

Parallelization Strategy for Wavefield Simulation with the Elastic Iterative Solver: Application to FWI

Mikhail Belonosov, *Vladimir Cheverda*, Victor Kostin, Dmitry Neklyudov and Sergey Soloviev

Institute of Petroleum Geology and Geophysics SB RAS &
Aramco Research Center - Delft, Aramco Overseas Company B.V.
Novosibirsk, Russia & Delft, the Netherlands

Contents

- Motivation
- Equations and parameters
- Parallelization
- Examples
 - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Contents

- **Motivation**
- Equations and parameters
- Parallelization
- Examples
 - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Full Waveform Inversion algorithm

Misfit functional

$$E(m) = \frac{1}{2} \|d^{obs} - d^{cal}(m)\|_D^2$$

d^{obs} - the receivers data;

m - the current model parameters;

$d^{cal}(m)$ - wavefield computed in receivers for the current model; in this talk we deal with frequency time domain 3D isotropic elastic media.

*Virieux, J. and Operto, S. An overview of full-waveform inversion in exploration geophysics, Geophysics, 74 (6), WCC1-WCC26.

In Frequency Domain, the misfit functional gradient (the case of scalar wave equation):

$$\nabla E(m) = -2 \operatorname{Re} \int_{\omega_1}^{\omega_2} \omega^2 U(x, y, z; \omega) \overline{W}(x, y, z; \omega) d\omega$$

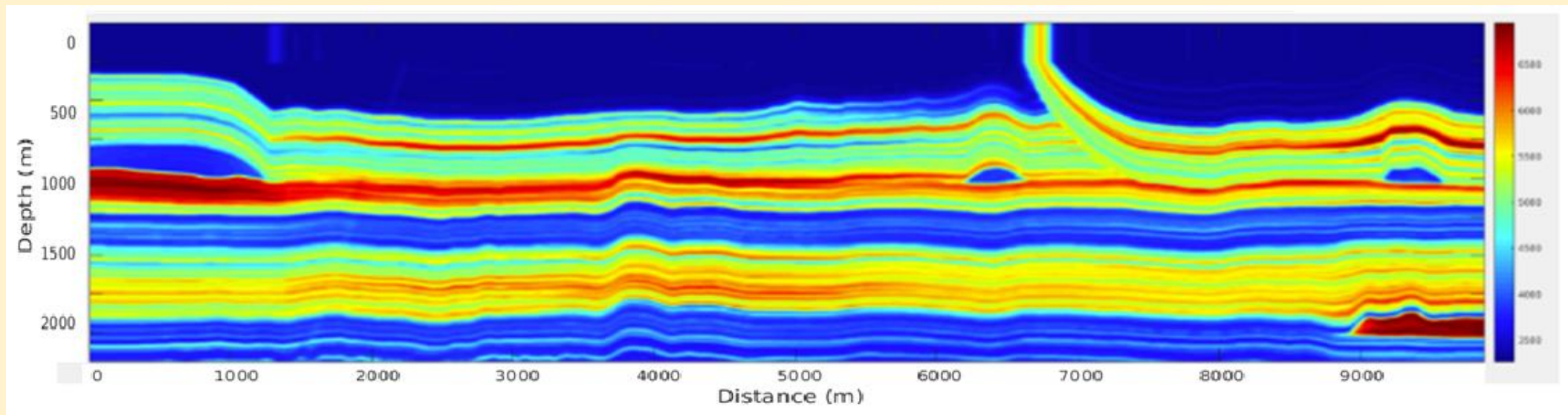
$U(x, y, z; \omega)$ - wavefield computed for the current model for a specific source position

$W(x, y, z; \omega)$ - wavefield for the current model for the registered wavefield taken as sources

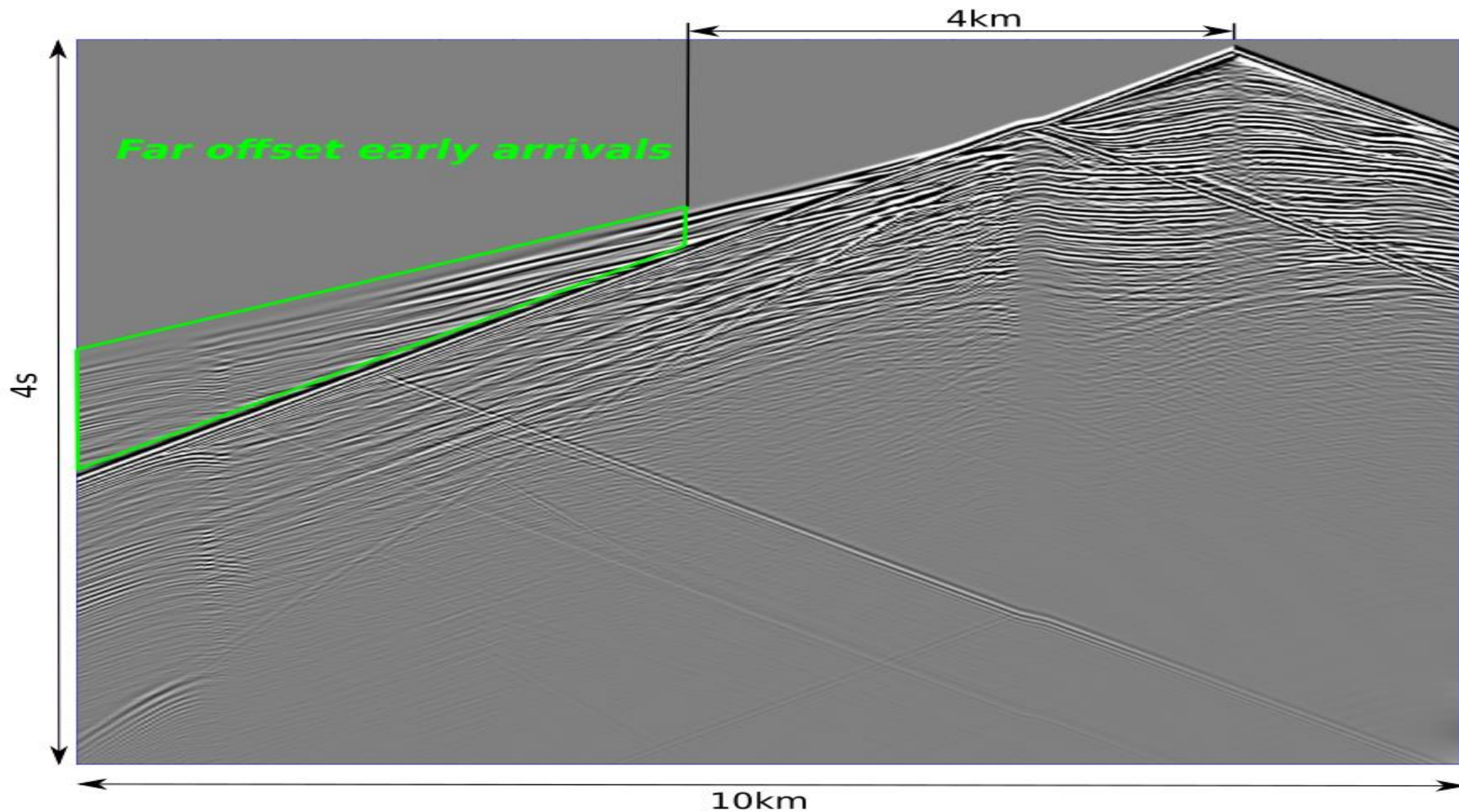
In a case of several sources the gradients to sum up.

Модель

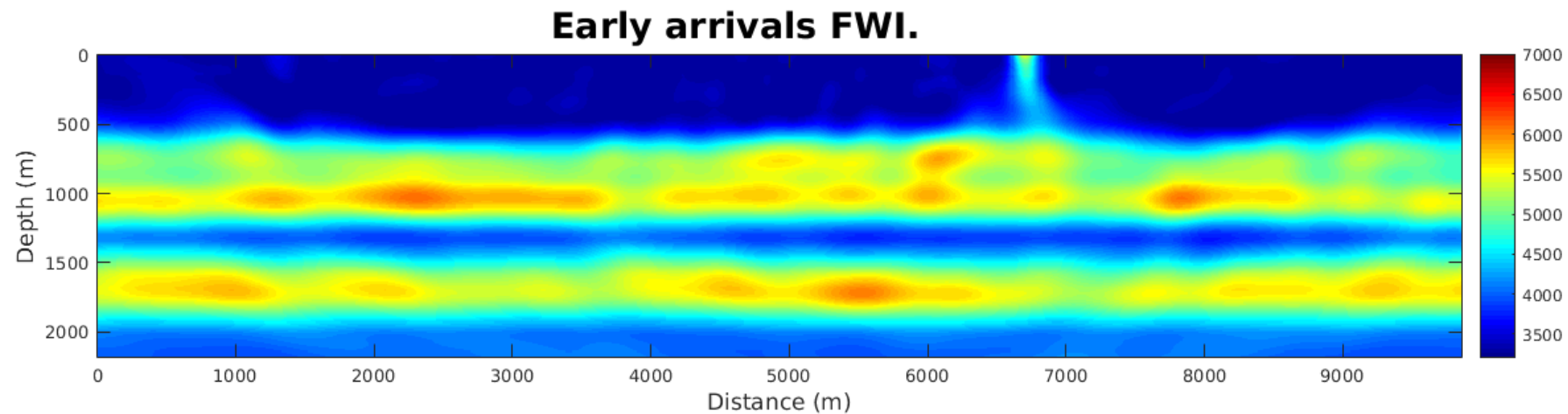
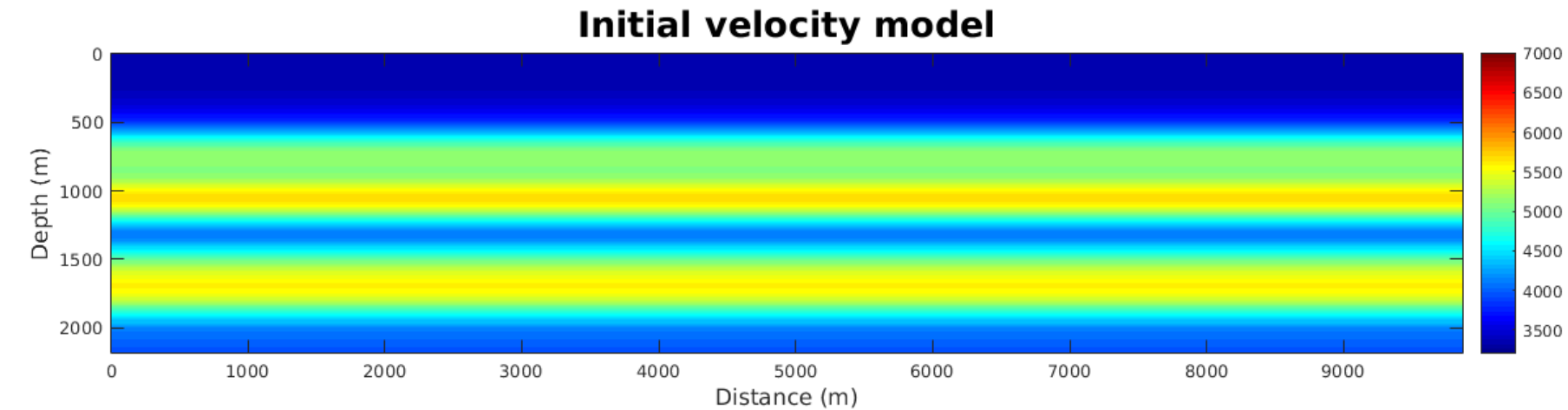
В наших численных экспериментах использовалась следующая модель:



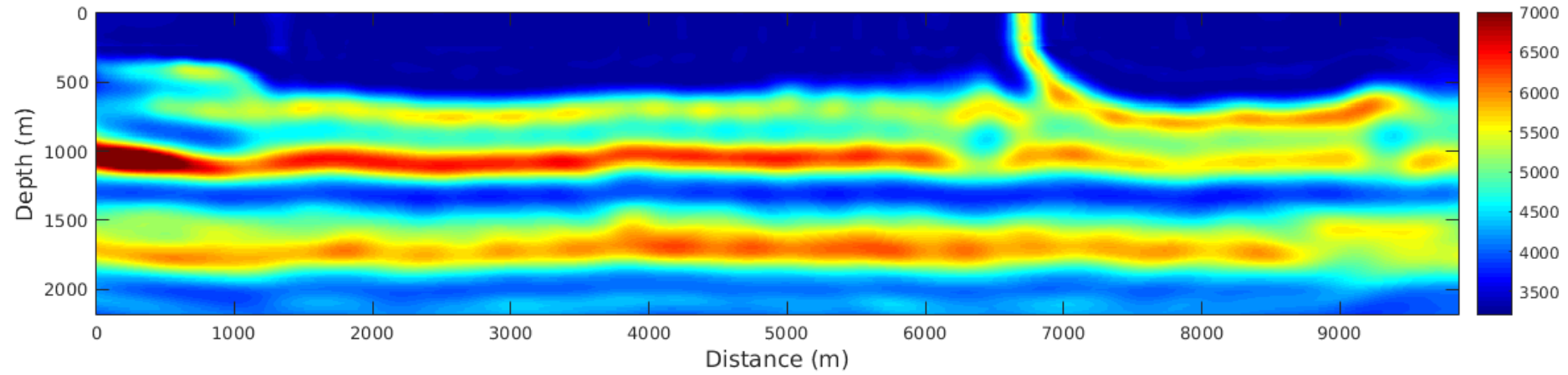
Дальние выносы: препроцессинг



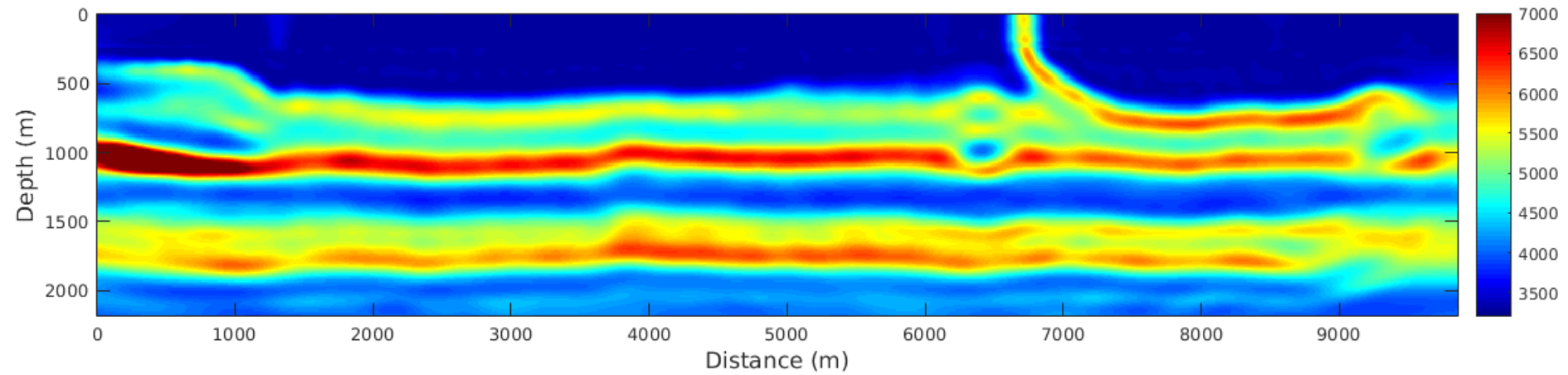
Дальние выносы: обращение



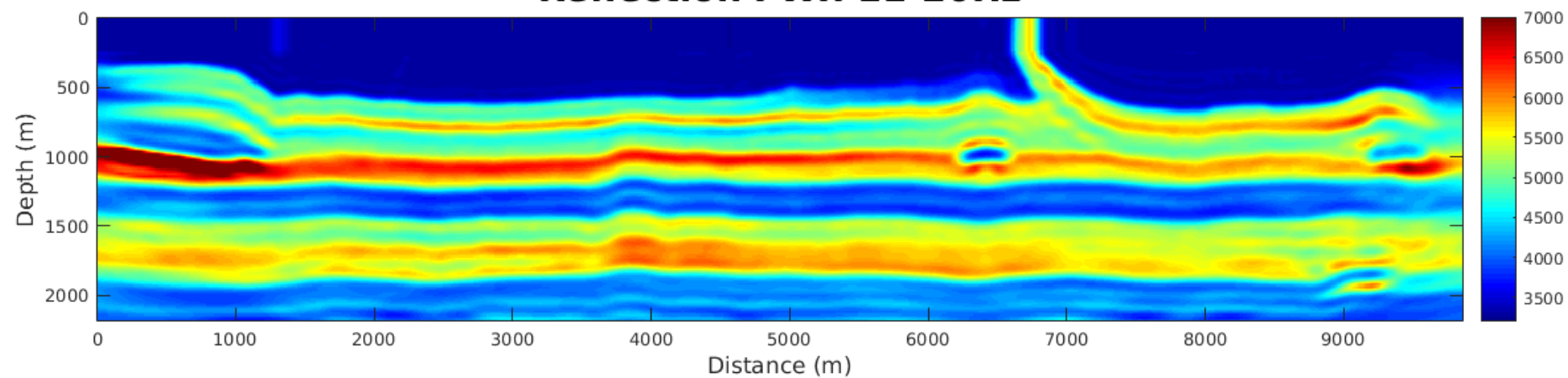
Reflection FWI. 5-7Hz



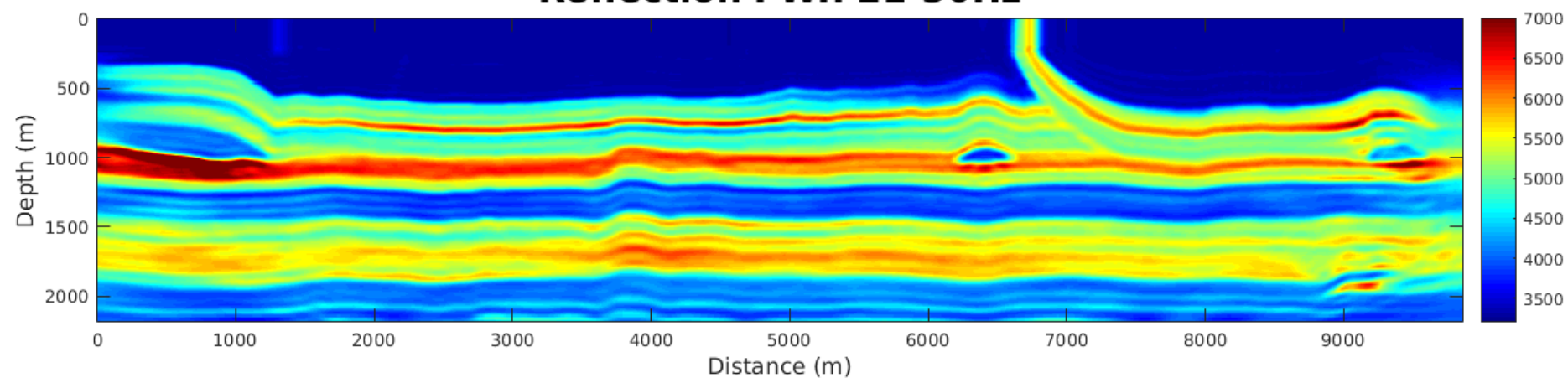
Reflection FWI. 8-10Hz



Reflection FWI. 11-20Hz



Reflection FWI. 21-30Hz



Contents

- Motivation
- Equations and parameters
- Parallelization
- Examples
 - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

- Needed a method for effective computing wavefields
- Input: model parameters, source/receivers positions, a set of frequencies
- Output: the wavefield in the target domain

Equations and parameters

We use elastic wave propagation equation for isotropic 3D media

$$\mathbf{L} \equiv \left[i\omega \begin{pmatrix} \rho \mathbf{I}_{3 \times 3} & 0 \\ 0 & \mathbf{S}_{6 \times 6} \end{pmatrix} - \begin{pmatrix} 0 & \hat{P} \\ \hat{P}^T & 0 \end{pmatrix} \frac{\partial}{\partial x} - \begin{pmatrix} 0 & \hat{Q} \\ \hat{Q}^T & 0 \end{pmatrix} \frac{\partial}{\partial y} - \gamma(z) \begin{pmatrix} 0 & \hat{R} \\ \hat{R}^T & 0 \end{pmatrix} \frac{\partial}{\partial z} \right] \mathbf{v} = \mathbf{f}.$$

where vector of unknowns $\mathbf{v}$ comprises nine components. These components include the displacement velocities (v_x, v_y, v_z) and components of the stress tensor $(\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{yz}, \sigma_{xz}, \sigma_{xy})$. ω is the real time frequency, $\rho(x, y, z)$ is the density, $\mathbf{I}_{3 \times 3}$ is 3 by 3 identity matrix, $\mathbf{S}_{6 \times 6}(x, y, z) = \begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix}$ is 6 by 6 compliance matrix and

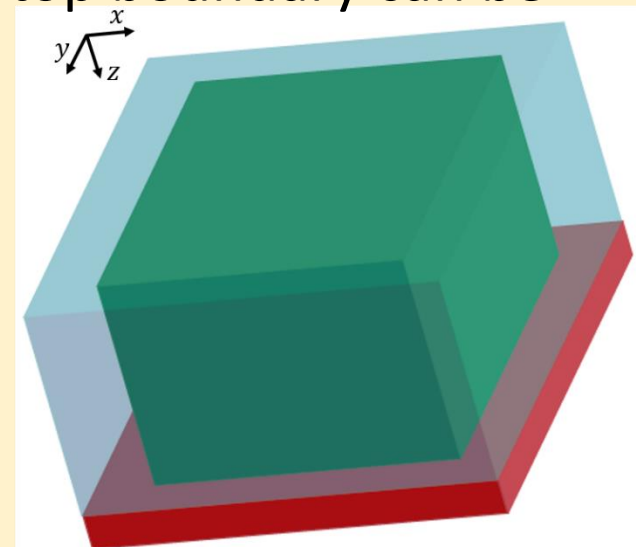
$$A = \begin{pmatrix} a & -b & -b \\ -b & a & -b \\ -b & -b & a \end{pmatrix}, C = \begin{pmatrix} c & 0 & 0 \\ 0 & c & 0 \\ 0 & 0 & c \end{pmatrix}, \hat{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}, \hat{Q} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Equations and parameters

Coefficients $a(x, y, z)$, $b(x, y, z)$ and $c(x, y, z)$ are related to the Lamé parameters λ and μ as follows $a = \frac{\lambda + \mu}{\mu(2\mu + 3\lambda)}$, $b = \frac{\lambda}{2\mu(2\mu + 3\lambda)}$, $c = \mu^{-1}$. $\mathbf{f}$ is the right-hand side representing the seismic source. In our experiments, we consider either a volumetric or vertical point-force source. $\gamma(z)$ may be either unity or the damping along z representing the Perfectly Matched Layer (PML)

Computational domain is a cuboid of $N_x \times N_y \times N_z$ points. This domain includes sponge layers [5] on the horizontal and PML on the vertical boundaries (top and bottom) imitating an elastic radiation condition at infinity. The top boundary can be also the free surface.

Computational domain (green), sponge layers (blue), PML (red)



Equations and parameters: preconditioning

Let L_0 be the same operator as L , but with

$$\rho(x, y, z) = \rho_0(z), \mathbf{S}_{6 \times 6}(x, y, z) = (1 + i\beta) \cdot \mathbf{S}_0(z)$$

where

$$\mathbf{S}_0(z) = \begin{pmatrix} A_0 & 0 \\ 0 & C_0 \end{pmatrix}, A_0 = \begin{pmatrix} a_0 & -b_0 & -b_0 \\ -b_0 & a_0 & -b_0 \\ -b_0 & -b_0 & a_0 \end{pmatrix}, C_0 = \begin{pmatrix} c_0 & 0 & 0 \\ 0 & c_0 & 0 \\ 0 & 0 & c_0 \end{pmatrix}$$

We use operator L_0 as the preconditioner and search for solution $\boldsymbol{v}$ of the original boundary value problem by solving the 2nd kind Fredholm integral equation

$$LL_0^{-1}\tilde{\boldsymbol{v}} = \boldsymbol{f} \quad (4)$$

with the same boundary conditions as for equation (1). Finally, we compute unknown $\boldsymbol{v}$ by formula $\boldsymbol{v} = L_0^{-1}\tilde{\boldsymbol{v}}$. Denoting $\delta L = L - L_0$ and substituting it into equation (4) we arrive at

$$(I - \delta LL_0^{-1})\tilde{\boldsymbol{v}} = \boldsymbol{f}, \quad (5)$$

where δL is the zero-order operator – pointwise multiplication by a matrix. This is valid because we consider equation (1) with the compliance matrix.

We solve equation (5) via a Krylov-type iterative method. From the variety of them, we choose the biconjugate gradient stabilized method (BiCGSTAB) [25] because of its moderate memory requirements.

Equations and parameters: preconditioning

This assumes computing several times per iteration (depending on a method) the product of the left-hand side operator of equation (5) by a particular vector w , i.e. computing $[w - \delta L L_0^{-1} w]$. This process breaks down into three computational steps

1. first, computing $q_1 = L_0^{-1} w$ by solving boundary value problem $L_0 q_1 = w$;
2. then, computing $q_2 = \delta L q_1$, that in the discrete case is a pointwise multiplication of a tridiagonal matrix by a vector;
3. finally, subtracting the two vectors $[w - q_2]$.

Equations and parameters: preconditioning

To solve $L_0 q_1 = w$ we assume that function $w(x, y, z)$ is expanded into a Fourier series with respect to the horizontal coordinates with coefficients $\hat{w}(k_x, k_y, z)$, where k_x and k_y are the respective spatial frequencies. These coefficients are solutions to the boundary value problems for ordinary differential equations (ODEs)

$$\left[i\omega \begin{pmatrix} \rho_0 \mathbf{I}_{3 \times 3} & 0 \\ 0 & \mathbf{S}_0 \end{pmatrix} - ik_x \begin{pmatrix} 0 & \hat{P} \\ \hat{P}^T & 0 \end{pmatrix} - ik_y \begin{pmatrix} 0 & \hat{Q} \\ \hat{Q}^T & 0 \end{pmatrix} - \gamma(z) \begin{pmatrix} 0 & \hat{R} \\ \hat{R}^T & 0 \end{pmatrix} \frac{\partial}{\partial z} \right] \hat{v} = \hat{w}, \quad (6)$$

We solve it numerically, applying a finite-difference approximation, that results in a system of linear algebraic equations (SLAEs) with a banded matrix, whose bandwidth depends on the order of the finite-difference scheme. In this case, computation of $\hat{w}(k_x, k_y, z)$ can be performed via the 2D Fast Fourier Transform (FFT) and after $\hat{w}(k_x, k_y, z)$ are found, $L_0^{-1} w$ can be computed via the inverse 2D FFT.

Contents

- Motivation
- Equations and parameters
- **Parallelization**
- Examples
 - Comparison to the time-domain solution
- Convergence speed-up
- Conclusions

Parallelization has three levels:

Highest level.

The FWI for macro velocity reconstruction involves simulations for different seismic sources at different low frequencies. This means, that in fact, many boundary value problems for equation (1) are solved at the same time, each having its own right-hand side f . Since they are solved independently of each other, we solve each one with a separate MPI process, assigned to a single node or a group of cluster nodes. This is the highest level of our parallelization strategy. There are no communications between these MPI processes. Assuming that all computational nodes have similar performance, this parallel process scales very well. This is why we do not mention this level of parallelization in subsequent tests and consider the case of one seismic source and one frequency only.

Second level:

Four computational processes including Krylov iteration method, the 2D forward and inverse FFTs and solving the boundary value problem for equation (6), mainly drive our solver. We decompose the computational domain along one of the horizontal coordinates and parallelize these processes via MPI. The main exchanges between the MPI processes are while performing FFTs. For computing them, we use the Intel MKL library [10] supporting the decomposition along one direction only. In principle, the decomposition along the second horizontal dimension may be also applied with minor corrections of the code using a 2D FFT realization, supporting this functionality. Decomposition along the z-direction is not that obvious, since this involves solving each boundary value problems for equation (6) in parallel.

Third level:

Following this strategy, each MPI process would independently solve its own set of $N_x \cdot N_y / N$ (N – the number of MPI processes) problems. We solve them in a loop, parallelized via OpenMP.

Schematically, our parallelization strategy is presented in Figure:

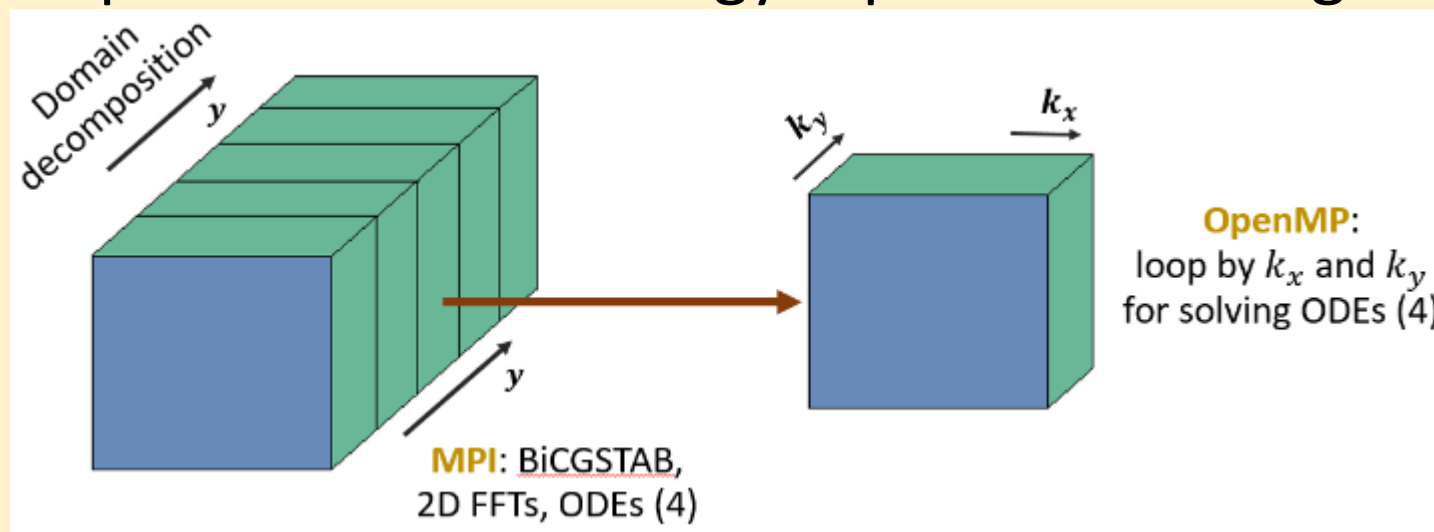


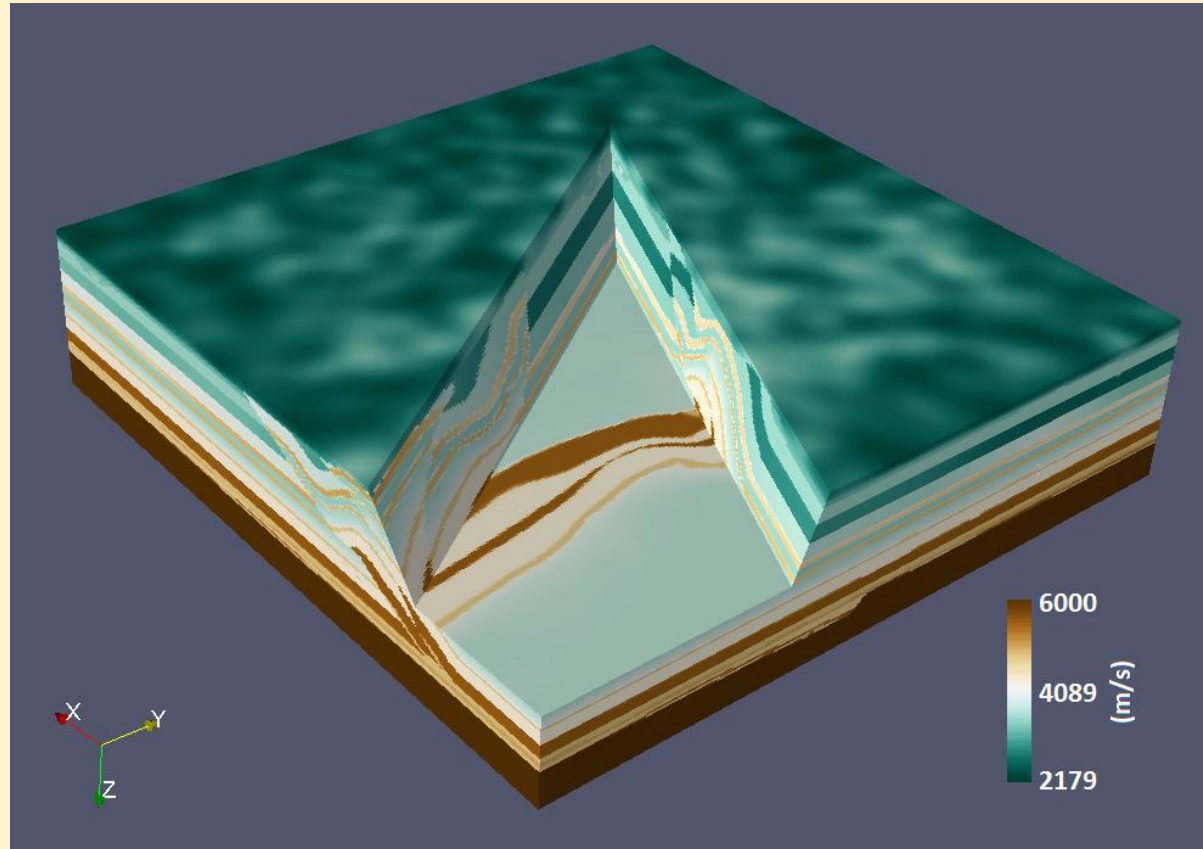
Figure. Parallelization strategy

Parallelization: Strong and Weak Scaling

Now I present the results of scaling analysis for both MPI and OpenMP. All results presented here have been computed on a HPC cluster comprising nodes with two Intel® Xeon® E5-2680v4 @ 2400 MHz CPUs and interconnected with 56 Gb FDR InfiniBand HCA. Double precision floating point format has been used in the computations.

This is necessary, when dealing with vectors of huge dimensions, for instance, for computing their dot product. As a stopping criterion for the BiCGSTAB, we used a 10^{-3} threshold for the relative residual of the L_2 -norm providing enough accuracy for FWI applications.

Parallelization: Strong and Weak Scaling

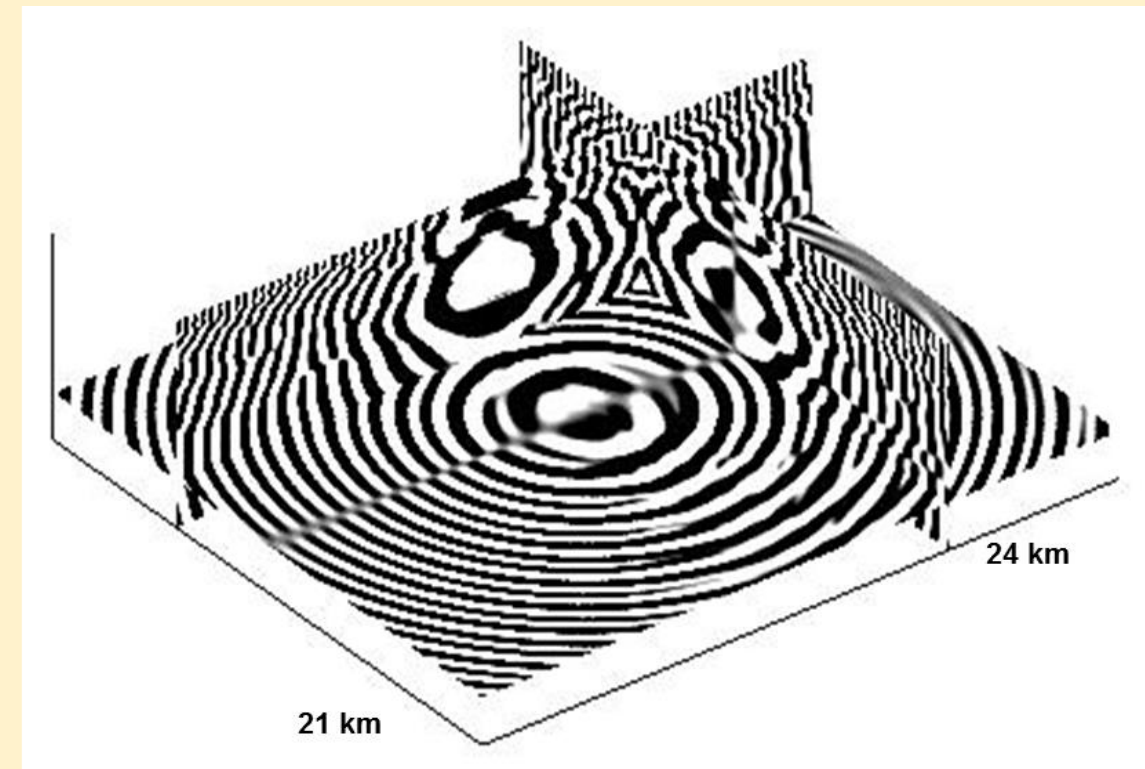
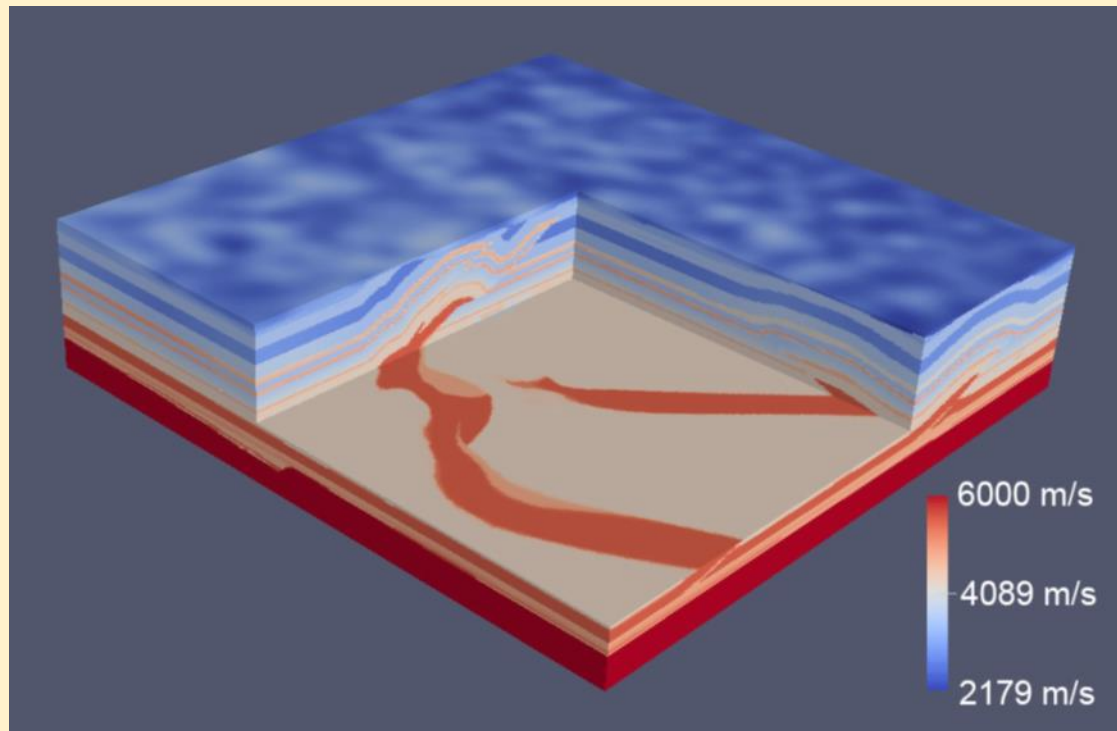


The model used for numerical experiments: 3D SEG/EAGE overthrust model
(19.8 x 19.8 x 4.65 km).

Parallelization: Strong and Weak Scaling

The model was discretized with a uniform grid of $957 \times 169 \times 651$ points with a lateral cell size of 25 m and a vertical cell size of 10 m. The source was placed in the middle of the area at 10 m depth. In the next slide we present a 3D view of the vertical velocity at 5 Hz and 10 Hz computed with the iterative solver. To obtain these results we used 18 computational nodes with 4 MPI processes per node and 7 OpenMP threads per MPI process. The total computational times for the 5 and 10 Hz solutions are 32 and 108 minutes, respectively.

Parallelization: Strong and Weak Scaling

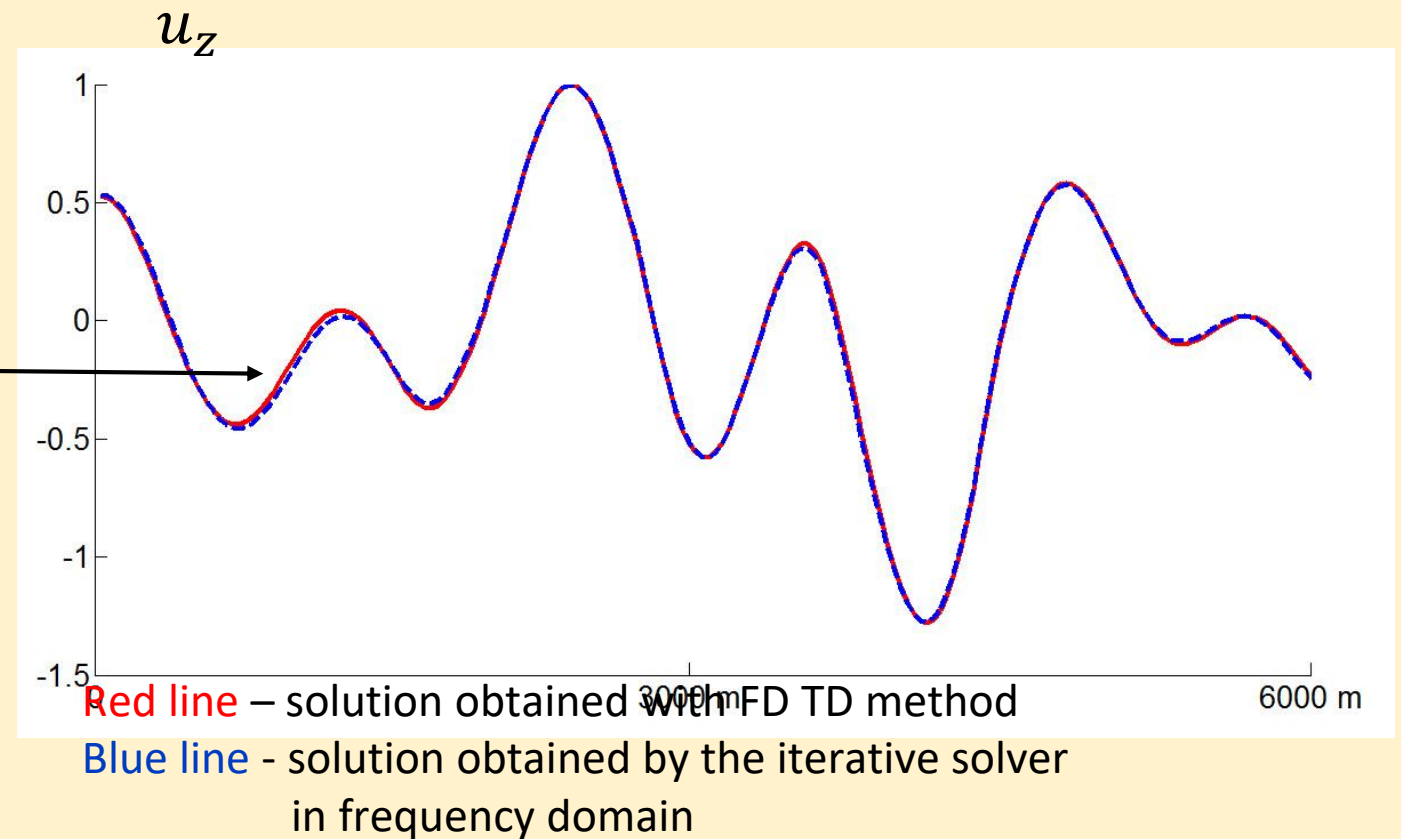
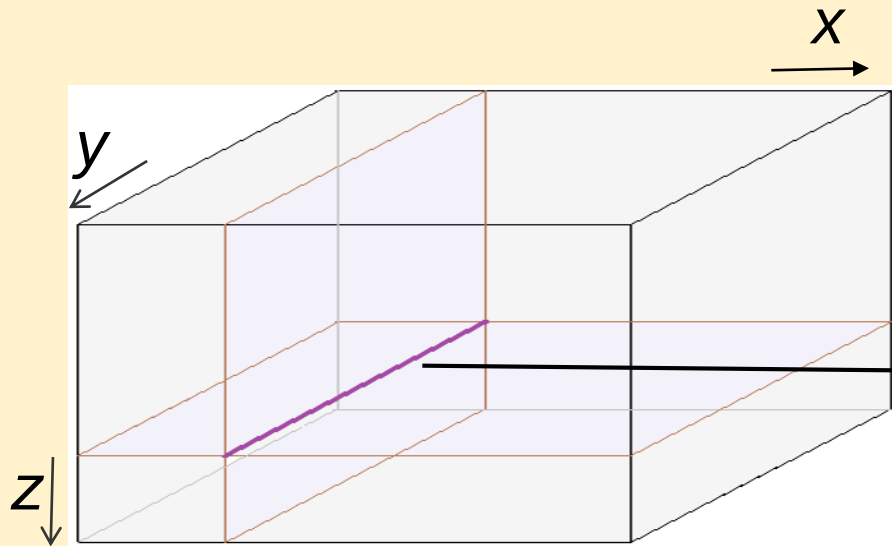


RAM needed: 420 GB.
of MPI processes: 75.
Time per one source: 84 min

66 iterations to converge

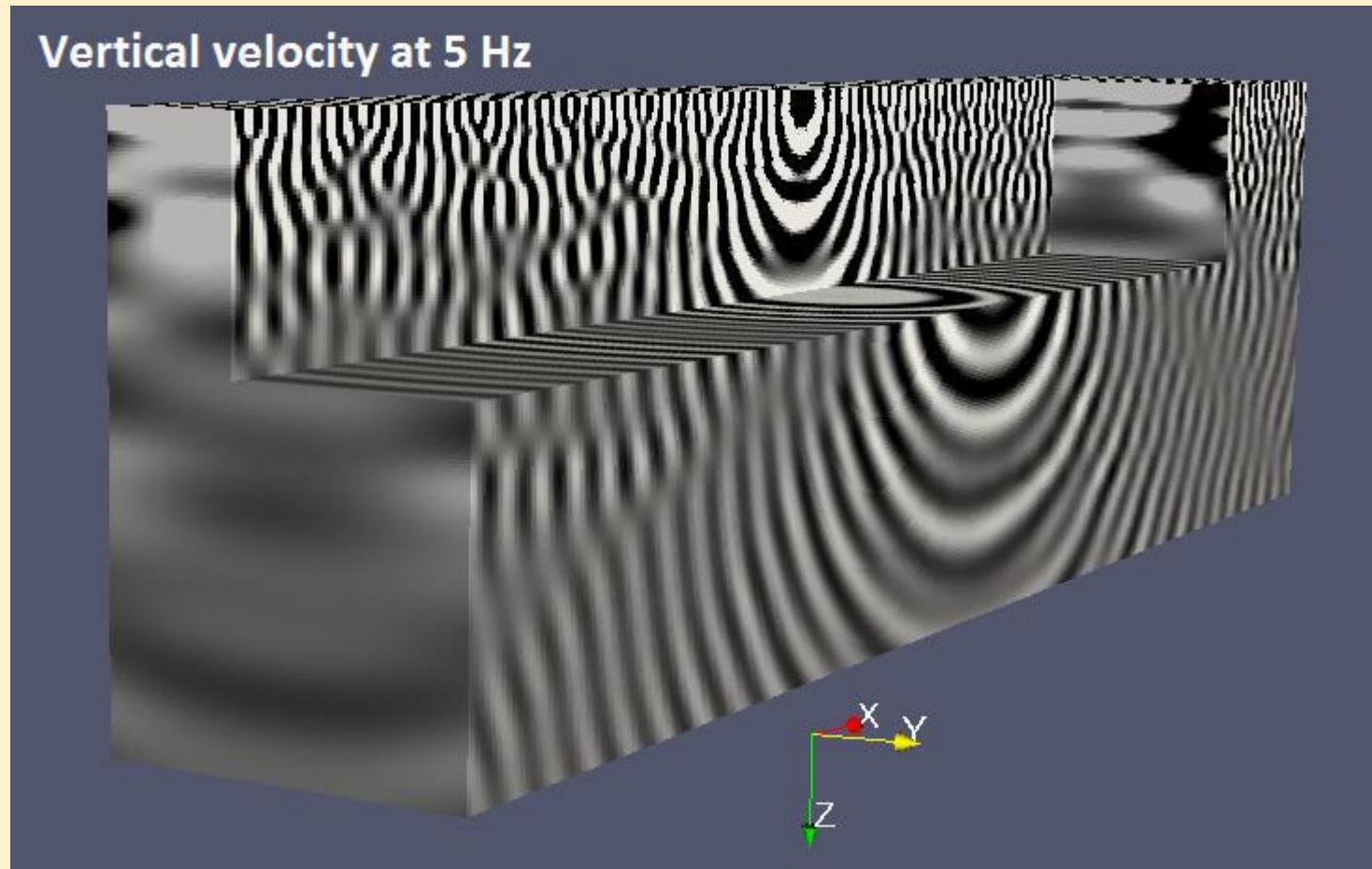
Parallelization: Strong and Weak Scaling

- SEG/EAGE overthrust submodel (5 Hz)

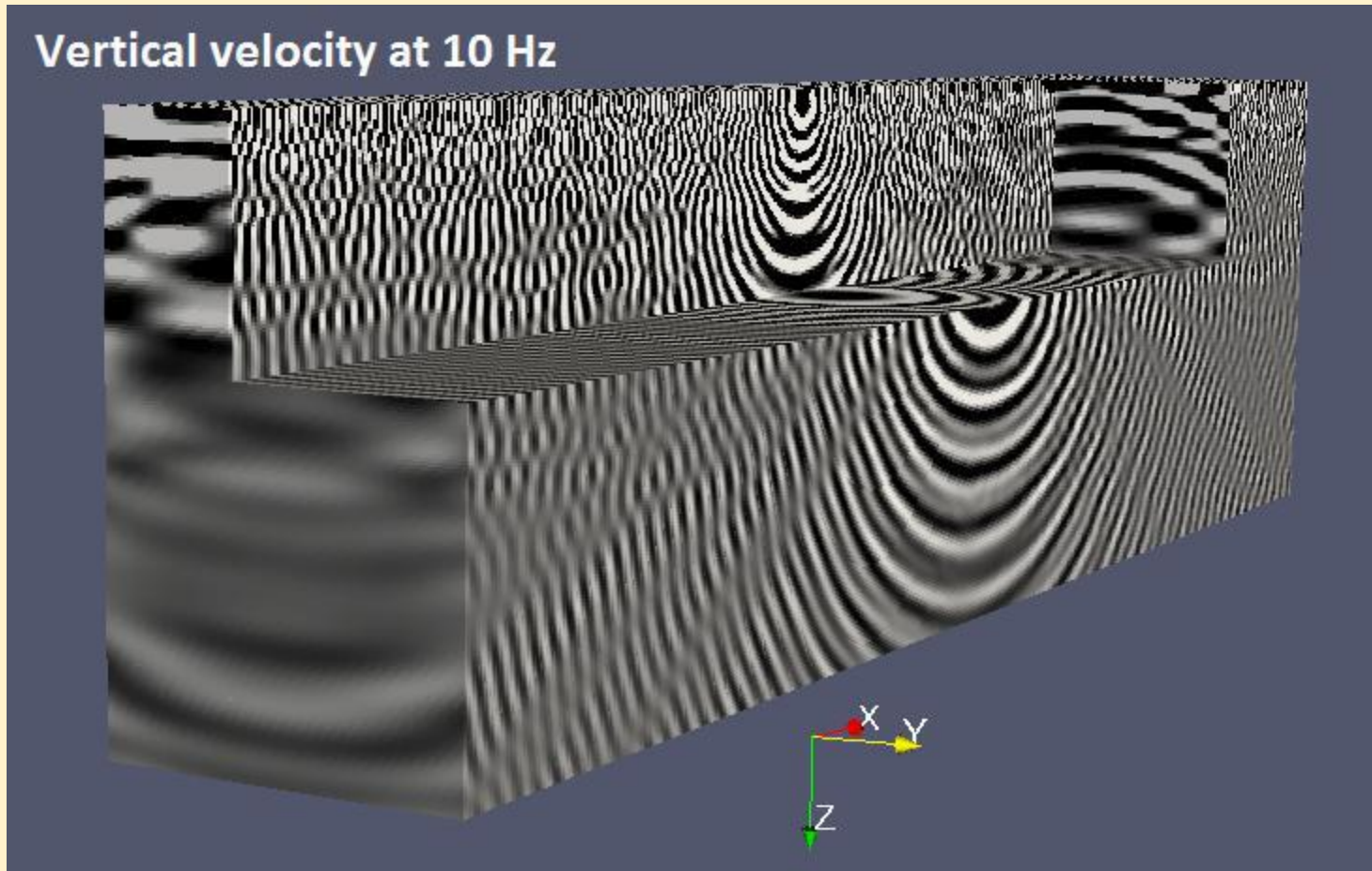


80 iterations to converge

Parallelization: Strong and Weak Scaling



Parallelization: Strong and Weak Scaling



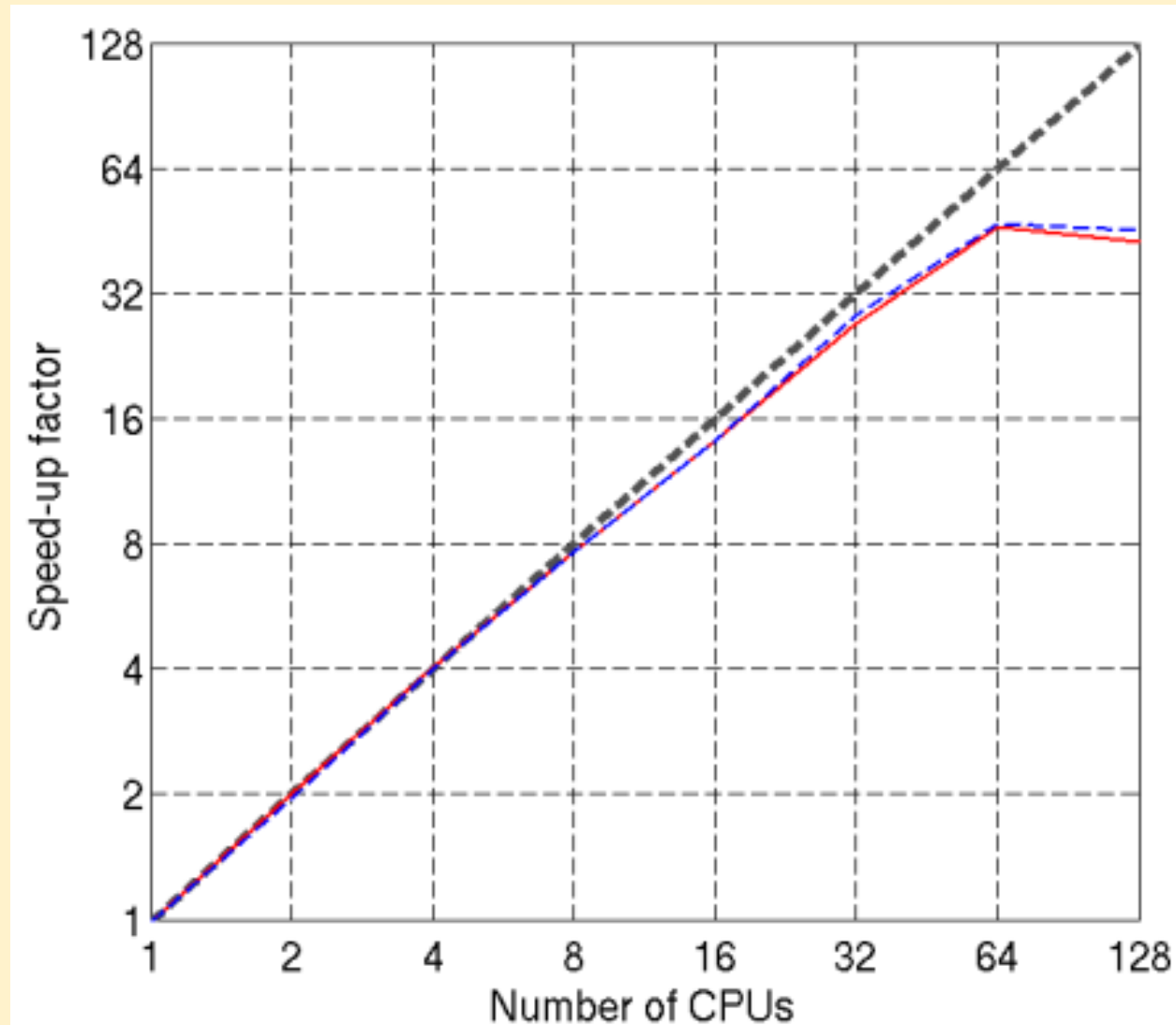
Parallelization: Strong and Weak Scaling

MPI strong scalability of the solver is defined as ratio t_M/t_N , where t_M and t_N are elapsed run times to solve the problem with N and $M>N$ MPI processes each corresponding to a different CPU. Using MPI, we parallelize two types of processes. First, those scaling ideally (solving problems (6)), for which the computational time with N processes is T/N . Second, the FFT, that scales as $T_{FFT}/\alpha(N)$, with coefficient $1<\alpha(N)<N$. The total computational time becomes $T/N+T_{FFT}/\alpha(N)$ (here we simplify, assuming no need of synchronization) with scaling coefficient $(T+T_{FFT})/(T/N+T_{FFT}/\alpha(N))$, that is greater than $\alpha(N)$. This is why, we expect very good scalability of the algorithm, somewhere between **the scalability of the FFT and the ideal scalability**.

Parallelization: Strong and Weak Scaling

Strong MPI scaling of the solver developed:

- The blue dashed line is the result for the Marmousi model,
- The red line - for the SEG/EAGE overthrust model.
- The dashed grey line is the ideal scalability.



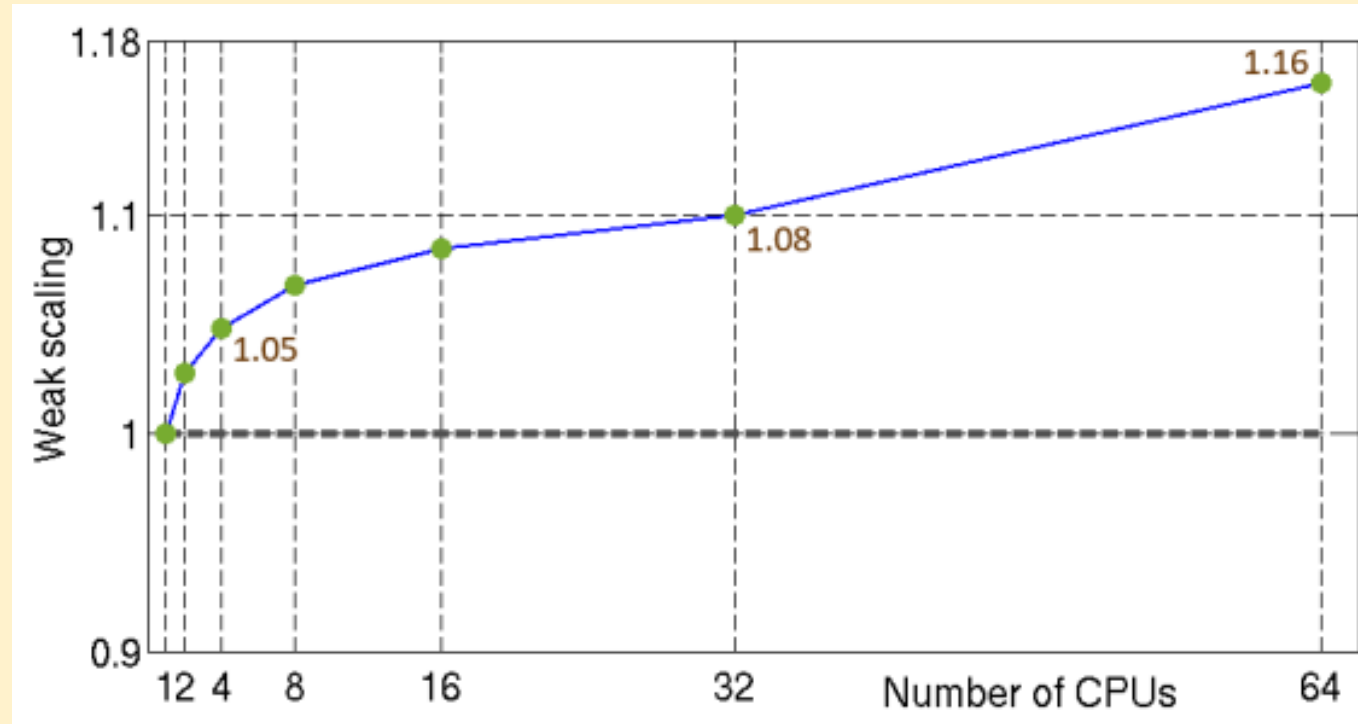
Parallelization: Strong and Weak Scaling

MPI Weak Scaling Analysis

For weak scaling estimation, we assign the computational domain to one MPI process and then extend the size of the computational domain along the y-direction, while increasing the number of MPI processes. Here, we use one MPI process per CPU. The load per CPU is fixed. For the weak scaling, we use function $f_{weak}(N)=T(N)/T(1)$, where $T(N)$ is the average computational runtime per iteration with N MPI processes. The ideal weak scalability corresponds to $f_{weak}(N)=1$.

Parallelization: Strong and Weak Scaling

MPI Weak Scaling Analysis



Weak scaling measurements: the blue line is the result of the iterative solver and the dashed grey line is the ideal weak scaling.

Parallelization: Strong and Weak Scaling

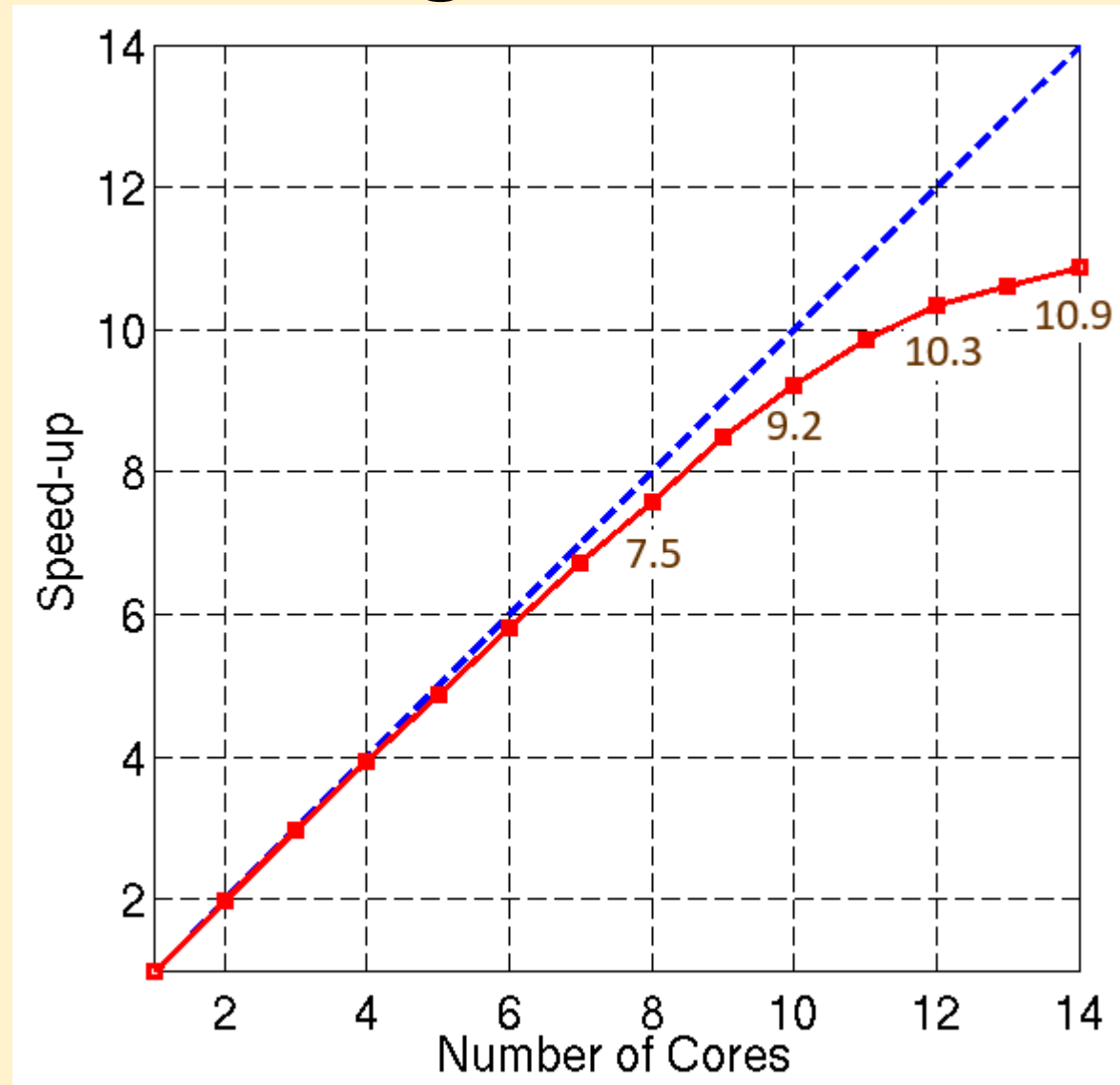
OpenMp Scaling Analysis

As already explained above, with OpenMP we parallelize the loop over spatial frequencies for solving the boundary value problems (6). To estimate the scalability of this part of our solver, we performed simulations in a small part of the SEG/EAGE overthrust model comprising $660 \times 50 \times 155$ points on a single CPU having 14 cores with hyper-threading switched off and without using MPI. Fig. 5 shows that our solver scales well for all threads involved in this example. It is worth mentioning, that we use OpenMP as an extra option applied when further increasing of the number of MPI processes doesn't improve performance any more, but the computational system is not fully loaded, i.e., there are free cores.

Parallelization: Strong and Weak Scaling

OpenMp Scaling Analysis

Strong scalability analysis on one CPU of the part parallelized via OpenMP: the dashed blue line is the ideal scalability and the red line is the iterative solver scalability.



Conclusions

We present a parallel iterative solver capable of modeling wavefields in 3D elastic land models of big size at low frequencies. The solver includes both MPI and OpenMP to reduce the computation time and shows good scalability. Further improvement of MPI scaling may be achieved by incorporating domain decomposition along the two horizontal directions into the current MPI parallelization scheme.

Road map

- Direct solvers!

Benchmarking 3D time- and frequency-domain solvers for FWI applications for different cluster sizes and variable number of sources

Andrey Bakulin^(*), Maxim Dmitriev^(*), Victor Kostin^(**), Sergey Solovyev^(**)

^(*)*Geophysics Technology, EXPEC Advanced Research Center, Saudi Aramco*

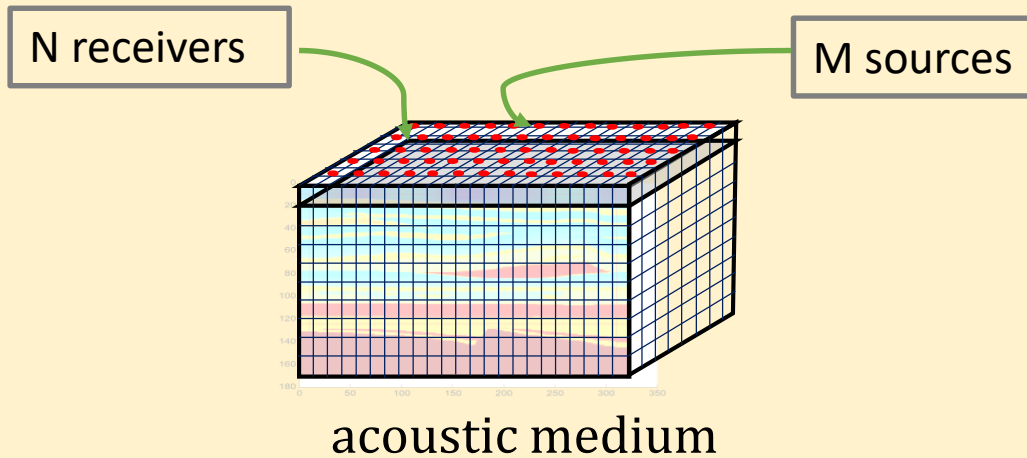
^(**)*Institute of Petroleum Geology and Geophysics, SB RAS, Novosibirsk, Russia*

Agenda

- Frequency Domain FWI
- The Solvers
- Numerical Results
- Conclusions

Frequency domain FWI

Measurement configuration



Governing equation

$$\frac{\partial^2 p}{\partial t^2} - c^2(x) \Delta p = f(t) \delta(x - x^s)$$

$c(x) = c(x_1, x_2, x_3)$ – sound velocity;

$x^s = (x_1^s, x_2^s, x_3^s)$ – point source position;

$p = p(t, x; x^s, c(x))$ – pressure (depending on the source position and the model).

Data

$x^r = (x_1^r, x_2^r, x_3^r)$ – receiver position;

$d = d(t; x^r, x^s, c(x)) = p(t, x^r; x^s, c(x))$

Acoustic waves are generated by point sources. Pressure is registered by receivers as data for inversion.

Frequency domain wave equation

$$\Delta \hat{p} + \frac{\omega^2}{c^2(x)} \hat{p} = g(\omega) \delta(x - x^s).$$

Here

$$\hat{p}(x; c, \omega, x^s) = \int e^{-i\omega t} p(t, x; x^s, c(x)) dt.$$

Minimization algorithms of (strongly nonlinear) objective function

$$\min_c \frac{1}{2} \sum_{\omega} \sum_{x^s} \sum_{x^r} |\hat{p}(x^r; c, \omega, x^s) - \hat{d}(x^r; c, \omega, x^s)|^2$$

require effective solver for the “forward problem”:

Given sound velocity $c(x)$, frequency ω and source function $f(x)$ find solution $u(x)$ to PDE

$$\Delta u + \frac{\omega^2}{c^2(x)} u = f(x).$$

To emulate boundary conditions at infinity the domain is surrounded with PMLs and the solution is subject to homogeneous boundary conditions on the outer boundary.

Time Domain Finite Difference solver (TD)

For comparison, we used the TD solver developed by Seiscope consortium

(<https://seiscope2.osug.fr>).

Features:

- 4th order explicit Finite Difference approximation of the wave equation

$$\frac{\partial^2 u}{\partial t^2} - c^2(x) \Delta u = f(t, x).$$

- To suppress reflections, Perfectly Matched Layers (PMLs) surround the computational domain.
- The Frequency Domain solution (for all frequencies at once) is found by the Fourier transform

$$\hat{u}(x; c, \omega, x^s) = \int e^{-i\omega t} u(t, x; x^s, c(x)) dt.$$

- For every particular source function the equation should be solved separately (easily parallelizable, ideally scaling).

Frequency Domain Finite Difference (FD) solver

Boundary Value Problem to solve is the Helmholtz equation

$$\Delta u + \frac{\omega^2}{c^2(x)} u = f(x)$$

in the computational domain surrounded with PMLs with homogeneous Dirichlet boundary conditions.

Finite Difference approximation (27 points grid stencil, optimized) of the BVP results in a system of linear equations

$$AU = F.$$

Columns U and F represent values of $u(x)$ and $f(x)$ at the grid points. A is sparse, complex valued (due to PMLs) and symmetric ($A = A^t$) but not Hermitian.

Direct solver outline

System of linear equations

$$AU = F.$$

- Permute the matrix (Nested Dissection)

$$\hat{A} = PAP^t$$

- Decompose the matrix

$$\hat{A} = L \cdot D \cdot L^t.$$

The most time consuming step. Compression of L by using Low-Rank approximation help to **reduce memory consumptions and flops count**. 5-6x reduction (memory and flops) is possible but leads to an approximate factorization

$$A \approx \tilde{L} \cdot \tilde{D} \cdot \tilde{L}^t.$$

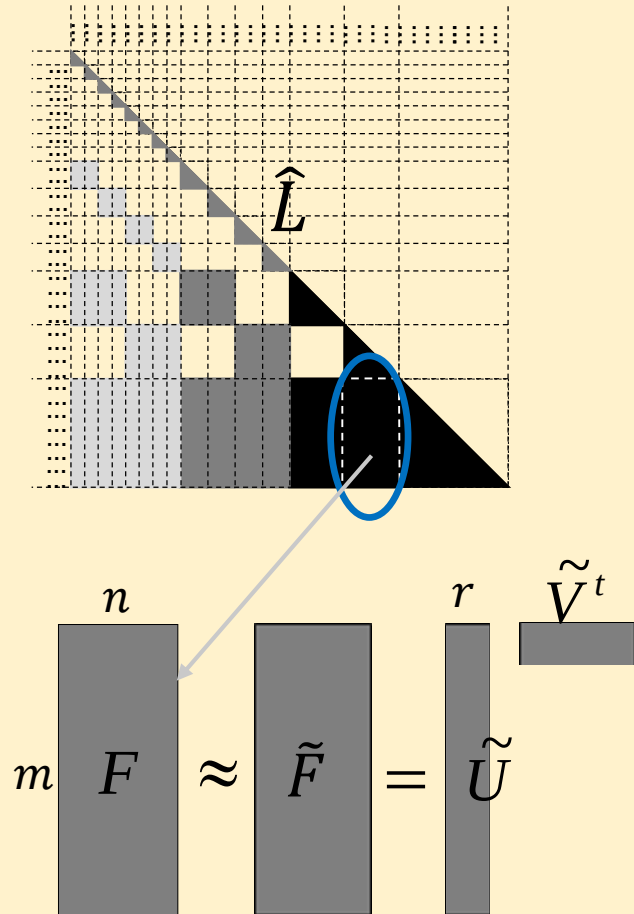
- Solve two systems of linear equations with triangular coefficient matrices

$$\begin{aligned} LV &= PF, \\ DL^t PU &= V. \end{aligned}$$

Replacement of L and D with $\tilde{L}$ and $\tilde{D}$ results in loss of accuracy.

Provided the compression is not too aggressive, the *iterative refinement* can help. Otherwise, the iterations may diverge.

Low-rank approximation



SVD-based solution

$$F = USV^t$$

$$S = \begin{pmatrix} s_1 & & & & \\ & \ddots & & & \\ & & s_r & & \\ & & & s_{r+1} & \\ & & & & \ddots \\ & & & & & s_n \end{pmatrix}$$

$$s_1 \geq s_2 \geq \dots \geq s_n \geq 0$$

- Given threshold ε find r : $\frac{s_{r+1}}{s_1} < \varepsilon$
- Define

$$S^r = \begin{pmatrix} s_1 & & \\ & \ddots & \\ & & s_r \end{pmatrix},$$

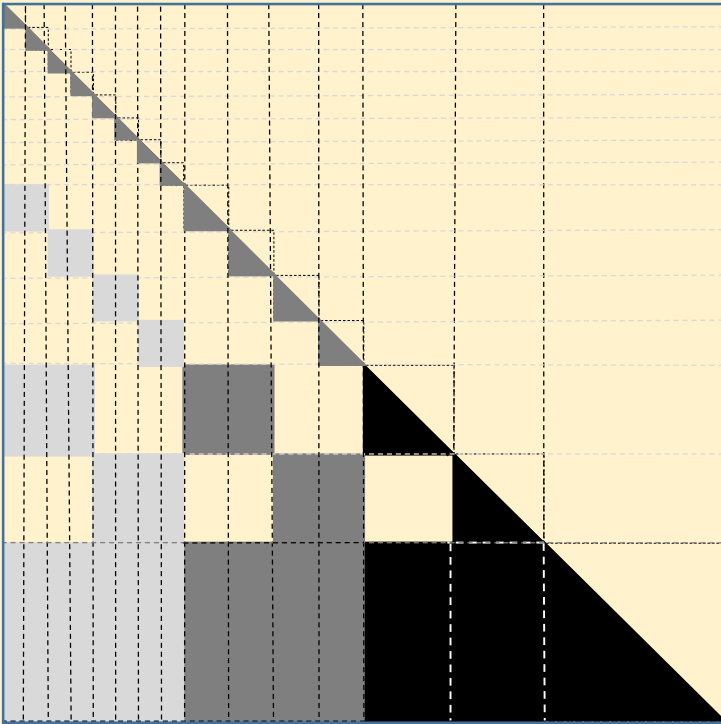
$$U = \begin{bmatrix} U^r & \end{bmatrix}$$

$$V = \begin{bmatrix} V^r & \end{bmatrix}$$

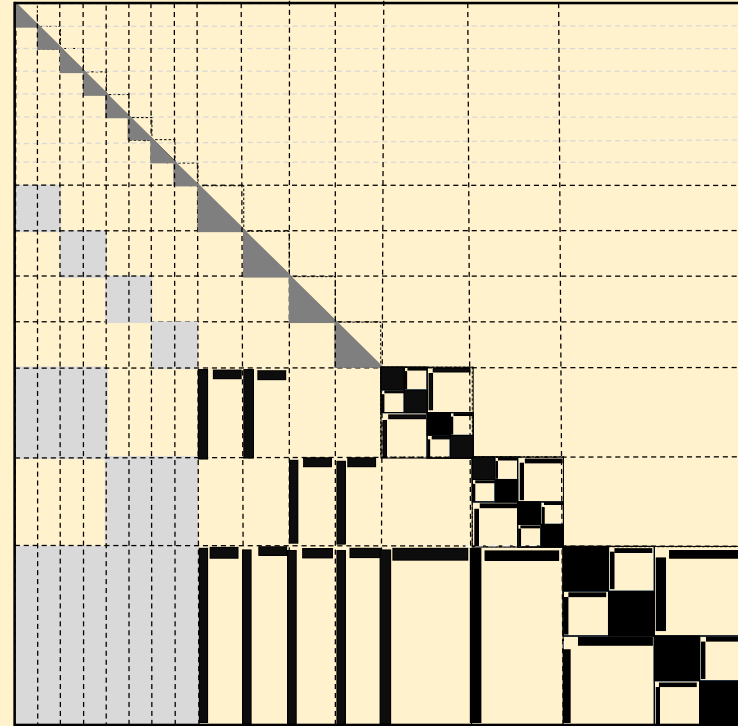
$$\begin{aligned} \tilde{U} &= U^r \\ \tilde{V} &= V^r \cdot S^r \end{aligned}$$

Compressed matrix structure

L



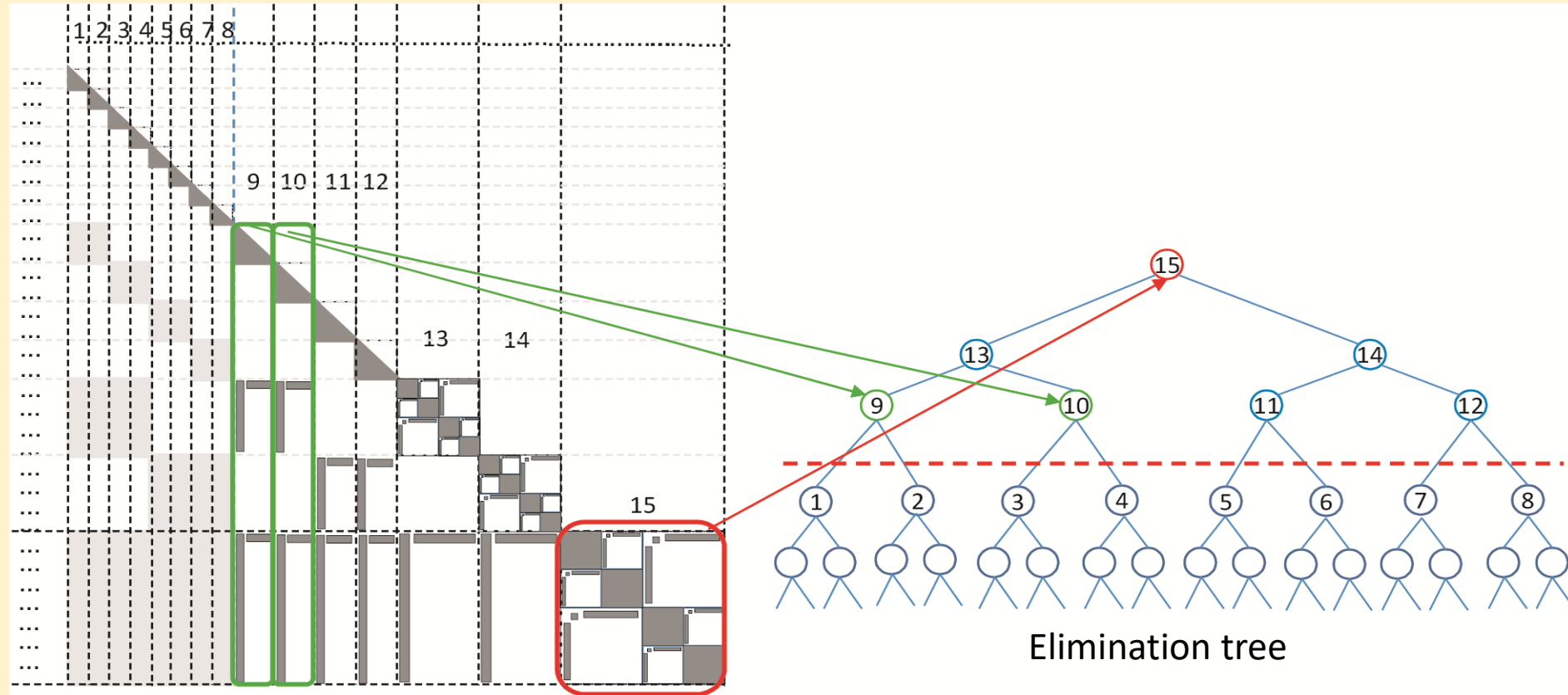
$\tilde{L}$



✓ Factorization time (5x speed up)

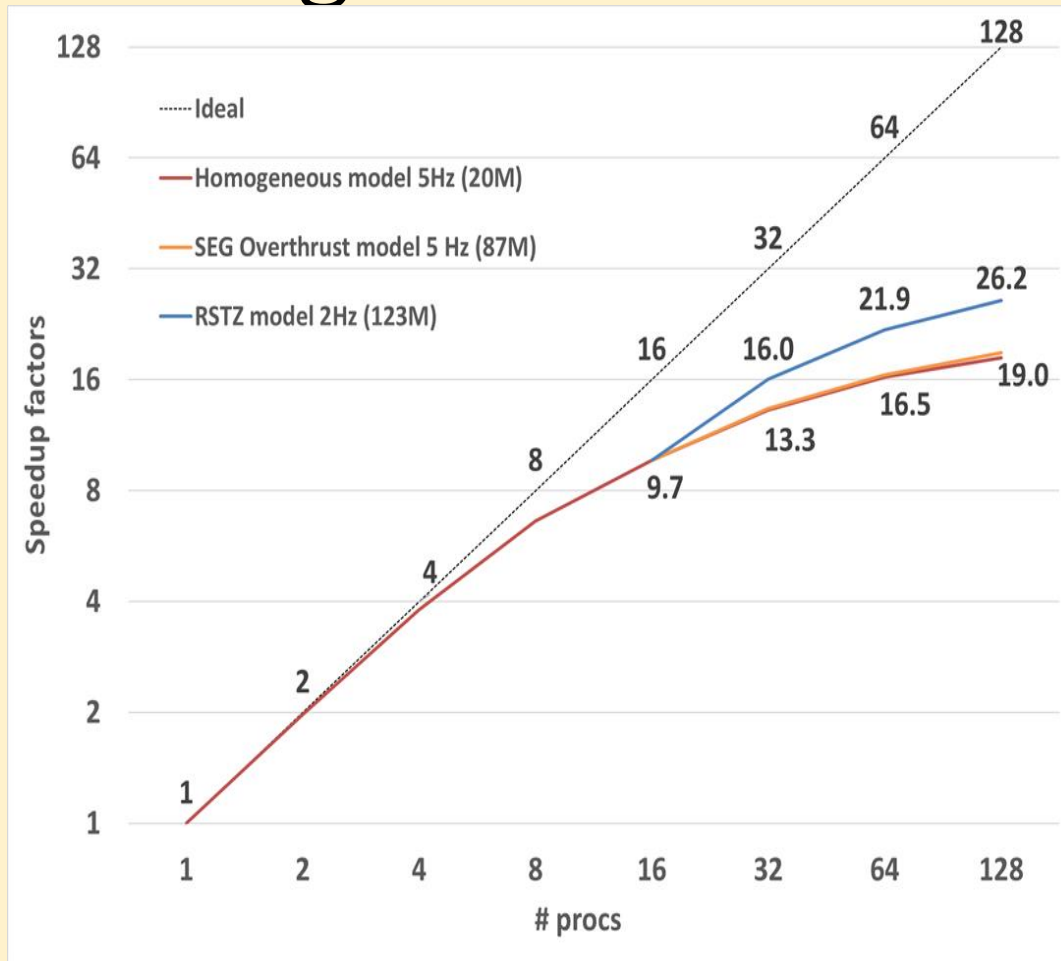
✓ Memory usage (5x compression)

Parallel LDLT decomposition

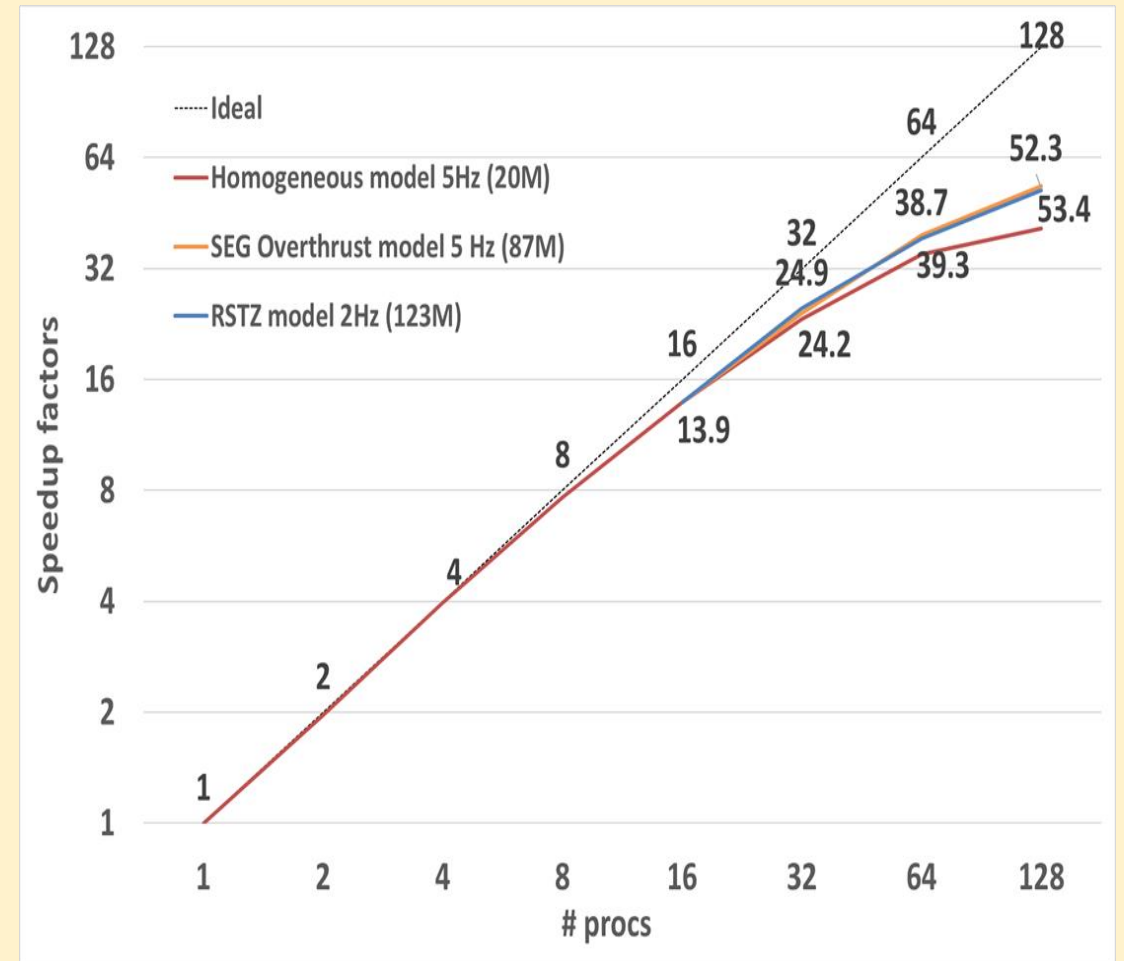


Structure of compressed L-factor

Scaling



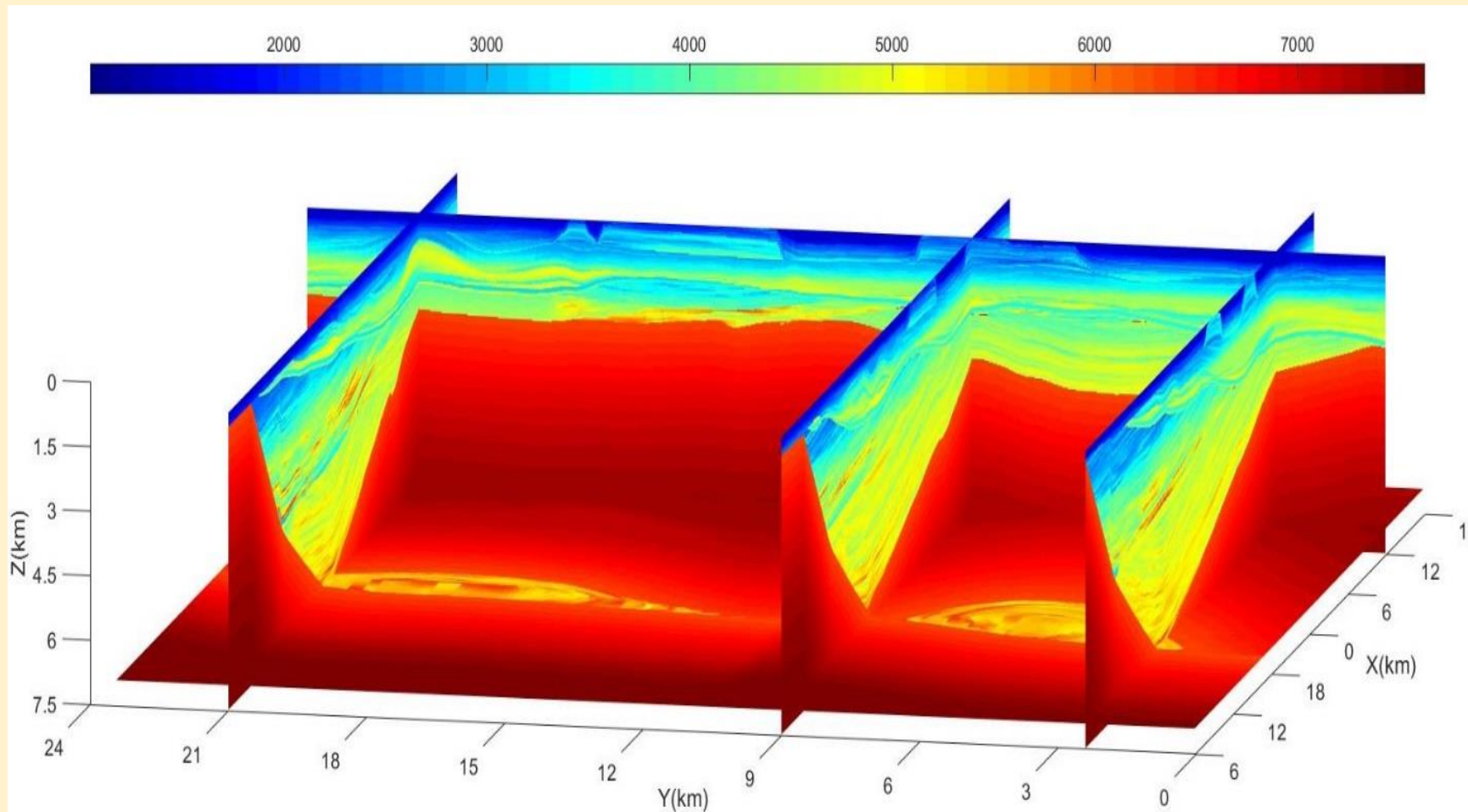
Matrix factorization



Solving step

Numerical results

Marine Transition Zone velocity model



Models sizes:
18000 x 23500 x 7000 m

Grid step 30 m

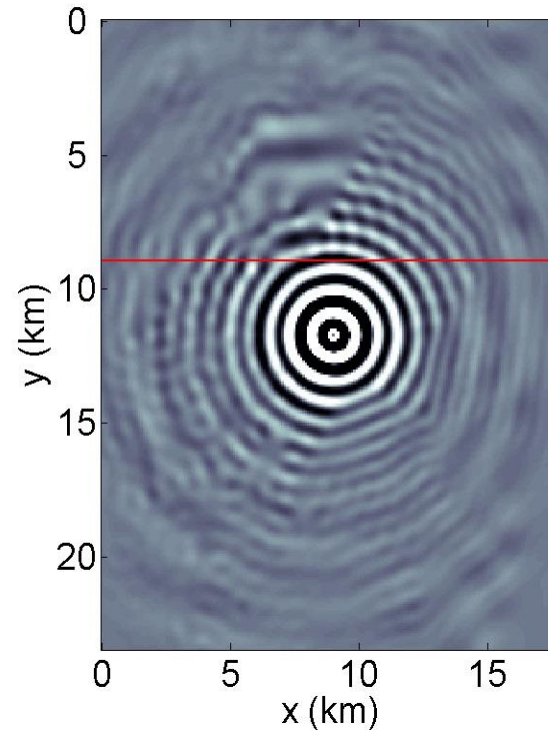
180 m of linear equations

Velocity varies in
[1042, 7626] m/s.

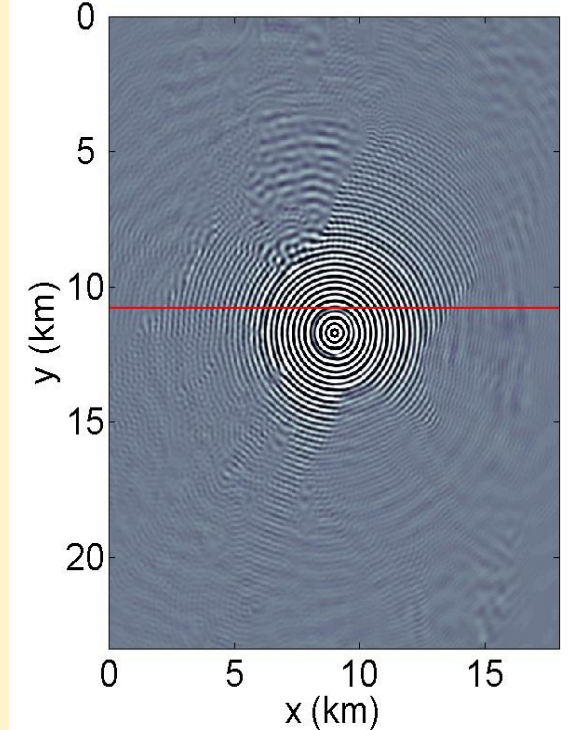
Numerical experiments

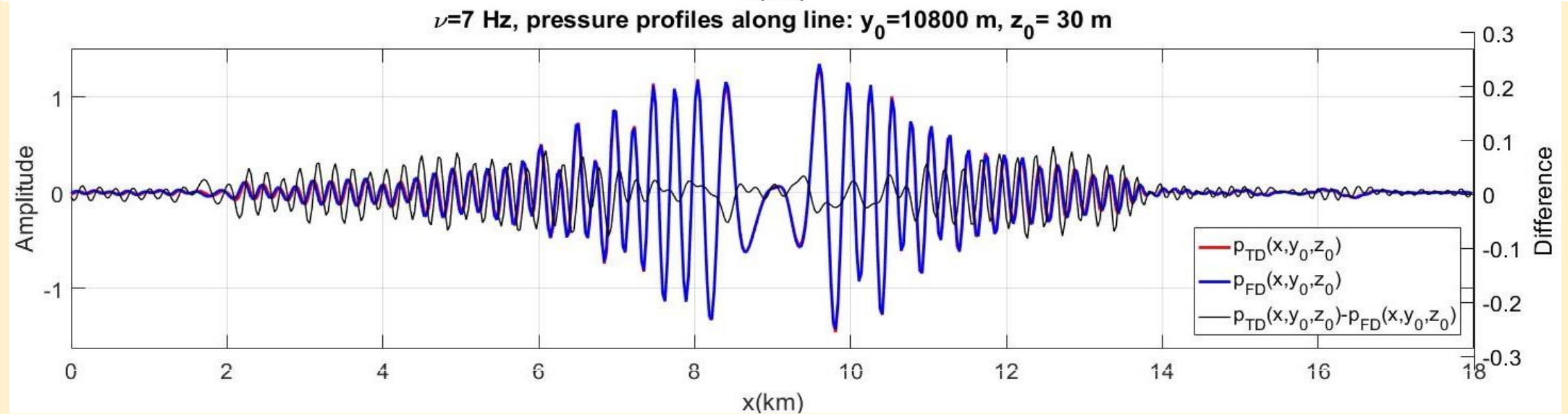
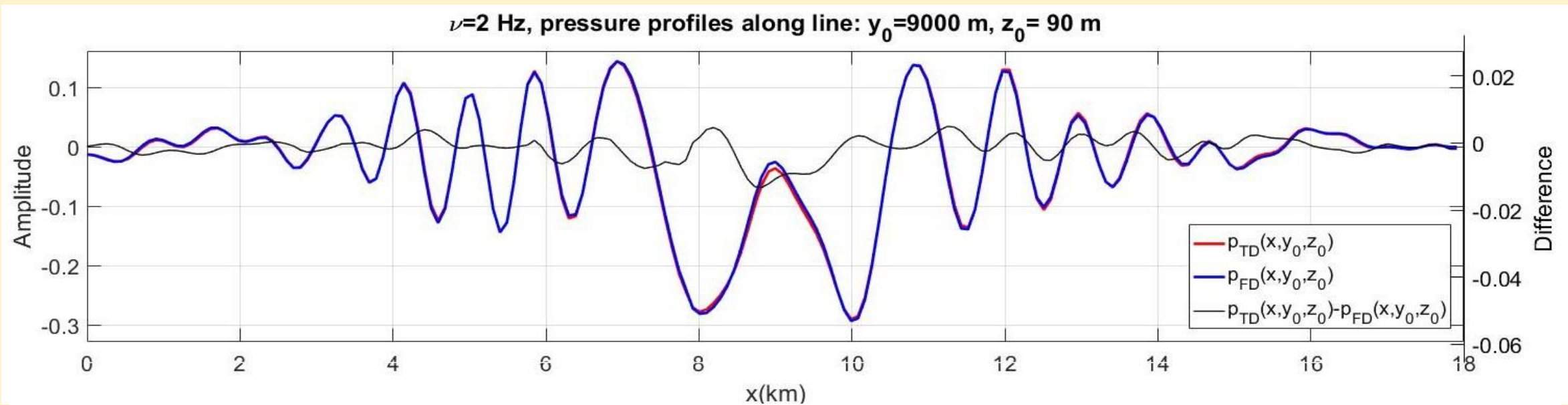
Parameters			
Freq (Hz)	2	3	7
Grid step (m)	90	60	30
ppw	5.8	5.8	4.0
Nx	200	300	600
Ny	261	391	781
Nz	77	116	231
N	$4 \cdot 10^6$	$1.4 \cdot 10^7$	10^8
TD solver time step (ms)	1.95		

$z_0 = 90$ m, $\nu = 2$ Hz: snapshot $\text{Re } p(x,y,z_0)$

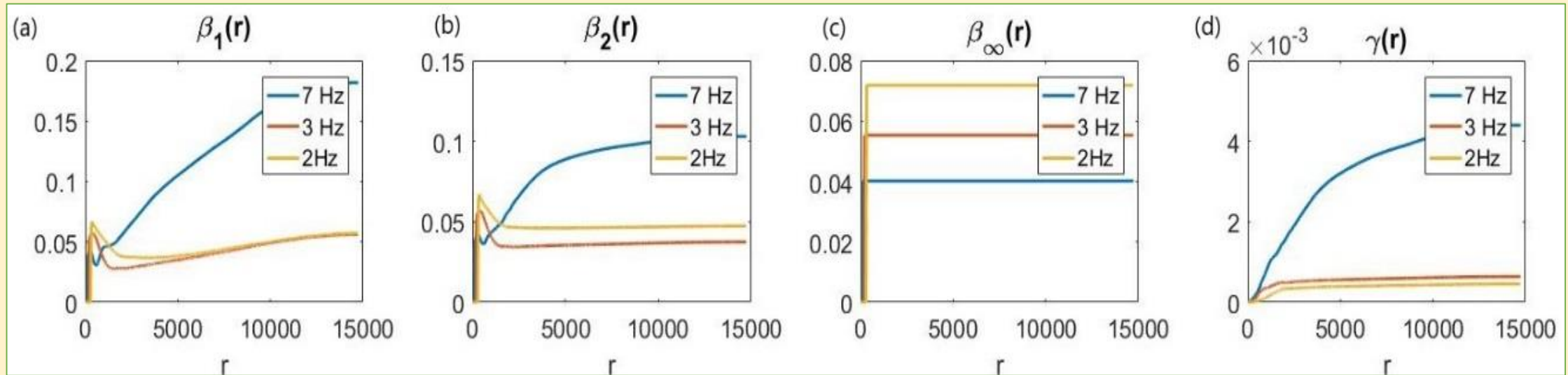


$z_0 = 30$ m, $\nu = 7$ Hz: snapshot $\text{Re } p(x,y,z_0)$





Compare solutions obtained with TD and FD solvers



$$\beta_k(u, v) = \frac{\|u - v\|_k}{\|u\|_k}, k = 1, 2, \infty, \quad \gamma(u, v) = \left| 1 - \left| \frac{(u, v)}{\|u\| \|v\|} \right| \right|$$

$$\beta_k(r) = \beta_k(u, v), \gamma(r) = \gamma(u, v) \quad \text{for } r_0 < \|x - x_s\| < r$$

u – the solution obtained with TD solver

v – the solution obtained with FD solver

Competitiveness

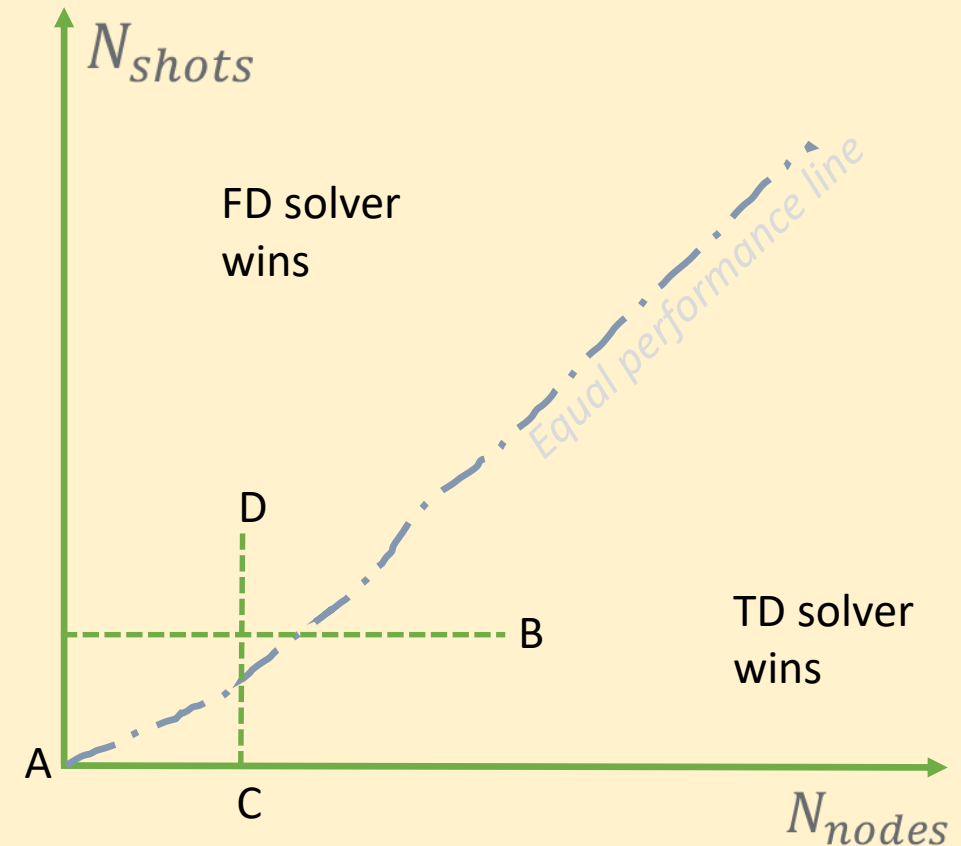
$\frac{t_{TD}}{t_{FD}}$		N_{shots}			
		1	128	1 280	12 800
N_{nodes}	32	17m 46s	1h 11m	11h 52m	122h
		2h 20m	2h 22m	2h 45m	6h 30m
	64	17m 46s	36m	5h 56m	61h
		1h 55m	1h 56m	2h 11m	4h 32m
	128	17m 46s	18m 19s	3h	30h 33m
		1h 41m	1h 42m	1h 54m	3h 49m
	256	17m 46s	18m 19s	1h 30m	15h 16m
		1h 41m	1h 42m	1h 47m	2h 45m

Some t_{TD} data are extrapolated, all t_{FD} are measured.

Runtimes in the table take into account three values of the frequency. For the FD solver it means

$$t_{FD} = t_{FD}(2\text{Hz}) + t_{FD}(3\text{Hz}) + t_{FD}(7\text{Hz}).$$

The TD solver solution contains all frequencies and only one run is needed.



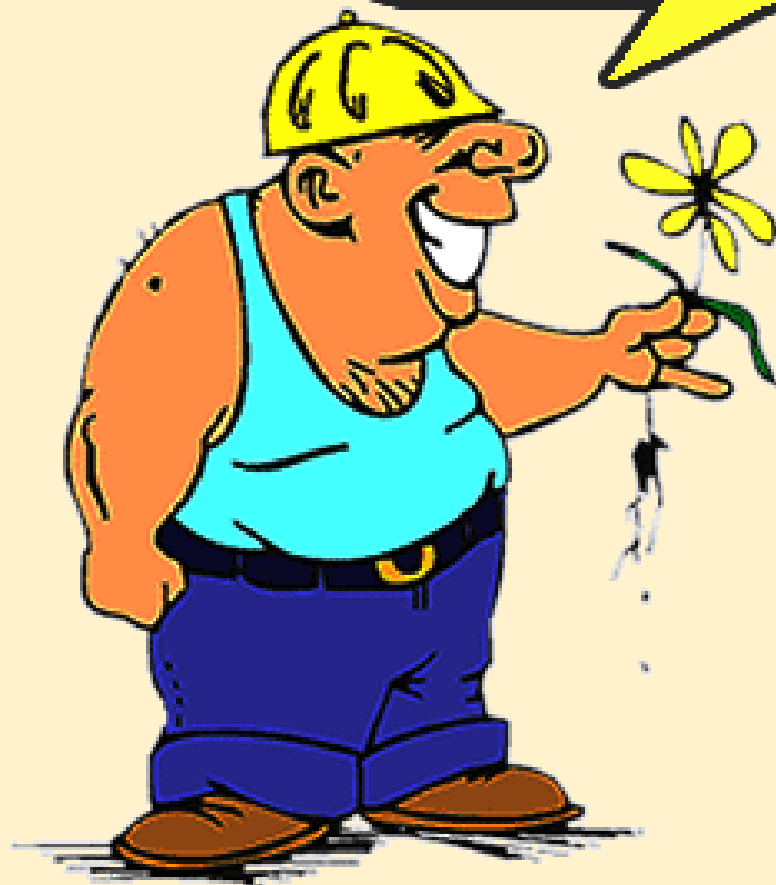
Conclusions

In the context of FWI, we compared two acoustic solvers for different cluster sizes and variable number of shots. The winner depends on available computing power and number of shots.

TD solver	FD direct HSS-based solver
Provides solution for all the frequencies at once;	Needs a separate run for a particular frequency;
Effort linearly increases with N_{shots} ;	Effort weakly depends on N_{shots} ;
Perfectly scales with respect to N_{nodes} .	Nonlinearly scales with respect to N_{nodes} .

An optimal FWI toolbox solver should contain both solvers.

THANK YOU!



Questions?

