3D frequency-domain elastic wave modeling with the spectral element method using a massively parallel direct solver

Yang Li¹, Romain Brossier¹, and Ludovic Métivier²

ABSTRACT

Complex topography, the free-surface boundary condition, and anelastic properties of media should be accounted for in the frame of onshore geophysical prospecting imaging, such as full-waveform inversion (FWI). In this context, an accurate and efficient forward-modeling engine is mandatory. We have performed 3D frequency-domain anisotropic elastic wave modeling by using the highly accurate spectral element method and a sparse multifrontal direct solver. An efficient approach similar to computing the matrix-vector product in the time domain is used to build the matrix. We validate the numerical results by comparing with analytical solutions. A parallel direct solver, the sparse direct multifrontal massively parallel solver (MUMPS), is used to solve the linear system. We find that a hybrid implementation of message passing interface and open multiprocess is more efficient in flops and memory cost. The influence of the deformed mesh, free-surface boundary condition, and heterogeneity of media on MUMPS performance is negligible. Complexity analysis suggests that the memory complexity of MUMPS agrees with the theoretical order $O(N^4)$ (or $O(N^{1.5})$ with an efficient matrix reordering method) for an $N^3$ grid when nontrivial topography is considered. With the available resources, we conduct a moderate scale modeling with a subset of the SEAM Phase II Foothills model, where 60 wavelengths in the $x$-axis are propagated. Computing one gradient of FWI based on this model using the frequency-domain modeling is shown to require similar or fewer computational resources than what would be required for a time-domain solver, depending on the number of sources, while larger memory is necessary. An estimation of the increasing trend indicates that approximately 20 Tb of memory would be required for a $50 \times 50 \times 50$ wavelength modeling. The limit of MUMPS scalability hinders the application to larger scale applications.

INTRODUCTION

Full-waveform inversion (FWI) is a high-resolution seismic imaging technique, used in academia for global and regional tomography and in industry for seismic exploration (Operto et al., 2006; Sirgue et al., 2009; Virieux et al., 2017). FWI is based on an optimization process that iteratively fits the synthetic and observed data to update the model parameters. The main computational cost comes from the repeated solution of wave equations to build the model updates. FWI was first developed in the time domain (Lailly, 1983; Tarantola, 1984). Frequency-domain FWI has been investigated using the acoustic approximation (Pratt, 1999; Operto et al., 2015; Górszczyk et al., 2017). However, the acoustic approximation can no longer be used for the inversion of land data because the anelastic and viscoelastic effects dominate. Moreover, the cost of large-scale 3D frequency-domain anisotropic viscoelastic wave modeling is relatively prohibitive. Thus, most current 3D imaging applications rely on time-domain forward modeling (Warner et al., 2013; Yang et al., 2018; Trinh et al., 2019) using two levels of parallelization over sources and domain decomposition, with a relatively modest memory requirement. Nevertheless, frequency-domain wave modeling possesses several advantages over its time-domain counterpart. First, the restriction of the Courant-Friedrichs-Lewy (CFL) (Courant, 1928) stability condition no longer exists in the frequency domain. This condition imposes a strict bound for the time marching step and may affect the efficiency of time-domain
modeling, especially when considering unconsolidated, low S-wave velocity zones. In addition, the seismic attenuation (viscosity) can be incorporated easily in the frequency domain by using the complex-valued elastic modulus (Carcione, 2015), whereas time-domain techniques rely on the generalized Maxwell/Zener body (GM/BGZ) or standard linear solid (SLS) mechanisms that increase the computational cost of the modeling. In the framework of time-domain FWI, introducing attenuation also incurs additional complexity related to gradient building. Trade-offs between memory and computation are generally found using checkpointing-based techniques (Griewank and Walther, 2000; Symes, 2007; Yang et al., 2016). This is not the case for frequency-domain FWI. To invert a realistic scale data set, time-domain FWI generally relies on source subsampling techniques. Conversely, frequency-domain FWI inverts for a limited number of discrete frequencies and can account for all of the sources, provided that a direct solver is used to solve the linear system associated with the frequency-domain wave equation. The resolution for different sources involves only forward and backward substitutions once the matrix factorization is done (Duff and Reid, 1983). These advantages, illustrated by the recent successful applications of 3D frequency-domain FWI for a viscoacoustic vertical transverse isotropy (VTI) medium in offshore environment (Operto et al., 2015; Operto and Miniussi, 2018), are our motivation to investigate the potential and limits of 3D frequency-domain anisotropic elastic wave modeling in the frame of elastic FWI.

The main discretization methods considered for wave equations are the finite-difference method (FDM) (Graves, 1996; Kristek and Moczo, 2003), discontinuous Galerkin method (DG and more recently hybridizable DG method, Käser et al., 2008; Etienne et al., 2010; Bonnasse-Gahot et al., 2018), and the spectral element method (SEM) (Komatitsch and Vilotte, 1998; Komatitsch and Tromp, 1999). FDM is popular for 3D elastic wave modeling due to its simplicity and high efficiency. However, it is required to refine the grid (Bohlen and Saenger, 2006) or use specific coordinate transform techniques (Tarrass et al., 2011; Zhang et al., 2012, 2016; de la Puente et al., 2014; Shragge, 2014; Chung et al., 2015; Cao et al., 2018) when dealing with complex topography and the free-surface boundary condition (FSBC). This would relatively counteract its efficiency in terms of computational and memory complexity. Konuk and Shragge (2019) recently show that topography can be included into viscoelastic anisotropic finite-difference modeling with comparable computational complexity to that of the Cartesian scenarios. The SEM, as a specific finite element method (FEM), has been investigated particularly in seismology and seismic imaging (Komatitsch and Vilotte, 1998; Komatitsch and Tromp, 1999; Trinh et al., 2019). The FSBC is naturally satisfied in the weak form of the wave equation, and an adaptive design of the mesh could further simplify the realization of complex topography and account for the variations in the media. The specific character of SEM consists in using a mesh of hexahedra in 3D and choosing the Gauss-Lobatto-Legendre (GLL) points for the integration, as well as the Lagrange polynomial for the interpolation. Using high-order Lagrange polynomials and Gauss quadrature on GLL points enables spectral convergence when solving smooth problems. Such high-order accuracy, combined with the capability of dealing with complex topography and FSBC, can reduce the size of the model, compared with other discretization techniques. In addition, general anisotropy can be considered without making an extra effort for SEM, which is not the case for conventional FDM.

To solve the large sparse linear system obtained from the discretization of the wave equation, one may apply direct methods, for example, lower-upper (LU) decomposition (Duff and Reid, 1983; Liu, 1992; Amestoy et al., 2000; Wang et al., 2012) or iterative methods (Erlangga et al., 2006; Plessix, 2007) or hybrid methods (Haidar, 2008; Sourbier et al., 2011). Due to the scale of the problem and the computational and memory cost, most 3D frequency-domain elastic wave modeling is carried out using iterative solvers with specifically designed preconditioners (Li et al., 2015; Rizziuti and Mulder, 2016; Gatto and Hesthaven, 2017; Baumann et al., 2018; Belonosov et al., 2018). In most cases, the number of iterations increases as the size of the model grows and efficient preconditioners are necessary to guarantee the convergence of iterative solvers in the 3D frequency-domain elastic wave modeling. However, it is hard to design a general preconditioner that is valid for different physical settings such as complex topography, FSBC, and high-contrast-medium variations. Direct solvers, on the contrary, do not have those convergence difficulties. The drawback is the huge memory cost and flops required during the matrix factorization. A tentative experiment has been carried out in which 3D frequency-domain elastic modeling is implemented using the discontinuous Galerkin method and the parallel sparse direct multifrontal massively parallel solver (MUMPS) a decade ago (Etienne et al., 2008). The result shows that the computer resources available at that time would make possible the propagation over only a few wavelengths. More recently, Gosselin-Cliche and Giroux (2014) give an example of performing the 3D frequency-domain elastic modeling using FDM and the parallel direct solver MUMPS. The number of wavelengths that they reach is approximately eight. As for the application of SEM in the frequency domain, Zhou and Greenhalgh (2011) propose an SEM-like method using a Gauss quadrature grid approach. Shi et al. (2016) also apply SEM to elastic and acoustic wave modeling to enable the acoustic-solid interaction. A sequential sparse direct solver, UMFPACK, is used as the linear solver for a relatively small linear system with fewer than one million unknowns. These studies, more methodologically oriented, do not tackle large-scale problems.

In this study, we present a feasibility study of 3D frequency-domain elastic wave modeling using SEM, taking into account general settings, the heterogeneity, anisotropy, viscosity, complex topography, and FSBC. The formulation of SEM for 3D frequency-domain elastic wave equation is given in the first part. Perfectly matched layers (PMLs) are implemented to avoid reflections from the artificial boundaries. A Cartesian-based fully deformed mesh based on the hexahedra is considered. The structure and building process of the impedance matrix are thoroughly explained. An efficient approach similar to computing the time-domain matrix-vector product is used to build the matrix row by row, MUMPS 5.1.2, the latest online version (full-rank [FR], MUMPS team, 2017), is used to solve the large sparse linear system. The numerical results are validated by comparing with the analytical solutions. The scalability of pure message passing interface (MPI) implementation of MUMPS is shown to be not very satisfactory for large-scale modeling (Amestoy et al., 2001; Mary, 2017). A hybrid implementation of MPI and Open Multi-Processing (OpenMP) presents better performance. The influence of deformed mesh and FSBC on MUMPS is also checked. A heuristic complexity analysis suggests that the computational and memory complexity of MUMPS agree well with the theoretical estimations of $O(N^6)$ and $O(N^4)$ for a 3D $N^3$ model.
The largest modelings achieved currently are presented for the Cartesian nondeformed mesh and vertically deformed mesh. A modeling, using a fifth-order SEM, with $30 \times 30 \times 30$ elements, that is, 30 wavelengths in each dimension, is performed using 384 cores in 2085 s. A modeling with a more realistic exploration-scale model (part of the SEAM Phase II Foothills model) is conducted. An estimation of the computational cost for FWI using the frequency-domain modeling method is given. For a limited number of sources (a few tens), it is shown to be similar, in scalar hours, as what would be required using a time-domain approach. However, the general trend as the number of sources increase is much more favorable to the frequency-domain approach: For a larger number of sources (a few hundreds), the difference in scalar hours between the two strategies reaches one order of magnitude. Comparison of computational cost between FDM and SEM is discussed. The increasing trend of computational cost for modelings with up to $50 \times 50 \times 50$ elements is estimated based on the obtained data. The increasing memory demand during the LU decomposition and the lack of scalability of MUMPS make it not feasible to investigate larger-scale modeling by now. The perspectives and further research will try to overcome this current limit.

### THEORY AND IMPLEMENTATION

#### Frequency-domain SEM for the 3D elastic wave equation

The 3D frequency-domain elastic wave equation in a general anisotropic medium reads (Aki and Richards, 1980)

$$\rho \omega^2 u_j + \frac{\partial}{\partial x_i} \left( c_{ijkl} \frac{\partial u_k}{\partial x_l} \right) + s_j(\omega, \mathbf{r}_x) = 0, \quad i, j, k, l = 1, 2, 3, \tag{1}$$

where $\rho$ is the density, $\omega$ is the angular frequency, $u_j$ is the displacement vector, $c_{ijkl}$ is the elastic modulus tensor, and $s_j(\omega, \mathbf{r}_x)$ is the point source force vector located at $\mathbf{r}_x$. Note that the Einstein convention is used in equation 1, which implies summation over all repeated indices. Seismic attenuation in viscoelastic media is easily incorporated by using complex-valued elastic moduli (Carcione, 2015).

To prevent artificial reflections from the boundaries of the limited computational domain, absorbing boundary conditions have to be incorporated. PMLs (Bérenger, 1994) are considered in this study. We use the anisotropic PML developed in Zhou and Greenhalgh (2011) and Shi et al. (2016) in the following experiments. The anisotropic PML could be directly realized by modifying the elastic parameters and the density. Details of developing the anisotropic PML is given in Zhou and Greenhalgh (2011) and Shi et al. (2016). Equation 1 with PML reads as follows, as derived in Shi et al. (2016):

$$\tilde{\rho} \omega^2 u_j + \frac{\partial}{\partial x_i} \left( \tilde{c}_{ijkl} \frac{\partial u_k}{\partial x_l} \right) + s_j(\omega, \mathbf{r}_x) = 0, \quad i, j, k, l = 1, 2, 3, \tag{2}$$

where the new parameters are defined as

$$\tilde{\rho} = \rho p_x p_x p_x, \quad \tilde{c}_{ijkl} = c_{ijkl} p_x p_x p_x, \tag{3}$$

Equation 1 with PML reads as follows, as derived in Shi et al. (2016):

$$\Omega \omega^2 \phi + \int_{\Omega} \frac{\partial}{\partial x_i} \left( c_{ijkl} \frac{\partial u_k}{\partial x_l} \right) \phi d\Omega + \int_{\Omega} s_j(\omega, \mathbf{r}_x) \phi d\Omega = 0, \tag{4}$$

where $p_x$ is the damping function defined in the coordinate stretching technique. Equation 2 has the same structure as the original wave equation 1. Because $p_x = 1$ outside the PML zone, equation 2 is then exactly equal to equation 1 therein. Within the PML zone, the complex parts of the modified stiffness tensor $\tilde{c}_{ijkl}$ ensure the absorption of outgoing waves. Note that this PML technique is also compatible with the implementation of attenuation in the frequency domain following, for instance, the Kolsky-Futterman model (Kolsky, 1956; Futterman, 1962), which implies a complex-valued stiffness tensor.

### SEM discretization

The weak form of equation 1 reads

$$\omega^2 \int_{\Omega} \rho u_j \phi d\Omega + \int_{\Omega} \frac{\partial}{\partial x_i} \left( c_{ijkl} \frac{\partial u_k}{\partial x_l} \right) \phi d\Omega + \int_{\Omega} s_j(\omega, \mathbf{r}_x) \phi d\Omega = 0, \tag{5}$$

where $\Omega$ is the volume of interest and $\phi$ is the test function. Through integration by parts (the Gauss divergence theorem), the second term splits into two and we have

$$\omega^2 \int_{\Omega} \rho u_j \phi d\Omega + \int_{\Omega} c_{ijkl} \frac{\partial u_k}{\partial x_l} \frac{\partial \phi}{\partial x_i} d\Omega - \int_{\partial \Omega} \phi \left( c_{ijkl} \frac{\partial u_k}{\partial x_l} \cdot n_i \right) dS + \int_{\Omega} s_j(\omega, \mathbf{r}_x) \phi d\Omega = 0, \tag{6}$$

where $\partial \Omega$ is the surface of $\Omega$ and $n_i$ is the component of the outward normal unit vector. The surface integral term vanishes naturally on the top surface due to the free-surface boundary condition. For the other surfaces of $\Omega$, applying the absorbing boundary conditions renders displacements $u_j$ decaying to zero on these surfaces and with enough elements in the absorbing layer the partial derivatives of $u_j$ are consequently small and negligible. We obtain

$$\omega^2 \int_{\Omega} \rho u_j \phi d\Omega + \int_{\Omega} c_{ijkl} \frac{\partial u_k}{\partial x_l} \frac{\partial \phi}{\partial x_i} d\Omega + \int_{\Omega} s_j(\omega, \mathbf{r}_x) \phi d\Omega = 0. \tag{7}$$

The specificity of SEM consists of using a mesh of quadrangles in 2D and hexahedra in 3D. The GLL points are selected for the Lagrange interpolation polynomials and Gauss quadrature. Using high-degree Lagrange polynomials and Gauss quadrature on GLL points enables enhanced accuracy when solving smooth problems (Deville et al., 2002).

The discretized wave equation is given as follows:

$$Au = s, \tag{8}$$

where $A = \omega^2 M + K$ is the impedance matrix, with $M$ being the mass matrix related to density $\rho$ and $K$ being the stiffness matrix related to elastic parameters $c_{ijkl}$. Vector $u$ is the discretized displacement, and $s$ represents the discretized source vector. Compared
with conventional FEM where the orthogonality between basis functions is uniquely due to the nonoverlapping local functions, the orthogonality in SEM is related to the topological nature and the analytical nature of the basis functions (Deville et al., 2002). This property leads to a diagonal mass matrix \( M \), whereas it is not the case for conventional FEM. This is specifically interesting for time-domain wave modeling based on explicit time stepping because no matrix inversion is required with SEM. In the frequency domain, the mass matrix and the stiffness matrix are combined as the final impedance matrix. As for the integration of partial derivative of function, it is performed by computing the derivative of the Lagrange polynomials, for which analytical formulas are available. Further details on SEM can be found in Komatitsch and Vilotte (1998), Komatitsch and Tromp (1999), and Schubert (2003).

**Stiffness matrix**

To explain the building process of the stiffness matrix, we take as an example the discretization of one term in the expended equation 7 to illustrate the connection between the node points, that is, the structure of the stiffness matrix. The detailed matrix building process is given in Appendix A. Without loss of generality, the elastic parameter \( c \) is used instead of a specific \( c_{ijkl} \). With test function \( \phi(x) = \ell(x) \ell(y) \ell(z) \), where \( \ell \) is the 1D Lagrange polynomial, we have

\[
\int \mathbf{c} \partial \mathbf{u} \partial \mathbf{u} \, d\Omega = \int \int \int \left[ \sum_{l,m,a} c_{lma} \ell_l(\xi) \ell_m(\eta) \ell_a(\zeta) \right] \partial_i \mathbf{u} = \int \sum_{l, j, k} c_{ijkl} \ell_l(\xi) \ell_j(\eta) \ell_k(\zeta) \partial_i \mathbf{u} \partial_i \mathbf{u} \, d\Omega
\]

where \( c_{lma} \) is the elastic parameter \( c \) at grid point \( (l, m, a) \). The Jacobian matrix at point \( (p, q, r) \) and \( w_pq \) are the weights of Gauss quadrature. The property of Lagrange polynomial \( \ell_l(\xi) = \delta_{lj} \) is used. This equation explains the distribution of nonzero entries in the row corresponding to point \( u_{pqr} \). The red terms in the sum indicate that one point in an element is connected to those points on three “planes,” which intersect at this point; that is, \( u_{pqr} \) is connected to \( u_{ij} \) for \( i = 1, \ldots, n_{GLL} \), \( u_{ik} \) for \( k = 1, \ldots, n_{GLL} \) and \( u_{jk} \) for \( j = 1, \ldots, n_{GLL} \), \( n_{GLL} \) is the number of GLL points in each dimension in one element. Thus, only the coefficients of these terms in row \( u_{pqr} \) are nonzero. The discretization of other terms in equation 7 has a similar formula and is omitted here. Thus, in general, the point with the largest number of connected points is a vertex of elements and is shared by eight surrounding elements. In this case, the point is connected to 12 “elemental planes.” Figure 1a shows the distribution of points in a fifth-order SEM element and a detailed illustration of the connection mode is given (Figure 1b). The blue cube at the center is an example of the mentioned point. The points on these three planes, that is, the 12 elemental planes, are connected to this point. If the order of SEM is \( p_{SEM} \), the number of connected points is \( 36 n_{SEM}^2 + 18 n_{SEM} + 3 \), that is, 993 for a fifth-order SEM. This number is much larger than that of an FDM, for instance, 51 for a fourth-order staggered-grid FDM (Li et al., 2015). The gain is the ability of dealing with complex topography, adaptive deformed mesh, and free-surface boundary condition as mentioned. It is worthwhile to remark that if SEM is implemented within a regular Cartesian mesh, which means no deformed point is considered, the Jacobian matrix is thus diagonal and the connection mode changes to three lines intersecting at the center point in Figure 1b. The number of connected points reduces to \( 18n + 3 \), that is, 93 for a fifth-order SEM.

Note that the mesh is designed based on the SEM46 package (Trinh et al., 2019), which is adapted to exploration-scale seismic imaging problems. We consider a Cartesian-based mesh; that is, the number of elements in each dimension is constant and the mesh is deformed only vertically to handle the complex topography. The unknowns \( u_{x1}, u_{x2}, u_{x3} \) are lexicographically ordered. Denoting \( N_1, N_2, \) and \( N_3 \) as the numbers of degrees of freedom (DOFs) in each dimension, the bandwidth of the generated matrix is \( 30 N_3 N_1 \), compared to \( 18 N_2 N_3 \) for the fourth-order staggered-grid FDM. The bandwidth refers to the largest distance between nonzero diagonals of the matrix, which influence the fill-in effect (the additional nonzeros introduced during the factorization), that is, the memory cost and the flops during the LU factorization of the matrix. The structures of matrices from a \( 3 \times 3 \times 3 \) element model for a fifth-order SEM and a \( 16 \times 16 \times 16 \) grid for a fourth-order staggered-grid FDM (with the same number of DOFs = 12,288) are shown in Figure 2. The SEM matrix is denser and has a bandwidth larger than that of FDM as expected. Further comparison of computational cost between FDM and SEM is given in the “Discussions” section.

**MUMPS**

MUMPS (MUMPS 5.1.2, full-rank version, MUMPS team, 2017) is used to solve the linear system expressed in equation 8. It is based on a multifrontal method (Duff and Reid, 1983), which recasts the original matrix into multiple frontal matrices and computes the LU decomposition of these smaller matrices to save memory and computational cost. The linear system is solved in three main steps. The first is an analysis process in which a reordering of the matrix and a symbolic factorization are performed. Reordering the matrix can drastically mitigate the fill-in effect. In this paper, we use the METIS algorithm for matrix reordering in the analysis step, which is based on the multilevel graph partitioning paradigm (Karypis, 2013). Experiments show that among the different methods, METIS provides the most efficient matrix reordering in terms of flops and memory cost. The symbolic factorization computes the so-called elimination tree, which is used to estimate the flops and memory cost for the factorization and resolution. The second step is LU factorization. The third step is to perform the forward and backward substitutions for each right-hand side or seismic source. Note that such substitutions are done efficiently with linear
Numerical Validation and Performance Analysis

Validation of numerical solutions

We validate the numerical solutions of frequency-domain SEM by comparing them with analytical solutions. Considering isotropic homogeneous elastic models, the frequency-domain analytical solution for a point force applied along the z-axis is given by (Pilant, 1979; Gosselin-Cliche and Giroux, 2014)

\[
\mathbf{u} = \frac{F(\omega)}{4\pi \rho c^2 \omega^2} \left[ k_x^2 e^{-ik_x R} \mathbf{e}_x + \frac{1}{k_x} \partial_z \left( e^{-ik_x R} \mathbf{e}_x \right) - k_y^2 e^{-ik_y R} \mathbf{e}_y + \frac{1}{k_y} \partial_z \left( e^{-ik_y R} \mathbf{e}_y \right) - k_z^2 e^{-ik_z R} \mathbf{e}_z + \frac{1}{k_z} \partial_z \left( e^{-ik_z R} \mathbf{e}_z \right) \right].
\]

where \( F \) is the Fourier transform of the source wavelet, \( k_x, k_y \) and \( k_z \) are the P and S wavenumbers, respectively, \( R = \| \mathbf{x} - \mathbf{x}_s \|_2 \) is the distance from the source, and \( \mathbf{e}_z \) is the unit vector of the z-axis.

The expanded expression reads

\[
\mathbf{u} = \frac{F(\omega)}{4\pi \rho c^2 \omega^2} \left[ ik_x (R^2 k_y^2 - 3 - 2ik_x R) e^{-ik_x R} + (3 + 3ik_x R - R^2 k_y^2) e^{-ik_x R} \mathbf{e}_x + \frac{ik_x R}{R} \partial_z (e^{-ik_x R} \mathbf{e}_x) + ik_y (R^2 k_z^2 - 3 - 2ik_y R) e^{-ik_y R} + (3 + 3ik_y R - R^2 k_z^2) e^{-ik_y R} \mathbf{e}_y + \frac{ik_y R}{R} \partial_z (e^{-ik_y R} \mathbf{e}_y) + ik_z (R^2 k_x^2 - 3 - 2ik_z R) e^{-ik_z R} + (3 + 3ik_z R - R^2 k_x^2) e^{-ik_z R} \mathbf{e}_z + \frac{ik_z R}{R} \partial_z (e^{-ik_z R} \mathbf{e}_z) \right].
\]

Note that the GLL points are not uniformly distributed. Thus, the calculation of equation 11 should be carried out on the GLL points, instead of the uniformly distributed finite-difference points.

Velocities \( V_P \) and \( V_S \), density \( \rho \), size of element, frequency \( f \), and the number of DOFs per wavelength are summarized in Table 1. With \( V_S = 2500 \text{ m/s} \) and \( f = 25 \text{ Hz} \), the wavelength \( \lambda \) is equal to 100 m. Thus, setting one element in each wavelength, that is, five grid points per wavelength, satisfies the accuracy requirement for the fifth-order SEM. The same setting is used in the following sections unless otherwise specified. The real parts of 3D elastic wavefields \( u_x, u_y \) and \( u_z \) are shown in Figure 3. As shown, the misfit between the analytical and numerical solutions is weak. In Figure 4, profiles are extracted from the 3D volumes, to present more details for comparison. The profiles are extracted at \( x = 664 \text{ m} \) (the first column) and \( z = 664 \text{ m} \) (the second column) in the x-z plane sliced at \( y = 1235 \text{ m} \). The relative error of the numerical wavefield compared with the analytical solutions is defined as

**Table 1. Parameter settings for frequency-domain elastic wave modeling.**

| \( N_e/\text{dim} \) | \( N_{PML} \) | \( |e| \text{ (m)} \) | \( V_P \text{ (m/s)} \) | \( V_S \text{ (m/s)} \) | \( \rho \text{ (g/cm}^3) \) | \( f \text{ (Hz)} \) | DOFs/\( \lambda \) |
|-----------------|-------------|----------------|----------------|----------------|----------------|----------|----------|
| 20              | 2           | 100            | 5000           | 2500           | 1              | 25       | 5        |

**Notes:** \( N_e/\text{dim} \), number of elements in each dimension; \( N_{PML} \), number of elements in PML; \( |e| \), element size; \( (V_P, V_S, \rho) \), physical parameters; \( f \), modeled frequency; and \( \text{DOFs}/\lambda \), the number of degrees of freedom per wavelength.
where $u$ could be $u_x$, $u_y$, or $u_z$. We have $\text{error}_{u_x} = 0.23\%$, $\text{error}_{u_y} = 0.22\%$, and $\text{error}_{u_z} = 0.90\%$, respectively. The relative errors of $u_z$ are larger because the source is a $z$-axis force. The numerical $u_z$ wavefields in the element where the source is activated deviate more from the analytical solutions as observed.

### Scalability of MUMPS

**Scalability for pure MPI implementation**

The strong scalability of MUMPS is investigated by increasing the number of cores (MPI) for modeling on a single model. A model with $20 \times 20 \times 20$ elements is selected. The physical parameters are the same as in Table 1 except for the number of elements in each dimension. Table 2 summarizes the number of DOFs, the number of nonzeros of the linear system ($N_{NNZ}$), and the number of MPI processes ($N_{MPI}$) used. The outputs of MUMPS, including the average and maximum memory cost ($M_A/M_M$), total memory cost ($M_T$), time for analysis ($T_A$), distribution of matrix ($T_D$), factorization ($T_F$), resolution for one RHS ($T_S$), and the total time ($T_T$) are summarized in Table 3. As expected, the total memory cost and the time for distribution of the matrix increase as the number of cores increases. The scalability curves of the factorization time are presented in Figure 5 with the red curves for MUMPS and blue for ideal scalability. As expected, the scalability of MUMPS for pure MPI implementation is not very satisfactory for more than 100 MPI processes (Amestoy et al., 2001; Mary, 2017). A hybrid implementation of MPI and OpenMP is investigated in the following section aiming at improving the MUMPS scalability.

**Scalability for hybrid implementation of MPI and OpenMP**

Two kinds of parallelism, referred to as tree parallelism (elimination tree) and node parallelism, are exploited in MUMPS (L’Éxcellent, 2012; Amestoy et al., 2016). Both levels of parallelism are exploited.
with MPI. In tree parallelism, frontal matrices in different subtrees are processed concurrently by different processes, whereas in node parallelism, large enough frontal matrices are mapped on several processes. Within each MPI process, node parallelism is exploited at a finer level with multithreading. This allows a better use of memory hierarchy and better performance, thanks to the blocking techniques used in BLAS routines (L’Excellent, 2012). The numerical settings are the same as in Table 1. We increase the total number of MPI processes and OpenMP threads from 96 to 256, and the number of OpenMP threads varies from one to eight to fit our hardware settings (two Intel E5-2670 processors per node, eight cores per processor). Figure 6 presents the corresponding factorization and total time of MUMPS with different numbers of OpenMP threads. The dashed lines indicate the ideal scalability, and the solid curves are real computing time. With a fixed total number of MPI processes and OpenMP threads, the more OpenMP threads we use, the better MUMPS scales. In Figure 7, the memory cost for each case is shown. As mentioned, using multithreaded BLAS yields a better usage of memory, which is also illustrated in Figure 7. Although the memory usage of eight threads is larger for smaller number of total MPI processes + OpenMP threads, the trend agrees well with the expectation when the number of total MPI processes + OpenMP threads increases.

Influence of complex topography, deformed mesh, and FSBC on performance of MUMPS

To investigate the influence of complex topography, deformed mesh, and FSBC on the performance of MUMPS, two sets of experiments are conducted: one with Cartesian nondeformed mesh and the other with vertically deformed mesh. The topography is designed as a sinusoidal function. The peak and valley of the topography are 0.3 and 240 m, respectively; thus, the distance is 239 m or approximately 2.4 wavelengths. We increase the model size from $10 \times 10 \times 10$ to $20 \times 20 \times 20$ elements. The parameter settings are the same as in Table 1 except for $N_e/\text{dim}$.

$N_e/\text{dim}$

<table>
<thead>
<tr>
<th>DOFs</th>
<th>$N_{\text{NNZ}}$</th>
<th>$N_{\text{MPI}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5,314,683</td>
<td>314,021,445</td>
</tr>
</tbody>
</table>

Notes: $N_e/\text{dim}$, number of elements in each dimension; DOFs, the number of degrees of freedom; $N_{\text{NNZ}}$, number of nonzeros in the matrix; $N_{\text{MPI}}$, number of MPI processes.

Figure 4. Profiles extracted from 3D numerical wavefields $u_x$ (a), $u_y$ (b), and $u_z$ (c) with PML, at $x = 664.26$ m (first column) and $z = 664.26$ m (second column) in the x-z plane sliced at $y = 1235.74$ m.

Table 3. MUMPS performance details.

<table>
<thead>
<tr>
<th>$N_{\text{MPI}}$</th>
<th>$M_A/M_M$ (MB)</th>
<th>$M_T$ (MB)</th>
<th>$T_A$ (s)</th>
<th>$T_D$ (s)</th>
<th>$T_F$ (s)</th>
<th>$T_S$ (s)</th>
<th>$T_T$ (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>35,304/45,183</td>
<td>564,864</td>
<td>450.17</td>
<td>27.80</td>
<td>11,521.57</td>
<td>6.65</td>
<td>12,070.32</td>
</tr>
<tr>
<td>32</td>
<td>21,354/25,241</td>
<td>683,328</td>
<td>415.70</td>
<td>30.40</td>
<td>5603.87</td>
<td>3.31</td>
<td>6083.99</td>
</tr>
<tr>
<td>64</td>
<td>10,608/13,420</td>
<td>678,912</td>
<td>414.60</td>
<td>38.59</td>
<td>3341.08</td>
<td>2.95</td>
<td>3827.83</td>
</tr>
<tr>
<td>128</td>
<td>5487/7125</td>
<td>702,336</td>
<td>428.69</td>
<td>57.82</td>
<td>1856.41</td>
<td>2.42</td>
<td>2377.16</td>
</tr>
<tr>
<td>256</td>
<td>3149/4384</td>
<td>806,144</td>
<td>453.68</td>
<td>104.39</td>
<td>1510.77</td>
<td>2.63</td>
<td>2107.37</td>
</tr>
<tr>
<td>512</td>
<td>2101/2873</td>
<td>1,075,712</td>
<td>452.65</td>
<td>227.24</td>
<td>1597.89</td>
<td>3.04</td>
<td>2342.31</td>
</tr>
</tbody>
</table>

Notes: $N_{\text{MPI}}$, number of MPI processes; $M_A/M_M$, average and maximum memory cost per working core; $M_T$, total memory cost; $T_A$, time for analysis; $T_D$, time for distribution of matrix; $T_F$, time for factorization; $T_S$, time for resolution for one RHS; and $T_T$, total elapsed time.
deformed mesh with nontrivial topography and to 30 for a Cartesian nondeformed mesh, to perform the complexity analysis of MUMPS memory cost and flops. This experiment is conducted on a second computing platform with 16 cores available per processor; therefore, we use 16 OpenMP threads in this case. Tables 4 and 5 summarize the outputs of MUMPS in each case, where \( N_{LU} \) and \( N_{ops} \) indicate the number of LU factors and operations during the factorization. Column DOFs of Table 5 is the same as in Table 4 and is thus omitted. The largest number of DOFs with nondeformed mesh is \( 1.41 \times 10^7 \) and \( 1.17 \times 10^7 \) with a vertically deformed mesh. Although the memory cost per working core varies in an irregular way, the total memory cost increases as expected. Note that the matrix is centralized on the host core and the matrix itself consumes a large amount of memory. Thus, MUMPS is set to forbid the host to participate in computations during the factorization and solve phases, as recommended by MUMPS team (2017). Therefore, the average and maximum memory cost is equivalent when \( N_{MPI} = 2 \). The time of matrix distribution increases as the matrix size becomes larger. The number of LU factors and operations during the factorization is listed in the last two columns and will be used to estimate the increasing trend of MUMPS cost.

Figure 9 shows the number of LU factors and floating-point operations during the factorization for different scale modeling and the corresponding increasing trend. The theoretical memory complexity of the LU factorization is \( O(N^4) \) if nested-dissection ordering is used and the number of operations is \( O(N^6) \) (Ashcraft and Liu, 1998; Operto et al., 2007). Through a curve fitting process on estimating the number of factors, we obtain a multiplying coefficient of 330 for the nondeformed mesh and of 347 for the vertically deformed mesh. Because the ordering method METIS is a combination of the hybrid multilevel nested-dissection and multiple minimum degree algorithm, there is some slight deviation between the output of MUMPS and the \( O(N^4) \) estimation. A curve based on \( O(N^{3.5}) \) fits better and is shown in the figure as well. Note that two complexity analysis of a 3D frequency-domain acoustic wave modeling (Operto et al., 2007) and a frequency-domain 3D elastic wave modeling (Gosselin-Cliche and Giroux, 2014), both using the finite-difference method, are included. The multiplying coefficients are 35 and 292, respectively. The number of factors, that is, the memory consumption, for the 3D elastic case is much larger than that for the acoustic case, but it is similar to the elastic case with FDM. A similar situation occurs for the estimation of the number of operations with \( O(N^6) \). A multiplying coefficient of 6903 is obtained for the nondeformed mesh and of 7385 for the vertically deformed mesh. Better curve fitting based on \( O(N^{5.5}) \) is achieved with the multiplying coefficients being 9875 and 9752 in each case.

We present in Figure 10 the factorization time (multiplied by the number of MPI processes) and effective total memory cost. As seen again, the factorization time for Cartesian nondeformed mesh and vertically deformed mesh is quite close to each other as expected. As seen in Figure 10b, the memory cost almost reaches the limit of the available amount of our computing environment.

Table 6 summarizes the largest scale modeling we have achieved with available resources, using FR MUMPS. The size of the linear
Table 4. MUMPS performance details with the Cartesian nondeformed mesh.

<table>
<thead>
<tr>
<th>$N_z$/dim</th>
<th>DOFs</th>
<th>$N_{NNZ}$</th>
<th>$N_{MPI}$</th>
<th>$M_A/M_M$ (GB)</th>
<th>$M_T$ (GB)</th>
<th>$T_A$ (s)</th>
<th>$T_D$ (s)</th>
<th>$T_F$ (s)</th>
<th>$T_S$ (s)</th>
<th>$N_{LU}$</th>
<th>$N_{ops}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>9.22e5</td>
<td>1.12e8</td>
<td>2</td>
<td>36.79/36.79</td>
<td>38.46</td>
<td>47.24</td>
<td>2.21</td>
<td>143.57</td>
<td>1.51</td>
<td>4.20e9</td>
<td>3.27e13</td>
</tr>
<tr>
<td>12</td>
<td>1.39e6</td>
<td>1.71e8</td>
<td>2</td>
<td>63.67/63.67</td>
<td>66.21</td>
<td>75.18</td>
<td>3.37</td>
<td>287.40</td>
<td>2.56</td>
<td>7.32e9</td>
<td>7.52e13</td>
</tr>
<tr>
<td>14</td>
<td>2.01e6</td>
<td>2.46e8</td>
<td>4</td>
<td>38.79/46.16</td>
<td>120.04</td>
<td>112.91</td>
<td>5.54</td>
<td>246.10</td>
<td>1.77</td>
<td>1.20e10</td>
<td>1.55e14</td>
</tr>
<tr>
<td>16</td>
<td>2.78e6</td>
<td>3.42e8</td>
<td>4</td>
<td>61.47/66.22</td>
<td>189.52</td>
<td>161.76</td>
<td>7.55</td>
<td>549.65</td>
<td>2.50</td>
<td>1.88e10</td>
<td>3.01e14</td>
</tr>
<tr>
<td>18</td>
<td>3.73e6</td>
<td>4.61e8</td>
<td>6</td>
<td>60.95/80.65</td>
<td>311.63</td>
<td>225.50</td>
<td>9.66</td>
<td>621.52</td>
<td>2.71</td>
<td>2.82e10</td>
<td>5.49e14</td>
</tr>
<tr>
<td>20</td>
<td>4.87e6</td>
<td>7.27e8</td>
<td>8</td>
<td>59.81/75.29</td>
<td>429.55</td>
<td>321.15</td>
<td>15.09</td>
<td>677.71</td>
<td>2.54</td>
<td>3.94e10</td>
<td>9.24e14</td>
</tr>
<tr>
<td>22</td>
<td>6.22e6</td>
<td>7.72e8</td>
<td>10</td>
<td>69.27/82.44</td>
<td>634.95</td>
<td>403.80</td>
<td>16.30</td>
<td>1054.21</td>
<td>2.97</td>
<td>5.57e10</td>
<td>1.53e15</td>
</tr>
<tr>
<td>24</td>
<td>7.81e6</td>
<td>9.71e8</td>
<td>12</td>
<td>76.76/92.97</td>
<td>858.87</td>
<td>528.14</td>
<td>20.38</td>
<td>1280.91</td>
<td>3.86</td>
<td>7.49e10</td>
<td>2.39e15</td>
</tr>
<tr>
<td>26</td>
<td>9.64e6</td>
<td>1.19e9</td>
<td>14</td>
<td>84.28/93.54</td>
<td>1113.61</td>
<td>666.33</td>
<td>24.86</td>
<td>1886.35</td>
<td>5.57</td>
<td>9.95e10</td>
<td>3.68e15</td>
</tr>
<tr>
<td>28</td>
<td>1.17e7</td>
<td>1.46e9</td>
<td>20</td>
<td>75.91/82.47</td>
<td>1464.21</td>
<td>840.48</td>
<td>31.22</td>
<td>1873.83</td>
<td>4.20</td>
<td>1.29e11</td>
<td>5.49e15</td>
</tr>
<tr>
<td>30</td>
<td>1.41e7</td>
<td>2.12e9</td>
<td>24</td>
<td>83.19/94.18</td>
<td>1945.22</td>
<td>1075.30</td>
<td>45.59</td>
<td>2085.97</td>
<td>5.17</td>
<td>1.65e11</td>
<td>7.93e15</td>
</tr>
</tbody>
</table>

Notes: $N_z$/dim, number of elements in each dimension; DOFs, the number of degrees of freedom; $N_{NNZ}$, number of nonzeros in the matrix; $N_{MPI}$, number of MPI processes; $M_A/M_M$, average and maximum memory cost per working core; $M_T$, total memory cost; $T_A$, time for analysis; $T_D$, time for distribution of matrix; $T_F$ time for factorization; $T_S$, time for resolution for one RHS; $N_{LU}$, number of LU factors; and $N_{ops}$, number of floating-point operations during the factorization.

Table 5. MUMPS performance details with the vertically deformed mesh.

<table>
<thead>
<tr>
<th>$N_z$/dim</th>
<th>$N_{NNZ}$</th>
<th>$N_{MPI}$</th>
<th>$M_A/M_M$ (GB)</th>
<th>$M_T$ (GB)</th>
<th>$T_A$ (s)</th>
<th>$T_D$ (s)</th>
<th>$T_F$ (s)</th>
<th>$T_S$ (s)</th>
<th>$N_{LU}$</th>
<th>$N_{ops}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>4.15e8</td>
<td>2</td>
<td>41.52/41.52</td>
<td>47.71</td>
<td>45.22</td>
<td>7.59</td>
<td>145.73</td>
<td>1.58</td>
<td>4.34e9</td>
<td>3.35e13</td>
</tr>
<tr>
<td>12</td>
<td>6.33e8</td>
<td>2</td>
<td>70.61/70.60</td>
<td>80.04</td>
<td>70.13</td>
<td>11.59</td>
<td>302.58</td>
<td>2.46</td>
<td>7.55e9</td>
<td>7.73e13</td>
</tr>
<tr>
<td>14</td>
<td>9.16e8</td>
<td>4</td>
<td>42.56/44.78</td>
<td>141.33</td>
<td>106.44</td>
<td>17.83</td>
<td>302.25</td>
<td>1.44</td>
<td>1.24e10</td>
<td>1.61e14</td>
</tr>
<tr>
<td>16</td>
<td>1.27e9</td>
<td>4</td>
<td>63.26/71.91</td>
<td>208.75</td>
<td>154.84</td>
<td>24.63</td>
<td>486.11</td>
<td>2.48</td>
<td>1.94e10</td>
<td>3.11e14</td>
</tr>
<tr>
<td>18</td>
<td>1.71e9</td>
<td>6</td>
<td>62.66/68.02</td>
<td>336.79</td>
<td>220.89</td>
<td>33.01</td>
<td>575.45</td>
<td>2.32</td>
<td>2.88e10</td>
<td>5.61e15</td>
</tr>
<tr>
<td>20</td>
<td>2.23e9</td>
<td>8</td>
<td>64.96/83.24</td>
<td>488.11</td>
<td>284.77</td>
<td>43.04</td>
<td>700.81</td>
<td>2.78</td>
<td>4.10e10</td>
<td>9.53e14</td>
</tr>
<tr>
<td>22</td>
<td>2.86e9</td>
<td>10</td>
<td>71.66/81.25</td>
<td>687.68</td>
<td>380.01</td>
<td>54.58</td>
<td>936.78</td>
<td>3.41</td>
<td>5.69e10</td>
<td>1.55e15</td>
</tr>
<tr>
<td>24</td>
<td>3.60e9</td>
<td>12</td>
<td>78.81/94.34</td>
<td>920.62</td>
<td>495.99</td>
<td>68.17</td>
<td>1245.28</td>
<td>4.53</td>
<td>7.71e10</td>
<td>2.46e15</td>
</tr>
<tr>
<td>26</td>
<td>4.45e9</td>
<td>14</td>
<td>86.44/94.98</td>
<td>1190.13</td>
<td>689.38</td>
<td>84.72</td>
<td>1739.36</td>
<td>4.78</td>
<td>1.01e11</td>
<td>3.72e15</td>
</tr>
<tr>
<td>28</td>
<td>5.42e9</td>
<td>20</td>
<td>76.68/86.53</td>
<td>1537.97</td>
<td>873.83</td>
<td>106.71</td>
<td>1828.17</td>
<td>4.59</td>
<td>1.32e11</td>
<td>5.54e15</td>
</tr>
</tbody>
</table>

Notes: $N_z$/dim, number of elements in each dimension; column DOFs is the same as in Table 4; $N_{NNZ}$, number of nonzeros in the matrix; $N_{MPI}$, number of MPI processes; $M_A/M_M$, average and maximum memory cost per working core; $M_T$, total memory cost; $T_A$, time for analysis; $T_D$, time for distribution of matrix; $T_F$ time for factorization; $T_S$, time for resolution for one RHS; $N_{LU}$, number of LU factors; and $N_{ops}$, number of floating-point operations during the factorization.
Computational cost estimation for a 3D moderate exploration-scale FWI model

A subset of the SEAM Phase II Foothills model (Regone et al., 2017; Trinh et al., 2019) is selected to estimate the computational cost for a 3D exploration-scale FWI model. The \( V_p \), \( V_S \), and density \( \rho \) models are given in Figure 11. This model, referred to as model I, is discretized into a mesh of \( 30 \times 72 \times 20 \) elements (the element size being 150 m), where the 30 elements in the \( z \)-axis are vertically deformed to acknowledge the topography (the maximal vertical variation being 1359 m). The \( z \)-, \( x \)-, and \( y \)-dimensions are 4.5, 10.8, and 3.0 km, respectively. The free-surface boundary condition is set on the topography, and PML is used on the other sides of the model. The frequency is 7.0 Hz, and the minimal \( V_S \approx 1300 \) m/s, which makes the minimal wavelength 180 m. Thus, we have 1.2 elements per wavelength to guarantee the accuracy of a fourth-order SEM implementation, and the number of propagated wavelengths in the longest dimension (\( x \)-axis) is approximately 60. An example wavefield of \( u_z \) is shown in Figure 12. The computational cost is summarized in Table 7. The results of modeling with a \( 28 \times 28 \times 28 \) vertically deformed mesh as in the previous section (referred to as model II) are listed as comparisons. Although the number of DOFs of model II is close to that of model I, the factorization time is much longer. The explanation is that the matrix bandwidth is \( 30N_2N_3 \), where \( N_2 \) and \( N_3 \) are the shorter dimensions. Thus, the bandwidth for model I is smaller, leading to less fill-in during the LU factorization and fewer flops. This indicates that our implementation may benefit from the model geometry in real applications, where the \( z \)-axis is always shorter. In addition, the modeling implementation is not restricted to homogeneous media. Theoretically, the heterogeneity of media will only affect the values of the matrix entries. The matrix structure stays the same as that of the homogeneous case. In addition, values of \( c_{ijkl} \) coefficients and density for subsurface structures will not lead to extreme matrix entry values and pivoting should normally be avoided. It is thus expected that accounting for heterogeneous media will not affect the performance of MUMPS compared with homogeneous media cases. The time of solving one source for model I is only approximately 1.97 s.

From this analysis, we can roughly estimate the computation time for one FWI gradient. Although the analysis step consumes a great part of the elapsed time, such analysis only needs to be performed once during the entire inversion process, even if the model changes significantly during the inversion because the matrix pattern in each inversion iteration stays the same. Therefore, we consider this as an offline computation in an FWI scenario and do not account for it in the gradient computation cost. Building one gradient of FWI requires the forward and adjoint wavefields; that is, one LU factorization and \( 2N_s \) solves for the incident and adjoint wavefields with \( N_s \) being the number of sources. The number of sources is chosen as 80. These operations require 0.34 h using 24 MPI processes and 16 OpenMP threads. This is equivalent to 130 scalar

![Figure 9. Number of factors (a) and floating-point operations (b) during the LU factorization with a nondeformed Cartesian mesh (Cart) and vertically deformed mesh (topo). The curves with stars in (a) are obtained from estimations by Operto et al. (2007) and Gosselin-Cliche and Giroux (2014) for acoustic and elastic wave modeling in the frequency domain.](image)

![Figure 10. Factorization time (a) and effective total memory cost (b) with a nondeformed Cartesian mesh (Cartesian) and vertically deformed mesh (with topo). The purple curve in (b) indicates the total memory available with the clusters nodes used.](image)

### Table 6. Current largest modeling with FR MUMPS (#OMP = 16).

<table>
<thead>
<tr>
<th>( N_s/N )</th>
<th>DOFs</th>
<th>( N_{NZ} )</th>
<th>( N_{MPS}/N_{core} )</th>
<th>( M_T ) (GB)</th>
<th>( T_F ) (s)</th>
<th>( T_S ) (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cartesian</td>
<td>30/171</td>
<td>1.41e7</td>
<td>2.12e9</td>
<td>24/384</td>
<td>1945.22</td>
<td>2085.97</td>
</tr>
<tr>
<td>Deformed</td>
<td>28/151</td>
<td>1.17e7</td>
<td>5.42e9</td>
<td>20/320</td>
<td>1537.97</td>
<td>1828.17</td>
</tr>
</tbody>
</table>

*Notes: \( N_s/N \), number of wavelength and grid-points in each dimension; DOFs, the number of degrees of freedom; \( N_{NZ} \), number of nonzeros in the matrix; \( N_{MPS}/N_{core} \), number of MPI processes and total cores used; \( M_T \), total memory cost during the factorization; \( T_F \), elapsed time for factorization; and \( T_S \), time for resolution for one RHS.*
hours. It is only considered as a reference because the actual entire computational overhead may be larger, taking into account other necessary calculations. As a comparison, we use a time-domain SEM code to estimate the computational cost of computing one inversion gradient as in Trinh et al. (2019). The same platform is used to conduct the modeling. The source is a Ricker wavelet centered at 2.8 Hz (maximum frequency 7.0 Hz). The time step is $7.5 \times 10^{-4}$ s to satisfy the CFL condition with respect to the maximum velocity and minimum spatial step. The number of time steps is $10^4$ (source duration 7.5 s) to account for the traveltimes of the longest offset. Constant attenuation ($Q_\text{P} = 100$, $Q_\text{S} = 100$) is approximated by three SLS mechanisms. We use one computing node for each source and 32 cores per node to perform domain decomposition. The elapsed time is 714 s if the intermediate wavefields are stored in memory. The total scalar hour for 80 sources is 507 h. Thus, the time required by the frequency-domain modeling is relatively smaller. As expected, the total memory cost of the time-domain modeling is approximately 83 Gb, much less than the 1.8 Tb required for the frequency-domain case. From this rough analysis, we see that for a small number of sources (a few tens) similar scalar hours are required when choosing frequency- or time-domain approaches. However, the time-domain computational time increases linearly with the number of sources, based on the total time of single source modeling (714 s). The frequency-domain time increases linearly as well, but only based on the solve time (1.97 s). Thus, the total computational time in scalar hours for the time-domain approach increases much faster using the time-domain approach than using the frequency-domain approach. Figure 13 shows the increasing trend of scalar hours for building one FWI gradient with respect to the number of sources. The source number axis is in log scale. For 800 sources, the total scalar hour is 431 h for the frequency-domain implementation, which is one magnitude order less than the 5077 h required using a time-domain implementation. This is an expected advantage of using a direct solver to solve the frequency-domain modeling. However, the main bottleneck for the frequency-domain approach is the significant increase of memory demand for larger scale targets, associated with a loss of scalability, which makes the frequency-domain approach still limited to small to moderate-scale problems if FWI is intended.

**DISCUSSION**

**PML stability**

PML is efficient in preventing the artificial reflections from boundaries. However, it can lead to instabilities for time-domain simulations when anisotropy is considered (Becache et al., 2003; Métivier et al., 2014). For frequency-domain anisotropic wave modeling, the instability of PML for specific anisotropic parameters is reported in Zhou et al. (2019). We have conducted several experiments using the anisotropic parameters in Zhou et al. (2019), and similar instabilities are observed. As an alternative to PML, the sponge layers (Cerjan et al., 1985; Shin, 1995) can be adapted to the anisotropic cases because it is theoretically stable but less efficient. The generalized stiffness reduction method (Zhou et al., 2019) could be investigated for future research as well.

**Comparison on computational cost between SEM and FDM**

As mentioned in the matrix-building section, the SEM matrix is denser and has a larger bandwidth than the FDM matrix. However, one should keep in mind that the higher order accuracy of SEM will decrease the number of grid points per wavelength for model sampling. Moreover, implementing FSBC would also require an FDM
grid fine enough to simulate the surface waves, whereas for SEM no extra treatment needs to be taken because FSBC is already taken into account by the weak form of the wave equation. Both of these factors largely reduce the size of the SEM-based matrix. Thus, the fact that the SEM matrix is denser is counteracted, and the solution of the linear system benefits from its smaller size. In practice, once a target model is chosen, a grid should be designed corresponding to the model sampling. To guarantee the accuracy for modeling the surface wave, at least 20 points per wavelength should be set for FDM with a complex topography and FSBC (Bohlen and Saenger, 2006). However, because of the low dispersion of SEM, a sampling with only four to five points per wavelength can be used with a negligible loss of accuracy for SEM of order four or greater (De Basabe and Sen, 2007). We denote \( N_\text{SEM} \) and \( N_\text{FDM} \) the number of grid points in each dimension for model sampling with FDM and SEM, respectively. For a fifth-order SEM and a fourth-order staggered-grid FDM, we have

\[
\frac{N_\text{FDM}}{N_\text{SEM}} = \frac{20}{5} = 4.
\]

Thus, the ratio between the size of corresponding linear systems is

\[
\frac{S_\text{FDM}}{S_\text{SEM}} = \frac{3N_\text{FDM}^3}{3N_\text{SEM}^3} = 64.
\]

The ratio between the matrix bandwidth is

\[
\frac{18N_\text{FDM}^2}{30N_\text{SEM}^2} = 9.6.
\]

According to the theoretical estimation of the memory and computational cost, we have the following comparison:

\[
\frac{\text{Memory}_{\text{FDM}}}{\text{Memory}_{\text{SEM}}} = O(\frac{N_{\text{FDM}}^4}{N_{\text{SEM}}^4}) = 256,
\]

\[
\frac{\text{Flops}_{\text{FDM}}}{\text{Flops}_{\text{SEM}}} = O(\frac{N_{\text{FDM}}^6}{N_{\text{SEM}}^6}) = 4096.
\]

Such a comparison shows that SEM is less demanding in computational resources when complex topography and FSBC have to be considered. Of course, it is just an estimation because detailed implementation of FDM and SEM will affect the coefficient before the \( O(\cdot) \) terms. Nevertheless, SEM seems to be less computationally demanding and more suitable for onshore applications where these physical factors are necessary.

**Limit in MUMPS scalability to tackle larger scale problems**

We discuss the increasing trend of MUMPS on the memory cost and flops. Extending the curves in Figure 9, we have the increasing trend of number of factors and floating-point operations during the LU factorization for larger-scale

### Table 7. Computational cost for the frequency-domain modeling with: I, the SEAM model, II, a model with a vertically deformed \( 28 \times 28 \times 28 \) mesh as in Table 5.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mesh</th>
<th>DOFs</th>
<th>( N_{\text{NNZ}} )</th>
<th>( N_{\text{MPI}}/N_{\text{core}} )</th>
<th>( M_T ) (TB)</th>
<th>( T_A ) (s)</th>
<th>( T_F ) (s)</th>
<th>( T_S ) (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>30 × 72 × 20</td>
<td>1.14e7</td>
<td>4.00e9</td>
<td>24/384</td>
<td>1.80</td>
<td>1037.83</td>
<td>895.28</td>
<td>1.97</td>
</tr>
<tr>
<td>II</td>
<td>28 × 28 × 28</td>
<td>1.17e7</td>
<td>5.42e9</td>
<td>20/320</td>
<td>1.53</td>
<td>873.83</td>
<td>1828.17</td>
<td>4.59</td>
</tr>
</tbody>
</table>

Notes: DOFs, the number of degrees of freedom; \( N_{\text{NNZ}} \), number of nonzeros in the matrix; \( N_{\text{MPI}}/N_{\text{core}} \), number of MPI processes and total cores used; \( M_T \), total memory cost during the factorization; \( T_A \), time for analysis; \( T_F \), elapsed time for factorization; and \( T_S \), time for resolution for one RHS.

---

Figure 13. Increasing trend of the scalar hour for building one FWI gradient with time- and frequency-domain wave modeling, with respect to the number of sources.

Figure 14. Increasing trend of number of factors (a) and floating-point operations (b) during the LU factorization with a nondeformed Cartesian mesh (Cart) and vertically deformed mesh (topo). The curves with stars in (a) are obtained from estimations by Operto et al. (2007) and Gosselin-Cliche and Giroux (2014) for acoustic and elastic wave modeling in the frequency domain. The shaded rectangles correspond to the data between \( \lambda = 10 \) and 30 as given in Figure 9.
modelings (up to a $50 \times 50 \times 50$ elements model) in Figure 14. The number of factors surpasses $10^{12}$, and the number of operations reaches $10^{17}$, which is a great increment compared with the data between $\lambda = 10$ and 30 shaded by the gray rectangles. In Figure 15, we perform curve fitting based on Figure 10. The dashed curves are from the estimation, and the solid ones are the outputs of MUMPS. Because there will always be a gap between the total memory requested and the total effective memory cost of the factorization, we also estimate the increasing trend of the total memory requested based on the memory limit of our computing environment. The number of flops in such a case is large, which means that we have to resort to larger and more efficient platforms. The memory cost is still demanding, for example, 20 Tb for a modeling with $\lambda = 50$, which, however, is not unacceptable with modern clusters. Take the current platform (192 Gb per node) as an example, approximately 130 computing nodes and 2500 s would be required with $\#\text{MPI} = 520$ and $\#\text{OMP} = 8$ if MUMPS could scale up to such a number of MPI processes. However, the bottleneck of MUMPS scalability will limit us from reaching such an elapsed time.

Figure 16 shows the memory cost for matrix building and storage. Due to the high order of SEM, the matrix is relatively dense, which leads to a larger memory cost for building and storing the matrix. In addition, dealing with nontrivial topography leads to at least a vertically deformed mesh, which will consequently increase the number of nonzeros in each matrix row. This could explain why the matrix storage with topography is larger than that with Cartesian nondeformed mesh. However, the building process and matrix storage could be totally parallelized by domain decomposition, which circumvents the large memory cost.

**Impact of matrix reordering**

Matrix reordering has a big impact on the memory cost and flops of the LU factorization (Guermouche et al., 2003). We use the sequential METIS (Karypis, 2013) that is quite efficient for the subsequent factorization in terms of time and memory. Figure 17 shows the time for matrix reordering using METIS (sequential). However, the current bottleneck is that the memory cost of METIS has reached the limit of one cluster node as the model size increases to approximately $30 \times 30 \times 30$ elements. The time of ordering is also quite long as shown in Figure 17. The parallel version of METIS (ParMETIS) and other ordering algorithms such as PT-SCOTCH (Pellegrini, 2018) has been tested as well. But the flops and memory cost of factorization with these ordering algorithms are not acceptable for our current computing environment. Analysis of a modeling with $10 \times 10 \times 10$ elements indicates that the total memory cost during the factorization using METIS (two MPI processes, $\#\text{OMP} = 16$), ParMETIS (four MPI processes, $\#\text{OMP} = 16$), and PT-SCOTCH (four MPI processes, $\#\text{OMP} = 16$) are 37,722 MB, 52,516 MB, and 69,752 MB, respectively. The memory cost is almost doubled when using ParMETIS or PT-SCOTCH. The factorization time is 142.2, 85.7, and 136.9 s, respectively. This estimation gives an insight on the bottleneck of the memory requirement for the $30 \times 30 \times 30$ elements case, where the memory cost might be more than doubled. Using the parallel reordering methods would avoid the bottleneck, but requires
Figure 18. The difference between the FR $\kappa_x$ wavefield and BLR results with (a) $\varepsilon = 10^{-5}$, (b) $10^{-4}$, and (c) $10^{-3}$, with the same color scale as in Figure 12.

CONCLUSION

We have investigated 3D frequency-domain anisotropic elastic wave modeling using the SEM and used a massively parallel direct solver MUMPS (FR version) to solve the corresponding linear system. Complex topography is well-handled by using a vertically deformed mesh. FSBC is naturally realized through the weak form of the wave equation. PML is incorporated to absorb the outgoing waves. The structure of the matrix is analyzed by expanding the discretization of the weak form of the wave equation. The matrix from frequency-domain SEM is denser and has a larger bandwidth compared to the one from FDM with the same discretization grid. However, frequency-domain SEM has higher order accuracy and can deal with complex topography and FSBC naturally. Therefore, frequency-domain SEM can be more economical in terms of computational and memory cost. The accuracy of the numerical results is validated by comparing with the analytical solutions. The wavefields with PML fit well with the analytical solutions.

The block low-rank (BLR) approximation of the frontal matrices could reduce the flops and memory imprint of the LU factorization (Amestoy et al., 2015). We adopt the BLR feature of MUMPS and use three low-rank thresholds $\varepsilon = 10^{-5}$, $10^{-4}$, and $10^{-3}$. The subset of the SEAM Phase II Foothills model and the same modeling settings are used. The difference between the FR wavefield and BLR wavefields with $\varepsilon = 10^{-5}$, $10^{-4}$, and $10^{-3}$ is shown in Figure 18 with the same color scale as in Figure 12. The BLR wavefield with $\varepsilon = 10^{-3}$ is almost the same as the FR one. For $\varepsilon = 10^{-4}$, the wavefield is close enough to the FR results. The difference for $\varepsilon = 10^{-5}$ is large. The scaled residuals, defined as $\|Au - d\|_\infty / \|A\mu\|_\infty \|d\|_\infty$ with $\mu$ being the numerical solution are $1.33 \times 10^{-6}$, $1.25 \times 10^{-6}$, $3.37 \times 10^{-6}$, and $3.57 \times 10^{-5}$, respectively. The same conclusion as shown in Figure 18 applies, and the residual for $\varepsilon = 10^{-5}$ is even smaller than that of the FR result. The reduction in factorization time, flops, and memory cost from the BLR approximation is summarized in Table 8. The theoretical operation counts during LU factorization are reduced to approximately 10% with the BLR approximation. For the real computational cost, nearly half of the factorization time and memory cost could be saved. Therefore, with the BLR approximation, a larger scale of modeling can be performed. However, the BLR feature of MUMPS cannot avoid the limit of its scalability and the huge memory imprint of the LU factorization as mentioned in the previous section.

We have investigated 3D frequency-domain anisotropic elastic wave modeling using the SEM and used a massively parallel direct solver MUMPS (FR version) to solve the corresponding linear system. Complex topography is well-handled by using a vertically deformed mesh. FSBC is naturally realized through the weak form of the wave equation. PML is incorporated to absorb the outgoing waves. The structure of the matrix is analyzed by expanding the discretization of the weak form of the wave equation. The matrix from frequency-domain SEM is denser and has a larger bandwidth compared to the one from FDM with the same discretization grid. However, frequency-domain SEM has higher order accuracy and can deal with complex topography and FSBC naturally. Therefore, frequency-domain SEM can be more economical in terms of computational and memory cost. The accuracy of the numerical results is validated by comparing with the analytical solutions. The wavefields with PML fit well with the analytical solutions.

For the performance of pure MPI implementation of MUMPS, the scalability deteriorates when the number of MPI processes surpasses approximately 100. MUMPS with a hybrid implementation of MPI and OpenMP presents better performance in terms of scalability, flops, and memory cost. The analysis of modeling with different combinations of OpenMP and MPI confirms this claim. In addition, using a vertically deformed mesh only introduces slight extra flops and memory cost compared with using a Cartesian non-deformed mesh. It is thus promising for onshore applications where complex topography has to be considered.

Modeling with a subset of the SEAM Phase II Foothills model shows its capability in handling realistic applications. An estimation of the computational cost for building one FWI gradient using the frequency-domain approach is shown to require similar scalar hours for few sources (a few tens) than what is required using the

### Table 8. Statistics of the FR and BLR results.

<table>
<thead>
<tr>
<th>$\varepsilon$</th>
<th>Factorization time (s)</th>
<th>Flop count LU</th>
<th>Mem LU (FR, TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^{-5}$</td>
<td>895.28</td>
<td>3.87e15</td>
<td>1.80</td>
</tr>
<tr>
<td>$10^{-4}$</td>
<td>477.76 (53%)</td>
<td>5.82e14 (15%)</td>
<td>1.06 (58%)</td>
</tr>
<tr>
<td>$10^{-3}$</td>
<td>428.25 (48%)</td>
<td>3.78e14 (9.8%)</td>
<td>1.01 (56%)</td>
</tr>
<tr>
<td>$10^{-2}$</td>
<td>404.26 (45%)</td>
<td>2.24e14 (5.8%)</td>
<td>0.92 (51%)</td>
</tr>
</tbody>
</table>

Notes: Flop count LU, number of theoretical operations during the LU factorization; Mem LU, memory cost for LU factors in TB. The metrics of BLR for different values of threshold $\varepsilon$ are given. The percentages with respect to the FR data are given in brackets.

The block low-rank approximation of the frontal matrices could reduce the flops and memory imprint of the LU factorization (Amestoy et al., 2015). We adopt the BLR feature of MUMPS and use three low-rank thresholds $\varepsilon = 10^{-5}$, $10^{-4}$, and $10^{-3}$. The subset of the SEAM Phase II Foothills model and the same modeling settings are used. The difference between the FR wavefield and BLR wavefields with $\varepsilon = 10^{-5}$, $10^{-4}$, and $10^{-3}$ is shown in Figure 18 with the same color scale as in Figure 12. The BLR wavefield with $\varepsilon = 10^{-3}$ is almost the same as the FR one. For $\varepsilon = 10^{-4}$, the wavefield is close enough to the FR results. The difference for $\varepsilon = 10^{-5}$ is large. The scaled residuals, defined as $\|Au - d\|_\infty / \|A\mu\|_\infty \|d\|_\infty$ with $\mu$ being the numerical solution are $1.33 \times 10^{-6}$, $1.25 \times 10^{-6}$, $3.37 \times 10^{-6}$, and $3.57 \times 10^{-5}$, respectively. The same conclusion as shown in Figure 18 applies, and the residual for $\varepsilon = 10^{-5}$ is even smaller than that of the FR result. The

The block low-rank (BLR) approximation of the frontal matrices could reduce the flops and memory imprint of the LU factorization (Amestoy et al., 2015). We adopt the BLR feature of MUMPS and use three low-rank thresholds $\varepsilon = 10^{-5}$, $10^{-4}$, and $10^{-3}$. The subset of the SEAM Phase II Foothills model and the same modeling settings are used. The difference between the FR wavefield and BLR wavefields with $\varepsilon = 10^{-5}$, $10^{-4}$, and $10^{-3}$ is shown in Figure 18 with the same color scale as in Figure 12. The BLR wavefield with $\varepsilon = 10^{-3}$ is almost the same as the FR one. For $\varepsilon = 10^{-4}$, the wavefield is close enough to the FR results. The difference for $\varepsilon = 10^{-5}$ is large. The scaled residuals, defined as $\|Au - d\|_\infty / \|A\mu\|_\infty \|d\|_\infty$ with $\mu$ being the numerical solution are $1.33 \times 10^{-6}$, $1.25 \times 10^{-6}$, $3.37 \times 10^{-6}$, and $3.57 \times 10^{-5}$, respectively. The same conclusion as shown in Figure 18 applies, and the residual for $\varepsilon = 10^{-5}$ is even smaller than that of the FR result. The reduction in factorization time, flops, and memory cost from the BLR approximation is summarized in Table 8. The theoretical operation counts during LU factorization are reduced to approximately 10% with the BLR approximation. For the real computational cost, nearly half of the factorization time and memory cost could be saved. Therefore, with the BLR approximation, a larger scale of modeling can be performed. However, the BLR feature of MUMPS cannot avoid the limit of its scalability and the huge memory imprint of the LU factorization as mentioned in the previous section.
time-domain implementation, with a favorable trend as the number of sources increases. The difference in computational cost between the frequency- and time-domain approaches reaches one order of magnitude for a few hundreds of sources. Moderate-scale FWI applications using frequency-domain modeling thus become feasible using this SEM/MUMPS approach. However, the huge memory imprint and the limit of scalability hinders the applications to larger scale problems. More scalable solvers would be desirable.

Other direct solvers such as the Watson Sparse Matrix Package (WSMP) and STRUCtured Matrix Package (STRUMPACK) or other low-rank techniques will be investigated in the future. Larger scale 3D frequency-domain elastic modeling may become feasible. Other parallel ordering methods should also be tested to avoid the current bottleneck on memory cost from sequential METIS. A new technique based on recasting the search for a harmonic solution of the elastodynamic equation as a time-domain controllability problem could also be investigated in the future. Such a technique has been generalized to settings adapted to the modeling of seismic waves and could be considered in the future.

**ACKNOWLEDGMENTS**

This study was partially funded by the SEISCOPE consortium (http://seiscope2.osug.fr), sponsored by AKERBP, CGG, CHEVRON, EQUINOR, EXXON-MOBIL, JGI, PETROBRAS, SCHLUMBERGER, SHELL, SINOPEC, and TOTAL. This study was also supported by the Rhône-Alpes region (GRANT CPERO7_13 CIRA), the OSUG@2020 labex (reference ANR10 LABX56), and the Equip@Meso project (reference ANR-10-EQPX-29-01) of the program Investissements d’Avenir supervised by the Agence Nationale pour la Recherche and the HPC resources of CINES/IDRIS under the allocation 046091 made by GENCI. The authors thank the MUMPS team for providing the MUMPS package (available at http://graal.ens-lyon.fr/MUMPS/index.html and http://mumps.enseeiht.fr) and appreciate the discussions with T. Mary, A. Buttari, J.-Y. L’Excellent, and P. Amestoy, which improve greatly our use of MUMPS. They also thank T.M. Irnaka for conducting the search for a harmonic solution of the elastodynamic equation as a time-domain controllability problem could also be investigated in the future. Such a technique has been generalized to settings adapted to the modeling of seismic waves and could be considered in the future.

**APPENDIX A**

**ELABORATION ON MATRIX BUILDING**

For finite-element-type methods, assembling the stiffness matrix is an essential component for discretization. We present the details of the three ways for building the stiffness matrix in this appendix. Similar to equation 9, discretization of all the terms in equation 7 is lengthy. Collection of the coefficients of $u_1$, $u_2$, $u_3$ separately requires lots of rearrangement of terms, even with symbolic computation. The first method for building the matrix is trying to collect the coefficients in a systematic way. We reformulate equation 9 as follows:

$$
\int_{\Omega} c \partial_i u \partial_i \phi d\Omega \approx \sum_{i,j,k} \int_{ij,k} \left( w_{ij,k} \phi_{ij} \right) \partial_i \left( J_{ij} \partial_i \phi_{ij} \right) + \sum_{i,j,k} \int_{ij,k} \left( w_{ij,k} \phi_{ij} \right) \partial_i \left( J_{ij} \partial_i \phi_{ij} \right) + \sum_{i,j,k} \int_{ij,k} \left( w_{ij,k} \phi_{ij} \right) \partial_i \left( J_{ij} \partial_i \phi_{ij} \right)
$$

Thus, the coefficient of each $u_{ijk}$ ($u_{1,ijk}$, $u_{2,ijk}$, or $u_{3,ijk}$) has been collected from the discretization of one term in equation 7. For the other terms, we only need to change the partial derivative $\partial_i$ and the elastic parameter $c$ to the corresponding ones. Note that although the coefficients obtained are for all of the $u_{ijk}$, $i$, $j$, $k = 1, \ldots, n$, only the coefficients of points connected to $u_{ijk}$ need to be calculated. Such a process is still an elementary way of calculating the coefficients. However, it is shown to be complicated and computationally demanding in our numerical tests.

The other two ways arise from the displacement-stress formulation of the wave equation (Schubert, 2003)

$$
\rho a^2 \mathbf{u}(\mathbf{x}) + \nabla \cdot \sigma(\mathbf{x}) + \mathbf{s}(\omega, \mathbf{r}_x) = 0,
$$

$$
\sigma(\mathbf{x}) = \mathbf{c}(\mathbf{x}) : \nabla \mathbf{u}(\mathbf{x}),
$$

where $\sigma$ is the stress tensor and $\mathbf{c}(\mathbf{x})$ is the elastic tensor. The corresponding weak form is

$$
\omega^2 \int_{\Omega} \rho \mathbf{u} \cdot \mathbf{d} \Omega + \int_{\Omega} \nabla \phi : \sigma d\Omega + \int_{\Omega} \phi \cdot \mathbf{s}(\omega, \mathbf{r}_x) d\Omega = 0.
$$

Here, $\phi$ is a 3C vector. The integrand of the second term could be expanded as follows:

$$
\nabla \phi : \sigma = \sum_{i,j=1}^{3} \sigma_{ij} \phi_i \partial_j \approx \sum_{i,k=1}^{3} \sum_{j=1}^{3} \sigma_{ij} \phi_i \partial_j \frac{\partial \phi_j}{\partial \xi_k}
$$

with

$$
= \sum_{i,k=1}^{3} B_{ik} \frac{\partial \phi_i}{\partial \xi_k},
$$

where $B_{ik}$ is the stiffness matrix.
\begin{equation}
B_{ik} = \sum_{j=1}^{3} \sigma_{ij} \partial_i \varepsilon_{jk},
\end{equation}
\begin{equation}
\sigma_{ij} = \sum_{k,l=1}^{3} c_{ijkl} \frac{\partial u_k}{\partial x_l}.
\end{equation}

Using Lagrange interpolation, we have
\begin{equation}
\frac{\partial u_k}{\partial x_l}_{\text{afy}} = \left( \frac{\partial u_k}{\partial x_l} \frac{\partial \varepsilon}{\partial x_l} + \frac{\partial u_k}{\partial x_l} \frac{\partial \eta}{\partial x_l} + \frac{\partial u_k}{\partial x_l} \frac{\partial \zeta}{\partial x_l} \right)_{\text{afy}}
\end{equation}
\begin{equation}
\approx \sum_{r=1}^{n_{u1}} u_{k,ryf} \epsilon'_r(\zeta_{n}) \frac{\partial \varepsilon}{\partial x_l}_{\text{afy}} + \sum_{s=1}^{n_{u2}} u_{k,asyf} \epsilon'_s(\eta_{n}) \frac{\partial \eta}{\partial x_l}_{\text{afy}} + \sum_{t=1}^{n_{u3}} u_{k,aftf} \epsilon'_t(\zeta_{n}) \frac{\partial \zeta}{\partial x_l}_{\text{afy}}.
\end{equation}

Then, the second term of equation A-4 could be evaluated as
\begin{align}
\int_{\Omega} \nabla \phi : \sigma d\Omega & \approx \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} \sum_{k=1}^{3} B_{ik} \frac{\partial \phi}{\partial x_l} J(\xi) d\Lambda \\
& \approx \sum_{l=0}^{n_{u1}} w_{l} J_{l yf} B_{11, l yf} \epsilon'_1(\xi_{l}) + \sum_{m=0}^{n_{u2}} w_{m} J_{m yf} B_{12, m yf} \epsilon'_2(\eta_{m}) + \sum_{n=0}^{n_{u3}} w_{n} J_{n yf} B_{13, n yf} \epsilon'_3(\zeta_{n}),
\end{align}
where \( \phi = (\phi_1, \phi_2, \phi_3), \phi_1 = \epsilon'(x) \epsilon'(y) \epsilon'(z), \phi_2 = 0, \phi_3 = 0. \)

Other selection of \( \phi \) could be computed similarly. Thus, building the stiffness matrix consists of six processes:
\begin{align}
\mathbf{u} & \rightarrow \mathbf{u} \partial \xi^i_{M_1} \mathbf{u} \partial \zeta^i_{M_2} \mathbf{c}_{ijkl} \mathbf{u} \partial \xi^j_{M_3} \mathbf{u} \partial \zeta^j_{M_4} B_{ik} \partial \mathbf{e}'(\xi) \partial \mathbf{e}'(\zeta) \partial \mathbf{e}'(\eta) \partial \mathbf{w}_{a, \text{afy}} \partial \mathbf{w}_{b, \text{afy}} \mathbf{m}_{\text{tmp}} \mathbf{m}_{\text{geo}} \int_{\Omega} \nabla \phi : \sigma d\Omega,
\end{align}
where \( \text{tmp} \) is a temporary variable and \( M_1, \ldots, M_6 \) are intermediate matrices whose entries are related to the terms over the arrows, respectively (one should adapt these terms with respect to the equations above). We have the stiffness matrix as follows:
\begin{equation}
\mathbf{K} = \mathbf{M}_6 \mathbf{M}_5 \mathbf{M}_4 \mathbf{M}_3 \mathbf{M}_2 \mathbf{M}_1.
\end{equation}

Note that all of the six matrices are sparse and the sparsity pattern is known in advance. Then, building \( \mathbf{K} \) should not be complex at first sight. However, the sparsity pattern of the matrices from the matrix-matrix multiplication, for example, \( \mathbf{M}_2 \mathbf{M}_1 \) or \( \mathbf{M}_1 \mathbf{M}_2 \mathbf{M}_1 \), is unknown, which necessitates the usage of dense matrices. It will surely reduce the efficiency of building \( \mathbf{K} \). This way of matrix building resembles the one mentioned by Zhou and Greenhalgh (2011) to some extent.

The third way of matrix building, that is, the most efficient one, still benefits from equations A-5 to A-11. In fact, it is the same as calculating \( \mathbf{Ku} \) in the time-domain modeling, apart from altering \( \mathbf{u} \) into a canonical basis vector \( \mathbf{e}^0 = (0, \ldots, 0, 1, 0, \ldots, 0) \) for the \( r \)th row. Hence, the matrix is built row by row. Such a process is performed separately for each element, which is efficient for numerical implementation as proven by experiments. Such a process resembles the highly efficient algorithm proposed by Deville et al. (2002), which recasts the matrix as the Kronecker products of smaller matrices. We have also adopted their techniques to benefit from the tensorial properties of hexahedral elements, the optimization of cache usage, and the combination of efficient loop vectorization and manual unrolling. As given in Deville et al. (2002), recasting the matrix as the Kronecker products of smaller matrices could reduce the computational cost from \( O(N^6) \) to \( O(N^3) \), where \( N \) (\( N_1, N_2, \) or \( N_3 \)) is the number of DOFs in one dimension. Numerical experiments indicate that performing the tensor-product evaluation is at least 10 times faster than the direct matrix-vector product. We have also conducted several experiments to compare the efficiency of these three ways of matrix building. The third one outperforms the others as expected. Building the matrix from a model of \( 30 \times 30 \times 30 \) elements, with the matrix size being \( 1.4e7 \), requires approximately 500 s for the third building way, but thousands of seconds for the other ways. The current building process is sequential. Parallelization through element-based domain decomposition will drastically reduce the elapsed time for matrix building. Although the element-based, row-by-row process is not optimal (computation with zeros in \( \mathbf{e}^0 \)), it is already efficient enough for stiffness matrix building. Compared with the stiffness matrix, mass matrix building is trivial because it is diagonal.

REFERENCES


