Parallelization Strategy for Wavefield Simulation with an Elastic Iterative Solver

Mikhail Belonosov[1], Vladimir Cheverda[2](✉), Victor Kostin[2] and Dmitry Neklyudov[2]

[1] Aramco Research Center - Delft, Aramco Overseas Company B.V., Delft, the Netherlands
mikhail.belonosov@aramcooverseas.com
[2] Institute of Petroleum Geology and Geophysics SB RAS, Novosibirsk, Russia
{cheverdava, kostinvi, neklyudovda}@ipgg.sbras.ru

Abstract. We present a parallelization strategy for our novel iterative method to simulate elastic waves in 3D land inhomogeneous isotropic media via MPI and OpenMP. The unique features of the solver are the preconditioner developed to assure fast convergence of the Krylov-type iteration method at low time frequencies and the way to calculate how the forward modeling operator acts on a vector. We successfully benchmark the accuracy of our solver against the exact solution and compare it to another iterative solver. The quality of the parallelization is justified by weak and strong scaling analysis. Our modification allows simulation in big models including a modified 2.5D Marmousi model comprising 90 million cells.

Keywords: Elastic equation · MPI · OpenMP · Preconditioner · Krylov iterations

1 Introduction

Advances in supercomputing technology make it feasible to solve big data problems. Large simulations seemed impossible in the recent past but have become commonplace today. In geosciences, supercomputers have been opening new horizons for understanding subsurface structures by allowing 3D imaging and velocity model estimation at high fidelity for fine-scale reservoir characterization. To keep up to date with cutting edge technologies, oil and gas companies are spending huge budgets to buy, lease and upgrade/maintain supercomputers (e.g., [1]). Dealing with high channel count field data processing and velocity estimation, the associated forward modeling and inversion algorithms may require petabytes of RAM and petaflops of computing power. The large datasets and high computation requirements are why modern algorithms have to be parallel.

Velocity reconstruction with frequency-domain full waveform inversion (FWI) ([19], [24], [26]) has been actively developing in the last decades. These days, even 3D elastic inversion, that may bring the most valuable information to interpreters, seems to be feasible. The most time consuming part of this process is forward modeling performed several times at each iteration. For macro velocity reconstruction, only a few low frequency (up to 10 Hz) monochromatic components of the seismic wavefield are used.

The common practice is to perform simulation in the time domain ([12], [16], [22]), extracting the needed frequencies via Fourier Transform applied on the fly with a Discrete Fourier Transform. Time-domain simulation is usually parallelized over sources via MPI, so that each MPI process is independent of the others. Then, within each MPI process, parallelization is carried out via OpenMP [9], or MPI through domain decomposition of the target area, involving exchanges between adjacent subdomains or groups of subdomains (e.g., [11] and [18]).

The alternative, 3D frequency-domain modeling becomes feasible with the appearance of computing technology able to operate with big data. Here, we model the wavefield in the frequency domain only for needed frequencies. There are certain theoretical advantages of frequency-domain modeling that might improve the overall efficiency. Two different approaches are distinguished, including: direct [17] and iterative approaches based on a Krylov-type iteration method [21]. The main bottleneck of the first one is the necessity to store LU factors of the forward modeling matrix, requiring hundreds of gigabytes of RAM in a 3D case even for the acoustic case. This process can be parallelized with MPI via Domain Decomposition (e.g., [4]), decreasing the memory requirements per MPI process. However, in big models this involves vast computational resources, making the method computationally too expensive. Other attempts to resolve this issue are based on applying data compression techniques using Hierarchically Semi-Separable formats for storing data and Low Rank approximation of matrices ([14], [27]).

The memory requirements of the iterative approach are much more modest, because matrix factorization is not performed. However, its indefiniteness in seismic applications leads to very slow convergence or even divergence of the Krylov-type iteration method. Attempts to overcome this issue involve an appropriate preconditioner. In case of a 3D elastic simulation, only a few preconditioners have been developed so far ([7], [15], [20]).

In this paper, we use our own preconditioner that we briefly introduced in [13]. It is a modification for the elastic case of the preconditioner presented in [5] that was designed for simulation of low frequency monochromatic components of a wavefield in a 3D acoustic medium. Here we do not illustrate that our method is superior to other aforementioned approaches. This is a feasibility study showing that the method is capable of performing simulation in 3D elastic models of big size at low frequencies needed for macro velocity reconstruction with FWI. This is currently only possible using parallelization that we developed using MPI and OpenMP.

2 Iterative Method to Solve a 3D Elastic Equation

2.1 Statement of the Problem

We solve the following elastic equation written in the velocity-stress form describing propagation of a monochromatic component of the wave in a 3D isotropic heterogeneous medium

$$\left[i\omega\begin{pmatrix}\rho \boldsymbol{I}_{\mathbf{3}\times\mathbf{3}} & 0\\ 0 & \boldsymbol{S}_{\mathbf{6}\times\mathbf{6}}\end{pmatrix}-\begin{pmatrix}0 & \hat{P}\\ \hat{P}^T & 0\end{pmatrix}\frac{\partial}{\partial x}-\begin{pmatrix}0 & \hat{Q}\\ \hat{Q}^T & 0\end{pmatrix}\frac{\partial}{\partial y}-\gamma(z)\begin{pmatrix}0 & \hat{R}\\ \hat{R}^T & 0\end{pmatrix}\frac{\partial}{\partial z}\right]\boldsymbol{v}=\boldsymbol{f}, \quad (1)$$

where vector of unknowns $\boldsymbol{v}$ 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, $\boldsymbol{I}_{\mathbf{3}\times\mathbf{3}}$ is 3 by 3 identity matrix, $\boldsymbol{S}_{\mathbf{6}\times\mathbf{6}}(x, y, z)=\begin{pmatrix}A & 0\\ 0 & C\end{pmatrix}$ is 6 by 6 compliance matrix and

$$A=\begin{pmatrix}a & -b & -b\\ -b & a & -b\\ -b & -b & a\end{pmatrix}, C=\begin{pmatrix}c & 0 & 0\\ 0 & c & 0\\ 0 & 0 & c\end{pmatrix}, \hat{P}=\begin{pmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 1 & 0\end{pmatrix}, \hat{Q}=\begin{pmatrix}0 & 0 & 0 & 0 & 0 & 1\\ 0 & 1 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 1 & 0 & 0\end{pmatrix}, \hat{R}=\begin{pmatrix}0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\end{pmatrix}. \quad (2)$$

Coefficients $a(x, y, z)$, $b(x, y, z)$ and $c(x, y, z)$ are related to the Lame parameters λ and μ as follows $a=\frac{\lambda+\mu}{\mu(2\mu+3\lambda)}$, $b=\frac{\lambda}{2\mu(2\mu+3\lambda)}$, $c=\mu^{-1}$. $\boldsymbol{f}$ 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) [6].

Equation (1) is solved in a cuboid domain of $N_x \times N_y \times N_z$ points. It is assumed that 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. Usage of the sponge layers assures the coefficients of the partial derivatives by x and y are constant matrices. We use this property in the next section, applying the Fourier Transform over these coordinates to construct our preconditioner. Equation (1) along with the boundary conditions on the outer boundaries of the absorbing layers ($v_z = \sigma_{yz} = \sigma_{xz} = 0$ on the horizontal and periodic conditions on the vertical faces) produce the boundary value problem that we solve by means of the Krylov iterations. Their straightforward application does not guarantee convergence. This is why in the next section we develop a special preconditioner.

2.2 Preconditioned Iterative Method

Denote by L the right-hand side operator in equation (1) that is subject to the same boundary conditions as the boundary value problem we solve. Let L_0 be the same operator as L, but with $\rho(x, y, z) = \rho_0(z)$, $\boldsymbol{S}_{\mathbf{6}\times\mathbf{6}}(x, y, z) = (1 + i\beta)\cdot \boldsymbol{S}_{\mathbf{0}}(z)$, where

$$\boldsymbol{S_0}(z) = \begin{pmatrix} A_0 & 0 \\ 0 & C_0 \end{pmatrix}, A_0 = \begin{pmatrix} a_0 & -b_0 & -b_0 \\ -b_0 & a_0 & -b_0 \\ -b_0 & -b_0 & a_0 \end{pmatrix}, C_0 = \begin{pmatrix} c_0 & 0 & 0 \\ 0 & c_0 & 0 \\ 0 & 0 & c_0 \end{pmatrix} \tag{3}$$

and a real positive number $\beta < 1$ introduces a complex shift by analogy with the shifted Laplacian [8]. Functions $\rho_0(z) > 0$, $a_0(z)$, $b_0(z)$ and $c_0(z)$ are the averaging of their 3D counterparts, providing proximity of operators L and L_0 to each other.

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

$$LL_0^{-1}\widetilde{\boldsymbol{v}} = \boldsymbol{f} \tag{4}$$

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

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

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. In principle, other methods of the same type are also applicable, for instance IDR [23]. 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 $\boldsymbol{w}$, i.e. computing $[w - \delta LL_0^{-1}w]$. 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 $[w - q_2]$.

To solve $L_0 q_1 = w$ we assume that function $\boldsymbol{w}(x, y, z)$ is expanded into a Fourier series with respect to the horizontal coordinates with coefficients $\widehat{\boldsymbol{w}}(k_x, k_y, z)$, 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)

$$\left[i\omega\begin{pmatrix} \rho_0 \boldsymbol{I_{3\times 3}} & 0 \\ 0 & \boldsymbol{S_0} \end{pmatrix} - ik_x\begin{pmatrix} 0 & \hat{P} \\ \hat{P}^T & 0 \end{pmatrix} - ik_y\begin{pmatrix} 0 & \hat{Q} \\ \hat{Q}^T & 0 \end{pmatrix} - \gamma(z)\begin{pmatrix} 0 & \hat{R} \\ \hat{R}^T & 0 \end{pmatrix}\frac{\partial}{\partial z}\right]\widehat{\boldsymbol{v}} = \widehat{\boldsymbol{w}}, \tag{6}$$

with the same boundary conditions as for equation (1) in the z-direction. 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 $\widehat{\boldsymbol{w}}(k_x, k_y, z)$ can be performed via the 2D Fast Fourier Transform (FFT) and after $\widehat{\boldsymbol{v}}(k_x, k_y, z)$ are found, $L_0^{-1}\boldsymbol{w}$ can be computed via the inverse 2D FFT.

It is worth mentioning, that here we assumed, that solutions to the boundary value problems (6) exist. This assumption is partly justified by numerous successful numerical tests that we've carried out.

3 Parallelization

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 $\boldsymbol{f}$. 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.

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.

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 Fig. 1.

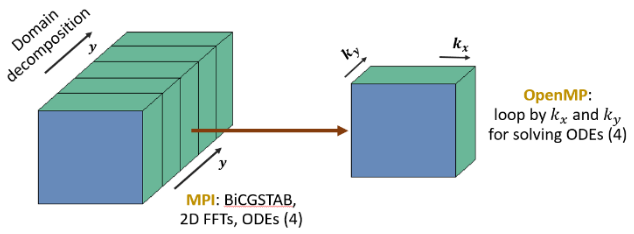

Fig. 1. Parallelization scheme.

Below, we 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. We assume a vertical point-force source as the source type unless explicitly stated to be volumetric.

For our tests, we construct a 2.5D land model (Fig. 2) from the open source 2D Marmousi model. It is discretized with a uniform grid of $551 \times 700 \times 235$ points. The horizontal cell sizes are 16 m and the vertical cell size is 8 m, corresponding to 5 and 10 points per minimal wavelength at frequency of 10 Hz respectively. Denser vertical sampling is required, because the finite-difference approximation along z-axis is only 4th order.

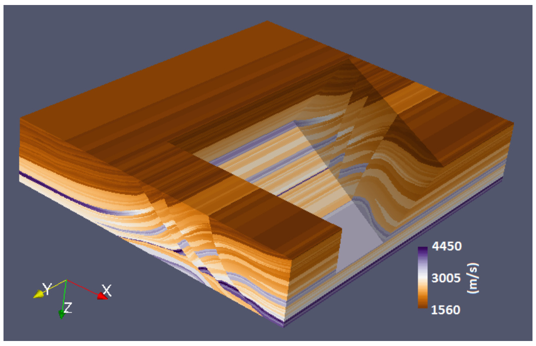

Fig. 2. 2.5D P-velocity model of 8.8 × 11.2 × 1.88 km, constructed from the Marmousi model.

3.1 MPI Strong Scaling Analysis

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 $\frac{T}{N}$. Second, the FFT, that scales as $\frac{T_{FFT}}{\alpha(N)}$, with coefficient $1 < \alpha(N) < N$. The total computational time becomes $\frac{T}{N} + \frac{T_{FFT}}{\alpha(N)}$ (here we simplify, assuming no need of synchronization) with scaling coefficient $\frac{T+T_{FFT}}{\frac{T}{N}+\frac{T_{FFT}}{\alpha(N)}}$, 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. We did not take into account OpenMP, which can be switched on for extra speed-up. It is worth noting, that we can not use MPI instead of OpenMp here, since then the scaling would degrade. MPI may have worked well if $T \gg T_{FFT}$, but this is not the case.

We estimate the strong scaling for modeling at 5 Hz in two different models. Each model comprises $200 \times 600 \times 155$ points. The first one is a subset of the model depicted in Fig. 2 with $V_{p_{max}}/V_{p_{min}} = 2.85$, $V_{s_{min}} = 867$ m/s and the second one is part of the SEG/EAGE overthrust model [2]: $V_{p_{max}}/V_{p_{min}} = 2.75$, $V_{s_{min}} = 1258$ m/s. Grid cells are 30 m. From Fig. 3 we conclude that our solver scales very well up to 64 MPI processes.

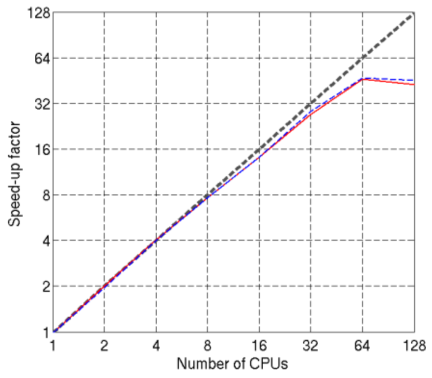

Fig. 3. Strong MPI scaling of our solver: blue dashed line is the result for the Marmousi model, the red line - for the SEG/EAGE overthrust model. The dashed grey line is the ideal scalability.

3.2 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 $f_{weak}(N) = \frac{T(N)}{T(1)}$, where $T(N)$ is the average computational runtime per iteration with N MPI processes. The ideal weak scalability corresponds to $f_{weak}(N) = 1$.

To estimate it in our case, we considered a part of the model presented in Fig. 2 of size $200 \times 25 \times 200$ points with a decreased 4 m step along the y-coordinate. After extending the model in the y-direction 64 times, we arrive at a model of size $200 \times 1600 \times 150$ points. Fig. 4 demonstrates that for up to 64 MPI processes, weak scaling of our solver has small variations around the ideal weak scalability.

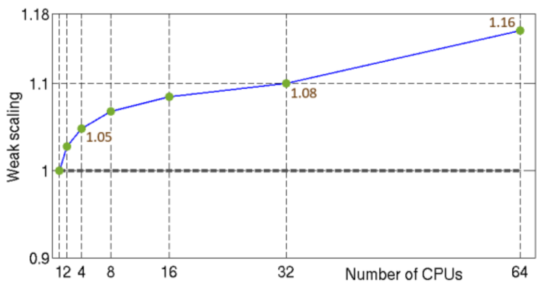

Fig. 4. Weak scaling measurements: the blue line is the result of the iterative solver and the dashed grey line is the ideal weak scaling.

3.3 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.

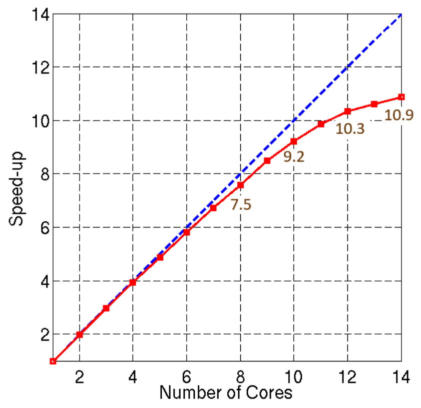

Fig. 5. Strong scalability analysis on one CPU of the part parallelized via OpenMP: the dashed blue line is the ideal scalability and the red line is the iterative solver scalability.

4 Numerical Experiments

4.1 Benchmarking

We verify our solver by comparison to the exact solution in a homogeneous medium. A frequency-domain vertical displacement $u_z(\boldsymbol{x}, \omega)$ in a homogeneous unbounded medium, resulting from a vertical point-force applied in the origin, has the analytical representation [3]:

$$u_z(\boldsymbol{x}, \omega) = \frac{S(\omega)e^{i\omega r/V_p}}{4\pi\rho V_p^2 r}\left[\gamma + (3\gamma - 1)\left(-\frac{V_p}{i\omega r}\right) + (3\gamma - 1)\left(-\frac{V_p}{i\omega r}\right)^2\right] - \frac{S(\omega)e^{i\omega r/V_s}}{4\pi\rho V_s^2 r}\left[\gamma - 1 + (3\gamma - 1)\left(-\frac{V_s}{i\omega r}\right) + (3\gamma - 1)\left(-\frac{V_s}{i\omega r}\right)^2\right], \quad (7)$$

where $S(\omega)$ is the monochromatic component of the source function, $r = \sqrt{x^2 + y^2 + z^2}$ is the radius vector, V_p and V_s are P and S-wave velocities respectively and $\gamma = \left(\frac{z}{r}\right)^2$. To get the frequency-domain vertical velocity, we multiply the displacement by $i\omega$ in the Fourier domain (for the derivative):

$$v_z(\boldsymbol{x}, \omega) = i\omega \cdot u_z(\boldsymbol{x}, \omega). \tag{8}$$

Our example comprises a volume of size $12 \times 12 \times 4.5$ km filled with constant elastic properties: $V_p = 2600$ m/s, $V_s = 1500$ m/s and $\rho = 2210$ kg/m^3. This volume is discretized using a uniform grid of 60 m in the horizontal directions and 15 m in the vertical direction. An analytical solution for vertical velocity is computed for a frequency of 10 Hz using formula (8), assuming that the origin is the middle of the volume. To obtain our solver solution, we surround the computational area with absorbing layers. In Fig. 6 the corresponding solutions along the vertical line $x = 6400$ m and $y = 3200$ m are given, showing good agreement between the results. We compute the root mean square (RMS) error in percent

$$\text{RMS}= 100 \cdot \frac{\|u_{solver} - u_{exact}\|_{L_2}}{\|u_{exact}\|_{L_2}}, \tag{9}$$

with u_{exact} and u_{solver} being the exact and solver solutions respectively. In our example, the RMS difference is 0.88%. This small error can be reduced by using denser vertical sampling or using a higher order finite-difference approximation for solving the boundary value problems (6).

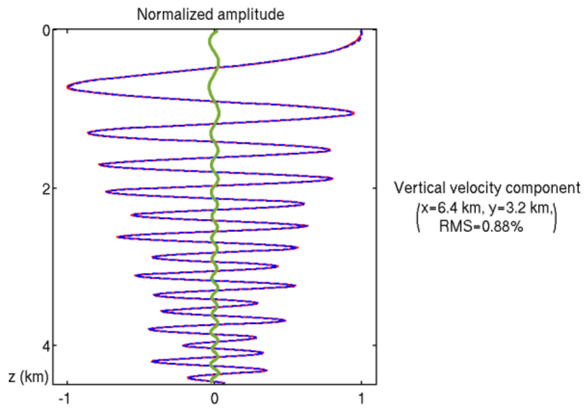

Fig. 6. 1D vertical profiles (real part) of V_z computed in the homogeneous model at receiver location x=6400 m, y=3200 m. The exact solution (dashed blue line) overlies our solver solution (solid red line) and the residuals (solid green line) are multiplied by 5.

We compare the convergence rate of our solver to another 3D elastic iterative solver – CARP-CG [15]. Generating the solution at 7.5 Hz using the homogeneous model our solver converges in 39 iterations in 794 seconds, whereas other solver converges in 1402 iterations in 1244.42 seconds (these data were taken from Table 9 in [15]). The number of cores involved in the computations were the same for both solvers. It is worth mentioning, that the computational time comparison is a bit unfair, since the hardware for running these solvers were slightly different.

4.2 Convergence Analysis in the Marmoussi Model

To understand how the convergence rate varies with increasing frequency we consider the full 2.5D model depicted in Fig. 2 containing more than 90 million cells. Simulations were performed at different low frequencies from 2 to 10 Hz, considered a sufficient range for macro velocity reconstruction with FWI. From Fig. 7 we infer, that for the higher frequencies the convergence is slower. In this particular case, the convergence curve varies around the linear increase with slope 10. It is worth mentioning, that pure BiCGSTAB would diverge in such a model (number of iterations > 10000) at any of those frequencies. The 10 Hz monochromatic component of the computed wavefield is presented in Fig. 8. Using 9 nodes with 7 MPI processes per node and 4 cores per process, the total computation time is 348 minutes.

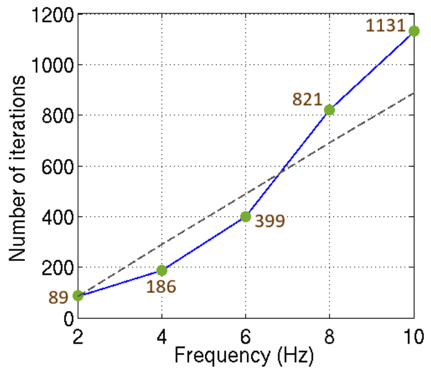

Fig. 7. Convergence versus frequency in the 2.5D Marmousi model: the blue line is the iterative solver convergence curve and the dashed grey line is the linear increase with slope 10.

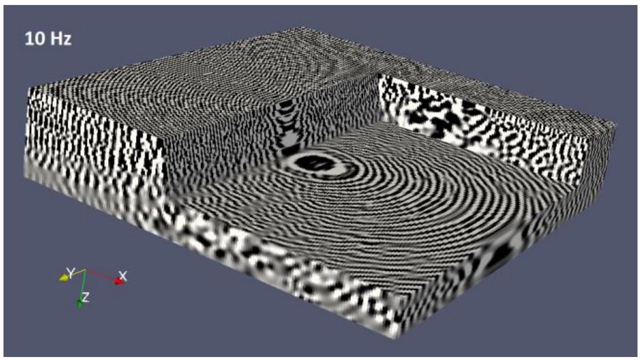

Fig. 8. A 3D view of the real part of a frequency-domain wavefield of the V_z component computed for the model depicted in Fig. 2.

5 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. Another strategy, that may be also considered, is parallelization using domain decomposition along the vertical direction for solving the boundary value problems for equation 4.

Acknowledgments

We are grateful to Vincent Etienne and Michael Jervis for reviewing of our manuscript. Special thanks to Maxim Dmitriev for useful discussions and advice on this topic. Two of the authors (Victor Kostin and Vladimir Tcheverda) have been sponsored by the Russian Science Foundation grant 17-17-01128.

References

1. Albanese C. and K. Gilblom: This oil major has a supercomputer the size of a soccer field: Bloomberg, January 18 (2018).
2. Aminzadeh, F., Brac, J. and T. Kuntz: 3-D salt and overthrust models: SEG/EAGE modelling series, n. 1, SEG Book Series, Tulsa, Oklahoma (1997)
3. Aki, K. and P.G. Richards: Quantitative seismology, Theory and methods, Volume 1: W.H. Freeman and Co (1980).
4. Belonosov, M., Kostov, C., Reshetova, G., Soloviev, S. and V. Tcheverda: Parallel Numerical Simulation of Seismic Waves Propagation with Intel Math Kernel Library: LNCS, 7782, 153-167 (2013).
5. Belonosov, M., Dmitriev, M., Kostin, V., Neklyudov, D. and V. Tcheverda: An iterative solver for the 3D Helmholtz equation: Journal of Computational Physics, 345, 330-344 (2017).
6. Berenger, J. P.: Three-dimensional perfectly matched layer for the absorption of electromagnetic waves: Journal of Computational Physics, 127, 363-379 (1996).
7. Darbas, M. and F. Louer: Analytic preconditioners for the iterative solution of elastic scattering problems: HAL, hal-00839653, 1-32 (2013).
8. Erlangga, Y.A. and R. Nabben: On a multilevel Krylov method for the Helmholtz equation preconditioned by shifted Laplacian: Electronic Transactions on Numerical Analysis, 31, 403-424 (2008).
9. Etienne, V., Tonellot, T., Thierry, P., Berthoumieux, V. and C. Andreolli: Optimization of the seismic modeling with the time-domain finite-difference method: 84th Annual International Meeting, SEG, Expanded Abstracts, 3536-3540 (2014).
10. Intel, 2018, Intel®Math Kernel Library (Intel®MKL): https://software.intel.com/en-us/intel-mkl.
11. Khajdukov, V., Korneev, V., Kostin, V., Kovalevsky, V., Malyshkin, V., Tcheverda, V. and D. Vishnevsky: Modelling of seismic waves propagation for 2D media (direct and inverse problems): Lecture Notes in Computer Sciences, vol. 1277, pp. 350–357 (1997).

12. Kostin, V., Lisitsa, V., Reshetova, G. and V. Tcheverda: Local time-space mesh refinement for simulation of elastic wave propagation in multi-scale media: Journal of Computational Physics, 281, 669 - 689 (2015).
13. Kostin, V., Neklyudov, D., Tcheverda, V., Belonosov, M. and M. Dmitriev: 3D elastic frequency-domain iterative solver for full-waveform inversion: 86th Annual International Meeting, SEG, Expanded Abstracts, 3825-3829 (2016).
14. Kostin, V., Solovyev, S., Liu, H. and A. Bakulin: HSS cluster-based direct solver for acoustic wave equation: 87th Annual International Meeting, SEG, Expanded Abstracts, 4017-4021 (2017).
15. Li, Y., Metivier, L., Brossier, R, Han, B. and J. Virieux: 2D and 3D frequency-domain elastic wave modeling in complex media with a parallel iterative solver: Geophysics, 80, T101-T118 (2015).
16. Lisitsa, V., Tcheverda, V. and C. Botter. Combination of discontinuous Galerkin method with finite differences for simulation of elastic wave: Journal of Computational Physics, 311, 142 - 157 (2016).
17. Operto, S., Virieux, J., Amestoy, P., L'Excellent, J., Giraud, L. and H. Hadj: 3D finite-difference frequency-domain modeling of visco-acoustic wave propagation using a massively parallel direct solver: A feasibility study: Geophysics 72, SM195-SM211 (2007).
18. Pissarenko, D, Reshetova, G. and V. Tcheverda: 3D finite-difference synthetic acoustic log in cylindrical coordinates: parallel implementation: J. of Computational and Applied Mathematics, vol. 234, n. 6, pp. 1766–1772 (2010).
19. Pratt, R.G.: Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model: Geophysics, 64, 888-901 (1999).
20. Rizzuti, G. and W.A. Mulder: A Multigrid-based Iterative Solver for the Frequency-domain Elastic Wave Equation: 77th EAGE Conference and Exhibition, Expanded Abstracts, 1-4 (2015).
21. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edition: SIAM, Philadelphia, PA (2003).
22. Sirgue, L., Etgen, J., Albertin, U. and S. Brandsberg-Dahl: System and method for 3D frequency domain waveform inversion based on 3D time-domain forward modeling: U. S. Patent, 11/756,384 (2007).
23. Sonneveld, P. and M.B. van Gijzen: IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations: SIAM J. Sci. Comput., 31, 1035-1062 (2008).
24. Symes, W.W.: Migration velocity analysis and waveform inversion: Geophysical Prospecting, 56, n. 6, 765-790 (2008).
25. Van Der Vorst, H.A.: BI-CGSTAB: a fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems: SIAM Journal on Scientific and Statistical Computing, 13, n. 2, 631-644 (1992).
26. Virieux J., Operto S., Ben-Hadj-Ali H., Brosssier R., Etienne V., Sourber F., Giraud, L. and A. Haidar: Seismic wave modeling for seismic imaging: The Leading Edge, 28, 538-544 (2009).
27. Wang, S.V., de Hoop, M., Xia, J. and X.S. Li: Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3-D anisotropic media: Geophysical Journal International, 191, 346-366 (2012).