3D Finite Difference Time-Domain Modeling of Acoustic Wave Propagation based on Domain Decomposition

UMR Géosciences Azur

CNRS-IRD-UNSA-OCA

Villefranche-sur-mer

Supervised by: Dr. Stéphane Operto

Jade Rachele S. Garcia

July 16, 2009

Outline

1. Introduction
2. Scope and Aim of the Work
3. The Forward Problem: Seismic Wave Modeling
4. Parallel Implementation
5. Numerical Results
6. Conclusion

Introduction

Seismic Exploration

Seismic Exploration – the search for subsurface deposits of crude oil, natural gas and minerals

Objective: to form a model of the subsurface

The basic processes in seismic exploration:

- Controlled sources emit elastic waves which propagate in the subsurface
- Record the wavefield propagated by the different layers
- Process the seismic data to produce some models of the subsurface

Full Waveform Inversion

Full Waveform Inversion – a data fitting procedure that utilizes the full information contained in the seismic data to produce high resolution models of the subsurface

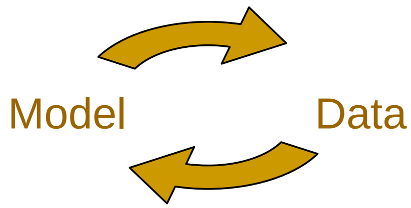
Two main ingredients: the **forward problem** and the **inverse problem**

Full Waveform Inversion

The Forward Problem

$$\frac{\omega^2}{\kappa(x,z)}P(x,z,\omega)+\frac{\partial}{\partial x}\left(\frac{1}{\rho(x,z)}\frac{\partial P(x,z,\omega)}{\partial x}\right)+\frac{\partial}{\partial z}\left(\frac{1}{\rho(x,z)}\frac{\partial P(x,z,\omega)}{\partial z}\right)=-S(x,z,\omega)$$

$$\mathbf{d}=g(\mathbf{m})$$



The Inverse Problem

$$\mathbf{m}=\mathbf{m}_0+\delta\mathbf{m}$$

$$\Delta\mathrm{d}(\mathrm{m_0})=\mathrm{d_{obs}}-\mathrm{g}(\mathrm{m_0})$$

$$\delta\mathbf{m}=\left\{\Re[\mathbf{J_0}^{\dagger}\mathbf{J_0}]+\Re[\frac{\partial\mathbf{J_0}^{\mathbf{t}}}{\partial\mathbf{m}^{\mathbf{t}}}(\boldsymbol{\Delta}\mathbf{d}^{*}...\boldsymbol{\Delta}\mathbf{d}^{*})]\right\}^{-1}\Re[\mathbf{J}_0^{\dagger}\boldsymbol{\Delta}\mathbf{d}]$$ where $$\mathbf{J_0}=\frac{\partial g(\mathbf{m_0})}{\partial\mathbf{m}}$$

Scope and Aim of the Work

- The focus of this paper : implement and validate the 3D parallel finite-difference time-domain code for acoustic wave modeling (part of the Forward Problem)

- Motivations:
  - Build a forward modeling engine in the time domain to perform 3D acoustic full-waveform inversion in the frequency domain.
  - Design an acoustic code with judicious stencil that will be easily extended to the 3D elastic case.
  - The 3D elastic code will be used:
    - 1. as forward modeling engine to perform 3D elastic full-waveform inversion
    - 2. to perform cross-validation with a Discontinuous Galerkin finite-element method developed by V. Etienne at Geosciences Azur
    - 3. Perform wave modeling for other kinds of application such as seismic hazards assessment.

The Forward Problem: Seismic Wave Propagation Modeling

The Acoustic Wave Equation: The Earth as a Fluid

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

- The acoustic wave equation describes sound waves in a liquid or gas.
- Acoustic wave equation: not very accurate for modeling wave propagation in solids but is relatively simple to solve
- Acoustic wave: essentially a pressure change. Since, fluids exhibit fewer restraints to deformation, the restoring force responsible for wave propagation is simply due to pressure change

Velocity-Stress Formulation of the Acoustic Wave Equation

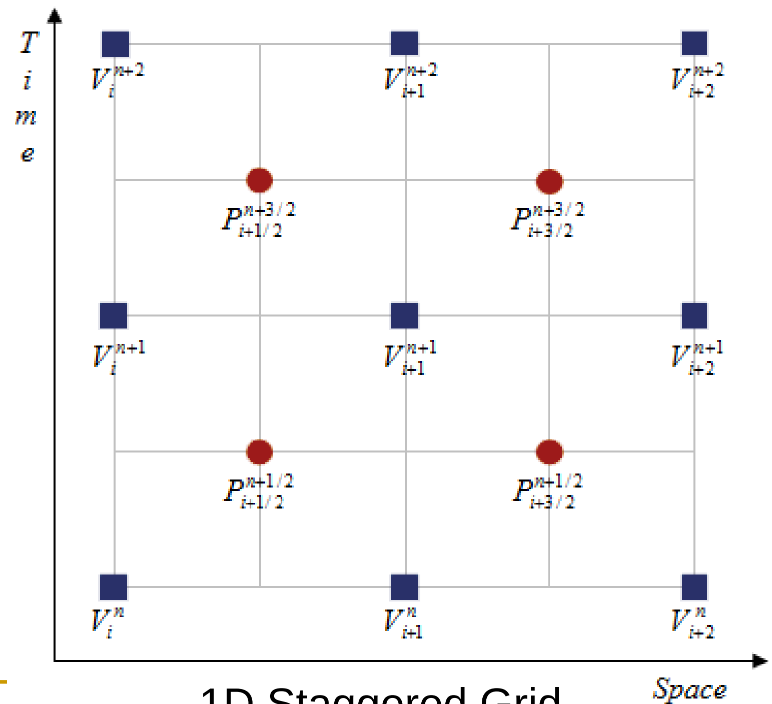
$$\frac{\partial P}{t} = \kappa \left(\frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} + \frac{\partial V_z}{\partial z} \right)$$
$$\frac{\partial V_x}{\partial t} = b \frac{\partial P}{\partial x} + f_x$$
$$\frac{\partial V_y}{\partial t} = b \frac{\partial P}{\partial y} + f_y$$
$$\frac{\partial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

- Initial conditions: P and V are zero at t=0
- Boundary conditions: Absorbing boundary conditions and free surface boundary condition

The Finite Difference Discretization of the 3D Acoustic Wave Equation

Staggered Grid Stencil

- Simple way to avoid odd-even decoupling between the pressure and the velocity.

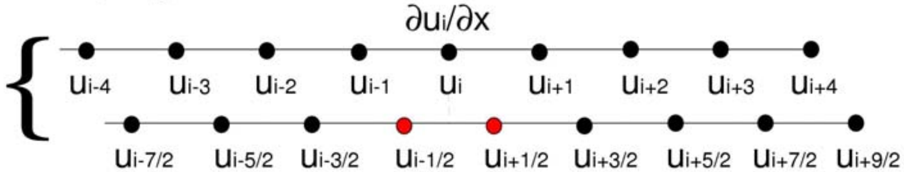


1D Staggered Grid

The Leapfrog Scheme

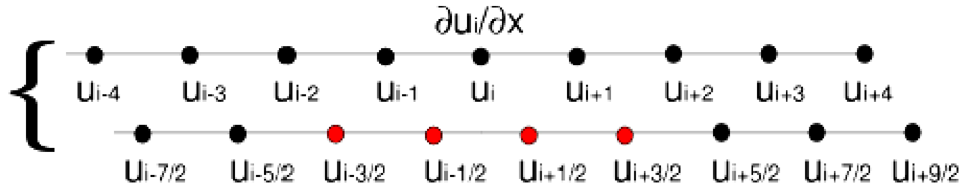
- Leapfrog scheme on staggered grids

Leapfrog second-order accurate central difference scheme



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

Leapfrog fourth-order accurate central difference scheme



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

The Discretized 3D Acoustic Wave Equation

- Using the 2nd order discretization for time and 4th order discretization for space in a staggered grid leads to:

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

Numerical Dispersion and Stability

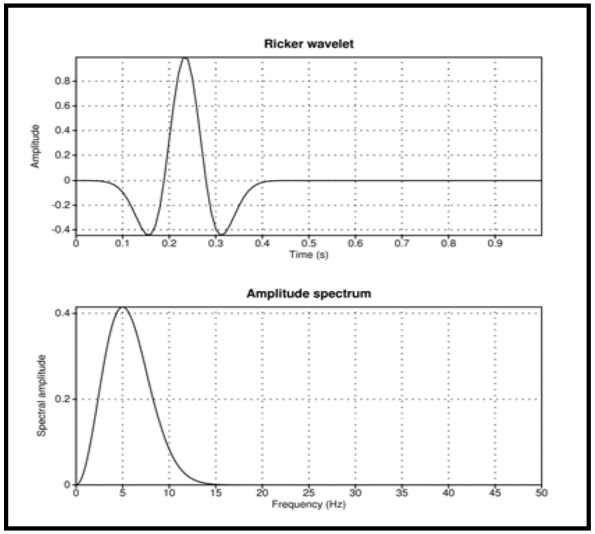
Numerical Dispersion

- variation of the numerical phase velocity as a function of frequency

Occurs if:

- Grid spacing is large
- Wavelength of the source is too short compared with the size of the grid

Numerical Dispersion



For a Ricker wavelet, the rule of the thumb given below is an effective criterion for nondispersive propagation

$$\Delta x = \frac{\lambda}{n}$$

Period (s) s: T
Frequency (Hz) Hz: f:
Angular frequency: $\omega = \frac{2\pi}{f}$

Wavelength (sparial period) in meter: $\lambda = \frac{c}{f}$
Wavenumber (spatial frequency) in $rad.m^{-1}$: $k = \frac{2\pi}{\lambda} = \frac{\omega}{c}$

n is the number of gridpoints per wavelength. For a 4^{th} order accurate scheme, it has been established to be 5-8 gridpoints per wavelength.

Numerical Stability

- **Numerical instability** – an undesirable property that may occur in explicit time-marching schemes, when the computed result spuriously increases without limit in time

- A stability condition for the time step is the Courant-Friedrichs-Levy (CFL) condition. For the scheme used here,

$$\Delta t = \xi \frac{\Delta x}{c_{\max}}, \text{ where } \xi = 0.48$$

An Illustration through the 1D Case

The 1D Scalar Wave Equation in Homogeneous Medium

- To illustrate the numerical analysis involved in finite difference discretization, we start with the simplest case, the one-dimensional homogeneous scalar wave equation

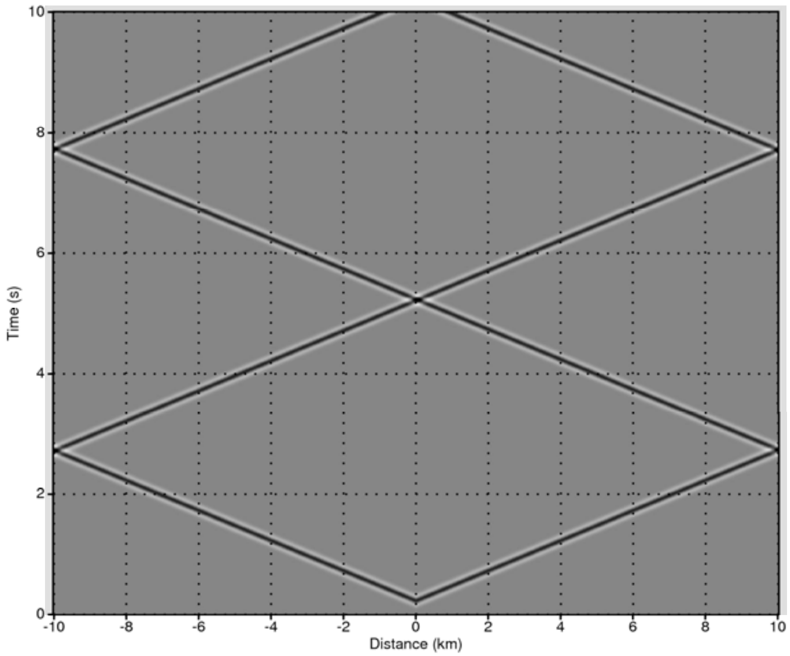
$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

- A fully explicit second-order accurate finite difference approximation of the wave equation

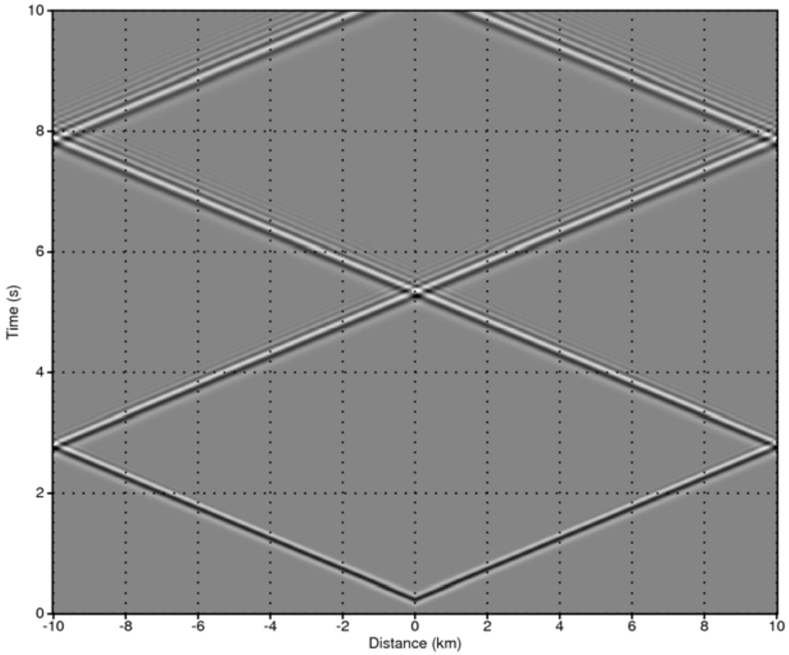
$$u_i^{n+1} = (c\Delta t)^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \right] + 2u_i^n - u_i^{n-1} + O\left((\Delta t)^2\right) + O\left((\Delta x)^2\right)$$

Example of Dispersionless and Dispersive Wave Propagation

Dispersionless case, S=1



Dispersive case, S=0.5



Velocity Field for Free Surface Boundary condition, *c* = 4000m/s

Simulation of an Unbounded Medium in 1D

- Radiation condition

- Sponge boundary condition

Radiation Condition

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, $(\tau = E(x)\frac{\partial u}{\partial x})$,

$$v(x, t) = -cf'(x - ct)$$
$$\tau(x, t) = E(x)f'(x - ct)$$

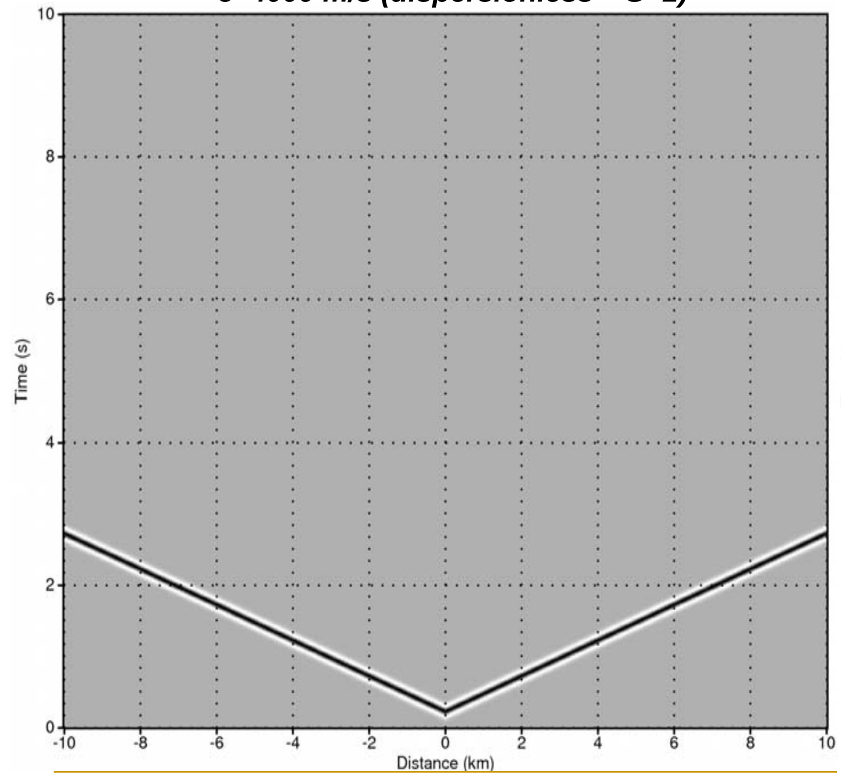
which leads to the radiation condition on the right edge:
$\tau(L, t) = -Z(l)v(L, t)$, where $Z = \rho c$ which is called the impedance.

Left edge

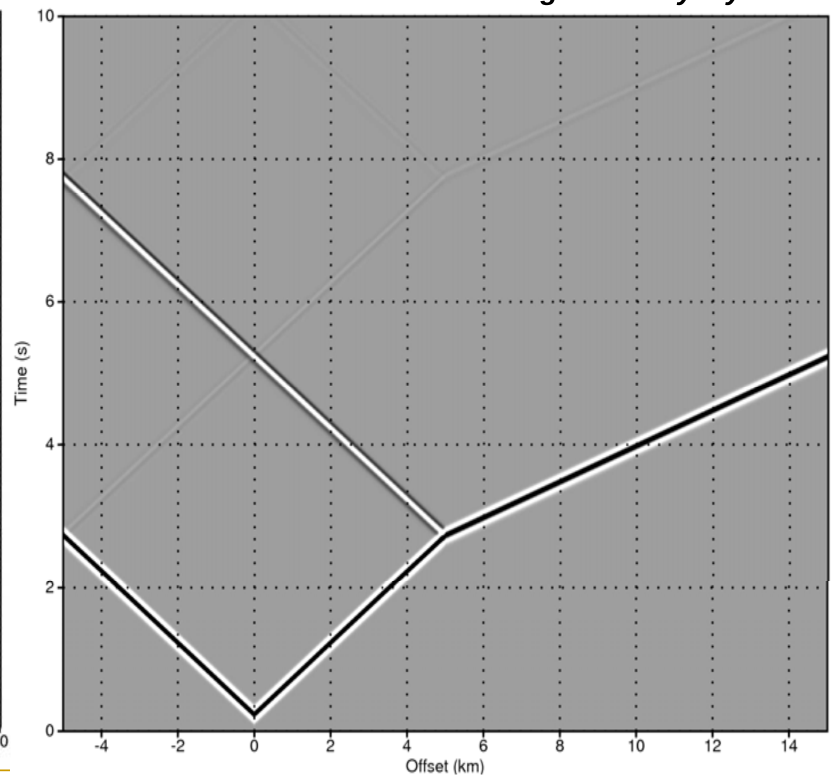
On the right edge, the wavefield must satisfy: $u(x, t) = f(x + ct)$, which leads to the radiation condition: $\tau(0, t) = Z(0)v(0, t)$.

Radiation Condition

Simulation in homogeneous medium
c=4000 m/s (dispersionless – S=1)



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

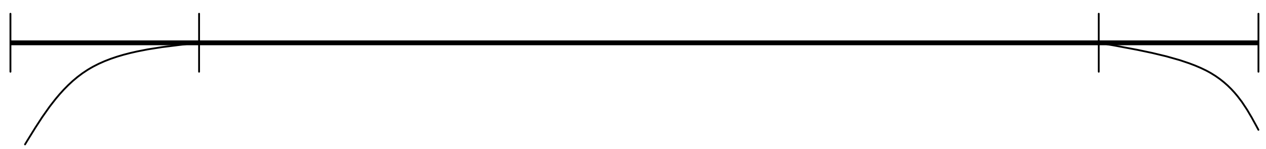


Sponge Boundary Condition

The sponge boundary condition captures the basic idea of behind the PML. Basically, the computational domain is augment with one absorbing (sponge) layer at each ends of the model.
The modified 1D wave equation, with an additional damping term is introduced

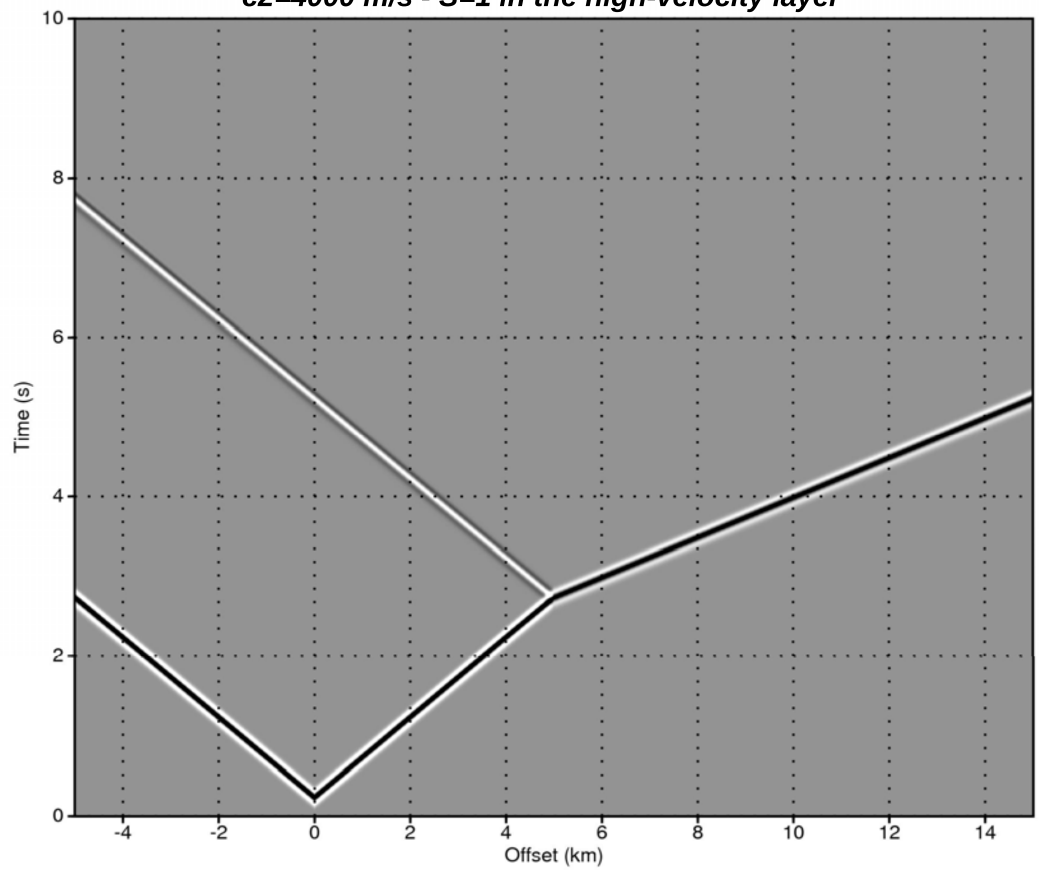
$$\begin{cases} \frac{\partial v(x,t)}{\partial t} + \gamma(x) v(x,t) = I(x) \frac{\partial \tau(x,t)}{\partial x} \\ \frac{\partial \tau(x,t)}{\partial t} + \gamma(x) \tau(x,t) = E(x) \frac{\partial v(x,t)}{\partial x} \end{cases}$$

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



Sponge Boundary Condition

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



Parallel Implementation

Methodology

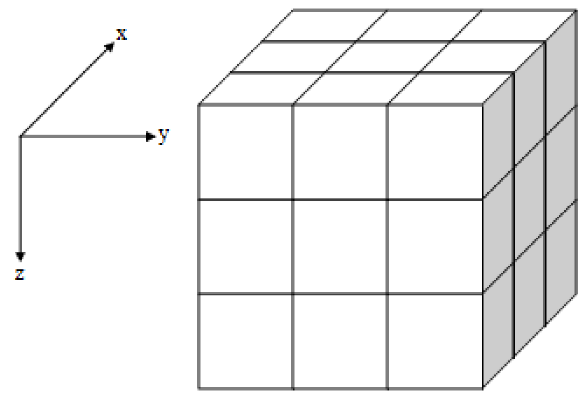
- A parallel version of the general algorithm based on the principle of domain decomposition for structured meshes is as follows:
    - Decompose the mesh into subdomains and assign each subdomain to a process
    - Determine the neighbors of each subdomain
    - Iterate time
    - Exchange messages among interfaces
    - calculate

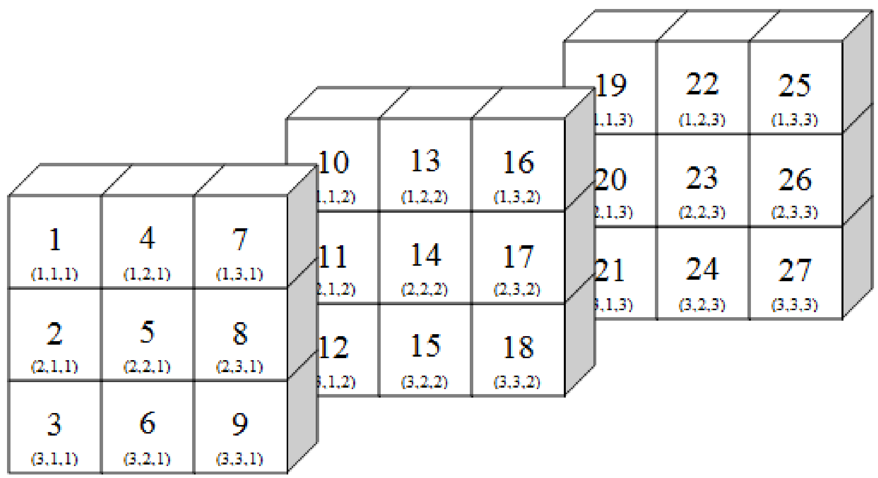
Subroutines

- SUBROUTINE init
- SUBROUTINE voisinage
- SUBROUTINE typage
- SUBROUTINE communication

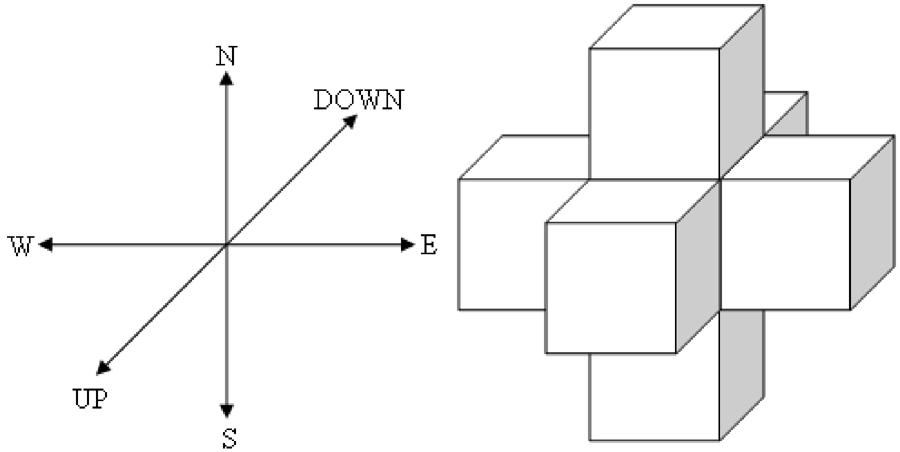
SUBROUTINE init

This procedure init executes the decomposition of the original domain into subdomains and the initialization of MPI





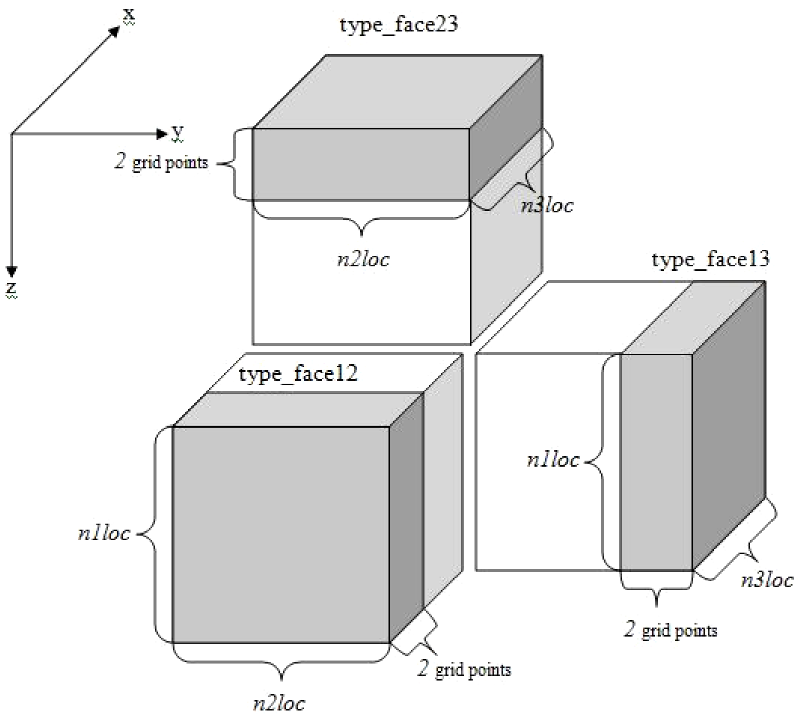
SUBROUTINE voisinage



This procedure determines the existing neighbors of a subdomain and which process they correspond to

SUBROUTINE typage

- This procedure defines the data blocks to be send in sending and receiving messages



SUBROUTINE typage

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(number of blocks, number of elements in each block,number of elements between the start of each block, old type, new type)
-- allows replication of a datatype into locations that consist of equally spaced blocks.

MPI TYPE HVECTOR
-- almost the same as MPI TYPE VECTOR except that the stride (3rd parameter) between the start of each block is in bytes instead of the number of elements

SUBROUTINE communication

- This procedure is done within the loop in time

- Its purpose is to send data blocks from the subdomain to the corresponding neighboring areas and to receive the same points in the relevant fields

- Use MPI_SENDRECV(initial address of sending, number of elements to be sent, type of elements to send, destination, initial address of reception, number of elements to receive, source)

SUBROUTINE communication

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

x(-1:n1loc+2,-1:n2loc+2,-1:n3loc+2)

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

Algorithm

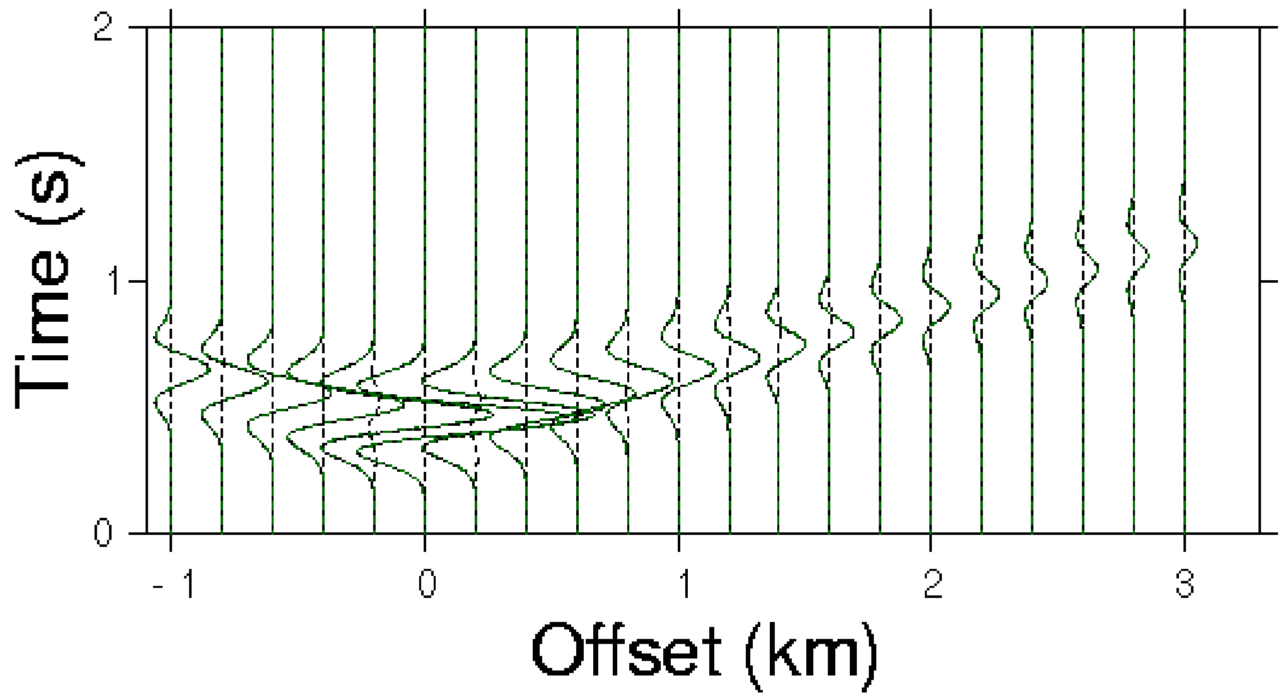
```
Intialize MPI
Read input file
Compute source wavelet
Decompose the domain and define data types for interface
Initialize p, vx, vy and vz to zero (Initial conditions)
Build the parameters b and kappa and C-PML parameters on each subdomain
Locate corresponding subdomain location of source, receivers 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
             dvz_dz = ( a0*(vz(i1,i2,i3)-vz(i1-1,i2,i3)) + a1*(vz(i1+1,i2,i3)-vz(i1-2,i2,i3)) ) / dz
             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
             Apply C-PML on dp_dx,dp_dy and dp_dz
             vx(i1,i2,i3) = vx(i1,i2,i3) + bu3loc(i1,i2,i3) * dp_dx
             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
```

Update p

Update vx, vy, vz

Numerical Results

Comparison with Analytical Solution in Homogeneous Medium

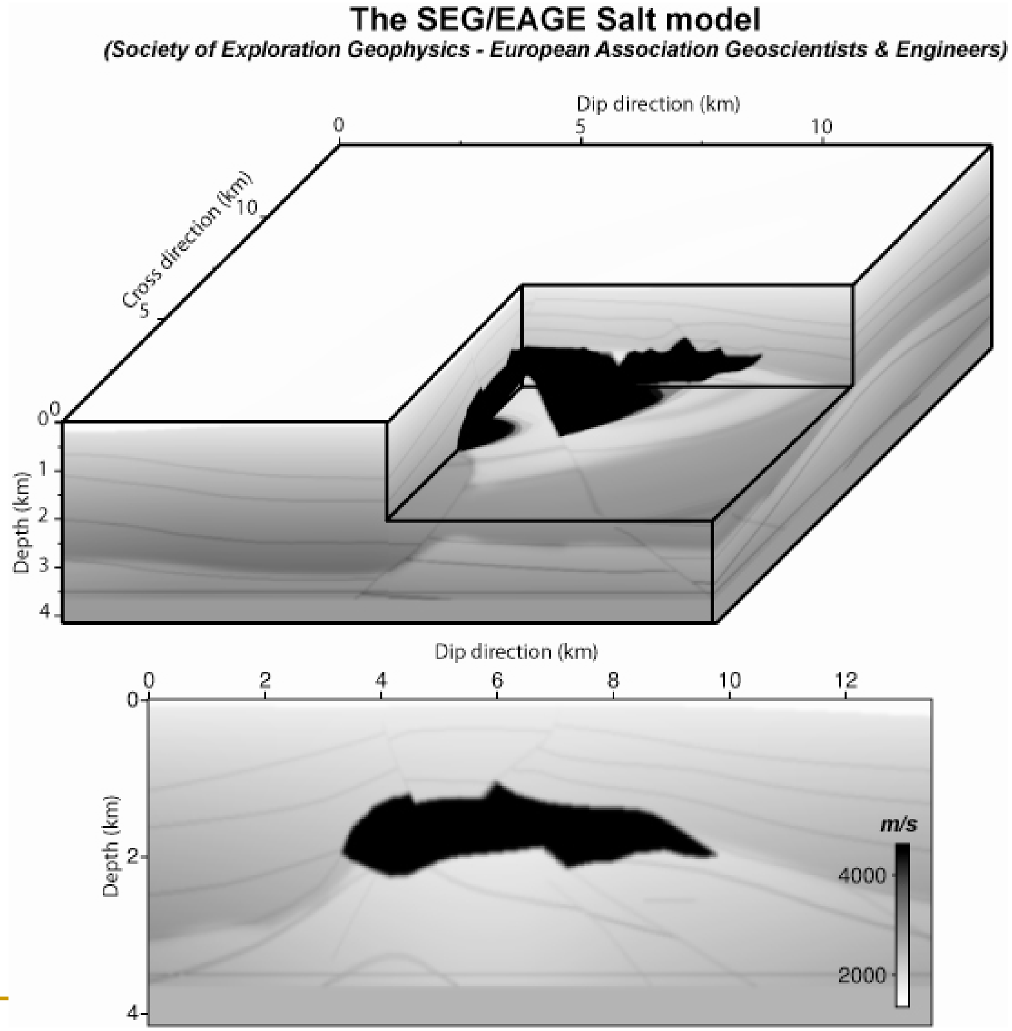


Simulation on a Realistic Model

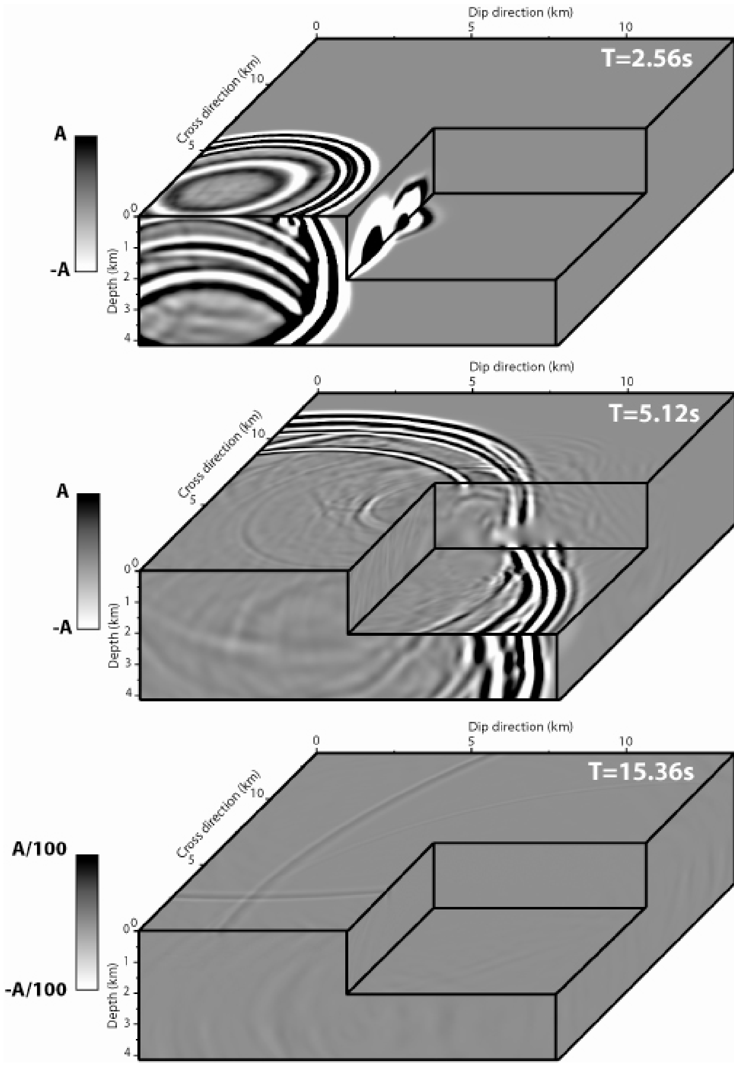
- Number of unknowns: 139x450x450
  = 28,147,500

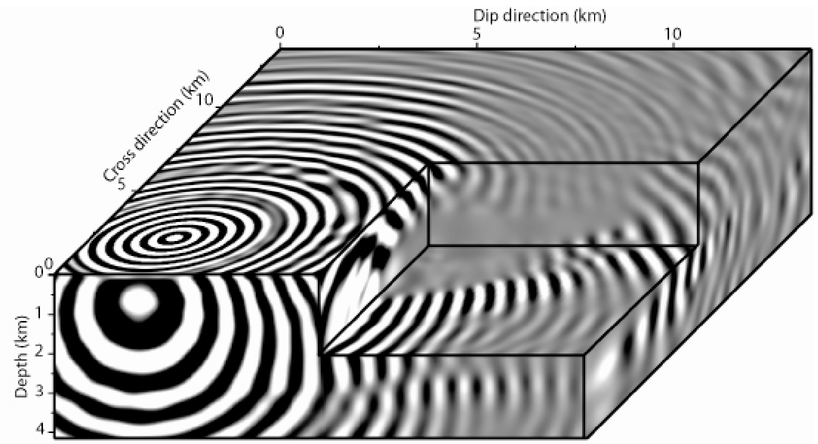
- Number of time steps=3000

- Time step =0.0032



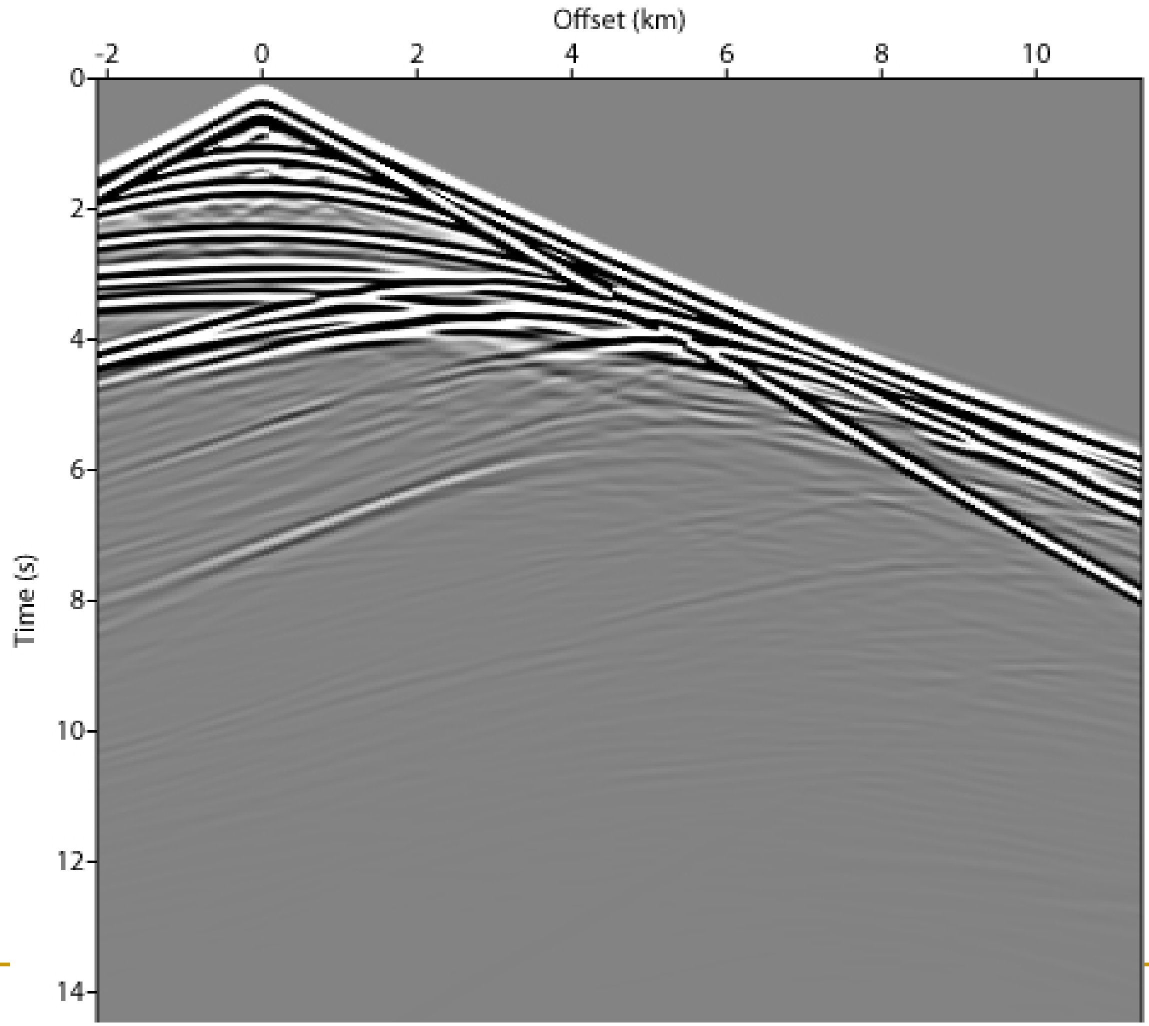
Simulation on a Realistic Model





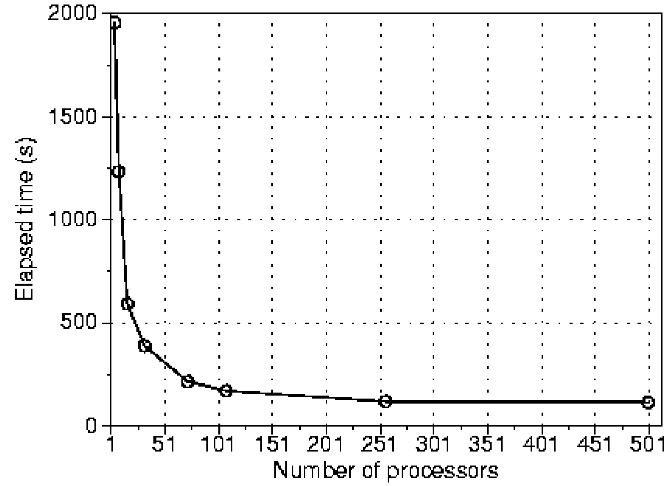
4-Hz monochromatic wavefield

Extraction of Seismograms



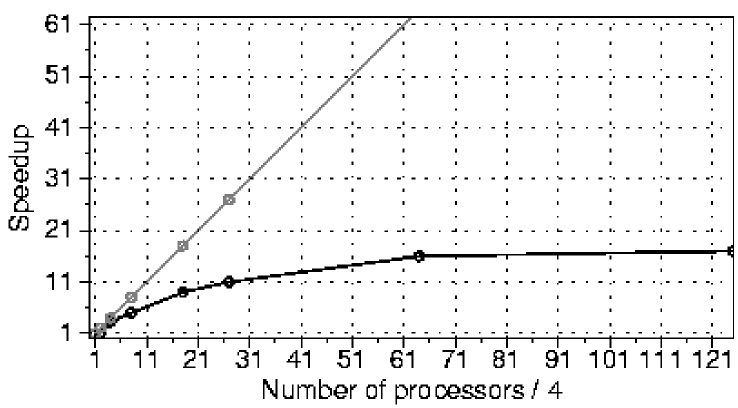
Scalability and Efficiency Analysis

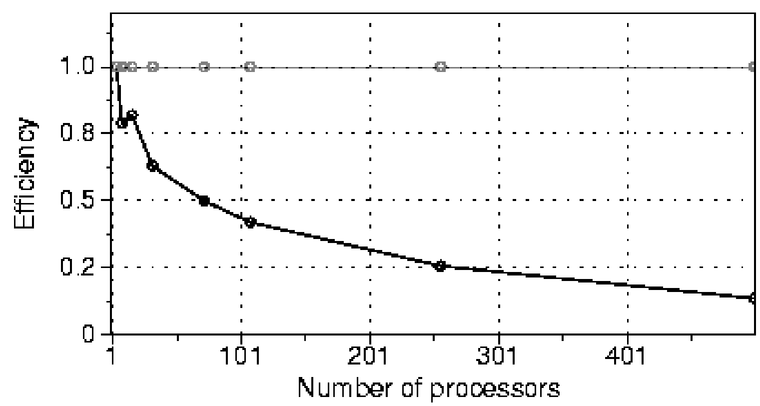
Simulations performed on the IBMPower6 of IDRIS



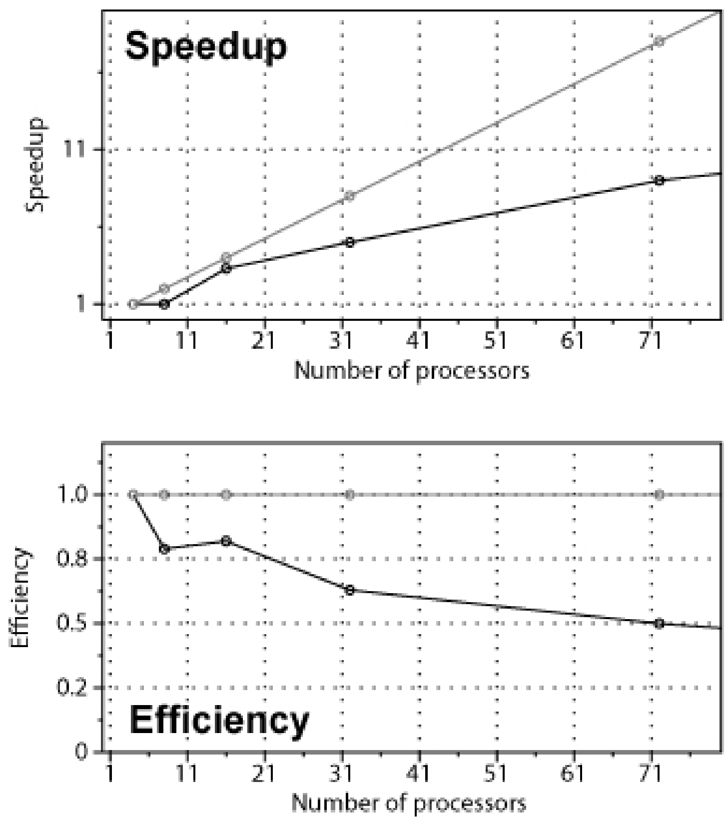
$$Speedup = \frac{T_{N_{\min}}}{T_N}, \quad \text{where } N_{\min} = 4$$

$$Efficiency = \frac{T_{N_{\min}} \times 4}{T_N \times N}$$





Scalability and Efficiency



Simulations performed on the IBM Power6 of IDRIS

Conclusions

Conclusion

- Validate the accuracy of the FDTD code
- Validate the efficiency of the absorbing boundary condition: C-PML
- Validate the computational efficiency of the code on realistic example computed on a large-scale distributed memory platform

Conclusion: we have a modeling engine which is ready to be implemented in a 3D acoustic Full Waveform Inversion code

Perspectives

- Extension to the elastic through rotated stencil
- Implementation of the FDTD code in FWI which can be viewed in two levels of parallelism:
  - Perform modeling in sequential and distribute the sources (rhs) over processors
  - Classical domain decomposition of the number of sources are much less than the number of processors

Thank you for listening!

Special thanks to Prof. Operto.