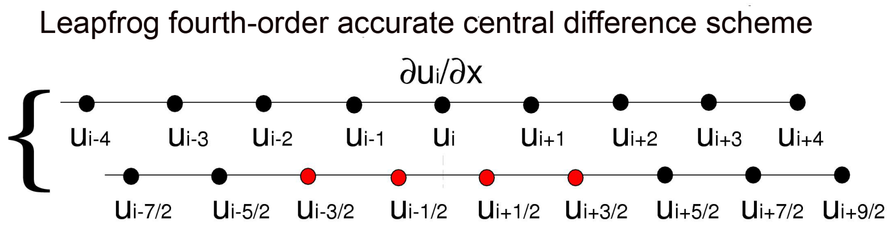
To discretize the differential operators of the acoustic wave equation (13), a *leapfrog scheme* on *staggered grids* is developed through the following:



$$\begin{aligned} u_{i+1/2} &= u_i + \frac{h}{2}\frac{\partial u_i}{\partial x} + \frac{h^2}{4}\frac{\partial^2 u_i}{\partial x^2} + \frac{h^3}{48}\frac{\partial^3 u_{i+\xi}}{\partial x^3} \\ u_{i-1/2} &= u_i - \frac{h}{2}\frac{\partial u_i}{\partial x} + \frac{h^2}{4}\frac{\partial^2 u_{i-\xi}}{\partial x^2} - \frac{h^3}{48}\frac{\partial^3 u_{i-\xi}}{\partial x^3} \end{aligned} \tag{14}$$

which gives

$$\frac{\partial u_i}{\partial x} = \frac{u_{i+1/2} - u_{i-1/2}}{h} + O(\frac{h^2}{48}) \tag{15}$$



For h/2,

$$\begin{aligned} u_{i+1/2} &= u_i + \tfrac{h}{2}\tfrac{\partial u_i}{\partial x} + \tfrac{h^2}{4}\tfrac{\partial^2 u_i}{\partial x^2} + \tfrac{h^3}{48}\tfrac{\partial^3 u_i}{\partial x^3} + \tfrac{h^4}{384}\tfrac{\partial^4 u_i}{\partial x^4} + O(h^5) \\ u_{i-1/2} &= u_i - \tfrac{h}{2}\tfrac{\partial u_i}{\partial x} + \tfrac{h^2}{4}\tfrac{\partial^2 u_{i-\xi}}{\partial x^2} - \tfrac{h^3}{48}\tfrac{\partial^3 u_i}{\partial x^3} + \tfrac{h^4}{384}\tfrac{\partial^4 u_i}{\partial x^4} + O(h^5) \end{aligned} \tag{16}$$

Taking the difference,

$$u_{i+1/2} - u_{i-1/2} = h\frac{\partial u_i}{\partial x} + \frac{h^3}{24}\frac{\partial^3 u_i}{\partial x^3} + O(h^5) \tag{17}$$

For 3h/2

$$\begin{aligned} u_{i+3/2} &= u_i + \tfrac{3h}{2}\tfrac{\partial u_i}{\partial x} + \tfrac{9h^2}{8}\tfrac{\partial^2 u_i}{\partial x^2} + \tfrac{27h^3}{48}\tfrac{\partial^3 u_i}{\partial x^3} + \tfrac{81h^4}{384}\tfrac{\partial^4 u_i}{\partial x^4} + O(h^5) \\ u_{i-3/2} &= u_i - \tfrac{3h}{2}\tfrac{\partial u_i}{\partial x} + \tfrac{9h^2}{8}\tfrac{\partial^2 u_{i-\xi}}{\partial x^2} - \tfrac{27h^3}{48}\tfrac{\partial^3 u_i}{\partial x^3} + \tfrac{81h^4}{384}\tfrac{\partial^4 u_i}{\partial x^4} + O(h^5) \end{aligned} \tag{18}$$

Taking the difference,

$$u_{i+3/2} - u_{i-3/2} = 3h\frac{\partial u_i}{\partial x} + \frac{9h^3}{8}\frac{\partial^3 u_i}{\partial x^3} + O(h^5) \tag{19}$$

To find an approximation of $\frac{\partial u_i}{\partial x}$ of the form

$$\frac{\partial u_i}{\partial x} = a_0(u_{i+1/2} - u_{i-1/2}) + a_1(u_{i+3/2} - u_{i-3/2}) \tag{20}$$

the following system is solved to obtain the necessary coefficients

$$\begin{cases} 1. = ha_0 + 3ha_1 \\ 0. = a_0\frac{h^3}{24} + a_1\frac{9h^3}{8} \end{cases} \tag{21}$$

Solving the above yields,

$$\begin{aligned} a_0 &= \tfrac{9}{8h} \\ a_1 &= -\tfrac{1}{24h} \end{aligned} \tag{22}$$

The *fourth-order accurate, central-difference approximation* to the first partial derivative of u is given by

$$\frac{\partial u_i}{\partial x} \approx \frac{1}{h}\left(\frac{9}{8}(u_{i+1/2} - u_{i-1/2}) - \frac{1}{24}(u_{i+3/2} - u_{i-3/2})\right) \tag{23}$$

By using discretized operators above, a second-order accurate in time and fourth-order accurate in space

$O\left((\Delta t)^2, (\Delta x)^4\right)$ discretization of the velocity-stress formulation of the acoustic wave equation (13) in three dimensions on a staggered grid is written as

$$\left\{\begin{array}{l}
\frac{P^{n+1}_{I,J,K}-P^{n}_{I,J,K}}{\Delta t} = E_{I,J,K}\frac{a_0\left(Vx^{n+1/2}_{I+1/2,J,K}-Vx^{n+1/2}_{I-1/2,J,K}\right)+a_1\left(Vx^{n+1/2}_{I+3/2,J,K}-Vx^{n+1/2}_{I-3/2,J,K}\right)}{\Delta x} \\
\qquad\qquad + E_{I,J,K}\frac{a_0\left(Vy^{n+1/2}_{I,J+1/2,K}-Vy^{n+1/2}_{I,J-1/2,K}\right)+a_1\left(Vy^{n+1/2}_{I,J+3/2,K}-Vy^{n+1/2}_{I,J-3/2,K}\right)}{\Delta y} \\
\qquad\qquad + E_{I,J,K}\frac{a_0\left(Vz^{n+1/2}_{I,J,K+1/2}-Vz^{n+1/2}_{I,J,K-1/2}\right)+a_1\left(Vz^{n+1/2}_{I,J,K+3/2}-Vx^{n+1/2}_{I,J,K-3/2}\right)}{\Delta z} \\
\frac{Vx^{n+1/2}_{I+1/2,J,K}-Vx^{n-1/2}_{I+1/2,J,K}}{\Delta t} = \frac{b_{I+1/2,J,K}}{\Delta x}\left[a_o\left(P^n_{I+1,J,K}-P^n_{I,J,K}\right)+a_1\left(P^n_{I+2,J,K}-P^n_{I-1,J,K}\right)\right]+Fx^n_{I+1/2,J,K} \\
\frac{Vy^{n+1/2}_{I,J+1/2,K}-Vy^{n-1/2}_{I,J+1/2,K}}{\Delta t} = \frac{b_{I,J+1/2,K}}{\Delta y}\left[a_o\left(P^n_{I,J+1,K}-P^n_{I,J,K}\right)+a_1\left(P^n_{I,J+2,K}-P^n_{I,J-1,K}\right)\right]+Fy^n_{I,J+1/2,K} \\
\frac{Vz^{n+1/2}_{I,J,K+1/2}-Vz^{n-1/2}_{I,J,K+1/2}}{\Delta t} = \frac{b_{I,J,K+1/2}}{\Delta z}\left[a_o\left(P^n_{I,J,K+1}-P^n_{I,J,K}\right)+a_1\left(P^n_{I,J,K+2}-P^n_{I,J,K-1}\right)\right]+Fz^n_{I,J,K+1/2}
\end{array}\right. \tag{24}$$

Multiplying both side of all equations by Δt and isolating the latest time update term onto the left side would yield a fully explicit scheme which can be easily encoded as subroutines of the main program. An important detail to take note of is that array indices in programming can only be integers and so it is important to keep in mind the staggered nature of the grid that is forced to be concealed by the integer indexing within the program.

3.3.3 Numerical Dispersion and Stability

The choice of grid spacing and time step cannot be done arbitrarily, as a number of conditions should be satisfied to avoid various numerical errors.

A common numerical error in wave propagation modeling occurs when the the cell size becomes too large in comparison with the wavelength of the source signal. When this happens, waves disperse with increasing traveltime. This phenomenon is known as grid dispersion. The grid dispersion is the variation of the numerical phase velocity as a property of frequency. What commonly occurs is that higher signal frequencies travel more slowly than lower signal frequencies. Consequently, substantial tailing of the signal arises with increasing traveltime.

Grid dispersion may occur if the grid spacing is large or the wavelength of the source is too short compared with the size of the grid. This translates into the importance of choosing the proper number of grid points per minimum wavelength of the source signal.

To avoid the first instance, the rule of the thumb in difference schemes for choosing an appropriate grid spacing based on the maximum frequency is expressed as

$$\Delta x = \frac{c_{min}}{f_n \cdot n} \tag{25}$$

where f_{max} is the maximum frequency, c_{min} is the minimum velocity of propagation, and n is number of points needed to cover the maximum frequency for nondispersive propagation.

In this report a Ricker wavelet, the second derivative of a Gaussian, is used as the synthetic source. For the aforementioned wavelet, this rule of the thumb is an effective criterion for nondispersive propagation[8]. The discretization criteria to reduce the effects of numerical dispersion have long been established (5-8) gridpoints per wavelength for a fourth-order accurate FD scheme.

Another condition is the stability condition for the time step which can be expressed as

$$\Delta t \leq \xi \frac{\Delta x}{c_{max}} \tag{26}$$

where ξ depends on the scheme. For the scheme used in this report, a working value for ξ is 0.48.

3.3.4 Absorbing Boundary Condition

In numerical modeling of wave propagation in seismic exploration, the actual domain usually extends farther than the domain of interest. This actually poses a problem on setting the boundary conditions since simple truncation of the wave field on the boundaries lead to unwanted artificial spurious wave reflections. Thus, to simulate an unbounded medium, nonreflecting conditions must be defined at the artificial boundaries of the computational domain so that energy will be absorbed by these boundaries.

A solution to this problem, however not in the context of seismic waves but in that of Maxwell's equation, was introduced by Berenger in 1994. He introduced a new condition called the perfectly matched layer (PML) which was named so because it had the notable property of having a zero reflection coefficient for all angles of incidence and all frequencies. This condition was proven to be more efficient than other conditions at that time and has become popular ever since.

However, the perfectly matched layer (PML) is not so perfect after discretization since, in the context of Maxwell's equations and elastodynamic equations, the reflection coefficient is not zero after discretization and even becomes very large at grazing incidence. This means that large amounts of energy are going to be sent from the PML layers to the main domain in the form of spurious reflected waves. Thus, in this case, the classical discretized PML is less efficient especially in cases like thin mesh slices, sources situated near the edge of the domain or receivers with very large offset location − cases which are common in seismic exploration.

An idea to improve the behavior of the discretized PML at grazing incidence was developed Kuzuoglu and Mittra (1996) and Roden and Gedney (2000) for Maxwell's equation. The main idea is to add to the complex coordinate transform used in classical PML, a frequency-dependent term that implements a Butterworth-type filter in the layer. Since it is frequency-dependent, for waves whose incidence is close to normal, this filter almost does not change anything while for waves with grazing incidence, which for geometrical reasons do not penetrate deeply enough into the PML but travel in a direction parallel to the PML layer, the filter will strongly attenuate these waves to prevent them from leaving the PML with significant energy. This situation is desirable since for the former kind of waves mentioned, absorption of the classical PML is already almost perfect. This modification of the classic PML is called convolutional-PML (C-PML) or complex frequency shifted-PML (CFS-PML).

This boundary condition is not restricted to the context of Maxwell's equation as it has also been developed and applied to seismic wave modeling. This formulation of PML is usually applied on a split formulation of the equations of elastodynamics. The formulation that was developed by Komatitsch and Martin(2007) has the advantage of not being split and the cost in terms of memory storage is similar to that of the classical PML[9]. This is the PML formulation applied in the code.

3.4 A More Detailed Illustration of the Discretization Method through the 1D Case

A physical justification of the finite difference method for modeling wave propagation is Huygens' principle, which basically states that every point of a wave front may be considered the source of secondary wavelets that spread out in all directions with a speed equal to the speed of propagation of the waves.

Thus in a finite difference grid, each point acts as a secondary source and the envelope of elementary diffractions provides the seismic response of the continuous medium, if this medium is discretized in a sufficiently fine way.

Starting of with the most basic example of the one-dimensional scalar wave equation to illustrate the numerical analysis involved in finite difference discretization[10], one can take similar steps and extend this for the actual equations of motion implemented in the code whose results will be discussed later.

It should be noted that an acoustic wave is a common example of a scalar field. It is a small disturbance in pressure and density which propagates in a compressible medium.

3.4.1 The 1D Wave Equation for Homogeneous Medium

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{27}$$

A second-order accurate centered scheme finite difference discretization of the second order partial differential operators are

$$\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i,t_n} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)} + O\left((\Delta x)^2\right) \tag{28}$$

$$\left.\frac{\partial^2 u}{\partial t^2}\right|_{x_i,t_n} = \frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\Delta t)^2} + O\left((\Delta t)^2\right) \tag{29}$$

Thus a second-order discretization of the wave equation is given by the following expression

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\Delta t)^2} + O\left((\Delta t)^2\right) = c^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + O\left((\Delta x)^2\right)\right] \tag{30}$$

By solving for the latest value of u at the spatial coordinate denoted by i, one gets a fully explicit second-order accurate finite difference approximation of the aforementioned wave equation:

$$u_i^{n+1} = (c\Delta t)^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}\right] + 2u_i^n - u_i^{n-1} + O\left((\Delta t)^2\right) + O\left((\Delta x)^2\right) \tag{31}$$

An interesting particular case is the case when the relation between the time-step, the grid interval and the wave propagation speed satisfies the relation $c\Delta t = \Delta x$

Plugging $c\Delta t = \Delta x$ in the discrete wave equation leads to,

$$u_i^{n+1} = u_{i+1}^n + u_{i-1}^n - u_i^{n-1} \tag{32}$$

Notice that the dependency with respect Δx and Δt has disappeared. This discrete form provides the error-free solution of the wave equation.

To verify this, the solution $F(x + ct) + G(x - ct)$ is plugged into the discrete wave equation

$$\begin{bmatrix} F(x_i + ct_{n+1}) \\ +G(x_i - ct_{n+1}) \end{bmatrix} = \begin{bmatrix} F(x_{i+1} + ct_n) \\ +G(x_{i+1} - ct_n) \end{bmatrix} + \begin{bmatrix} F(x_{i-1} + ct_n) \\ +G(x_{i-1} - ct_n) \end{bmatrix} - \begin{bmatrix} F(x_i + ct_{n-1}) \\ +G(x_i - ct_{n-1}) \end{bmatrix} \tag{33}$$

On the right hand side, the coordinates are written as a function of Δx and Δt

$$RHS = \begin{bmatrix} F((i+1)\Delta x + cn\Delta t) \\ +G((i+1)\Delta x - cn\Delta t) \end{bmatrix} + \begin{bmatrix} F((i-1)\Delta x + cn\Delta t) \\ +G((i-1)\Delta x - cn\Delta t) \end{bmatrix} - \begin{bmatrix} F(i\Delta x + c(n-1)\Delta t) \\ +G(i\Delta x - c(n-1)\Delta t) \end{bmatrix} \tag{34}$$

Using $c\Delta t = \Delta x$ and doing some simplifications yield

$$RHS = F\left[(i + 1 + n)\Delta x\right] + G\left[(i - 1 - n)\Delta x\right] \tag{35}$$

These two waves match the left hand side, that is, the exact solution of the wave equation at position x_i and time t_{n+1}. Therefore, the exact solution of the 1D wave equation in homogeneous media can be numerically computed for any value of Δx provided that $\Delta t = \Delta x/c$. Unfortunately, this non-intuitive property is only satisfied for 1D homogeneous media.

3.4.2 Dispersion Analysis

Dispersion is defined as the variation of a propagating wave's wavelength λ with frequency f. It may also be presented as the variation of the propagating wave's wavenumber $k = 2\pi/\lambda$ with the angular frequency $\omega = 2\pi f$.

The dispersion relation for the 1D wave equation is obtained by substituting a continuous sinusoidal-traveling-wave solution of (27) written in phasor form

$$u(x,t) = e^{j(\omega t - kx)} \tag{36}$$

(where ω is the angular frequency, k is the wavenumber and $j = \sqrt{-1}$)

to the 1D wave equation.

This yields

$$(j\omega)^2 e^{j(\omega t - kx)} = c^2(-jk)^2 e^{j(\omega t - kx)} \tag{37}$$

and by eliminating the complex exponential term on both sides, and solving for k, the dispertion relation is obtained

$$k = \pm\omega/c \tag{38}$$

The above relation states that the wavenumber is linearly proportional to the sinusoidal frequency, with the signs designated to the corresponding direction of propagation along the x axis.

From this relation, one can get an expression for the wave phase velocity, the rate at which the phase of the wave propagates in space, which is defined as either $v_p = \lambda/T$ (where T is the period) or equivalently $v_p = \omega/k$:

$$v_p = \frac{\omega}{k} = \pm c \tag{39}$$

This means that for all frequencies, the magnitude of the phase velocity is constant and the same for all. Propagating waves having a dispersion relation of the form of (38) which results to a constant phase velocity of (39) are said to be *dispersionless*. This means that their waveshape does not change after arbitrarily large propagation distances for arbitrary modulation envelopes or pulse shapes.

The same procedure can be applied to obtain the *numerical dispersion* relation of the finite difference approximation of the 1D wave equation:

$$u_i^{n+1} \approx (c\Delta t)^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}\right] + 2u_i^n - u_i^{n-1} \tag{40}$$

Plugging a numerical sinusoidal traveling wave at the discrete space-time point (x_i, t_n):

$$u_i^n = u(x_i, t_n) = e^{j(\omega n\Delta t - \tilde{k} i\Delta x)} \tag{41}$$

where $\tilde{k}$ is the wavenumber of the numerical sinusoidal traveling wave of angular frequency ω that is present in the finite difference grid. This $\tilde{k}$ is in general different from the wavenumber k of the corresponding continuous physical wave whose angular frequency is the same ω. The difference between the numerical $\tilde{k}$ and the actual physical k leads to numerical phase and group velocities that depart from the exact values. This phenomenon is called numerical dispersion, one of the many possible causes of numerical errors.

Like in the continuous and physical case, the sinusoidal traveling wave is also substituted into the discretized version of the wave equation, the complex exponential term is then factored out. After doing a few manipulations, one may arrive at a trigonometric expression for the numerical dispersion relation corresponding to the finite difference algorithm (30)

$$\cos(\omega \Delta t) = \left(\frac{c\Delta t}{\Delta x}\right)^2 \cdot \left[\cos(\tilde{k}\Delta x) - 1\right] + 1 \tag{42}$$

It is important to notice that using the relation ($c\Delta t = \Delta x$) in the numerical dispersion expression will lead to

$$\cos(\omega \Delta t) = 1 \cdot \left[\cos(\tilde{k}\Delta x) - 1\right] + 1 = \cos(\tilde{k}\Delta x) \tag{43}$$

which is true if and only if $\tilde{k}\Delta x = \pm\omega\Delta t$. Solving for $\tilde{k}$ and using the relation ($c\Delta t = \Delta x$) again leads to

$$\tilde{k} = \pm\frac{\omega}{c} \tag{44}$$

This means that if ($c\Delta t = \Delta x$), the numerical dispersion relation reduces to that of the corresponding continuous dispersion relation. This is important because it means that the numerical solution is an exact discretization of the continuous one.

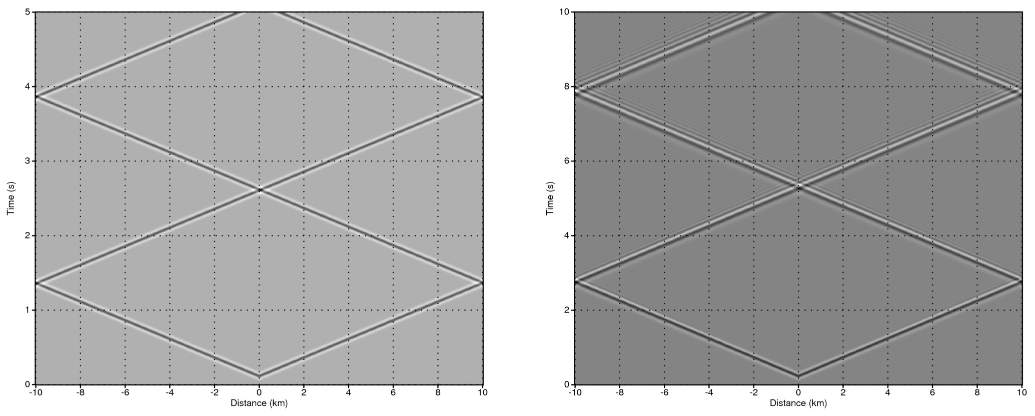


Figure 2: **Left:**Free Surface Condition c=4000 m/s (nondispersive) Right:Free Surface Condition c=4000 m/s (dispersive)

3.4.3 Numerical Stability

Another important issue of numerical schemes, specifically of explicit time marching algorithms is numerical stability. This is affected by the choice of the time step Δt, in which if it is chosen to be too large, the computed result will spuriously increase without limit as time-marching continues.

A restricting condition on the time step was founded by Courant, Friedrichs and Levy (CFL). An intuitive interpretation of the idea behind this is to apply a sinusoidal wave whose frequency can be complex and to find the restriction on the time step that will not give an amplification factor for this sinusoidal wave. This amplification factor leads to instability since due to this, the sinusoidal wave will increase without limit as the time-marching continues.

To start off with the analysis, define the frequency as a complex frequency $\tilde{\omega} = \tilde{\omega}_r + j\tilde{\omega}_j$ and assuming that the wavenumber is real, the expression of the same harmonic wave from the previous section becomes,

$$u_i^n = e^{j(\tilde{\omega}n\Delta t - \tilde{k}i\Delta x)} = e^{-\tilde{\omega}_j\Delta} \cdot exp^{j(\tilde{\omega}_r n\Delta t - \tilde{k}i\Delta x)} \tag{45}$$

Repeating a derivation similar from that of the dispersion analysis yields,

$$cos(\tilde{\omega}\Delta t) = \left(\frac{c\Delta t}{\Delta x}\right)^2 \left[cos(\tilde{k}\Delta x) - 1\right] + 1 \tag{46}$$

Solving for the complex frequency leads to the expression,

$$\tilde{\omega} = \frac{1}{\Delta t} cos^{-1}(\xi) = \frac{1}{\Delta t}\left[\frac{\pi}{2} - sin^{-1}(\xi)\right] \tag{47}$$

with $\xi = S^2 \left[cos(\tilde{k}\Delta x) - 1\right] + 1$, where $S = \left(\frac{c\Delta t}{\Delta x}\right)$ which is called the Courant number.

For a real $\tilde{k}$, $1 - 2S^2 \leq \xi \leq 1$. There are two cases to be considered. The first case is when $-1 \leq \xi \leq 1$, which gives $0 \leq S \leq 1$

In this case, $sin^{-1}(\xi)$ is real-valued and the imaginary part of the frequency is zero. There is no amplification of the solution, and the numerical scheme is stable.

The second case is when $1 - 2S^2 \leq \xi \leq -1$, which gives $S > 1$. One should notice that the most negative possible value of ξ is reached for $cos(k\Delta x) = -1$. This expression is equivalent to $\lambda = 2\Delta x$, which corresponds to the Nyquist condition whose consequence leads to λ being undersample or having not enough gridpoints per minimum wavelength.

Now, substituting the expression $sin^{-1}(\xi) = -j\ln\left(\mathrm{j}\xi + \sqrt{1-\xi^2}\right)$ in the expression of the complex frequency gives,

$$\tilde{\omega} = \frac{1}{\Delta t}\left\{\pi + j\ln\left(-\xi - \sqrt{\xi^2 - 1}\right)\right\}, \tag{48}$$

that provides the amplification factor associated with the imaginary part of the frequency.

$$u_i^n = \left(\frac{1}{-\xi - \sqrt{\xi^2 - 1}} \right)^n e^{j\left[(\pi/\Delta t)(n\Delta t) - \tilde{k} i \Delta x\right]} \tag{49}$$

For the coarser sampling, $\Delta x = \pi/\tilde{k}$, which leads to $\xi = 1 - 2S^2$, the amplification factor is maximal and is given by,

$$u_i^n = \left(S + \sqrt{S^2 - 1} \right)^2 e^{j\left[(\pi/\Delta t)(n\Delta t) - \tilde{k} i \Delta x\right]} \tag{50}$$

The above confirms that for the case $S = 1$, there is no amplification. Therefore, the solution is stable if $S \leq 1$ which means that the restriction on the time step is $\Delta t \leq \frac{\Delta x}{c}$

3.4.4 1D Wave Equation in Heterogeneous Media

The 1D wave equation in heterogeneous 1D media can be written as

$$\rho(x) \frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x} \left(E(x) \frac{\partial u}{\partial x} \right) \tag{51}$$

where u is the displacement, ρ is the density, E is the bulk modulus, t is time and x is the spatial coordinate. The propagation velocity is given by $c =\sqrt{\frac{E}{\rho}}$.

Introducing the auxiliary stress field

$$\begin{cases} \rho(x) \frac{\partial^2 u}{\partial t^2} = \frac{\partial \tau(x,t)}{\partial x} \\ \tau(x,t) = E(x) \frac{\partial u(x,t)}{dx} \end{cases} \tag{52}$$

allows the reduction of the order of the spatial derivative.

The first equation is the equation of motion and the second one is Hooke's law. Introducing the particle displacement velocity, $v(x) = \frac{\partial u(x)}{\partial t}$ yields:

$$\begin{cases} \frac{\partial v(x,t)}{\partial t} = b(x) \frac{\partial \tau(x,t)}{\partial x} \\ \frac{\partial \tau(x,t)}{\partial x} = E(x) \frac{\partial v(x,t)}{dx} \end{cases} \tag{53}$$

where b is the buoyancy.

This approach reformulates the original second order wave equation into a first-order hyperbolic system. The reduction of the order of the partial derivatives by introducing the auxiliary stress field is attained at the expense of increasing the number of variables. The first-order hyperbolic system can then be discretized with a second-order accurate centered staggered-grid stencil.

$$\begin{cases} v_i^{n+1} = v_i^n + \frac{\Delta t}{h} b_i \left[\tau_{i+\frac{1}{2}}^{n+\frac{1}{2}} - \tau_{i-\frac{1}{2}}^{n+\frac{1}{2}} \right] \\ \tau_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \tau_{i+\frac{1}{2}}^{n+\frac{1}{2}} + \frac{\Delta t}{h} E_{j+\frac{1}{2}} \left[v_{i+1}^{n+1} - v_i^{n+1} \right] \end{cases} \tag{54}$$

where n and i denote the temporal and spatial indices, respectively. After discretization, it is possible to come back to a second-order partial differential equation by eliminating the auxiliary stress field. This elimination procedure after discreitzation is the so-called parsimonious approach (Luo and Schuster, 1990) Injecting the expression $\tau_{i+\frac{1}{2}}^{n+\frac{1}{2}}$ and $\tau_{i-\frac{1}{2}}^{n+\frac{1}{2}}$ in the first equation,

$$\begin{cases} \tau_{i+\frac{1}{2}}^{n+\frac{1}{2}} = \tau_{i+\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\Delta t}{h} E_{j+\frac{1}{2}} \left[v_{i+1}^{n} - v_i \right] \\ \tau_{i-\frac{1}{2}}^{n+\frac{1}{2}} = \tau_{i-\frac{1}{2}}^{n-\frac{1}{2}} + \frac{\Delta t}{h} E_{i-\frac{1}{2}} \left[v_i^n - v_{i-1}^{n} \right] \end{cases} \tag{55}$$

results to,

$$v_i^{n+1} = v_i^n + \left(\frac{\Delta t}{h} \right)^2 \left[E_{i+\frac{1}{2}} \left[v_{i+1}^n - v_i^n \right] - E_{i-\frac{1}{2}} \left[v_i^n - v_{i-1}^n \right] \right] + \frac{\Delta t}{h} b_i \left[\tau_{i+\frac{1}{2}}^{n-\frac{1}{2}} - \tau_{i-\frac{1}{2}}^{n-\frac{1}{2}} \right] \tag{56}$$

Using the relation $v_i^n = v_i^{n-1} + \frac{\Delta t}{h} b_i \left(\tau_{i+\frac{1}{2}}^{n-\frac{1}{2}} - \tau_{i-\frac{1}{2}}^{n-\frac{1}{2}} \right)$ the following expression is obtained:

$$\frac{v_i^{n+1} - 2v_i^n + v_i^{n-1}}{\Delta t^2} = b_i \frac{E_{i+\frac{1}{2}} v_{i+1}^n - E_{i+\frac{1}{2}} + E_{i-\frac{1}{2}} v_i^n + E_{i-\frac{1}{2}} v_{i-1}^n}{h^2} \tag{57}$$

3.4.5 Simulation of Infinite Medium

Radiation

The solution of the 1D wave equation in homogeneous media is $u(x,t) = f(x-ct) + f(x+ct)$, where f is a function describing the waveform over time and space. The partial solutions $f(x-ct)$ and $f(x+ct)$ describe propagation in two opposite directions.

To mimic an infinite medium radiation conditions on the left and right boundaries should be imposed as the following:

Right edge

On the right edge, the wavefield must satisfy: $u(x,t) = f(x-ct)$, which gives according to Hooke's law, ($\tau = E(x)\frac{\partial u}{\partial x}$),

$$\begin{aligned} v(x,t) &= -cf'(x-ct) \\ \tau(x,t) &= E(x)f'(x-ct) \end{aligned} \tag{58}$$

which leads to the radiation condition on the right edge: $\tau(L,t) = -Z(l)v(L,t)$, where $Z = \rho c$ called the impedance.

Left edge

On the left edge, the wavefield must satisfy: $u(x,t) = f(x + ct)$, which leads to the radiation condition: $\tau(0,t) = Z(0)v(0,t)$

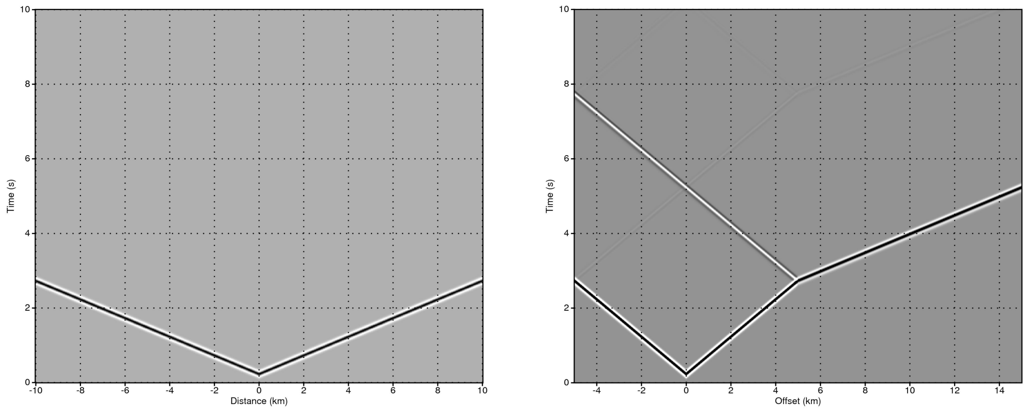


Figure 3: **Left:**Radiation Condition c=4000 m/s (nondispersive) **Right:** Simulation in a two-layer medium c1=2000 m/s c2=4000 m/s - S=1 in the high-velocity layer

Sponge

The sponge boundary condition can give a basic idea of what the Perfectly Matched Layer does to mimic an unbounded medium. Basically, the computational domain is augment with one absorbing (sponge) layer at each ends of the model. Then a modified 1D wave equation, with an additional damping term is introduced

$$\left\{ \begin{array}{l} \frac{\partial v(x,t)}{\partial t} + \gamma(x)v(x,t) = l(x)\frac{\partial \tau(x,t)}{\partial x} \\ \frac{\partial \tau(x,t)}{\partial t} + \gamma(x)\tau(x,t) = E(x)\frac{\partial v(x,t)}{\partial x} \end{array} \right. \quad (59)$$

where $\gamma(x)$ are functions, the values of which are 0 in the medium and progressively increase in the absorbing layers.

Discretizing this system leads to,

$$\left\{ \begin{array}{l} \frac{v_i^{n+1} - v_i^n}{\Delta t} + \frac{1}{2}\gamma_i(v_i^{n+1} + v_j^n) = \frac{1}{\Delta x} b_i \left[\tau_{i+1/2}^{n+1/2} - \tau_{i-1/2}^{n+1/2} \right] \\ \frac{\tau_{i+1/2}^{n+3/2} - \tau_{i+1/2}^{n+1/2}}{\Delta t} + \frac{1}{2}\gamma_{i+1/2}(\tau_{i+1/2}^{n+3/2} + \tau_{i+1/2}^{n+1/2}) = \frac{1}{\Delta x} E_{i+1/2} \left[v_{i+1}^{n+1} - v_i^{n+1} \right] \end{array} \right. \quad (60)$$

After grouping terms with respect to time and using the approximation $1 \pm \frac{\Delta t\gamma(x)}{2} \approx e^{\pm j\frac{\Delta t\gamma(x)}{2}}$, the following is obtained,

$$\left\{ \begin{array}{l} v_i^{n+1} = e^{-\Delta t\gamma_i} v_i^n + e^{-\frac{\Delta t\gamma_i}{2}} \frac{\Delta t b_i}{\Delta x} \left[\tau_{i+1/2}^{n+1/2} - \tau_{i-1/2}^{n+1/2}\right] \\ \tau_{i+1/2}^{n+3/2} = e^{-\Delta t\gamma_{i+1/2}} \tau_{i+1/2}^{n+1/2} + e^{-\frac{\Delta t\gamma_{i+1/2}}{2}} \frac{\Delta t E_{i+1/2}}{\Delta x} \left[v_{i+1}^{n+1} - v_i^{n+1}\right] \end{array} \right. \tag{61}$$

For $\Delta t = \Delta x/c$, waves $e^{-x\gamma(x)/c}v(x)$ and $e^{-x\gamma(x)/c}, \tau(x)$ propagate without dispersion but with a decreasing amplitude with distance, with an attenuation rate $\frac{\gamma}{c}$ independent to the frequency.

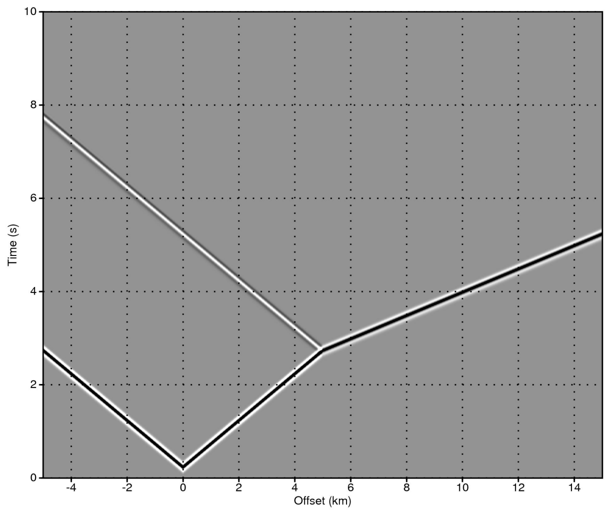


Figure 4: Simulation in a two-layer medium c1=2000 m/s c2=4000 m/s - S=1 in the high-velocity layer

4 Parallel Implementation

Because of the coarse-grained nature of finite difference method and the massive scale of the domain of the modeling problem, parallel implementation of the code is not only a must but also seems to come easily. A simple way to divide the work among several processes is through classical domain decomposition of the 3D block domain. Since finite difference computations are done per grid point which only require information from some specified neigboring gridpoints, only data along the overlaps of the subdomain with its neighbors is needed by each subdomain to complete the computation for all the gridpoints belonging to it. The size of the interface area is much smaller than the size of the overall problem meaning the cost of communicating data is usually almost negligible compared to the computational cost of the overall problem.

Basically, the main idea of parallel implementation through domain decomposition is to split the original domain of computation into subdomains, compute local simplified solutions and to communicate the data needed from and by the neighboring domains[11]. This approach has the advantage of lending itself well to the use of local memory.

A parallel version of the general algorithm based on the principle of domain decomposition that is suitable for structured meshes, is as follows:

- decompose the mesh into subdomains and assign each subdomain to a process
- for each subdomains determine its neighboring subdomains
- iterate time
- exchange messages among interfaces
- calculate

The programming model chosen is the SPMD (Single Program Multiple Data), meaning a single code is ran on all processes. The use of MPI in this type of domain decomposition is very convenient since there are several predefined functions in MPI that make it possible to create a virtual grid of processes, to determine the process and its neighbors, to name a few. The following section discusses the MPI commands and their usage in implementing the domain decomposition. This is based on the the

----

4.1 Methodology

The size of the whole domain in kilometers, the grid spacing (dx,dy,dz), the number of subdomains per z, y and x axis ($nd1$,$nd2$,$nd3$) and the number of PML points are given by the user.

4.1.1 SUBROUTINE init

The procedure **init** executes the decomposition of the original domain into subdomains and the initialization of MPI.

The domain is cut in the z, y and x directions depending on the number of subdomains per direction specified by the user. The number of gridpoints ($n1loc, n2loc, n3loc$) in each subdomain per direction is determined by dividing the total number of grid points (main domain + pml layer) by the number of subdomains in that direction and assigning the excess final points in the latter subdomains. Each process is identified by its local coordinates, which can either be a single number that corresponds to the $nd1 \times nd2 \times nd3$ coordinate system or three numbers which correspond to the (z, y, x) (since z is the fast index) coordinate system of the processes. There are simple formulas used to switch back and forth between these two coordinate systems.

The assignment of subdomain to processes is well-ordered, governed by the indexing illustrated below. This makes determining the process handling the neighboring subdomains of a current subdomain easy.

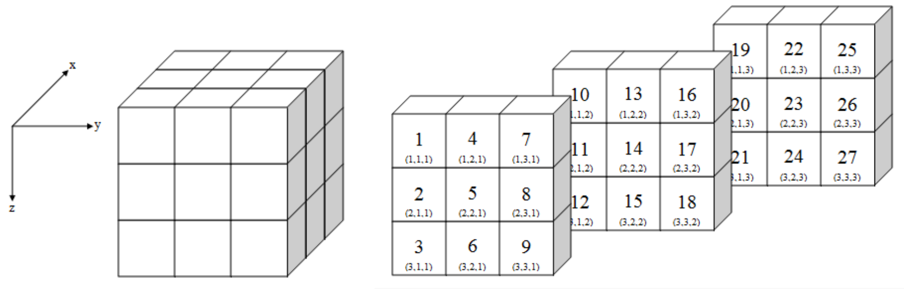


Figure 5: Example of Indexing of Processes

4.1.2 SUBROUTINE voisinage

The procedure voisinage determines the existing neighbors of a subdomain and which processes they correspond to. The 3-coordinate indexing makes checking for neighbors quite convenient since subdomains that contain a part of the boundary of the whole domain would either have, in any of the three directions, either a 1 or the maximum number of subdomains in that direction as an index value.

A domain that has neighbors on all edges has six neighbors (N,S,W,E,Up,Down) all in all. Below is an illustration of these six neighbors.

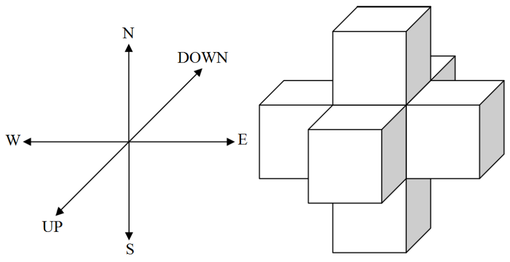


Figure 6: Six Possible Neighbors of a Subdomain

4.1.3 SUBROUTINE typage

The procedure **typage** defines data blocks to be used in sending and receiving messages. MPI offers the advantage to optimize the communications by allowing the user to define customized data types that can be simultaneously sent and received. This saves the user from hardcoding the exact extent of the array needed to be sent and where it would be received. With customized data types, one only needs to specify the initial address of that certain block of data needed. To pass data to and from the overlaps of subdomains, data types for each type of face are defined by using the following MPI functions:

MPI_TYPE_VECTOR allows replication of a datatype into locations that consist of equally spaced blocks. Each block is obtained by concatenating the same number of copies of the old datatype. The spacing between blocks is a multiple of the extent of the old datatype.

1st parameter: number of blocks
2nd parameter: number of elements in each block
3rd parameter: number of elements between start of each block
4th parameter: old type
5th parameter: new type

MPI_TYPE_HVECTOR is almost the same as MPI_TYPE_VECTOR except that the stride (3rd paramenter) between the start of each block is in bytes instead of the number of elements. For this, the function **MPI_TYPE_EXTENT** is used to return the extent of the data type *real* .

The function **MPI_TYPE_COMMIT** commits the datatype, that is, the formal description of a communication buffer, not the content of that buffer. Thus, after a datatype has been committed, it can be repeatedly reused to communicate the changing content of a buffer or, indeed, the content of different buffers, with different starting addresses.

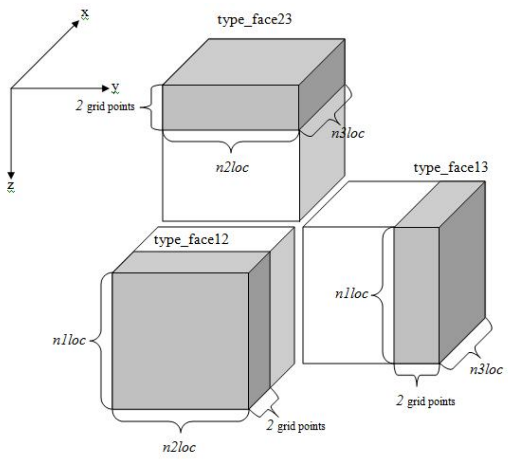


Figure 7: The Three Different Types of Interface

Figure 7 shows the three new datatypes to be constructed.

The first step to construct the face data types is to construct a base type using MPI_TYPE_VECTOR of corresponding length in the z-axis. MPI_TYPE_VECTOR can be used since z is the fast index, meaning data is contiguous in memory along this axis.

The next step is to construct another base type from the previous one using MPI_TYPE_HVECTOR of corresponding number of elements in the y-axis. The stride between these elements are $(n1loc + 4) * size\_of\_real$ since there are $n1loc$ grid points of the domain for the z-axis plus 4 more, an additional 2 for each side, due to the domain overlap resulting from the 4th order accurate stencil.

The type face is then constructed from the latest base type with the corresponding number of elements in the x-axis. The stride between these elements is $(n1loc+4)*(n2loc+4)*size\_of\_real$ since there are $(n1loc+4)*(n2loc+4)*size\_of\_real$ bytes in between each element along the same x-axis in terms of memory.

4.1.4 SUBROUTINE communication

The procedure **communcation** is done within the loop in time since the data between interfaces need to be communicated at each calculation. The purpose of this procedure is to send data blocks from the subdomain to the corresponding neighboring areas and to receive the same points in the relevant fields. For this, the function MPI_SENDRECV, a blocking send-receive function which allows simultaneous sending and receiving of data, is used.

The arguments of the function MPI_SENDRECV are:

1st parameter: initial address of the sending
2nd parameter: number of elements to send
3rd parameter: type of elements to send
4th parameter: destination
6th parameter: initial address of the reception
7th parameter: number of elements to receive
8th parameter: type of elements to receive
9th parameter: source

For each subdomain, a three-dimensional array (to be called **x**) is allocated as such

$$\mathbf{x}(-1:n1loc+2,-1:n2loc+2,-1:n3loc).$$

The original extents of the subdomain $(1:n1loc, 1:n2loc, 1:n3loc)$ are augmented with 2 more grid-points on each side in all three directions, hence the indexing of **x**.

Below is a table that summarizes the communication procedure:

SEND-RECEIVE,	Send Address	Receive Address	Type
send to N-receive from S	x(1, 1, 1)	x(1, $n2loc+1$, 1)	type_face13
send to W-receive from E	x(1, 1, 1)	x(1, 1, $n3loc+1$)	type_face12
send to UP-receive from DOWN	x(1, 1, 1)	x($n1loc+1$, 1, 1)	type_face23
send to DOWN-receive from UP	x($n1loc-1$, 1, 1)	x(-1, 1, 1)	type_face23
send to E-receive from W	x(1, 1, $n3loc-1$)	x(1, 1, -1)	type_face12
send to S receive from N	x(1, $n2loc-1$, 1)	x(1, -1, 1)	type_face13

```
Intialize MPI
Read input file
Compute source wavelet
Decompose the domain and define data types for interface
Initialize p, vx, vy and vz to zero (Initial conditions)
Build the parameters b and kappa and C-PML parameters on each subdomain
Locate corresponding subdomain location of source, receivers and topography points
Do it=1,nt
       Take snapshots of pressure wavefield
       Record pressure values at each time step for each receiver
       Communicate vx, vy and vz between subdomains
       Do i3=1,n3loc
         Do i2=1,n2loc
           Do i1=1,n1loc
             dvx_dx = ( a0*(vx(i1,i2,i3)-vx(i1,i2,i3-1)) + a1*(vx(i1,i2,i3+1)-vx(i1,i2,i3-2)) ) / dx
             dvy_dy = ( a0*(vy(i1,i2,i3)-vy(i1,i2-1,i3)) + a1*(vy(i1,i2+1,i3)-vy(i1,i2-2,i3)) ) / dy    Update
             dvz_dz = ( a0*(vz(i1,i2,i3)-vz(i1-1,i2,i3)) + a1*(vz(i1+1,i2,i3)-vz(i1-2,i2,i3)) ) / dz      p
             Apply C-PML on dv_dx,dv_dy and dv_dz
             p(i1,i2,i3) = p(i1,i2,i3) + kappaloc(i1,i2,i3) * (dvx_dx + dvy_dy + dvz_dz)
           End do
         End do
       End do
       Increment Source
       Communicate p between subdomains
       Do i3=1,n3loc
         Do i2=1,n2loc
           Do i1=1,n1loc
             dp_dx = ( a0*(p(i1,i2,i3+1)-p(i1,i2,i3)) + a1*(p(i1,i2,i3+2)-p(i1,i2,i3-1)) ) / dx
             dp_dy = ( a0*(p(i1,i2+1,i3)-p(i1,i2,i3)) + a1*(p(i1,i2+2,i3)-p(i1,i2-1,i3)) ) / dy
             dp_dz = ( a0*(p(i1+1,i2,i3)-p(i1,i2,i3)) + a1*(p(i1+2,i2,i3)-p(i1-1,i2,i3)) ) / dz         Update
             Apply C-PML on dp_dx,dp_dy and dp_dz                                                      vx,
             vx(i1,i2,i3) = vx(i1,i2,i3) + bu3loc(i1,i2,i3) * dp_dx                                    vy, vz
             vy(i1,i2,i3) = vy(i1,i2,i3) + bu2loc(i1,i2,i3) * dp_dy
             vz(i1,i2,i3) = vz(i1,i2,i3) + bu1loc(i1,i2,i3) * dp_dz
           End do
         End do
       End do
End do
Write  snapshots and seismograms to a file
```

Figure 8: Outline of the Code

4.2 Algorithm

5 Numerical Implementation and Results

Sources are explosions. The source wavelet is a Ricker wavelet or the primitive of a Ricker wavelet. *A posteriori* convolution with a band-limited wavelet will provide dispersion-free seismograms if the bandwidth of the wavelet is consistent with the grid interval (5 grid points per wavelength). Absorbing boundary conditions are Perfectly Matched Layers. The original FD grids are augmented with layers of *npml* points along the 6 surface edges of the 3D finite difference grid. A non-split formulation of PML is used (Komatitsch and Martin, 2007) wherein 20 pml points each were added to augment the six faces of the domain. Spatial differential operators are discretized with fourth-order accurate staggered-grid stencils. A grid interval of at least 5 grid points per minimum wavelength is required to compute seismograms with an acceptable amount of numerical dispersion. Temporal differential operators are discretized with second-order accurate staggered-grid stencil. Stable simulations are obtained when using $\Delta t < 0.3h/c_{max}$ where h is the spatial grid interval, c_{max} is the maximum velocity in the model and Δt is the time interval.

5.1 Validation with the Analytical Solution in Homogeneous Medium

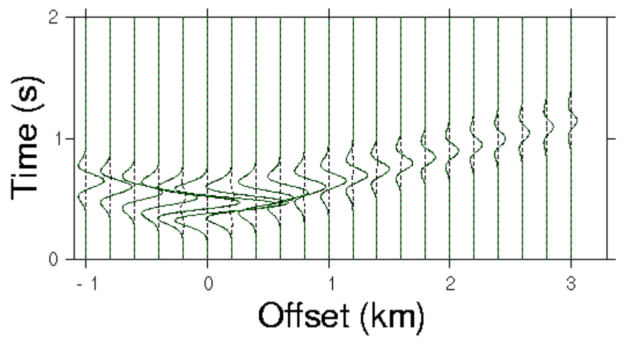
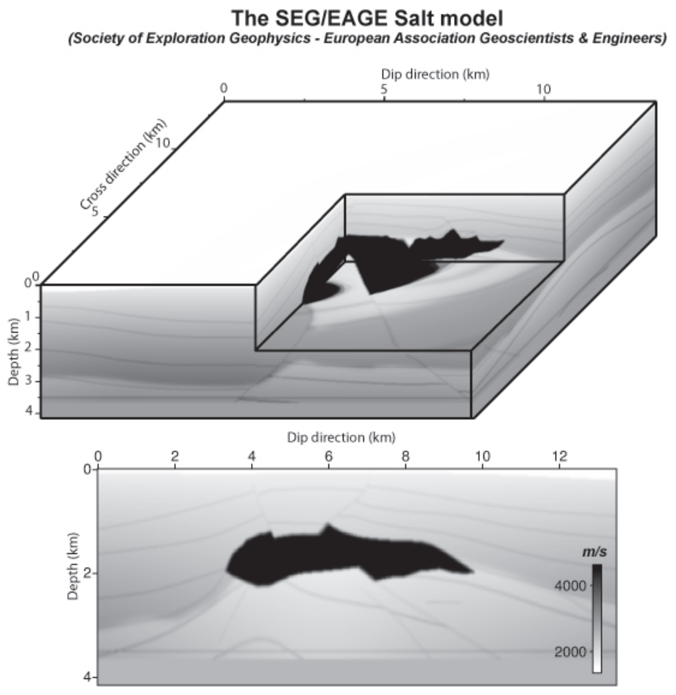


Figure 9: Comparison of Seismograms from Analytic and Numerical Solution in Homogeneous Medium

The figure illustrates the comparison of the numerical solution (solid black seismograms) with the analytical solution(dotted green seismograms). One can see that the residuals (dotted black lines) are just slightly perturbed in some seismograms, meaning that the numerical solution corresponds well to the analytical one.

5.2 Simulation on Realistic Model



The 3D SEG/EAGE salt model is used to test the finite difference algortihm on a realistic model. The salt body (black portion) shows steep flanks and a rough surface on the top which makes it a good test-case for accurate 3-D imaging methods.

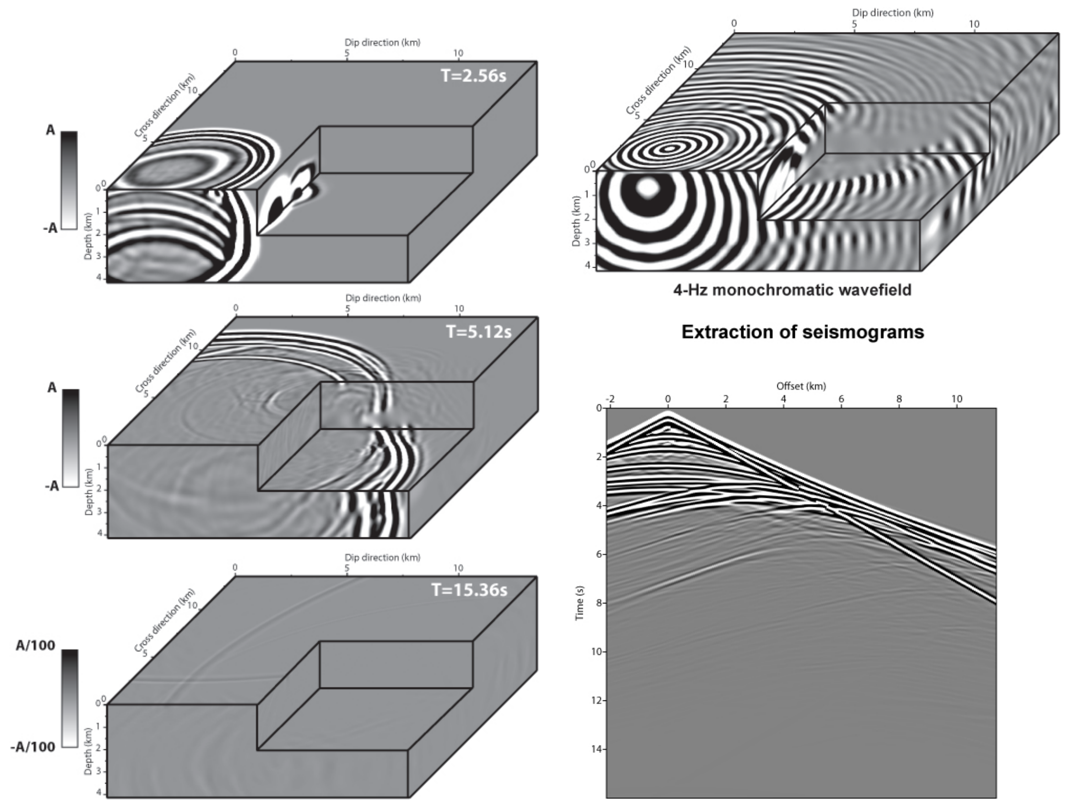


Figure 10: Snapshots of the monochromatic wavefield at different times, the 4-Hz monochromatic wavefield and seismogram acquired from a receiver line passing through the source in the dip direction

The figures on the left column show snapshots of the monochromatic pressure wavefield at subsequent times. The monochromatic range represents the amplitude of the wave. The second snapshot clearly shows how the different layers of the subsurface perturb the propagation of the acoustic wave within the medium. The range is then scaled by a factor of 1/100 to qualitatively assess the performance of the PML. Without this scaling factor, nothing would be seen since at this time, the waves have already propagated beyond the domain. This validates the efficiency of the PML. The efficiency of the PML may also be seen from the extracted seismograms wherein no artificial spurious reflections can be seen from the boundary.

5.3 Scalability

The key issue in the parallel processing of a single application is the speedup achieved, especially its dependence on the number of processors used. The speedup (S) is defined as the factor by which the execution time for the application changes: that is,

$$S = \frac{T_{seq}}{T_N}$$

where T_{seq} is the sequential execution time for one processor while T_N is the execution time for N processors.

However, there are some applications that cannot be ran sequentially. In this case, $T_{N_{min}}$ is then used instead of T_{seq} , where $T_{N_{min}}$ is the execution time for the least number of processors (in this case, N_{min}=4) needed to implement the application in parallel.

Another issue in parallel processing of a single application is its efficiency (E) expressed as:

$$E = \frac{T_{N_{min}} \times N_{min}}{T_N \times N}$$

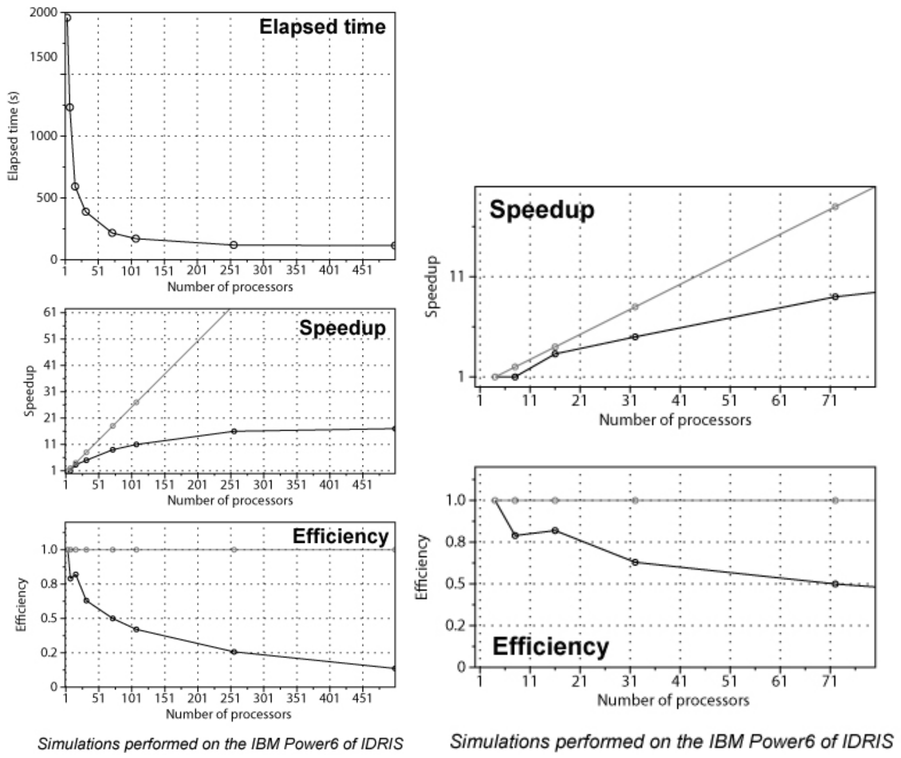


Figure 11: **Left:**Graph of Scalability and Efficiency Test on the IBM Power6 of IDRIS, **Right:**Zoom-in view of the same graphs

Number of Processors	nd1×nd2×nd3
4	1×2×2
8	2×2×2
16	1×4×4
32	2×4×4
72	2×6×6
108	3×6×6
256	4×8×8
500	5×10×10

The above figures are the result of a scalability and efficiency analysis of the parallel implementation of the code. The elapsed time is measured starting from the first iteration of the loop in time until the end of the loop in time (see section **4.2** for the Algorithm). By assessing the graphs, the code does not seem to scale very well. The explanation for this is not yet clear, however it is possible that the way the domain is decomposed is one of the factors which is summarized on the table given above.

6 Conclusions and Remarks

With the results above, the following steps have been achieved:

- validated the accuracy of a finite difference time-domain code for three dimensional acoustic wave propagation in homogeneous medium
- validated the efficiency of the absorbing boundary condition implemented in the code which is the non-split formulation of the Convolutional Perfectly Matched Layers (C-PML) by Komatitsch and Martin
- validated the computational efficiency of the code on a realistic example (SEG/EAGE Salt model) computed on a large-scale distributed memory platform (IBM Power6)

From the following, it is reasonable to conclude that the code is now ready to be utilized as the forward modeling engine for the three-dimensional acoustic Full Waveform Inversion code.

Pespectives

As mentioned before, the acoustic code is implemented with the aim of being able to extend it to the elastic case. However, the staggered grid stencil of Virieiux implemented in the code does not extend easily to the elastic case. Therefore, to do the extension to the elastic case, there are three major aspects of the code to be modified, the stencil (adapting the rotated stencil for the preferable rotated axes of the differential operators), the equations of motion and the neighbors of a subdomain (from the original six neighbors to the complete twenty-six possible neighbors of a subdomain that is needed to fully augment a subdomain).

A possible modification to the parallel implementation of the code is by distributing the sources (which corresponds to the right hand side of the equations of motion) to the processes and running the code sequentially. However, if the number of sources is much less than the number of processors, the parallel implementation through classical domain decomposition implemented in the code is still justified.

References

[1] J. Virieux and S. Operto. Full waveform inversion: Theory, algorithms and applications. *in press*, 2009.

[2] P. Moczo, J. Kristek, and L. Halada. *The Finite-Difference Method for Seismologists: An Introduction*. Comenius University Bratislava, 2004.

[3] K. Marfurt. Accuracy of finite-difference and finite-elements modeling of the scalar and elastic wave equation. *Geophysics*, 49:533, 1984.

[4] F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J. Y. L'Excellent. Fwt2d: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic data- part 1 algorithm. *Computers and Geosciences*, 35:487–495, 2009.

[5] F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J. Y. L'Excellent. Fwt2d: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic data- part 2 algorithm. *Computers and Geosciences*, 35:496–514, 2009.

[6] R. G. Pratt. Frequency-domain elastic wave modelling by finite-differences: A tool for crosshole seismic imaging. *Geophysics*, 55:626, 1990.

[7] J. Virieux. P-sv wave propagation in heterogeneous media:velocity-stress finite-difference method. *Geophysics*, 51:889–901, 1986.

[8] M.A. Dablain. The application of high-order differencing to the scalar wave equation. *Geophysics*, 51:54–66, 1986.

[9] D. Komatitsch and R. Martin. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72:1259, 2007.

[10] A. Taflove. *Computational Electrodynamics: the finite-difference time-domain method.* Springer-verlag, New York, 1995.

[11] IDRIS Service Documentation. *Utilisation de MPI en dcomposition de domaine.* CNRS - IDRIS, 1996.