Computational Geosciences (2023) 27:793–804
https://doi.org/10.1007/s10596-023-10222-5

ORIGINAL PAPER

Modeling 3-D anisotropic elastodynamics using mimetic finite differences and fully staggered grids

Harpreet Sethi[1] · **Fatmir Hoxha**[2] · **Jeffrey Shragge**[1] · **Ilya Tsvankin**[1]

Received: 28 October 2022 / Accepted: 15 May 2023 / Published online: 26 July 2023

Abstract
Accurate modeling of elastic wavefields in 3-D anisotropic media is important for many seismic processing and inversion applications. However, efficient wavefield simulation for tilted transversely isotropic (TTI) media and, especially, for orthorhombic and lower symmetries remains challenging. Finite-difference (FD) implementations using centered Taylor-series coefficients on singly staggered grids suffer from reduced numerical accuracy due to problems in computing the partial wavefield derivatives in TTI or tilted orthorhombic (TOR) media, as well as in enforcing the free-surface (zero-traction) boundary conditions. To address these issues, we develop a 3-D mimetic FD (MFD) algorithm for arbitrarily anisotropic media that uses a fully-staggered-grid strategy. This CUDA-based algorithm is implemented on graphics processing units (GPUs) to leverage the massive parallelism of this computer architecture. For multi-GPU parallelization, we employ the CUDA-aware message passing interface (MPI) library to exploit the remote direct memory access (RDMA) feature for buffer transfers. Weak- and strong-scaling tests on up to eight DGX NVIDIA A100 nodes (64 GPUs in total) demonstrate that the proposed multi-GPU implementation achieves a quasi-linear computational speedup with over 98% efficiency for large industrial-scale models of size in excess of 1.7×10^{10} grid points.

Keywords Elastic media · Anisotropy · Orthorhombic symmetry · Wave propagation · Finite-difference modeling · GPU · Boundary conditions

Mathematics Subject Classification (2010) 86-08 · 86-10

1 Introduction

Generating accurate wavefield solutions for large-scale 3-D anisotropic elastic models is a computationally challenging problem. There is a need for efficient modeling engines for arbitrarily anisotropic media that maintain high accuracy of the overall numerical scheme. An additional complexity is in implementing the correct boundary conditions at interfaces (e.g., at a free surface, fluid/solid interface) with pronounced property contrasts. Inaccurate handling of the boundary conditions with lower-order numerical approximations can produce significant wavefield distortions, especially for surface waves [1]. Here, we aim to improve the performance of 3-D elastic anisotropic wavefield solvers by developing a computationally efficient propagator with accurate high-order implementation of the free-surface boundary conditions (FSBC) using multiple graphic processing unit (GPU) cards [2].

The advent of GPUs as accelerators in elastic wavefield modeling, imaging, and inversion is due to their single-instruction multiple-data (SIMD) nature. Numerous methodologies for modeling wave propagation have been developed and ported to GPUs. A GPU-based spectral-element method for simulating elastic wave propagation at the continental scale is introduced in [3]. That work employs nonblocking

✉ Harpreet Sethi
hsethi@mines.edu

Fatmir Hoxha
fhoxha@nvidia.com

Jeffrey Shragge
jshragge@mines.edu

Ilya Tsvankin
itsvanki@mines.edu

[1] Department of Geophysics, Colorado School of Mines, 1500 Illinois Street, Golden 80401, CO, US

[2] NVIDIA Corporation, 2788 San Tomas Expressway, Santa Clara 95050, CA, US

communication based on a message passing interface (MPI) framework to overlap the data transfer with computations on GPUs across the network via the peripheral component interconnect express (PCIe) bus. The portability of a higher-order discontinuous Galerkin method for modeling of elastic wave propagation on a single GPU using unstructured tetrahedral meshes is studied in [4]. Articles [5] and [6] discuss the GPU implementation of finite-difference (FD) algorithms designed to solve the acoustic wave equation. The solutions of the 3-D isotropic elastic wave equation on multiple GPUs are presented in [7]. A FD algorithm for modeling elastic wave propagation in anisotropic media is introduced in [8], which also discusses both single- and multi-GPU implementations. FD solutions of the anisotropic elastic wave equation using singly and fully staggered grids (SSGs and FSGs, respectively) are discussed in [9].

FD methods are often employed for modeling seismic wavefields in exploration geophysics because of their advantages in terms of implementation, code parallelization, compact stencils, and moderate computational cost. However, they suffer from reduced numerical accuracy in anisotropic media without a horizontal symmetry plane due to the challenge of interpolating the results of calculating the partial wavefield derivatives, especially near the free surface and fluid/solid interfaces [10].

Using FD methods based on mimetic (MFD) operators with fully staggered grids (FSGs) overcomes this issue by preserving the underlying physics of the employed partial differential equations (PDE) in the discretization process [11]. Applying FSGs obviates the need for interpolating partial wavefield derivatives, thus making it easier to implement the algorithm for arbitrarily anisotropic media [12]. The performance of the MFD method for acoustic wave propagation on CPUs and GPUs is analyzed in [13]. They discuss 1-D and 2-D implementations using a second-order formulation of the acoustic wave equation and show that the performance of the GPU implementations is consistently better (i.e., they are 6-12 faster) compared to their optimized CPU counterparts.

In this work, we employ GPUs to solve the 3-D anisotropic elastic wave equation using the MFD+FSG approach. Our implementation involves a velocity-stress formulation with FSGs that can handle arbitrarily anisotropic media without losing global numerical discretization accuracy. The MFD+FSG algorithm and MFD operators are applied only to the near-surface model components to reduce the thread divergence of CUDA kernels. Absorbing boundary conditions are used on the five remaining model faces. Multi-GPU communications are handled with a CUDA-aware MPI library both within single and across multiple nodes.

The paper begins with a review of the theory of elastic wave propagation in anisotropic media, the free-surface boundary conditions, and the velocity-stress formulation of the elastic wave equation for orthorhombic media [14] with a horizontal symmetry plane. Next, we describe the major algorithmic steps of the MFD+FSG approach, as well as our CUDA-based multi-GPU implementation. Then, we present numerical examples using large-scale models and benchmark our solution with the spectral-element method. Code-profiling and scaling performance tests are performed for up to 64 GPUs on eight Nvidia A100 nodes. Finally, we present the roofline plot illustrating the performance of the algorithm and discuss future improvements of this method.

2 Elastodynamics theory

The 3-D elastic wave equation in a heterogeneous anisotropic medium can be written as:

$$\rho \dot{v}_i = \partial_j \sigma_{ij} + f_i, \tag{1}$$

where ρ is the density, v_i is the ith component of the particle velocity, the dot marks the temporal derivative, σ_{ij} is the stress tensor, ∂_j denotes the differentiation with respect to the coordinate x_j, and $\mathbf{f}_i$ is the body force per unit volume. We use the following linear constitutive relationship (i.e., generalized Hooke's law) between the temporal derivatives of the stress (σ_{ij}) and strain (ϵ_{kl}) tensors,

$$\dot{\sigma}_{ij} = C_{jkl}\, \dot{\epsilon}_{kl}, \tag{2}$$

where C_{ijkl} is the (time-independent) stiffness tensor and

$$\dot{\epsilon}_{kl} = \frac{1}{2} \left(\partial_l v_k^s + \partial_k v_l^s \right). \tag{3}$$

The free-surface boundary conditions (FSBC) require a vanishing traction vector across the free surface,

$$\sigma_{ij} n_j = 0, \tag{4}$$

where n_j are the directional cosines for a unit vector orthogonal to the area (surface) element. Assuming a flat free surface at $x_3 = 0$, Eq. 4 yields the following three conditions:

$$\sigma_{13} = \sigma_{23} = \sigma_{33} = 0. \tag{5}$$

3 3-D MFD+FSG approach

Generating *globally* high-order [i.e., $O(\Delta x^4)$ or greater] accurate implementations of the FSBC specified in Eq. 5 remains challenging for standard FD methods even for simple Cartesian geometries. Numerical solutions that satisfy the (anisotropic) elastic wave equations and the boundary conditions with the same accuracy tend to become unstable. In addition, many lower-order-accuracy boundary-condition

implementations introduce unphysical ghost points to ensure the continuity of the wavefield derivatives obtained with two-sided FD operators [15]. In addition, wave-equation discretizations based on standard Taylor-series FD operators do not honor such underlying physical concepts as the conservation laws and tensorial calculus properties that are naturally satisfied by their continuum counterparts [16]. These shortcomings lead to numerical instabilities and errors in the simulated wavefields.

An efficient way to address these issues is by employing mimetic operators [16–18]. The MFD divergence and gradient operators, **D** and **G**, honor global conservation laws [16] and can be constructed with $O(\Delta x^4)$ (or greater) accuracy throughout the entire computational domain including at and near boundaries and partitioned interfaces [10, 11, 18]. We use fully staggered grids (FSG) to implement the MFD operators because such grid layout increases the numerical stability by making all partial wavefield derivatives available at each grid point [12, 19]. In contrast to SSGs, this approach eliminates the need for costly high-order interpolation of partial derivatives [12, 20]. Employing an FSG system is particularly useful for lower-symmetry anisotropic models including the most general, triclinic symmetry characterized by 21 independent stiffnesses [20].

A 3-D FSG grid includes four complementary SSGs composed of one base grid and three others staggered by half-grid spacing in the three orthogonal directions. Figure 1a shows the distribution of the particle-velocity variables defined on four SSG grids that form the FSG system: [**v**, **v**, **v**], [**v**, **f**, **f**], [**f**, **v**, **f**] and [**f**, **f**, **v**], where $\mathbf{f} \in \mathbb{R}^{N+2}$, $\mathbf{v} \in \mathbb{R}^{N+1}$ and N represents the number of points in the model along the $\mathbf{x} = [x_1, x_2, x_3]$ coordinate directions. (Note that in following figures we use $\mathbf{x} = [x, y, z]$ in accordance with common geophysical convention.) In Fig. 1a, one also can see the number of points corresponding to each SSG grid that contribute to the injection and extraction of the velocity variables in a single FSG cell. The [**v**, **v**, **v**] grid contributes eight points at the corners in an FSG system, whereas the [**v**, **f**, **f**], [**f**, **v**, **f**], and [**f**, **f**, **v**] grids contribute two points each at the face center in Fig. 1a.

Similarly, the stress variables have to be defined on four SSG grids: [**f**, **f**, **f**], [**v**, **f**, **v**], [**v**, **v**, **f**], and [**f**, **v**, **v**]. Figure 1b shows the number of points corresponding to each employed SSG grid. The [**f**, **f**, **f**] grid contributes one point in the cell center. The [**v**, **f**, **v**], [**v**, **v**, **f**], and [**f**, **v**, **v**] grids contribute four points each in Fig. 1b. For injection and extraction of the velocity and stress wavefields, different weighting coefficients need to be used for different grids as explained below.

Mimetic operators $\mathbf{D_i}$ and $\mathbf{G_i}$ act on the complementary SSG grids in 1-D fashion but along the three coordinate directions. Operator $\mathbf{G_i}$ is applied to the field variables defined on the **f**-grid and maps them to a vector defined on the **v**-grid. Operator $\mathbf{D_i}$ acts on the field variables defined on the **v**-grid and maps them to a vector defined on the **f**-grid. The next section describes the discretization of the stress-velocity formulation with MFD operators for orthorhombic media with a horizontal symmetry plane. However, one can extend this formalism to arbitrarily anisotropic media in a straightforward fashion.

3.1 Stress-field updating

The stress-updating equation for 3-D orthorhombic media with a horizontal symmetry plane can be written as (see Eqs. 2 and 3):

$$\dot{\sigma}_{11} = c_{11}\frac{\partial v_1}{\partial x_1} + c_{12}\frac{\partial v_2}{\partial x_2} + c_{13}\frac{\partial v_3}{\partial x_3}, \tag{6}$$

$$\dot{\sigma}_{22} = c_{12}\frac{\partial v_1}{\partial x_1} + c_{22}\frac{\partial v_2}{\partial x_2} + c_{23}\frac{\partial v_3}{\partial x_3}, \tag{7}$$

$$\dot{\sigma}_{33} = c_{13}\frac{\partial v_1}{\partial x_1} + c_{23}\frac{\partial v_2}{\partial x_2} + c_{33}\frac{\partial v_3}{\partial x_3}, \tag{8}$$

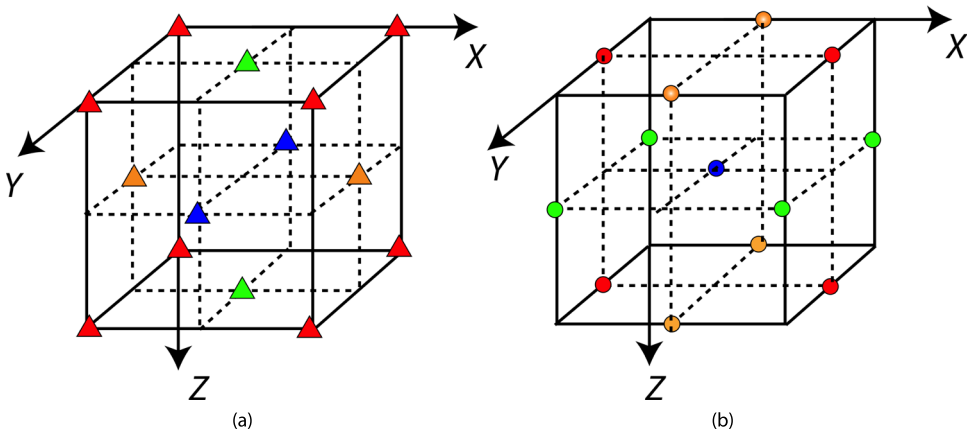

Fig. 1 Coupled system of SSGs inside a 3-D MFD+FSG cell. (a) The [**v**, **v**, **v**], [**v**, **f**, **f**], [**f**, **v**, **f**], and [**f**, **f**, **v**] grid points are in red, green, blue, and yellow, respectively. (b) The [**f**, **f**, **f**], [**v**, **f**, **v**], [**v**, **v**, **f**], and [**f**, **v**, **v**] grid points are in blue, green, red and yellow, respectively

$$\dot{\sigma}_{12} = c_{66}\left(\frac{\partial v_2}{\partial x_1} + \frac{\partial v_1}{\partial x_2}\right), \tag{9}$$

$$\dot{\sigma}_{13} = c_{55}\left(\frac{\partial v_3}{\partial x_1} + \frac{\partial v_1}{\partial x_3}\right), \tag{10}$$

$$\dot{\sigma}_{23} = c_{44}\left(\frac{\partial v_3}{\partial x_2} + \frac{\partial v_2}{\partial x_3}\right), \tag{11}$$

where c_{ij} are the C_{ijkl} stiffness coefficients from Eq. 2 represented in the two-index (Voigt) notation. All stress variables are defined on the four complementary staggered grids mentioned above. For example, the $\dot{\sigma}_{ij}$ updates, herein assumed to be computed on the half time step $n + \frac{1}{2}$, are obtained on the grid [**f**, **f**, **f**] using MFD operators:

$$\dot{\sigma}_{11}^{[f,f,f]} = c_{11}\mathbf{D_1}v_1^{[v,f,f]} + c_{12}\mathbf{D_2}v_2^{[f,v,f]} + c_{13}\mathbf{D_3}v_3^{[f,f,v]}, \tag{12}$$

$$\dot{\sigma}_{22}^{[f,f,f]} = c_{12}\mathbf{D_1}v_1^{[v,f,f]} + c_{22}\mathbf{D_2}v_2^{[f,v,f]} + c_{23}\mathbf{D_3}v_3^{[f,f,v]}, \tag{13}$$

$$\dot{\sigma}_{33}^{[f,f,f]} = c_{13}\mathbf{D_1}v_1^{[v,f,f]} + c_{23}\mathbf{D_2}v_2^{[f,v,f]} + c_{33}\mathbf{D_3}v_3^{[f,f,v]}, \tag{14}$$

$$\dot{\sigma}_{12}^{[f,f,f]} = c_{66}\left(\mathbf{D_1}v_2^{[v,f,f]} + \mathbf{D_2}v_1^{[f,v,f]}\right), \tag{15}$$

$$\dot{\sigma}_{13}^{[f,f,f]} = c_{55}\left(\mathbf{D_1}v_3^{[v,f,f]} + \mathbf{D_3}v_1^{[f,f,v]}\right), \tag{16}$$

$$\dot{\sigma}_{23}^{[f,f,f]} = c_{44}\left(\mathbf{D_2}v_3^{[f,v,f]} + \mathbf{D_3}v_2^{[f,f,v]}\right), \tag{17}$$

where the variables v_i are assumed to be computed on the whole time step n. The superscripts denote the SSG grid where the variables are defined. Table 1 shows the stress updates for the other complementary SSG grid staggerings.

3.2 Velocity-field updating

The velocity-updating equation for 3-D orthorhombic media defined in Eq. 1 can be obtained using

$$\rho\,\dot{v}_1 = \frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{12}}{\partial x_2} + \frac{\partial \sigma_{13}}{\partial x_3}, \tag{18}$$

$$\rho\,\dot{v}_2 = \frac{\partial \sigma_{21}}{\partial x_1} + \frac{\partial \sigma_{22}}{\partial x_2} + \frac{\partial \sigma_{23}}{\partial x_3}, \tag{19}$$

$$\rho\,\dot{v}_3 = \frac{\partial \sigma_{31}}{\partial x_1} + \frac{\partial \sigma_{32}}{\partial x_2} + \frac{\partial \sigma_{33}}{\partial x_3}, \tag{20}$$

where the components $\dot{v}_i$ and σ_{ij} are again assumed to be obtained on the whole (n) and half-time ($n+\frac{1}{2}$) steps, respectively.

Similarly, all velocity variables are defined on the four complementary staggered grids mentioned above. For example, the update of the velocity fields on the [**v**, **v**, **v**]-grid can be computed using MFD operators as follows:

$$\rho\dot{v}_1^{[v,v,v]} = \mathbf{G_1}\sigma_{11}^{[f,v,v]} + \mathbf{G_2}\sigma_{12}^{[v,f,v]} + \mathbf{G_3}\sigma_{13}^{[v,v,f]}, \tag{21}$$

$$\rho\dot{v}_2^{[v,v,v]} = \mathbf{G_1}\sigma_{21}^{[f,v,v]} + \mathbf{G_2}\sigma_{22}^{[v,f,v]} + \mathbf{G_3}\sigma_{23}^{[v,v,f]}, \tag{22}$$

$$\rho\dot{v}_3^{[v,v,v]} = \mathbf{G_1}\sigma_{31}^{[f,v,v]} + \mathbf{G_2}\sigma_{32}^{[v,f,v]} + \mathbf{G_3}\sigma_{33}^{[v,v,f]}. \tag{23}$$

The superscripts denote the SSG grids where the variables are defined. The velocity updates obtained using MFD operators for the other complementary SSG grid staggerings are displayed in Table 2.

3.3 Free-surface implementation

The mimetic nodes at the free surface are updated using the strategy described in [10] by employing the zero-traction FSBC (Eq. 5). The velocities $v_1^{[f,f,v]}$, $v_2^{[f,f,v]}$ and $v_3^{[f,f,v]}$ at the free-surface mimetic points for orthorhombic media are obtained by setting $\dot{\sigma}_{33}^{[v,f,v]} = \dot{\sigma}_{23}^{[v,f,v]} = \dot{\sigma}_{13}^{[v,f,v]} = 0$ at $x_3 = 0$:

$$c_{13}\,\mathbf{G_1}v_1^{[f,f,v]} + c_{23}\,\mathbf{D_2}v_2^{[v,v,v]} + c_{33}\mathbf{G_3}v_3^{[v,f,f]} = 0, \tag{24}$$

$$\mathbf{G_3}v_2^{[v,f,f]} + \mathbf{D_2}v_3^{[v,v,v]} = 0, \tag{25}$$

$$\mathbf{G_1}v_3^{[f,f,v]} + \mathbf{G_3}v_1^{[v,f,f]} = 0. \tag{26}$$

Next, we split up the gradient operators according to $\mathbf{G_1}v_1 = \mathbf{G_1}[\mathbf{0}]v_1^M + \mathbf{G_1^\dagger}v_1^\dagger$ etc. where the superscript M represents mimetic boundary points, and the symbol † indicates all points excluding the mimetic boundary points. (See Appendix A for fourth-order examples of $\mathbf{G_i}$ and $\mathbf{D_i}$.) This

Table 1 Updates of the stress variables obtained from the velocity variables defined on four different staggered grids (SGs) using cyclic MFD operators

Updated variable σ	SG1 [**v**, **v**, **v**]	SG2 [**v**, **f**, **f**]	SG3 [**f**, **v**, **f**]	SG4 [**f**, **f**, **v**]
[**f**, **f**, **f**]	—	$\mathbf{D_1}$	$\mathbf{D_2}$	$\mathbf{D_3}$
[**f**, **v**, **v**]	$\mathbf{D_1}$	—	$\mathbf{G_3}$	$\mathbf{G_2}$
[**v**, **f**, **v**]	$\mathbf{D_2}$	$\mathbf{G_3}$	—	$\mathbf{G_1}$
[**v**, **v**, **f**]	$\mathbf{D_3}$	$\mathbf{G_2}$	$\mathbf{G_1}$	—

Table 2 Updates of the velocity variables from the stress variables defined on four different staggered grids (SGs) using cyclic MFD operators

Updated variable **v**	SG1 [**f**, **f**, **f**]	SG2 [**f**, **v**, **v**]	SG3 [**v**, **f**, **v**]	SG4 [**v**, **v**, **f**]
[**v**, **v**, **v**]	—	$\mathbf{G_1}$	$\mathbf{G_2}$	$\mathbf{G_3}$
[**v**, **f**, **f**]	$\mathbf{G_1}$	—	$\mathbf{D_3}$	$\mathbf{D_2}$
[**f**, **v**, **f**]	$\mathbf{G_2}$	$\mathbf{D_3}$	—	$\mathbf{D_1}$
[**f**, **f**, **v**]	$\mathbf{G_3}$	$\mathbf{D_2}$	$\mathbf{D_1}$	—

definition allows us to rearrange Eqs. 24-26 to obtain

$$v_1^M = \frac{1}{c_{13}\mathbf{G_1}[\mathbf{0}]}\left(-c_{13}\mathbf{G}_\mathbf{1}^\dagger v_1^{[f,f,v]} - c_{23}\mathbf{D_2}v_2^{[v,v,v]} - c_{33}\mathbf{G_3}v_3^{[v,f,f]}\right), \quad (27)$$

$$v_2^M = -\frac{1}{\mathbf{G_3}[\mathbf{0}]}\left(-\mathbf{G}_\mathbf{3}^\dagger v_2^{[v,f,f]} - \mathbf{D_2}v_3^{[v,v,v]}\right), \quad (28)$$

$$v_3^M = -\frac{1}{\mathbf{G_1}[\mathbf{0}]}\left(-\mathbf{G}_\mathbf{1}^\dagger v_3^{[f,f,v]} - \mathbf{G_3}v_1^{[v,f,f]}\right), \quad (29)$$

where $\mathbf{G_1}[\mathbf{0}]$ and $\mathbf{G_3}[\mathbf{0}]$ denote the coefficients of the mimetic gradient operator at $x_1 = 0$ and $x_3 = 0$, respectively. Mimetic velocity nodes $v_1^{[f,v,f]}$, $v_2^{[f,v,f]}$, and $v_3^{[f,v,f]}$ are updated in a similar fashion.

3.4 Absorbing boundary conditions

We implemented the convolutional perfectly matched layer (C-PML) boundary conditions [21, 22] on the remaining five model sides. The implementation of the absorbing conditions is straightforward; however, due to the intertwining of four staggered grids, the C-PML conditions must be applied on all four grids for the velocity and stress field variables alike.

3.5 Source injection and wavefield extraction

The coupling of multiple grids requires distributed source injection and wavefield extraction in an FSG system. The distribution of stress nodes on four SSGs which are involved in the cell-centered injection and extraction in the FSG system is shown in Fig. 1b. The stress nodes on $[\mathbf{f}, \mathbf{f}, \mathbf{f}]$ are injected with a unit weight, whereas the nodes at the three other SSGs are injected with a weight of 0.25. The stress nodes on $[\mathbf{f}, \mathbf{f}, \mathbf{f}]$ are extracted with a 0.25 weight, and the rest of the three SSGs are extracted with a weight of 0.0625. Figure 1a illustrates the distribution of the velocity nodes on the four SSGs involved in the cell-centered velocity injection and extraction. The velocity nodes on $[\mathbf{v}, \mathbf{v}, \mathbf{v}]$ are injected with a weight of 0.125 and the three other SSGs are injected with a weight of 0.5. The velocity nodes on $[\mathbf{v}, \mathbf{v}, \mathbf{v}]$ are extracted with a weight of 0.03125 and other three SSGs are extracted with a weight of 0.125.

4 GPU Implementation

Concurrently running thousands of threads on GPUs leads to significant speedups compared to CPU implementations. To simplify the GPU code, we make several algorithmic modifications. First, all four SSGs are adjusted to coincide with the $[\mathbf{f}, \mathbf{f}, \mathbf{f}]$ grid [i.e., $(N + 2, N + 2, N + 2)$]. Second, to minimize the thread divergence we apply MFD operators to compute the partial derivatives only near the free surface for implementation of the FSBC; the absorbing boundary conditions are applied on the other five model sides.

Table 3 Pseudocode that outlines the main steps of the 3-D MFD+FSG numerical solution and references the relevant equations

Step	Substep	Instruction	Equation(s)
0		Read stiffness matrix, source wavelet, and density fields.	—
1		Initialize data structures on GPU and copy the data from CPU to GPU.	—
2		For all time steps:	—
3		Update mimetic stress nodes	6-11
4		Update stress field	6-11
	4a	Update stress halos	—
	4b	Asynchronously communicate stress halos and compute stress in the middle.	—
5		Inject source	—
6		Update mimetic velocity nodes	18-20
7		Update velocity field	18-20
	7a	Update velocity halos	—
	7b	Asynchronously communicate velocity halos and compute velocity in the middle.	—
8		Apply FSBC	5 and 27-29
9		Iterate steps 1-8	—

The major algorithmic steps, including updating velocity and stress wavefield variables, source injection and wavefield extraction, and FSBC application are implemented as separate GPU kernels. The C-PML absorbing boundary conditions are applied as part of updating the stress and velocity kernels. Table 3 shows the major steps in our MFD+FSG algorithm; Steps 3-8 are implemented as GPU kernels. We use the strategy proposed in [5] for updating the field variables to reduce global memory usage and read redundancy. The order of axes from fast to slow is x_1, x_2, and x_3. The data in the fast $[x_1, x_2]$-plane are loaded into the shared memory, while the variables along x_3-axis are held in registers. Domain decomposition is performed along the x_3-axis (Fig. 2) to facilitate the application of FSBC and transfer the data only in the x_3-direction.

The intertwining of the four SSGs introduces greater algorithmic complexity. For example, $v_1^{[v,f,f]}$ is only involved in the update of $\sigma_{11}^{[f,f,f]}$, $\sigma_{22}^{[f,f,f]}$, and $\sigma_{33}^{[f,f,f]}$, but not in updating the other stress components of the same SSG. However, the overall redundancy of the algorithm is same as that for SSGs. The $v_1^{[v,f,f]}$ still contributes to the partial update of the shear-stress ($\sigma_{23}^{[v,f,v]}$ and $\sigma_{13}^{[v,v,f]}$) components of the intertwined SSGs. First, we read and load $v_1^{[v,f,f]}$ to shared

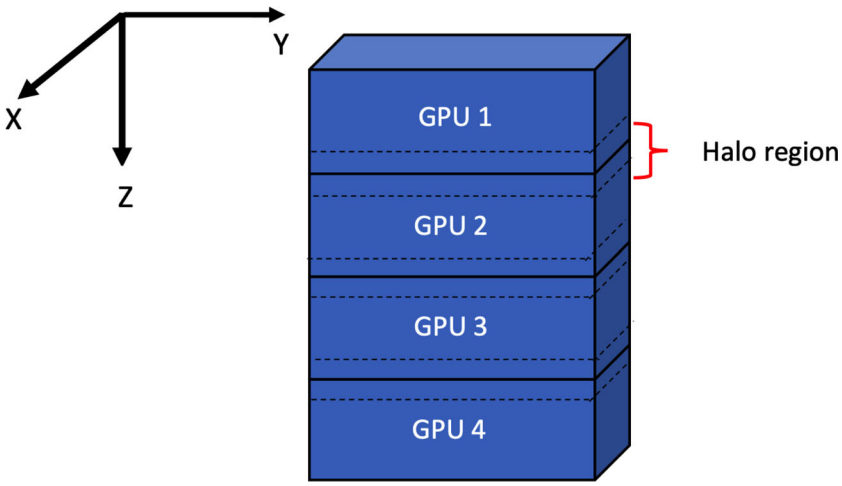

Fig. 2 1-D domain decomposition along the x_3-direction for the example of four GPUs. The dashed line marks the domain-boundary halo regions that need to be communicated to the GPU(s) hosting adjacent compute domains

memory, and then partially update the three stress components to reduce the number of global memory accesses. Updating the shear stress components requires the field v_1 to be defined on different SSGs; however, such redundancy is not feasible for the velocity update kernel that relies on different stress components.

The partial derivatives are first stored in the registers before applying Hooke's law to facilitate application of the absorbing boundary conditions. The C-PML is applied during the stress and velocity update and the size of the C-PML must be an integer multiple of 16 to reduce the thread divergence. The damping coefficient and other state variables associated with C-PML are stored in constant memory along with the mimetic operators. However, the convolutional variables for C-PML are stored in the device memory because of their large size.

We use CUDA streams to compute the FD halo regions, update mimetic nodes, and communicate data in halo regions across GPUs for both single- and multi-node implementation. Figure 3 shows the CUDA code profile for two GPUs. The communication of the halo region across the GPUs in Fig. 2 completely overlaps with velocity and stress computation in the internal region. For communication, we employ a CUDA-Aware MPI library, which can directly use the GPUDirect technologies and the RDMA feature. That feature permits direct inter-GPU communication in the GPU buffers without first staging them to the CPU through a connector. NVIDIA GPUDirect technologies provide high-bandwidth, low-latency communications with NVIDIA GPUs and cover all types of inter-rank communications (intra-node, inter-node, and RDMA inter-node).

5 Numerical examples

For numerical tests, we use the $O(\Delta x^4)$ MFD divergence and gradient operators. The algorithm is applied to an orthorhombic model [23] with the symmetry planes that coincide with the Cartesian coordinate planes. The model size is $[N_x, N_y, N_z] = [600, 500, 320]$ with a uniform 10 m

Fig. 3 Overlap of computation and communication across two GPUs. The upper and lower parts of the plot represent the workflow of two MPI processes corresponding to two GPUs. The blue boxes mark the computational kernels and the brown ones mark the communication between the GPUs. The exchange of the velocity and stress halos completely overlaps with computation of the internal velocity and stress nodes, respectively

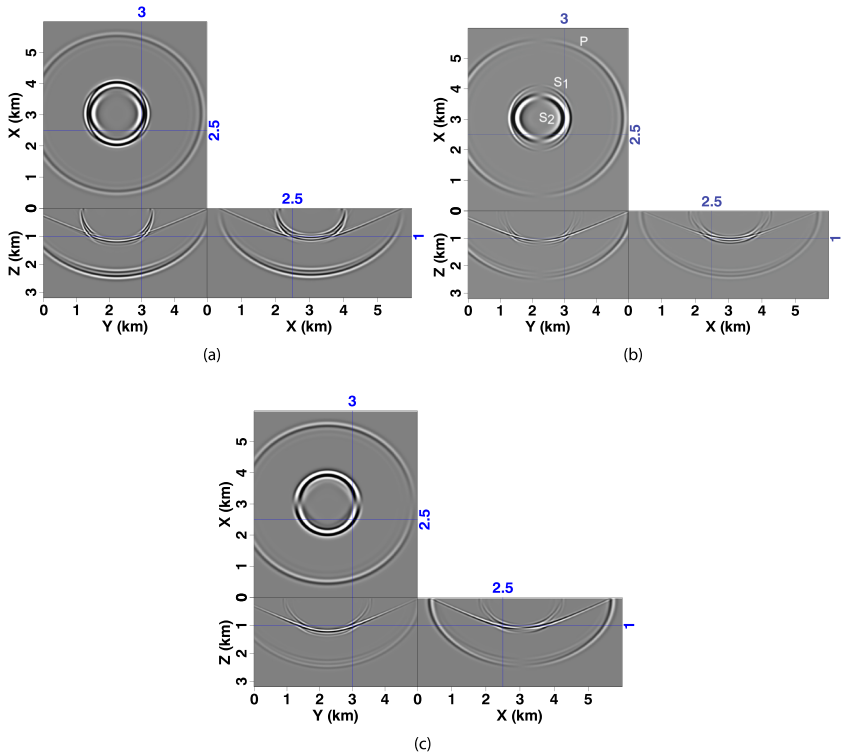

Fig. 4 Snapshots of the velocity components (a) v_1, (b) v_2 and (c) v_3 extracted at t=0.4 s. The model is a homogeneous orthorhombic solid with the following Tsvankin's [14, 23] parameters: V_{P0} = 2.0 km/s, V_{S0} = 1.2 km/s, $\epsilon_1 = 0.2$, $\epsilon_2 = 0.15$, $\delta_1 = 0.1$, $\delta_2 = 0.07$, $\delta_3 = 0.05$, $\gamma_1 = 0.1$, and $\gamma_2 = 0.2$. The P, S_1, and S_2 annotations on plot (b) denote the compressional, fast shear, and slow shear waves, respectively

discretization spacing. We inject a Ricker wavelet with a 20 Hz peak frequency and time increment Δt = 0.5 ms at $[x, y, z] = [2.25, 2.25, 1.0]$ km. After running the simulation for 3000 time steps, we observe both P- and S-wavefields that include free-surface reflections and mode conversions, as illustrated in Fig. 4a-c. Also, the shear-wave splitting into the fast (S_1) and slow (S_2) modes is visible in the $[x, y]$-plane (Fig. 4b).

Next, we consider a model composed of two horizontal orthorhombic layers with the parameters listed in Table 4. The model size is $[N_x, N_y, N_z] = [600, 500, 400]$ with a uniform 10 m grid spacing. We inject a Ricker wavelet with a peak frequency of 20 Hz and Δt = 0.2 ms at $[x, y, z] = [3.0, 3.0, 1.0]$ km. Figure 5 displays particle-velocity snapshots for all three components with the layer boundaries underlain. We observe the S-wave splitting, as well as P- and S-wave reflections from the layer boundaries and the free surface. A good match between the MFD (red) and spectral-element [24] (black) solutions confirms the numerical accuracy of our GPU algorithm (Fig. 6).

For the final test, we consider a heterogeneous model based on a section from the Northern Carnarvon Basin (NCB) located on Western Australia's North West Shelf [25, 26]. We modify the original isotropic model to make it orthorhombic and obtain the Tsvankin parameters [14, 23] by scaling the P- and S-wave velocities and density (see Fig. 7). The entire model is subsampled to $[N_x, N_y, N_z] = [1000, 800, 480]$ with a uniform 5 m grid spacing. A Ricker wavelet with a 20 Hz peak frequency and Δt = 0.01 ms is injected at $[x, y, z] = [2.5, 2.0, 1.2]$ km. Figure 8 shows the particle-velocity components that exhibit complex sea-floor scattering nodes (Fig. 8b), surface waves, and P- and S-wave reflections.

6 Performance analysis

We test the code on GPU nodes of a DGX A100 system, which consists of eight A100 GPUs (80 GB per card) per node interconnected via NVLINK. Figure 9a shows the weak-

Table 4 Tsvankin's parameters for the model with two horizontal orthorhombic layers. The depth of the interface is 2 km

Layers	V_{P0}	V_{S0}	ρ	ϵ_1	ϵ_2	δ_1	δ_2	δ_3	γ_1	γ_2
Unit	km/s	km/s	g/cm^3	—	—	—	—	—	—	—
Top	2.5	1.3	1.0	0.2	0.1	0.15	0.1	0.07	0.1	0.2
Bottom	3.0	1.6	2.0	0.25	0.17	0.12	0.05	0.05	0.1	0.2

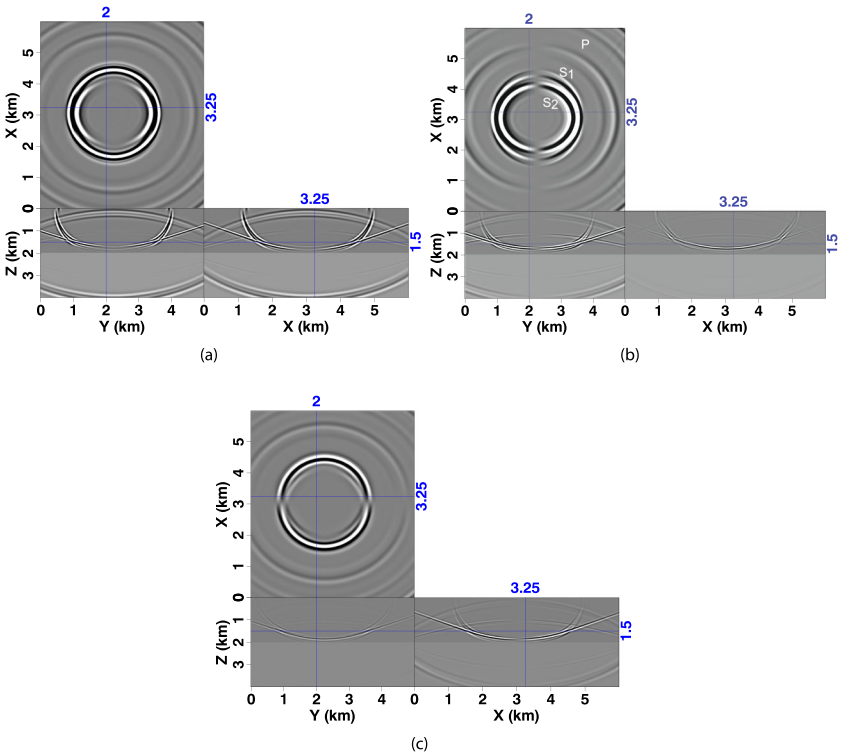

Fig. 5 Snapshots of the velocity components extracted at t=0.4 s for the orthorhombic model from Table 4: (a) v_1, (b) v_2, and (c) v_3

scaling efficiency tests (i.e., the model size per GPU is the same) when using up to 64 GPUs (or 8 DGX A100 nodes). The model size is $[N_x, N_y, N_z] = [1024, 1024, 256 \times N]$, where N is the number of GPUs. We see a minor drop in the

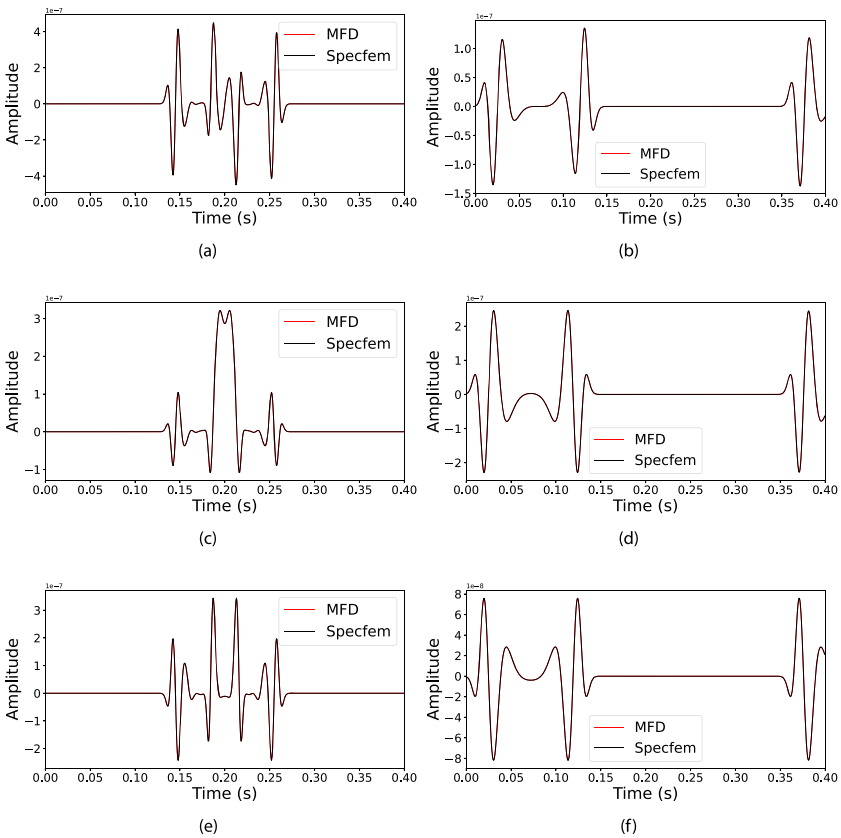

Fig. 6 Comparison of the seismograms computed by our algorithm ("MFD") and the spectral-element method ("Specfem"). The particle-velocity components recorded at $[x, y, z] = [3.0, 2.0, 0.0]$ km [(a) v_1, (c) v_2, and (e) v_3]; and at $[x, y, z] = [2.0, 3.0, 0.0]$ km [(b) v_1, (d) v_2, and (f) v_3]

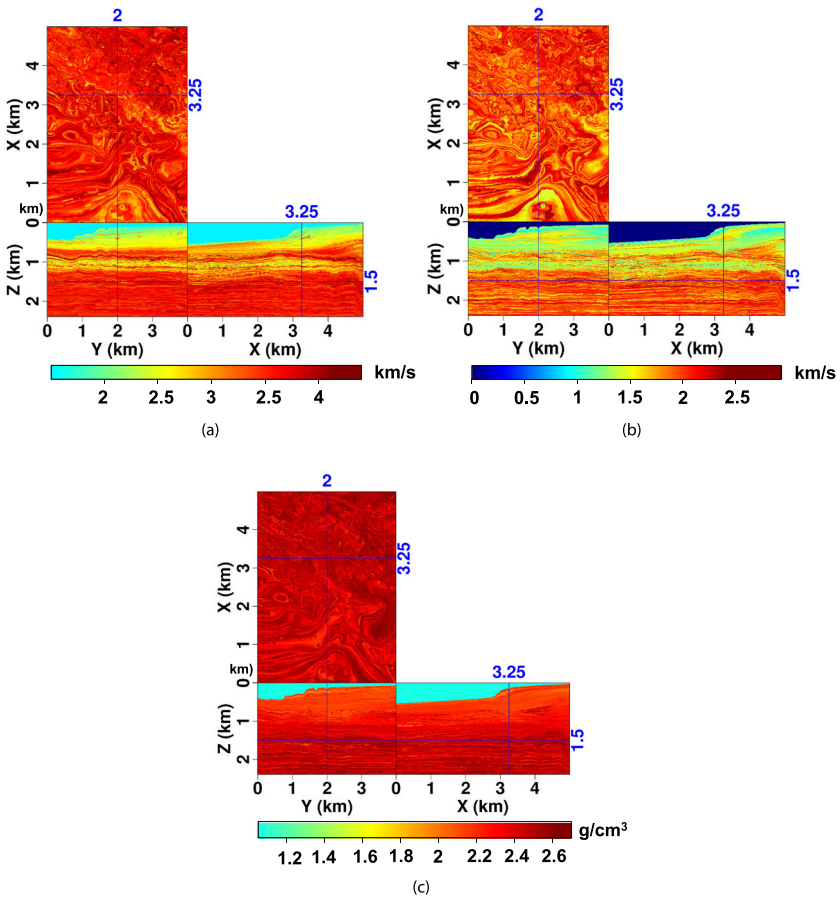

Fig. 7 Parameters of an isotropic medium from the Western Australia Modeling project (WAMO) based on a model developed for the Carnarvon Basin on the Western Australian Shelf: (a) V_P, (b) V_S, and (c) ρ [25, 26]

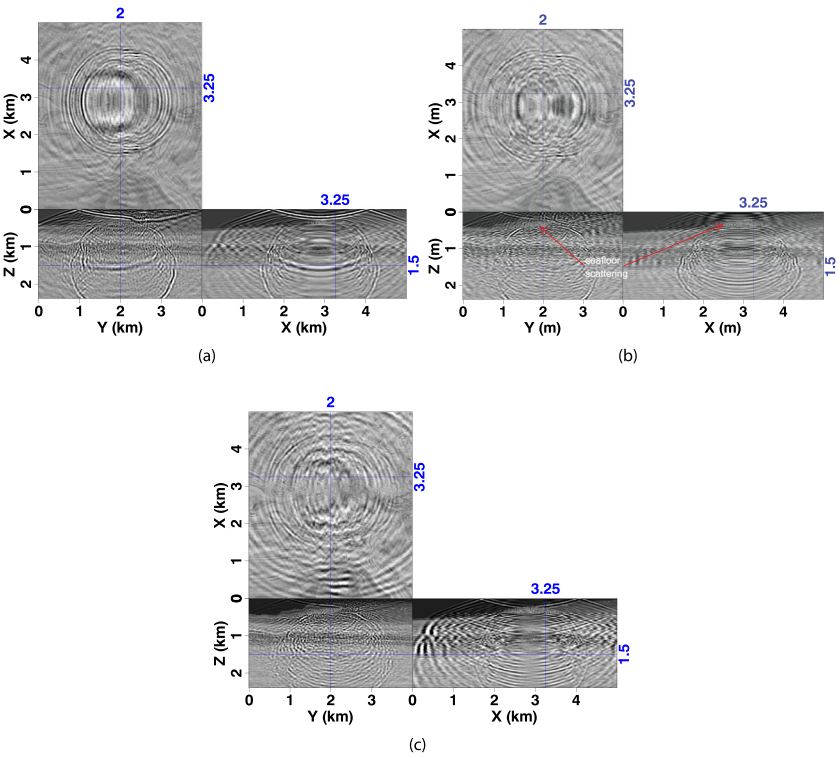

Fig. 8 Snapshots of the velocity components (a) v_1, (b) v_2, and (c) v_3 extracted at t=0.8 s. The superimposed model represents an orthorhombic extension of the isotropic WAMO model from Fig. 7

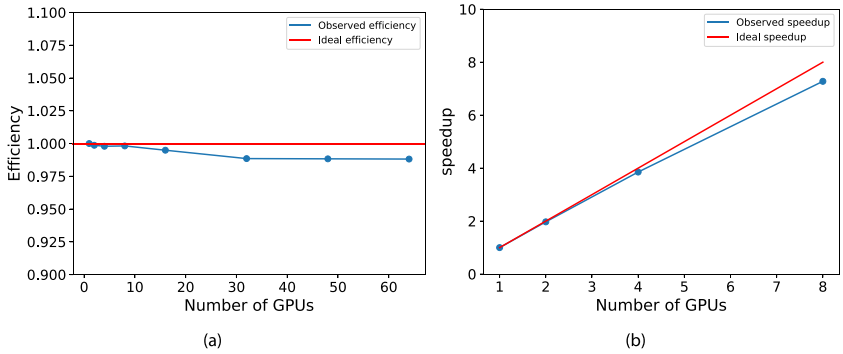

Fig. 9 (a) Weak-scaling efficiency for $[N_x, N_y, N_z]$=[1024, 1024, 256×N]. (b) The strong-scaling efficiency for $[N_x, N_y, N_z]$=512^3

performance after moving from eight GPUs on one node to 64 GPUs on eight nodes (Fig. 9a). However, the efficiency remains above 98% in comparison with single-node GPU runs. The blue line in Fig. 9a suggests a negligible communication overhead incurred by NVLINK for single and InfiniBand for multi-node tests.

For the strong-scaling test we use the model size $[N_x, N_y, N_z] = 512^3$ and increase the number of GPUs, while fixing the model dimensions. Figure 9b shows the results of the speedup test as we increase the number of A100 GPUs from one to eight within a single node. Overall, we observe quasi-linear speedup in the strong-scaling efficiency test.

The "roofline plot" in Fig. 10 illustrates the peak performance of the algorithm expected with the given GPU architecture. The blue curves indicate the maximum single- and double-precision performance that can be achieved on Nvidia A100 GPUs with the given arithmetic intensity of the developed algorithm. Overall, the performance of the implemented MFD+FSG algorithm is close to the achievable peak value.

7 Discussion

The MFD+FSG algorithm scales well with increasing number of GPUs using the MPI+CUDA framework. However, we have not considered such optimizations as the 1-D kernel-level decomposition of the model instead of the current 2-D, utilization of constant memory, grid transposition, and other specific GPU architecture improvements, which would be interesting to explore in the future.

The roofline plot in Fig. 10 also indicates that the arithmetic intensity of the MFD+FSG algorithm is insufficient to fully utilize the maximum resources on A100 GPUs. Efficiency improvements could likely be achieved by using higher-order MFD stencils that involve additional floating point operations per grid point. In general, FSG-based algorithms (including the one presented here) achieve higher accuracy than SSG-based codes using the same grid spacing, but are more memory-intensive due to the use of four times as many computational grids.

8 Conclusions

We developed an efficient GPU-based MFD+FSG algorithm using the velocity-stress formulation of the anisotropic elastic wave equation. The algorithm can be used for large-scale models (i.e., $N_x \times N_y \times N_z > 512^3$) with orthorhombic and even lower symmetry. The weak- and strong-scaling tests demonstrate that the communication overhead using the CUDA-Aware MPI library is negligible for both single-

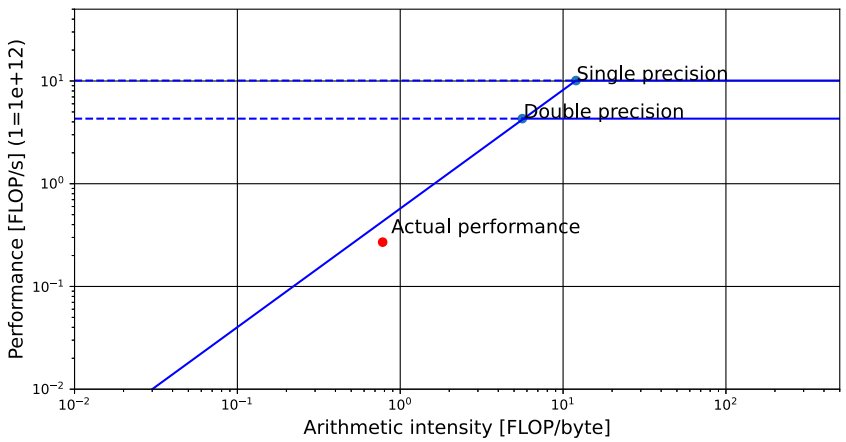

Fig. 10 Roofline plot for the MFD+FSG algorithm. The horizontal axis represents the number of floating point operations (FLOPs) performed per byte and the vertical axis represents the overall performance in FLOPs/s. The red dot marks the actual performance of our MFD+FSG algorithm relative to the theoretical performance shown in blue

and multi-node GPU systems. Comparison of the simulated wavefields with those generated by the spectral-element method confirms the numerical accuracy of our algorithm. We also tested the algorithm using up to 64 A100 GPU cards on eight nodes and achieved a quasi-linear computational speedup with over 98% efficiency.

Appendix A - MFD operators

The fourth-order 1-D mimetic divergence operator [18] $\mathbf{D} \in \mathbb{R}^{(N+2,N+1)}$ operating on the N-point grid is given by:

$$\mathbf{D}_i = \frac{1}{h}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \dots & \dots & \dots \\ \frac{-11}{12} & \frac{17}{24} & \frac{3}{8} & \frac{-5}{24} & \frac{1}{24} & 0 & \dots & \dots & \dots \\ 0 & \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots & \dots & \dots \\ 0 & 0 & \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \dots & \dots & 0 & \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots \\ \dots & \dots & \dots & 0 & \frac{-1}{24} & \frac{5}{24} & \frac{-3}{8} & \frac{-17}{24} & \frac{11}{12} \\ \dots & \dots & \dots & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{30}$$

where the dots indicate the repetition of the stencil coefficients in the corresponding direction, and h is the spatial discretization along the ith direction.

The fourth-order 1D mimetic gradient operator $\mathbf{G} \in \mathbb{R}^{(N+1,N+2)}$ [18] for the N-point grid can be written as:

$$\mathbf{G}_i = \frac{1}{h}\begin{bmatrix} \frac{-352}{105} & \frac{35}{8} & \frac{-35}{24} & \frac{21}{40} & \frac{-5}{56} & 0 & \dots & \dots & \dots \\ \frac{16}{105} & \frac{-31}{24} & \frac{29}{24} & \frac{-3}{40} & \frac{1}{168} & 0 & \dots & \dots & \dots \\ 0 & \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \dots & \dots & 0 & \frac{1}{24} & \frac{-9}{8} & \frac{9}{8} & \frac{-1}{24} & 0 & \dots \\ \dots & \dots & \dots & 0 & \frac{-1}{168} & \frac{3}{40} & \frac{-29}{24} & \frac{31}{24} & \frac{-16}{105} \\ \dots & \dots & \dots & 0 & \frac{5}{56} & \frac{-21}{40} & \frac{35}{24} & \frac{-35}{8} & \frac{352}{105} \end{bmatrix}. \tag{31}$$

By examining the first row of $\mathbf{G_i}$, one can identify $\mathbf{G_i}[\mathbf{0}] = -\frac{352}{105}$ and the non-zero components of $\mathbf{G_i}^{\dagger} = \left[\frac{35}{8}, \frac{-35}{24}, \frac{21}{40}, \frac{-5}{56}\right]$.

Acknowledgements This work was supported by the sponsors of the Consortium Project on Seismic Inverse Methods for Complex Structures at the Center for Wave Phenomena at Colorado School of Mines. The spectral-element solutions are computed using the SpecFEM3D package (https://github.com/geodynamics/specfem3d). We would like to thank NVIDIA for providing the computational resources for running the weak-scaling tests.

Funding This research was sponsored by the Center for Wave Phenomena research consortium at Colorado School of Mines.

Code availability The code will be available via NVIDIA Energy SDK (https://github.com/NVIDIA/energy-sdk/mimeticfd) once published.

Declarations

Competing interests The authors have no competing interests to declare that are relevant to the content of this article.

References

1. De Basabe, J.D., Sen, M.K.: A comparison of finite-difference and spectral-element methods for elastic wave propagation in media with a fluid-solid interface. Geophys. J. Int. **200**(1), 278–298 (2014)
2. Foltinek, D., Eaton, D., Mahovsky, J., Moghaddam, P., McGarry, R.: Industrial-scale reverse time migration on GPU hardware. In: SEG Technical Program Expanded Abstracts 2009, pp. 2789–2793. Society of Exploration Geophysicists, Tulsa (2009)
3. Komatitsch, D., Erlebacher, G., Göddeke, D., Michéa, D.: High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster. J. Comput. Phys. **229**(20), 7692–7714 (2010)
4. Mu, D., Chen, P., Wang, L.: Accelerating the discontinuous Galerkin method for seismic wave propagation simulations using the graphic processing unit (GPU)–single-GPU implementation. Comput. Geosci. **51**, 282–292 (2013)
5. Micikevicius, P.: 3D Finite Difference Computation on GPUs using CUDA. In: Proceedings of 2nd workshop on general purpose processing on graphics processing units, pp. 79–84. (2009)
6. Abdelkhalek, R., Calandra, H., Coulaud, O., Roman, J., Latu, G.: Fast seismic modeling and reverse time migration on a GPU cluster. In: 2009 International Conference on High Performance Computing & Simulation, pp. 36–43. IEEE (2009)
7. Nakata, N., Tsuji, T., Matsuoka, T.: Acceleration of computation speed for elastic wave simulation using a graphic processing unit. Explor. Geophys. **42**(1), 98–104 (2011)
8. Weiss, R.M., Shragge, J.: Solving 3D anisotropic elastic wave equations on parallel GPU devices. Geophysics **78**, F7–F15 (2013)
9. Rubio, F., Hanzich, M., Farrés, A., de la Puente, J., Cela, J.M.: Finite-difference staggered grids in GPUs for anisotropic elastic wave propagation simulation. Comput. Geosci. **70**, 181–189 (2014)
10. Sethi, H., Shragge, J., Tsvankin, I.: Mimetic finite-difference coupled-domain solver for anisotropic media. Geophysics **86**, T45–T59 (2021)
11. Castillo, J.E., Miranda, G.F.: Mimetic Discretization Methods. Chapman and Hall/CRC, London (2013)
12. de la Puente, J., Ferrer, M., Hanzich, M., Castillo, J.E., Cela, J.M.: Mimetic seismic wave modeling including topography on deformed staggered grids. Geophysics **79**(3), T125–T141 (2014)
13. Otero, B., Francés, J., Rodriguez, R., Rojas, O., Solano, F., Guevara-Jordan, J.: A performance analysis of a mimetic finite difference scheme for acoustic wave propagation on GPU platforms. Concurr. Comput. Pract. Experience **29**(4), e3880 (2017)
14. Tsvankin, I.: Seismic signatures and analysis of reflection data in anisotropic media, 3rd edn. Society of Exploration Geophysicists, Tulsa (2012)
15. Sun, Y.C., Zhang, W., Xu, J.K., Chen, X.: Numerical simulation of 2-D seismic wave propagation in the presence of a topographic fluid-solid interface at the sea bottom by the curvilinear grid finite-difference method. Geophys. J. Int. **210**(3), 1721–1738 (2017)
16. Castillo, J.E., Grone, R.: A matrix analysis approach to higher-order approximations for divergence and gradients satisfying a global conservation law. SIAM J. Matrix Anal. Appl. **25**(1), 128–142 (2003)

17. Rojas, O.: Mimetic finite difference modeling of 2D elastic P-SV wave propagation. Qualifying examination report, Computational Science Research Center, San Diego State University (2007)
18. Corbino, J., Castillo, J.: Computational Science & Engineering. San Diego State University, San Diego (2017)
19. Shragge, J., Tapley, B.: Solving the tensorial 3D acoustic wave equation: A mimetic finite-difference time-domain approach. Geophysics **82**(4), T183–T196 (2017)
20. Lisitsa, V., Vishnevskiy, D.: Lebedev scheme for the numerical simulation of wave propagation in 3D anisotropic elasticity. Geophys. Prospect. **58**(4), 619–635 (2010)
21. Roden, J.A., Gedney, S.D.: Convolution PML (CPML): An efficient FDTD implementation of the CFS-PML for arbitrary media. Microw. Opt. Technol. Lett. **27**(5), 334–339 (2000)
22. Martin, R., Komatitsch, D.: An unsplit convolutional perfectly matched layer technique improved at grazing incidence for the viscoelastic wave equation. Geophys. J. Int. **179**(1), 333–344 (2009)
23. Tsvankin, I.: Anisotropic parameters and P-wave velocity for orthorhombic media. Geophysics **62**(4), 1292–1309 (1997)
24. Komatitsch, D., Tromp, J.: Introduction to the spectral element method for three-dimensional seismic wave propagation. Geophys. J. Int. **139**(3), 806–822 (1999)
25. Shragge, J., Bourget, J., Lumley, D., Giraud, J., Wilson, T., Iqbal, A., Emami Niri, M., Whitney, B., Potter, T., Miyoshi, T., Witten, B.: The Western Australia Modeling Project. Part 1: Geomodel Building. Interpretation **7**(4), T773–T791 (2019a)
26. Shragge, J., Lumley, D., Bourget, J., Potter, T., Miyoshi, T., Witten, B., Giraud, J., Wilson, T., Iqbal, A., Emami Niri, M.: The Western Australia Modeling Project. Part 2: Seismic Validation. Interpretation **7**(4), T793–T807 (2019b)