L-Sweeps: A scalable, parallel preconditioner for the high-frequency Helmholtz equation

Matthias Taus[a,b], Leonardo Zepeda-Núñez[c,d,e], Russell J. Hewett[f,g], Laurent Demanet[b]

[a] *Institute of Analysis and Scientific Computing, Vienna University of Technology*
[b] *Department of Mathematics, Massachusetts Institute of Technology*
[c] *Computational Research Division, Lawrence Berkeley National Laboratory*
[d] *Department of Mathematics, University of California Berkeley*
[e] *Department of Mathematics, University of Wisconsin-Madison*
[f] *Total E&P Research and Technology, USA*
[g] *Department of Mathematics, Virginia Tech*

Abstract

We present the first fast solver for the high-frequency Helmholtz equation that scales optimally in parallel, for a single right-hand side. The L-sweeps approach achieves this scalability by departing from the usual propagation pattern, in which information flows in a 180° degree cone from interfaces in a layered decomposition. Instead, with L-sweeps, information propagates in 90° cones induced by a checkerboard domain decomposition (CDD). We extend the notion of accurate transmission conditions to CDDs and introduce a new sweeping strategy to efficiently track the wave fronts as they propagate through the CDD. The new approach decouples the subdomains at each wave front, so that they can be processed in parallel, resulting in better parallel scalability than previously demonstrated in the literature. The method has an overall $O((N/p)\log\omega)$ empirical run-time for $N = n^d$ total degrees-of-freedom in a d-dimensional problem, frequency ω, and $p = O(n)$ processors. We introduce the algorithm and provide a complexity analysis for our parallel implementation of the solver. We corroborate all claims in several two- and three-dimensional numerical examples involving constant, smooth, and discontinuous wave speeds.

Acknowledgements

The authors thank Total SA for support and for permission to release the example code. LD is also supported by AFOSR grant FA9550-17-1-0316. The BP model is provided courtesy of BP and Fréderic Billette. We thank NERSC for computation resources. MT thanks the Institute of Applied Mathematics at Graz University of Technology for hosting him for part of this research.

1. Introduction

The Helmholtz equation, a time-harmonic form of the wave equation, arises in modeling many physical phenomena, including electromagnetic and subsurface wave propagation. Such applications are of interest when solving related inverse problems that require solutions at high frequency to recover fine-grained details, e.g., ultrasound and subsurface recovery. In subsurface recovery, the propagation medium tends to be extremely complex and the asymptotic approximations ubiquitous in other modalities are not sufficiently accurate, thus the numerical solution of the wave equation is required. Such computations are the backbone of the full-waveform inversion (FWI) method for subsurface recovery [81, 63]. In the context of time-harmonic wave equations, such as the Helmholtz equation, accurate reconstructions of the subsurface via FWI require solutions at a wide range of frequencies. The finest recoverable detail is determined by the highest frequency for which the wave equation can be solved. Consequently, the efficient solution of time-harmonic wave equations at high-frequency is extremely important in scientific and industrial applications.

We consider the Helmholtz equation with variable wave speed and constant density on an open domain $\Omega_{\texttt{bulk}}$ with absorbing boundary conditions,

$$\begin{aligned} -\Delta u - \omega^2 m u &= f \quad \text{in } \Omega_{\texttt{bulk}}, \\ &+ \text{A.B.C on } \partial\Omega_{\texttt{bulk}}, \end{aligned} \tag{1}$$

where for $x \in \Omega_{\texttt{bulk}}$, $m(x) = 1/c^2(x)$ is the squared slowness for the p-wave speed c, u is the solution wavefield, ω is the characteristic frequency, and f is the source density. We consider $\Omega_{\texttt{bulk}}$ to be a square or a cube, but this is not a limiting assumption. While we consider only problems of this form in the ensuing developments, the method we propose is a framework that can be applied to other time-harmonic formulations that model more complex physics.

In particular, we consider discretizations of (1) in the high-frequency regime. This means that the coarsest structure in the spatial discretization (mesh) has to scale as $1/\omega$. In this regime the solution of the discrete system is notoriously difficult [29]. It is well-established that, independent of the discretization, the spectrum of the resulting system matrix deteriorates as the frequency ω increases [57, 69]. Therefore, it is not feasible to solve systems arising from high-frequency problems without the use of specialized solution strategies which directly address this issue.

A number of solvers for solving the resulting linear systems are available, including direct methods (e.g., [82]), domain decomposition methods [78], and preconditioned iterative methods (including, e.g., multigrid and shifted-Laplacian methods) [67, 39]. We consider a solver or a preconditioner to be *sequentially scalable* if, up to logarithmic factors, it can compute a solution in $O(N)$ run-time in a sequential computational environment where N is the total number of degrees-of-freedom in the discrete problem. We consider a solver or a preconditioner to be *parallel scalable* if, up to logarithmic factors, it can compute a solution in $O(N/p)$ run-time in a parallel computational environment with $p = O(n)$ processors, for a single right-hand side. Here, n is the number of degrees of freedom in one spatial direction, i.e., $n = O(N^{\frac{1}{d}})$ where d is the problem dimension. In this paper, we present what we believe to be the first *parallel scalable* preconditioner for the high-frequency Helmholtz equation.

Currently, no scalable direct method, sequential or parallel, is available for the high-frequency regime. Standard domain decomposition methods (DDMs) localize the problem to subdomains and transfer information between subdomains. Domain decomposition methods can be applied as a direct solver (e.g., in the context of optimal Schwarz methods [35, 33]), but the resulting solver is not scalable. Domain decomposition methods can yield scalable preconditioners, both in parallel and sequentially. However, the resulting preconditioned iterative solver is not scalable because it requires $O(\omega)$ iterations. This shows an important aspect for the construction of solvers based on preconditioning: a quality preconditioner has to be scalable but also the resulting iterative method has to converge with limited growth in iterations as ω increases. In this regard, we call a preconditioner *effective* if the resulting preconditioned system can be solved in $O(\log \omega)$ iterations. Classical DDMs exhibit sub-optimal behavior for two reasons. First, artificial and spurious reflections are induced by imprecise information transfer between subdomains. Second, long-range wave-material interactions are not tracked consistently.

Sweeping preconditioners have been introduced [26, 80, 16, 85, 32] to alleviate these drawbacks, while preserving the advantages of classical DDMs. Sweeping preconditioners make use of layered domain decompositions, accurate transmission conditions, and a layer-by-layer sweeping strategy. In particular, the layered domain decomposition provides scalability by controlling growth in computational cost and memory footprint. The accurate transmission conditions allow information to flow between the subdomains without numerical artifacts, e.g., artificial reflections. Finally, the sweeping strategy consistently tracks and propagates long-range wave-material interactions. The resulting approach can be interpreted as an approximate block-LU factorization, where the blocks correspond to the local problem in each subdomain. In particular, using sparse direct solvers on blocks that arise from sufficiently thin layers yields a method with quasi-linear (i.e., linear with poly-logarithmic factors) sequential complexity for a single right-hand side [26, 25, 86, 53, 72, 16, 80, 56]. In the presence of many right-hand sides, the layered domain decomposition allows for optimal parallelization [89].

Efforts to improve the performance of sweeping preconditioners have focused on obtaining better and more accurate factorizations [80], reducing the over-all cost by restricting the unknowns to the interfaces [85, 86], and accelerating the computation on local subproblems (blocks in the approximate factorization) with compression or parallelization [62]. Several approaches aim specifically to sparsify those blocks, thus decreasing the sequential costs [53, 86]. However, while leveraging parallelism to accelerate the solve for local blocks decreases the run-times, it does not result in a parallel scalable solver for a single right-hand side.

The main bottleneck of current sweeping preconditioners is a lack of parallel scalability for a single right-hand side. The difficulty arises because the Helmholtz problem is inherently sequential, independent of domain decomposition strategy. This sequential nature is dictated by the hyperbolic nature of wave equations and is manifested in the need to accurately resolve the long-range interactions. The key to accurately resolving these interactions is a consistent information transfer between subdomains, which has precluded more general domain decompositions (i.e., beyond layered subdomains) and consequently inhibited parallelization.

In this work, we address these issues by departing from standard layered domain decompositions and introducing a checkerboard domain decomposition (CDD) along with a new sweeping strategy that is only viable on such domain decompositions. Our novel sweeping preconditioner consistently and efficiently tracks the wavefield across subdomains. The preconditioner can be interpreted as an approximate LU factorization, where parallelism arises because the diagonal blocks themselves have a block-diagonal structure. For a single right-hand side, the resulting algorithm has a $O(N/p)$ run-time, up to logarithmic factors, where, independent of geometric dimension, $p = O(n)$ is the number of processors. The new algorithm therefore provides the first parallel scalable solver for the Helmholtz equation at high-frequency for a single right-hand side.

Our approach can be applied to two- and three-dimensional problems. In two dimensions, the algorithm exhibits good weak parallel scalability. That is, as the frequency increases, and consequently the problem is refined, we refine the CDD so that the local problems in each subdomain have constant size. In three dimensions, we apply the same strategy as for the two-dimensional case in two of the spatial dimensions and extend the subdomains along the third spatial dimension resulting in beam-shaped quasi-one-dimensional local problems. In either case, we employ off-the-shelf direct solvers to solve the local problems and obtain parallel complexities of $O(n \log \omega)$ and $O(n^2 \log \omega)$ in two and three dimensions, respectively. The 3D complexity can be reduced to $O(n \log \omega)$ by employing existing parallel direct solvers [64, 50] for the quasi-one-dimensional problems in each subdomain, but we do not exploit these tools here.

The new algorithm therefore results in an $O(n \log \omega)$ complexity for a single right-hand side, regardless of the geometric dimension. This new algorithm requires sweeps across the CDD in both cardinal and diagonal directions, and we can exploit parallelism in all directions orthogonal to the current sweep direction. While this new strategy increases parallelization, the sweeps are still inherently sequential and cannot be parallelized. Thus, we do not anticipate further reduction in complexity, for a single right-hand, due exclusively to modification of the domain decomposition. In the presence of $O(n \log \omega)$ right-hand-sides, it has been shown, however, that the sequential nature of the sweeps can be mitigated by pipelining multiple right-hand sides [89]. Using pipelining, solutions for all $O(n)$ right-hand-sides can be computed with $O(n \log \omega)$ parallel complexity, i.e., the average parallel complexity per right-hand-side is $O(\log \omega)$.

1.1. Related Work

Our method is inspired by the method of polarized traces, which is well-established in the literature and, in contrast to other proposed preconditioners, has been proven applicable for problems with high-order discretizations [87, 73] and highly heterogeneous (and even discontinuous) wave speed distributions [86].

Standard linear algebra techniques such as nested dissection [40] and multi-frontal solvers [24, 83], coupled with $\mathcal{H}$-matrices [9], have been applied to the Helmholtz problem [42, 19, 82, 1]. While these methods take advantage of compressed linear algebra to gain more efficiency (e.g., [6]), in the high-frequency regime they still suffer from the same sub-optimal asymptotic complexity as standard multi-frontal methods (e.g., [22, 2, 18]).

Multigrid methods (e.g., [48, 13, 28, 67, 58, 28, 3]), once thought to be inefficient for the Helmholtz problem, have been successfully applied [14, 45, 71]. Approaches stemming from the complex-shifted Laplacian [28] can be advantageous if properly tuned. However, in general, they either require an expensive solver for the shifted problem or require a large number of iterations to reach convergence, depending on the scaling between the complex shift and the frequency [39]. Although these algorithms do not result in a lower computational complexity, they are highly parallelizable, resulting in low run-times.

Within the geophysical community, the analytic incomplete LU (AILU) method was explored in [60, 59]. A variant of Kaczmarz preconditioners [43] has been studied and applied to time-harmonic wave equations by [51]. Another class of methods, called hybrid direct-iterative methods, have been explored by [68]. Although these solvers have, in general, relatively low memory consumption they tend to require many iterations to converge, thus hindering practical run-times.

Domain decomposition methods for solving partial differential equations (PDEs) have a long history [66, 52]. The first application of domain decomposition to the Helmholtz problem was proposed by [23], which inspired the development of various domain decomposition algorithms, which are now classified as Schwarz algorithms[1]. However, the convergence rate of such algorithms is strongly dependent on the boundary conditions prescribed at the interfaces between subdomains [35]. The subsequent introduction of the optimized Schwarz framework in [33], which uses optimized boundary conditions to obtain good convergence, has inspired several competing approaches, including, but not limited to [34, 11, 37, 38, 36].

Absorbing boundary conditions for domain decomposition schemes for elliptic problems are introduced by [27] and the first application of such techniques to the Helmholtz problem traces back to the AILU factorization [31]. The sweeping preconditioner, introduced in [25, 26], was the first algorithm to show that those ideas could yield algorithms with quasi-linear complexity, leading to several related algorithms with similar claims such as the source transfer preconditioner [16], the rapidly converging domain decomposition [70] and its extensions [72], the double sweep preconditioner [80], and the method of polarized traces [85]. For an extensive review on sweeping-type methods we direct the reader to [32].

To our knowledge, all domain decomposition methods and sweeping preconditioners have been based on layered domain decompositions, with one exception. In [49], a $q \times q$ CDD is considered in the context of the source transfer method. Their proposed strategy requires $p = q^2$ processors to obtain the sub-optimal parallel complexity of $O((N/\sqrt{p}) \log N)$, in both 2D and 3D. The primary difference between this method and our method is the sweeping strategy. The strategy employed in [49] is inspired by classical domain decomposition methods, while our method is inspired by sweeping preconditioners. Consequently, our method inherits a superior sweeping strategy.

1.2. Model Problem and Discretization

We consider problems formulated within the framework of (1), where we choose to model absorbing boundary conditions using perfectly matched layers (PMLs) [7, 46]. To preserve the solution in all of $\Omega_{\texttt{bulk}}$, using a PML requires the domain to be extended, which in turn also implies that the solution, material property, and source spaces must also be extended. The extended domain, $\Omega_{\texttt{extended}}$, contains all of $\Omega_{\texttt{bulk}}$, as illustrated in Figure 1a and a new boundary value problem is formulated on $\Omega_{\texttt{extended}}$,

$$\begin{aligned} -\operatorname{div}(\Lambda \nabla u) - \omega^2 m u &= f \quad \text{in } \Omega_{\texttt{extended}}, \\ u &= 0 \quad \text{on } \partial\Omega_{\texttt{extended}}, \end{aligned} \tag{2}$$

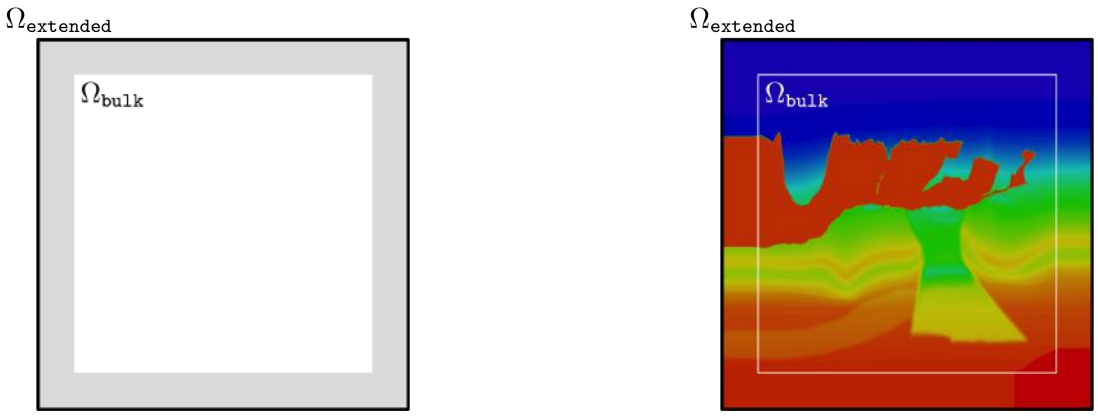
where, for brevity, we re-use the symbols u, m, and f to represent the extended solution wavefield, slowness, and source distribution. As is customary for PMLs in domains with varying wave speed, we assume that the source density, f, is extended by zero into $\Omega_{\texttt{extended}}$ and the squared slowness, m, is extended into $\Omega_{\texttt{extended}}$ along the normal direction of $\partial\Omega_{\texttt{bulk}}$ in a constant fashion, as illustrated in Figure 1b. The PML-extended Helmholtz equation (2) is reduced to (1) in $\Omega_{\texttt{bulk}}$ by imposing that Λ is the identity matrix in $\Omega_{\texttt{bulk}}$. In the PML region, $\Omega_{\texttt{extended}} \backslash \Omega_{\texttt{bulk}}$, Λ is a complex valued diagonal matrix which depends on the PML formulation, and m and f are complex-valued functions, obtained from imposing the PML. Details of the precise formulation used in our developments are provided in Appendix A. Equation (2) is effectively a complex-valued boundary-value problem in $\Omega_{\texttt{extended}}$ with homogeneous Dirichlet boundary conditions. In the remainder of the discussion, we consider (2) to be the canonical problem and therefore, for simplicity of notation, denote $\Omega_{\texttt{extended}}$ as Ω.

When (2) is discretized, we obtain the linear algebraic system

$$\mathbf{A}\mathbf{u} = \mathbf{f} \tag{3}$$

where $\mathbf{A}$ is the model-dependent system matrix, $\mathbf{u}$ is the solution vector, and $\mathbf{f}$ is the vector of the source density f. In this paper we restrict our discussion of absorbing boundary conditions to PMLs and discretization to finite-difference methods. These restrictions are merely to simplify the exposition: other transparent boundary conditions, such as absorbing layers or sponge layers, and other discretizations, such as higher-order finite differences and those derived

[1] For a review on classical Schwarz methods see [15, 78]; and for other applications of domain decomposition methods for the Helmholtz equations, see [20, 41, 55, 17, 54, 10, 4].



(a) (b)

Figure 1: (a) $\Omega_{\texttt{bulk}}$ (white) and $\Omega_{\texttt{extended}}$ (white and gray) and (b) part of the BP 2004 [8] wave speed ($\Omega_{\texttt{bulk}}$) extended to $\Omega_{\texttt{extended}}$ via the normal extension.

from finite element methods, may be used in this framework. For 2D problems we use standard 5-point finite difference stencils and in 3D, we use 9-point stencils. As a result, the discretization has N global degrees-of-freedom and $n = N^{\frac{1}{2}}$ and $n = N^{\frac{1}{3}}$ degrees-of-freedom in each spatial dimension for two- and three-dimensional problems respectively.

For the PMLs we use a cubic PML profile function. As we increase the frequency of the problem, we do not increase the width of the PML. Instead, we choose the PML width so that the number of wavelengths is constant in the PML region, and increase the absorption constant logarithmically with the frequency. This is motivated by an analysis of the PML [12], where this choice is rigorously justified for a more complex PML-profile. In our work, we employ the same strategy for the cubic PML-profile and obtain satisfactory results. Details on the construction of the linear system and the PMLs are given in Appendix A.

1.3. Continuous Polarization

The method of polarized traces was introduced as a solver [85] and then as a preconditioner [88] for the linear system (3). At its core, the method of polarized traces spatially subdivides the discrete degrees-of-freedom in Ω into layers and computes an approximate solution to the global wavefield by sweeping over the layers and solving a local discrete half-space problem in each layer. Following [85], the solutions of the half-space problems are called polarized wavefields. In this work, we make extensive use of this concept and therefore provide a brief review, in the continuous setting, in this section. In Section 1.4, we present a similar treatment in the discrete context.

Consider the boundary-value problem (2) with a source density function f supported only in $\Omega_{\texttt{bulk}}$. Let Γ be an interface dividing Ω into two regions, Ω_1 and Ω_2, such that the support of f lies entirely within Ω_1. For example, Γ could be a straight line (Figure 2a) or an L-shaped line (Figure 2b).

The wavefield in Ω_2 can be computed from the representation formula

$$u(x) = \int_\Gamma \mu(y) G(x,y) ds_y - \int_\Gamma \lambda(y) \left[n(y) \cdot \left(\Lambda(y) \nabla_y G(x,y) \right) \right] ds_y \quad x \in \Omega_2, \tag{4}$$

where

$$\lambda = u|_\Gamma, \quad \text{and} \quad \mu = [n \cdot (\Lambda \nabla u)]|_\Gamma$$

are the Dirichlet and Neumann traces on Γ, and $G(x,y)$ is the Green's function corresponding to the problem (2), i.e., for $x \in \Omega$

$$\begin{aligned} -\operatorname{div}_y \left(\Lambda(y) \nabla_y G(y,x) \right) - \omega^2 m(y) G(y,x) &= \delta(x-y) \qquad y \in \Omega, \\ G(y,x) &= 0 \qquad\qquad y \in \partial\Omega. \end{aligned}$$

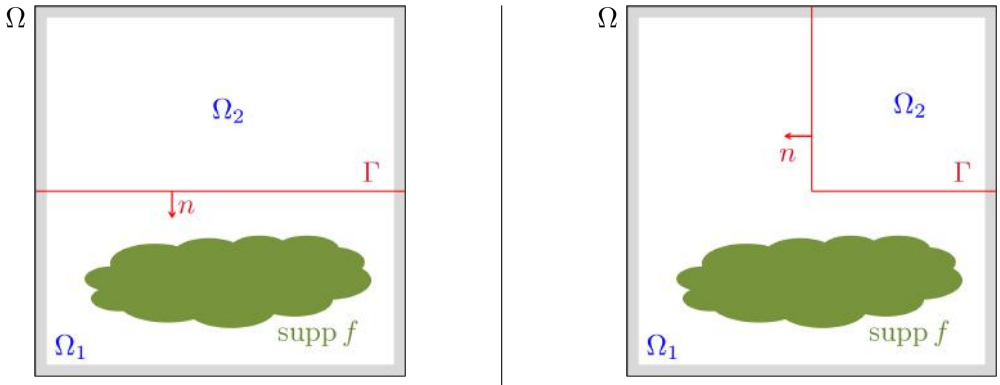


Figure 2: A schematic representation of the truncated (a) half- and (b) quarter-space problem.

This formula directly follows from the divergence theorem and properties of the Green's function, assuming that Λ is a diagonal matrix.

Equation (4) requires knowledge of the Dirichlet and Neumann traces of the solution almost everywhere on Γ. Thus, corners in Γ are admissible, as in the quarter-space problem, even though the normal is not uniquely defined. Using (4), the solution u can then be computed on Ω_2. Extending equation (4) to all of Ω,

$$U(x) = \int_\Gamma \mu(y) G(x,y) ds_y - \int_\Gamma \lambda(y) \left[n(y) \cdot \left(\Lambda(y) \nabla_y G(x,y) \right) \right] ds_y \quad x \in \Omega, \tag{5}$$

it can be shown that U vanishes on Ω_1,

$$U(x) = 0 \quad \text{for } x \in \Omega_1. \tag{6}$$

Following the terminology of [85], we call (6) the annihilation condition, Γ the polarization interface, and U a polarized wavefield. A proof of the annihilation condition (6) is provided in Appendix B.

1.4. Discrete Polarization

A discrete counterpart of the polarized wavefield U can be derived by constructing a discrete solution that satisfies the discrete analogue to the annihilation condition in (6). Consider the discretization points corresponding to degrees-of-freedom in (3), as well as an interface Γ which does not intersect with any discretization point, as illustrated in Figure 3 for both the half-space and quarter-space subdomains. Given a discretization-dependent distance δ (e.g., $\delta = 1$ for a classical 5-point finite difference stencil in 2D), such an interface divides the degrees-of-freedom into four sets, as labeled in Figure 3:

1. $\Gamma_\mathbf{1}$, the set of all degrees-of-freedom physically contained in Ω_1 and δ-adjacent to Γ;
2. $\Omega_\mathbf{1}$, the set of all degrees-of-freedom physically contained in Ω_1, excluding $\Gamma_\mathbf{1}$;
3. $\Gamma_\mathbf{2}$, the set of all degrees-of-freedom physically contained in Ω_2 and δ-adjacent to Γ;
4. $\Omega_\mathbf{2}$, the set of all degrees-of-freedom physically contained in Ω_2, excluding $\Gamma_\mathbf{2}$.

Upon reordering the degrees-of-freedom with respect to these four sets, the discrete system (3) can be rewritten as

$$\begin{pmatrix} \mathbf{A}_{\Omega_1\Omega_1} & \mathbf{A}_{\Omega_1\Gamma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{\Gamma_1\Omega_1} & \mathbf{A}_{\Gamma_1\Gamma_1} & \mathbf{A}_{\Gamma_1\Gamma_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\Gamma_2\Gamma_1} & \mathbf{A}_{\Gamma_2\Gamma_2} & \mathbf{A}_{\Gamma_2\Omega_2} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{\Omega_2\Gamma_2} & \mathbf{A}_{\Omega_2\Omega_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_{\Omega_1} \\ \mathbf{u}_{\Gamma_1} \\ \mathbf{u}_{\Gamma_2} \\ \mathbf{u}_{\Omega_2} \end{pmatrix} = \begin{pmatrix} \mathbf{f}_{\Omega_1} \\ \mathbf{f}_{\Gamma_1} \\ \mathbf{f}_{\Gamma_2} \\ \mathbf{f}_{\Omega_2} \end{pmatrix}, \tag{7}$$

where $\mathbf{A}_{\Gamma_1\Omega_1}$ denotes the slice of the matrix $\mathbf{A}$ generated from the rows corresponding to $\Gamma_\mathbf{1}$ and the columns corresponding to $\Omega_\mathbf{1}$. The remaining submatrices are similarly generated. In the same way, the vector $\mathbf{u}_{\Omega_2}$ (and

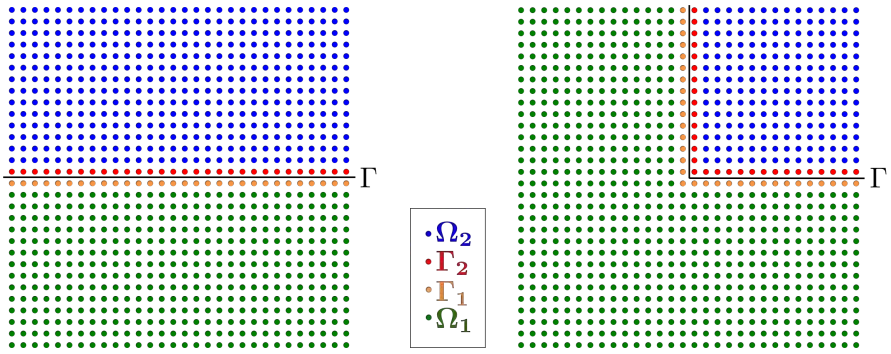


Figure 3: A schematic representation of four sets of degrees-of-freedom Ω_1, Ω_2, Γ_1 and Γ_2.

similarly all other vectors) denotes the slice of the vector $\mathbf{u}$ with respect to Ω_2. As in the continuous case, we consider the case without sources in Ω_2, i.e., $\mathbf{f}_{\Omega_2} = \mathbf{0}$ and $\mathbf{f}_{\Gamma_2} = \mathbf{0}$. Then, due to the invertibility of $\mathbf{A}$, it is easy to see that the solution $\mathbf{U}$ of the linear system

$$\begin{pmatrix} \mathbf{A}_{\Omega_1\Omega_1} & \mathbf{A}_{\Omega_1\Gamma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{\Gamma_1\Omega_1} & \mathbf{A}_{\Gamma_1\Gamma_1} & \mathbf{A}_{\Gamma_1\Gamma_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\Gamma_2\Gamma_1} & \mathbf{A}_{\Gamma_2\Gamma_2} & \mathbf{A}_{\Gamma_2\Omega_2} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{\Omega_2\Gamma_2} & \mathbf{A}_{\Omega_2\Omega_2} \end{pmatrix} \mathbf{U} = \begin{pmatrix} \mathbf{0} \\ \mathbf{A}_{\Gamma_1\Gamma_2}\mathbf{u}_{\Gamma_2} \\ -\mathbf{A}_{\Gamma_2\Gamma_1}\mathbf{u}_{\Gamma_1} \\ \mathbf{0} \end{pmatrix} \tag{8}$$

satisfies

$$\mathbf{U} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{u}_{\Gamma_2} \\ \mathbf{u}_{\Omega_2} \end{pmatrix}.$$

Consequently, the discrete counterpart of the polarized wavefield $\mathbf{U}$ is

$$\mathbf{U} = \begin{pmatrix} \mathbf{A}_{\Omega_1\Omega_1} & \mathbf{A}_{\Omega_1\Gamma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{\Gamma_1\Omega_1} & \mathbf{A}_{\Gamma_1\Gamma_1} & \mathbf{A}_{\Gamma_1\Gamma_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{\Gamma_2\Gamma_1} & \mathbf{A}_{\Gamma_2\Gamma_2} & \mathbf{A}_{\Gamma_2\Omega_2} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{\Omega_2\Gamma_2} & \mathbf{A}_{\Omega_2\Omega_2} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{0} \\ \mathbf{A}_{\Gamma_1\Gamma_2}\mathbf{u}_{\Gamma_2} \\ -\mathbf{A}_{\Gamma_2\Gamma_1}\mathbf{u}_{\Gamma_1} \\ \mathbf{0} \end{pmatrix}. \tag{9}$$

From the right-hand side of (8), one can easily see that knowledge of both $\mathbf{u}_{\Gamma_1}$ and $\mathbf{u}_{\Gamma_2}$ is required in order to compute $\mathbf{U}$. By construction, these sets contain information about the discrete wavefield and its normal derivative in the vicinity of Γ. Thus, as with the continuous case, the discrete case also requires information about the Dirichlet and Neumann traces to compute the polarized wavefield $\mathbf{U}$ in Ω_2. In fact, [85] demonstrates, using similar techniques, that a discrete counterpart of the representation formula (4) can be derived.

Finally, our technique is easily extended to more general discretization techniques, as long as they allow for a reordering of the degrees-of-freedom to the block-tridiagonal system (7). Depending on the discretization, change in the selection of the sets Γ_1 and Γ_2 may be required. Similar schemes have been applied for many different discretizations such as high-order finite difference methods [89], finite element methods [84], enriched finite element methods [30], discontinuous Galerkin methods [74], and integral representations [87]. For example, for higher-order finite difference methods the stencils centered at the points in Ω_2 (respecting Γ_2, Γ_1, or Ω_1) cannot involve discretization points in Γ_1 (respecting Ω_1, Ω_2, Γ_2). This is easily enforced by defining an appropriate δ, e.g., $\delta = 2$ for a 9-point, 5x5, stencil in 2D. Similarly, in finite element or discontinuous Galerkin methods, the sparsity of the system matrices can be exploited in order to obtain suitable sets of degrees-of-freedom.

1.5. Organization

In Section 2, we introduce the algorithm to compute an approximate global solution u of the boundary value problem (2) with constant squared slowness m. We introduce the algorithm first on the continuous level in Section 2.1

and then extend it to the discrete level in Section 2.2. In Section 3, we show how the algorithm can be used to precondition the linear system (3). This opens the possibility of using the algorithm as part of an optimally parallel scaling solver based on a preconditioned GMRES method. We conclude Section 3 with a complexity analysis of this solver with regards to computational and communication effort. In Section 4, we analyze the effects of heterogeneous wave speeds on the effectiveness of the preconditioner. In Section 5, we provide several numerical examples in two- and three-dimensions for constant and non-constant wave speeds to corroborate all claims. The paper is concluded by a discussion where we briefly summarize our results and discuss possible extensions. Additional details and pseudo-code for the proposed algorithms are provided in the Appendices.

2. L-sweeps: Reconstruction of wavefields

In this section, we introduce the algorithm to compute the global solution u of the boundary value problem (2). This solution is obtained from local solutions of local problems defined over a CDD. The concept of polarization introduced in Section 1 plays a crucial role in this procedure. We introduce the algorithm for constant wave speeds at the continuous and the discrete level in Sections 2.1 and 2.2. The procedure can be applied to problems with non-constant wave speeds in an analogous fashion. The effects of these heterogeneous wave speeds, in particular discontinuous ones, are addressed in Section 4.

2.1. Continuous formulation

Consider a decomposition of Ω into a CDD[2], with q rows and r columns, of non-overlapping open subdomains. For example, see Figure 4, where $q = r = 5$. We first define the local problems associated with each subdomain, Ω_{ij}. To this end, we define extended domains $\Omega^{\varepsilon}_{ij,\texttt{bulk}}$ and $\Omega^{\varepsilon}_{ij,\texttt{extended}}$. The domain $\Omega^{\varepsilon}_{ij,\texttt{bulk}}$ is obtained from extending Ω_{ij} by a ε-layer along interior edges of the CDD, and $\Omega^{\varepsilon}_{ij,\texttt{extended}}$ is an extension of $\Omega^{\varepsilon}_{ij,\texttt{bulk}}$ to impose absorbing boundary conditions via PMLs. In $\Omega^{\varepsilon}_{ij,\texttt{extended}}$ we define the local squared slowness $m_{ij} := m|_{\Omega^{\varepsilon}_{ij,\texttt{extended}}}$. For a given source density f_{ij} in $\Omega^{\varepsilon}_{ij,\texttt{extended}}$, we define the local problem

$$\begin{aligned} -\operatorname{div}\left(\Lambda_{ij}\nabla u_{ij}\right) - \omega^2 m_{ij} u_{ij} &= f_{ij} \quad \text{in } \Omega^{\varepsilon}_{ij,\texttt{extended}}, \\ u_{ij} &= 0 \quad \text{on } \partial\Omega^{\varepsilon}_{ij,\texttt{extended}}, \end{aligned}$$

where u_{ij} denotes the local solution. Due to the local PML, Λ_{ij} is a complex-valued diagonal matrix, and m_{ij} and f_{ij} are complex-valued functions in the PML region $\Omega^{\varepsilon}_{ij,\texttt{extended}} \backslash \Omega^{\varepsilon}_{ij,\texttt{bulk}}$. Further details of the PML formulation are provided in Appendix A. For the sake of brevity we denote the PML-adjusted squared slowness and source density still as m_{ij} and f_{ij}, respectively. Note that in contrast to (2), we do not set the squared slowness to be constant using a normal extension on $\partial\Omega^{\varepsilon}_{ij,\texttt{bulk}}$ in the PML region, rather we use the squared slowness inherited from the global problem in these regions. This definition makes our approach more accurate, since we use the local problem to compute sections of the global problem.

Note that the construction of the local problems consists of two subsequent extensions of Ω_{ij}: we first add an additional ε-layer around Ω_{ij}, and then further extend the subdomain by a PML region. While the latter is clearly needed to avoid artificial reflections in the local solutions, the former appears to be ad-hoc at this stage, but for reasons that will be explained in the sequel, the first extension by an ε-layer is crucial for a consistent exchange of information between subdomains. The resulting local problems are Dirichlet boundary-value problems defined on the extension $\Omega^{\varepsilon}_{ij,\texttt{extended}}$ of Ω_{ij}. As before, we denote the domain $\Omega^{\varepsilon}_{ij,\texttt{extended}}$ as $\Omega^{\varepsilon}_{ij}$. Examples of these local problems are illustrated in Figure 5.

In what follows, we introduce our method for computing global solutions to (2) by considering four scenarios, illustrated in Figure 4, each increasingly more general:

1. the source density is supported in the interior of a corner subdomain of the CDD,
2. the source density is supported in the interior of an arbitrary subdomain,

[2]Note that the CDD is chosen so that neglecting the PML regions (shown in gray in Figure 4) each subdomain has the same size.

3. the source density is supported in the interior of an arbitrary number of subdomains such that its support does not intersect the skeleton of the CDD, and
4. the source density has arbitrary support in $\Omega_{\texttt{bulk}}$.

In the subsequent developments, we consider these scenarios using source distributions constructed from unions of point sources. However, the developments do not depend on an assumption that the source distributions are point-sources – any source density is admitted as long as it satisfies the conditions on its support. In particular, scenario 4 allows for arbitrary source distributions, including those which intersect the CDD skeleton, making our approach widely applicable.

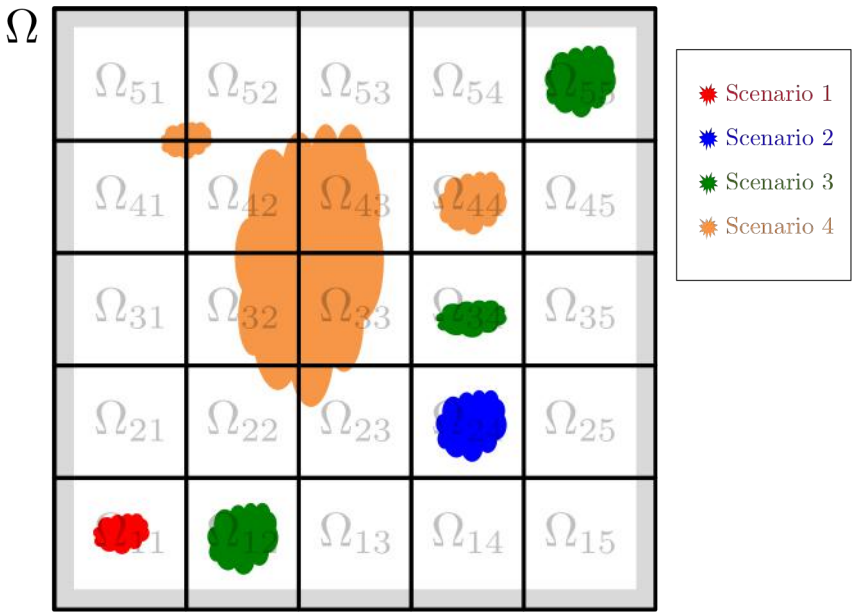


Figure 4: The domain decomposition, where the colored regions show examples of source distributions for each of the four scenarios.

2.1.1. Scenario 1: A source density supported in a corner subdomain

Without loss of generality, we introduce the algorithm for a source density supported in Ω_{11}. Source densities supported in any other corner subdomain can be constructed in an analogous way. The computation of the global solution is performed in three stages:

1. compute an approximation of the global solution in Ω_{11} using the local problem associated with Ω_{11},
2. compute the approximate global solution in the first row and column of the CDD, and,
3. compute the global solution in the rest of the subdomains.

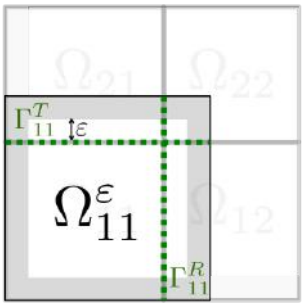
Ultimately, using these three stages, we are able to compute the global solution, up to PML induced errors, using three sweeps: one vertical, one horizontal, and one diagonal.

Stage 1: Local solution

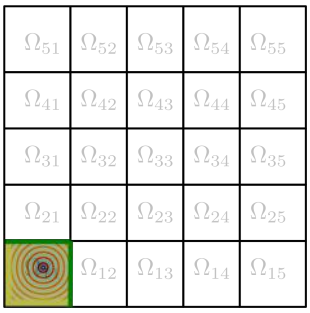
The local solution u_{11} defined over $\Omega_{11}^{\varepsilon}$ can be computed by solving the local problem associated with Ω_{11} with the source density $f_{11} := f|_{\Omega_{11}^{\varepsilon}}$. Since f is supported in Ω_{11} and the wave speed is constant, the global solution u restricted to Ω_{11} coincides with u_{11} up to errors induced by the PMLs. We therefore simply set the global solution u to u_{11} in Ω_{11}: $u|_{\Omega_{11}} := u_{11}|_{\Omega_{11}}$.

We define the straight lines containing the top and right boundary of Ω_{11} to be Γ_{11}^{T} and Γ_{11}^{R}, as depicted in Figure 5a. On Γ_{11}^{T} and Γ_{11}^{R}, we extract the Dirichlet and Neumann traces of u_{11}:

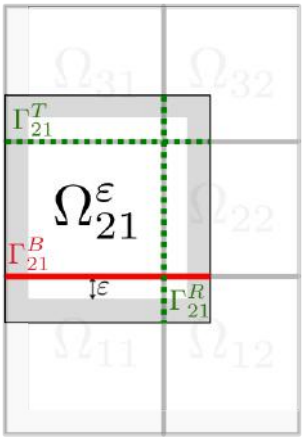
$$\begin{aligned} \lambda_{11}^{T} &:= u_{11}|_{\Gamma_{11}^{T}}, \quad \mu_{11}^{T} := [n \cdot (\Lambda_{11} \nabla u_{11})]\,|_{\Gamma_{11}^{T}}, \\ \lambda_{11}^{R} &:= u_{11}|_{\Gamma_{11}^{R}}, \quad \mu_{11}^{R} := [n \cdot (\Lambda_{11} \nabla u_{11})]\,|_{\Gamma_{11}^{R}}. \end{aligned} \tag{10}$$



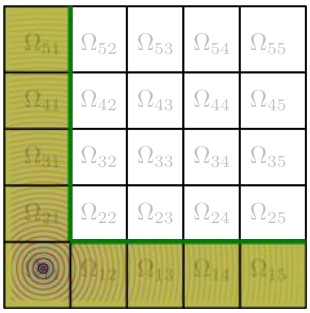
(a) Example of a local problem in stage 1.



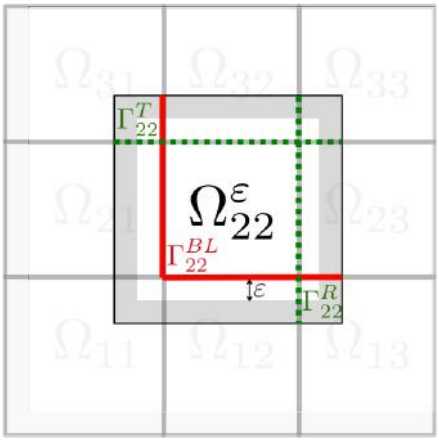
(b) Solution after stage 1.



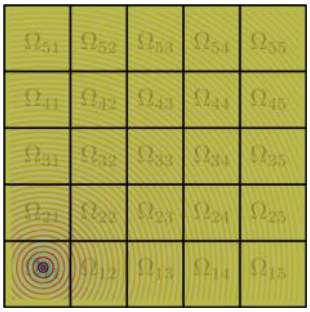
(c) Example of a local problem in stage 2.



(d) Solution after stage 2.



(e) Example of a local problem in stage 3.



(f) Solution after stage 3.

Figure 5: Illustration of local problems and the computed solution after each stage of a sweep. The green lines in the wavefields depict extracted trace information used in the next stage.

These traces provide the necessary information to compute good approximations of the global solution in the neighboring subdomains Ω_{21} and Ω_{12} in stage 2. The solution computed after stage 1 is shown in Figure 5b, where the trace information that is used in stage 2 is shown in green.

Stage 2: Global solution in the same row/column
By construction, Γ_{11}^T lies entirely inside Ω_{21}^ε and the source density f is zero on Γ_{11}^T. Thus, the traces λ_{11}^T and μ_{11}^T can be used to compute a polarized wavefield in Ω_{21}^ε using $\Gamma_{21}^B := \Gamma_{11}^T$ as the polarization interface, and

$$u_{21}(x) = \int_{\Gamma_{21}^B} \mu_{11}^T(y) G_{21}(x,y) ds_y - \int_{\Gamma_{21}^B} \lambda_{11}^T(y) \left[n(y) \cdot \left(\Lambda_{21} \nabla_y G_{21}(x,y) \right) \right] ds_y. \tag{11}$$

Here, $G_{21}(x,y)$ is the Green's function corresponding to the local problem defined on Ω_{21}^ε.

Following the same reasoning used for u_{11} in stage 1, the global solution restricted to Ω_{21} coincides with u_{21}, up to errors induced by the PMLs, and we set $u|_{\Omega_{21}} := u_{21}|_{\Omega_{21}}$. We also extract the Dirichlet and Neumann traces λ_{21}^T and μ_{21}^T of u_{21} on Γ_{21}^T in the same way as in (10) and repeat the process to compute an approximation of the global solution in the entire first column of the CDD. In addition, for each of these polarized wavefields, we extract the Dirichlet and Neumann traces λ_{i1}^R and μ_{i1}^R on Γ_{i1}^R. These traces are needed in stage 3 to compute approximations of the global solution in the rest of the subdomains. Similarly, we compute the global solution in the first row of the CDD and extract the traces λ_{1j}^T and μ_{1j}^T on Γ_{1j}^T for further use in stage 3. The solution after stage 2 and the trace information extracted for stage 3 are shown in Figure 5d.

For the extension of the solution into the neighboring subdomains to be accurate, the local problems associated with Ω_{i1} and $\Omega_{(i+1)1}$ need to coincide in an ε-tube around $\Gamma_{i1}^T = \Gamma_{(i+1)1}^B$. This is ensured by the extension of the local problem by the additional ε-layer.

Stage 3: Global solution in the remaining subdomains
Consider Ω_{22}, where, by construction, $\Gamma_{22}^B := \Gamma_{12}^T$ and $\Gamma_{22}^L := \Gamma_{21}^R$ lie entirely inside Ω_{22}^ε. We combine these lines to form an L-shaped line, Γ_{22}^{BL}, so that Ω_{22} is entirely contained in the quadrant defined by Γ_{22}^{BL} as shown in Figure 5e. In addition, we combine the trace information on Γ_{21}^R and Γ_{12}^T to define Dirichlet and Neumann traces on Γ_{22}^{BL} as

$$\lambda_{22}^{BL}(x) := \begin{cases} \lambda_{21}^R(x) & x \in \Gamma_{22}^{BL} \cap \Gamma_{22}^L, \\ \lambda_{12}^T(x) & x \in \Gamma_{22}^{BL} \cap \Gamma_{22}^B, \end{cases} \tag{12}$$

$$\mu_{22}^{BL}(x) := \begin{cases} \mu_{21}^R(x) & x \in \Gamma_{22}^{BL} \cap \Gamma_{22}^L, \\ \mu_{12}^T(x) & x \in \Gamma_{22}^{BL} \cap \Gamma_{22}^B. \end{cases} \tag{13}$$

The traces can be used to compute a polarized wavefield given by

$$u_{22}(x) = \int_{\Gamma_{22}^{BL}} \mu_{22}^{BL}(y) G_{22}(x,y) ds_y - \int_{\Gamma_{22}^{BL}} \lambda_{22}^{BL}(y) \left[n \cdot \left(\Lambda \nabla_y G_{22}(x,y) \right) \right] ds_y, \tag{14}$$

where $G_{22}(x,y)$ is the Green's function corresponding to the local problem defined on Ω_{22}^ε. In particular, (14) holds due to the fact that the local problems defined on Ω_{12}^ε and Ω_{21}^ε coincide with the local problem in Ω_{22}^ε in the vicinity of Γ_{22}^B and Γ_{22}^L, respectively.

It is clear that the polarized wavefield u_{22} coincides with the global wavefield u in Ω_{22} up to errors induced by the PMLs, and we simply set $u|_{\Omega_{22}} := u_{22}|_{\Omega_{22}}$. Similarly to (10), we extract the Dirichlet and Neumann traces of u_{22} on Γ_{22}^T and Γ_{22}^R and we use them to compute the approximations of the global wavefield in the other subdomains. Following this pattern, the wavefield is computed in the remaining subdomains by sweeping diagonally from the bottom-left corner to the top-right corner. At each step of the sweep, a diagonal perpendicular to the direction of the sweep is updated. This diagonal consists of subdomains only touching each other at a corner. For one diagonal, this procedure is summarized in Figure 6.

2.1.2. Scenario 2: A source density supported in an arbitrary subdomain

In this scenario, we consider a point source supported in an arbitrary subdomain. Without loss of generality, for illustrative purposes we select the subdomain Ω_{24}. Source densities supported in any other subdomain can be

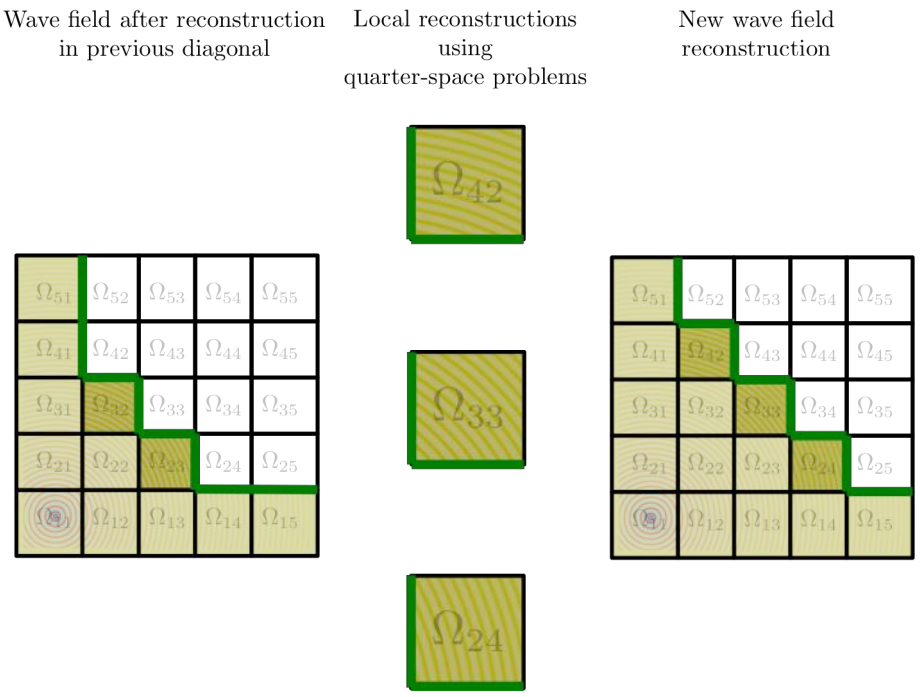


Figure 6: Summary of the computed solution in one diagonal sweeping step. Updated subdomains are diagonally adjacent in a direction perpendicular to the bottom-left to top-right sweep direction.

treated analogously. First we restrict the problem to the quadrant in the top right corner of the domain, inclusive of Ω_{24}, which we denote Ω_{TR} and illustrate in Figure 7, by truncating the global problem with PMLs. The interior boundary of Ω_{TR} is extended by a layer of thickness ε so that the local problems in Ω_{TR} are precisely the same as in Section 2.1.1. Similarly, we define domains Ω_{TL}, Ω_{BL}, and Ω_{BR} in the top-left, bottom-left, and bottom-right corner. These definitions reduce the problem posed on Ω to four sub-problems, each of which have a point source supported in a corner subdomain. Thus, we can readily apply the procedure introduced in Section 2.1.1 to each of them. The computed solutions are shown in Figure 8.

The global solution may be obtained from summing these four solutions, taking care to avoid counting the contributions from the overlapped region multiple times. Executing this procedure as described requires many redundant computations. We achieve an equivalent method by generalizing the algorithm presented in Section 2.1.1, which we describe in three stages:

1. compute the local solution in the subdomain where the source density is supported and extract the traces on all interior boundaries of the domain,
2. extend the solution into the subdomains in the same row/column following stage 2 of Section 2.1.1, and extract the traces required for stage 3, and,
3. extend the solution into the rest of the subdomains by computing the local solutions in diagonal rows, perpendicular to the direction of the sweep, similarly to stage 3 of Section 2.1.1.

Stage 3 is illustrated in Figure 9. In this scenario, the full algorithm requires a total of eight sweeps. Stage 2 requires four sweeps over the domain: up, down, left, right. Stage 3 also requires four sweeps: bottom-left to top-right, top-right to bottom-left, bottom-right to top-left, top-left to bottom-right.

2.1.3. Scenario 3: Arbitrary source distributions not intersecting the CDD skeleton

Global solutions for arbitrary source distributions that do not intersect the skeleton of the CDD can be computed from a union of distinct source densities, each supported in a single subdomain. Therefore, the global solution can be naïvely computed by applying the procedure from scenario 2 to each of the localized source densities and summing the results. This approach is not computationally efficient because much of the work is redundant.

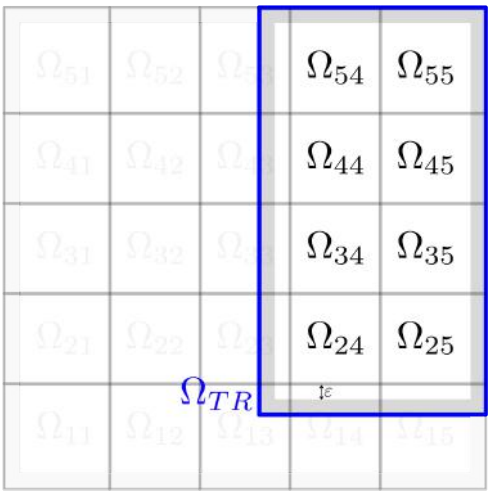


Figure 7: The definition of the subproblem by truncation with absorbing boundary conditions for scenario 2.

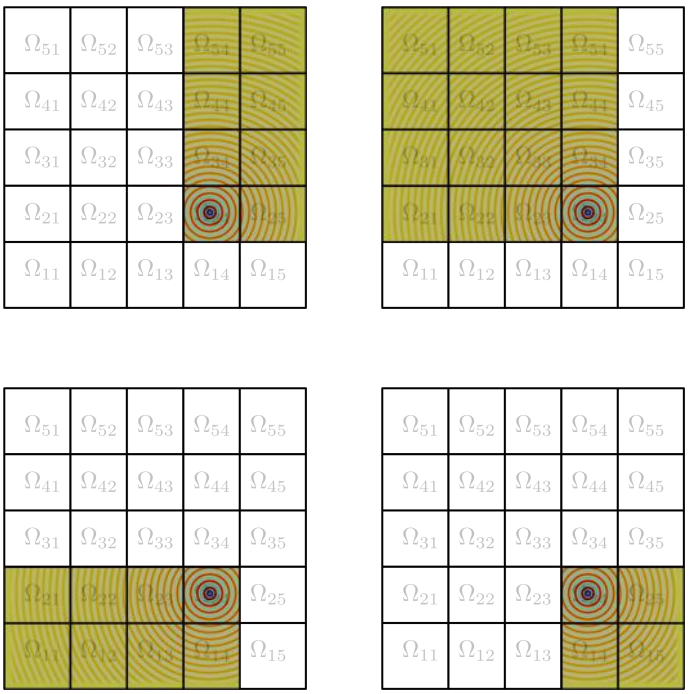
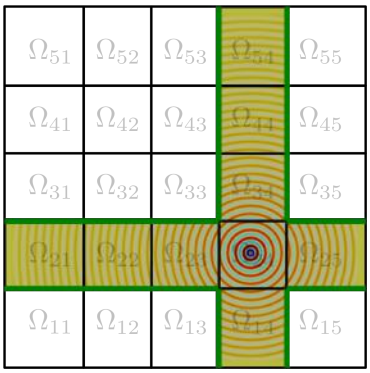
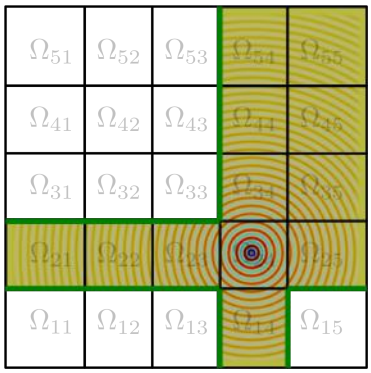


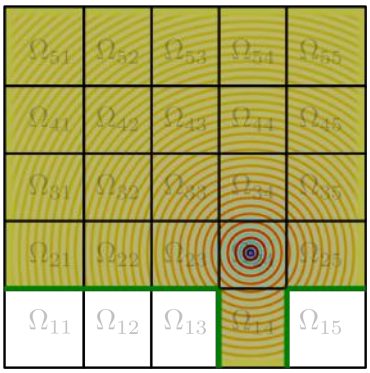
Figure 8: The solutions in each quadrant in scenario 2.



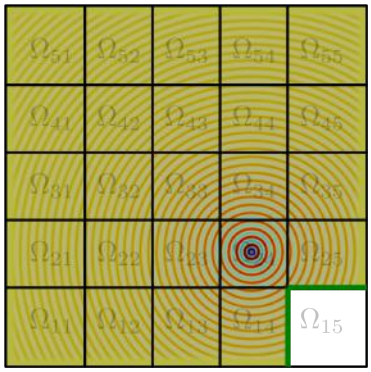
(a) Before stage 3.



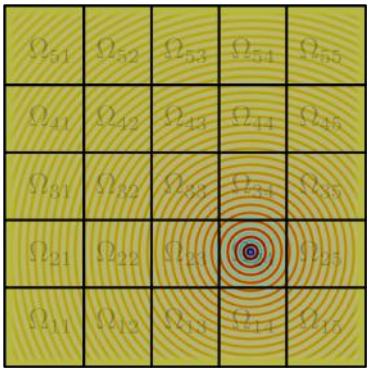
(b) After bottom-left to top-right sweep.



(c) After bottom-right to top-left sweep.



(d) After top-right to bottom-left sweep.



(e) After top-left to bottom-right sweep.

Figure 9: Summary of stage 3 for scenario 2. The plots show the computed wave field after stage 2, and the computed wave field after each sweep from corner to corner.

In this section, we show how to compute the global solution without redundancy, by applying each of the eight sweeps of scenario 2 only once. This allows for an efficient computation of the global wavefield, regardless of the number of subdomains containing components of the source. The algorithm can still be performed in three stages, which are detailed below. For a succinct summary in pseudo-code we direct the reader to Appendix C.

In the first stage, we restrict the source density to each subdomain. Within each subdomain, we use this restricted source density to compute the corresponding local solution, from which we extract the traces on all interior interfaces Γ_{ij}^{l}, $l = B, R, T, L$. This stage is illustrated in Figure 10.

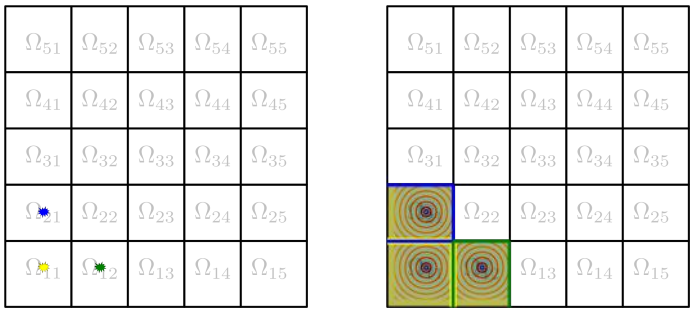


Figure 10: Left: The setup of the problem in scenario 3, with multiple point sources shown in red, yellow and blue. Right: The corresponding local solutions in each subdomain. The extracted trace information is shown using the same colors as their corresponding point sources.

In the second stage, we use the traces extracted in stage 1 to extend the local solutions into the same column and row as subdomains containing sources. This can be done using four sweeps: up, down, left, and right. We consider the upwards sweep in the first column in detail, all other sweeps are performed analogously. In the upward sweep, we use traces on the bottom of each element to compute polarized wavefields and update local solutions. For example, in Figure 10, the subdomain Ω_{11} has no incoming bottom trace. The outgoing top trace is therefore simply taken from stage one and transferred to Ω_{21}. In Ω_{21}, this trace is the incoming bottom trace and used to compute a polarized wavefield. We then update the local wavefield in Ω_{21} by adding the computed polarized wavefield. The outgoing trace on the top is then extracted from the updated local wavefield and used as the incoming trace in Ω_{31}. Continuing this procedure, we update the local wavefields in the entire first column which concludes the upwards sweep. The procedure is illustrated in Figure 11. Then, we perform the remaining three cardinal sweeps.

Applying these four sweeps computes an intermediate wavefield. This wavefield is updated in stage 3. As part of the vertical and horizontal sweeps we also extract the vertical and horizontal traces of the subdomains needed in stage 3. The individual contributions from each of the four sweeps are shown in Figure 12. The intermediate solution obtained from the sum of the contributions from stages 1 and 2 is illustrated in Figure 13. In both figures the traces extracted in preparation for stage 3 are shown by the colored lines.

We generalize the third stage of the algorithm in a similar way by computing polarized wavefields from the trace information coming into a subdomain, and updating the local solutions by adding these polarized wavefields. We then extract trace information from the updated local wavefields and use them in the neighboring subdomains as incoming traces.

As in Section 2.1.2, stage 3 is realized by sweeping over the CDD from corner to corner, updating subdomains that are diagonally adjacent, perpendicular to the sweep direction. Computed wavefields in three example diagonals for the sweep from the bottom-left to the top-right corner are illustrated in Figure 14. The same technique can also be employed for the other three diagonal sweeps: top-right corner to bottom-left corner, bottom-right to top-left corner, and top-left to bottom-right corner. These sweeps are illustrated in Figures 15 and 16.

Using these three stages, we compute the global solution in a total of eight sweeps, independently of the source distribution. In the next section, we show that applying this procedure to source densities appropriately windowed on four slightly different domain decompositions, the algorithm can be extended to entirely arbitrary source densities in a straightforward way.

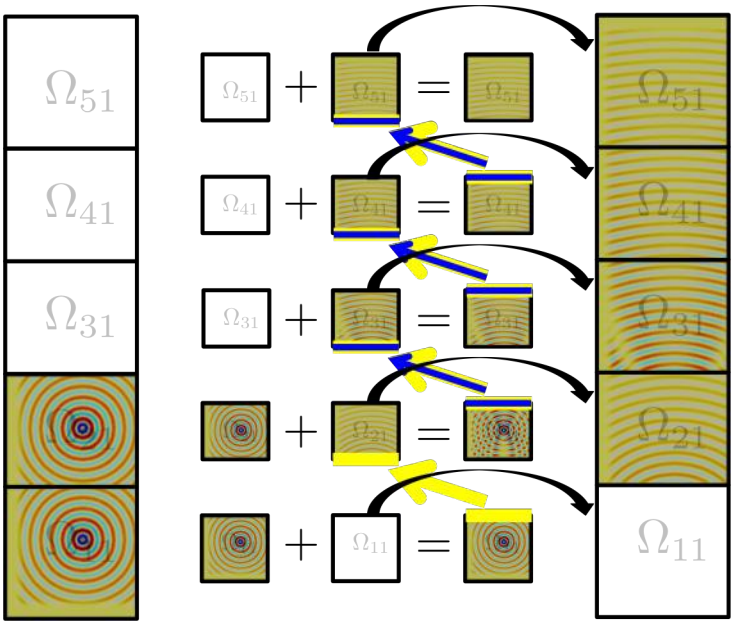


Figure 11: Summary of the upwards sweep, for scenario 3, in one column in the presence of several point sources.

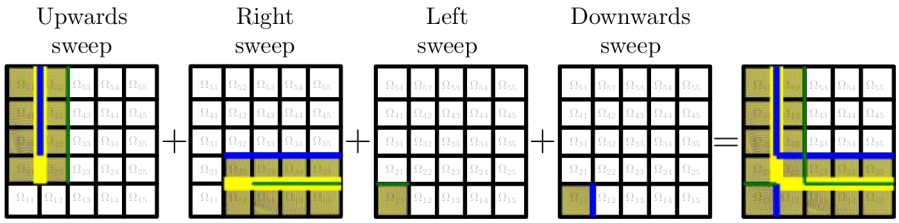


Figure 12: The contributions to the intermediate wavefield from the sweeps along each row and column in scenario 3. The generated trace information that still needs to be propagated is shown by the colored lines.

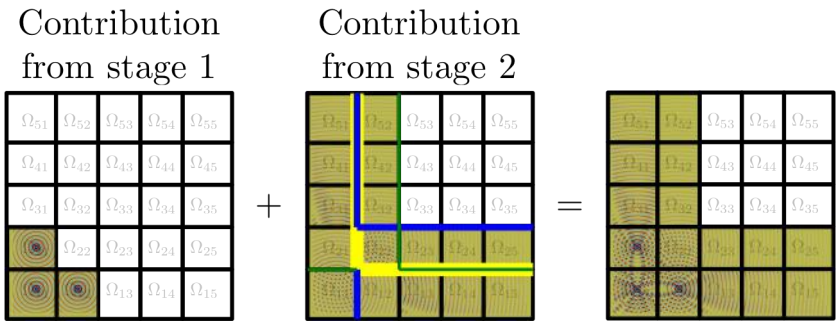


Figure 13: The reconstructed wavefield composed of the contributions from stage 1 and 2 of scenario 3, for the problem in Figure 10.

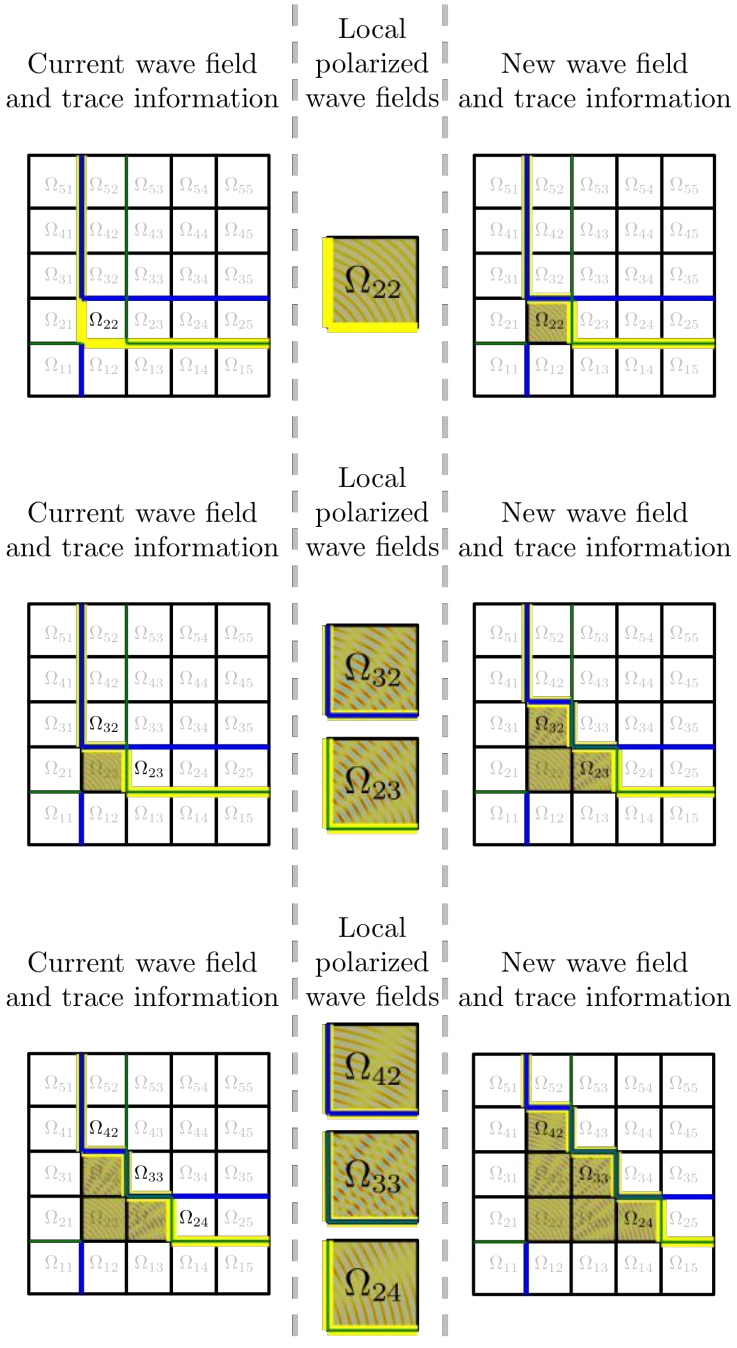
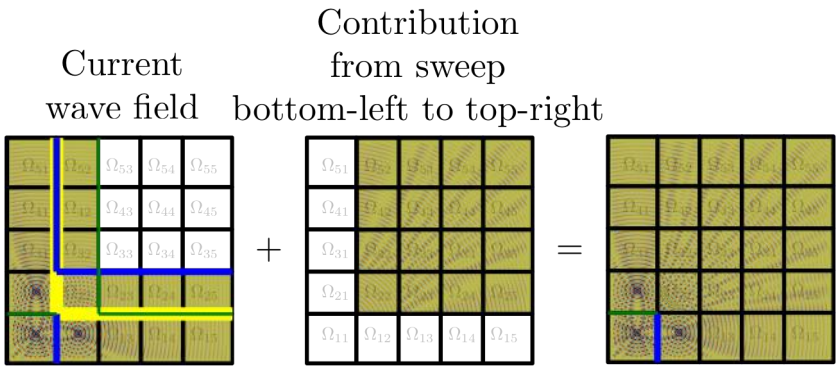
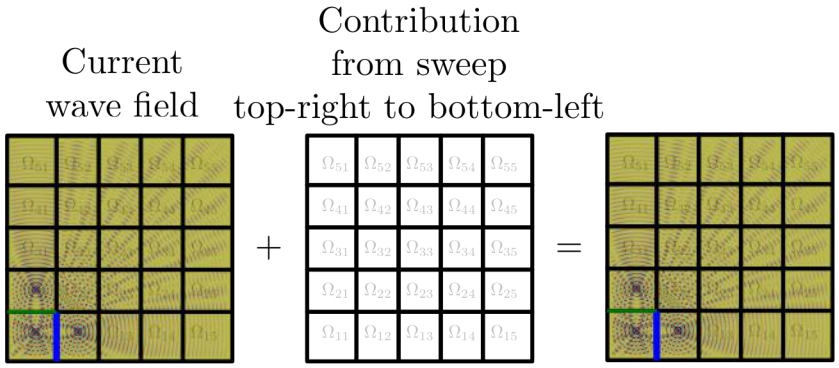


Figure 14: Illustration of the sweep from the bottom-left to the top-right corner in scenario 3.

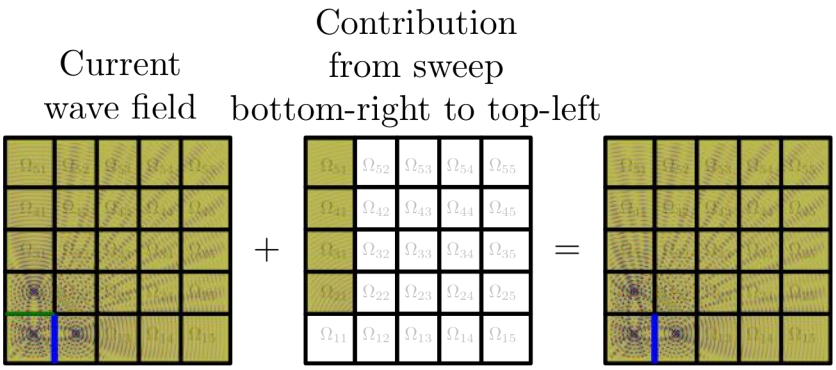


(a) The sweep from the bottom-left to the top-right corner.

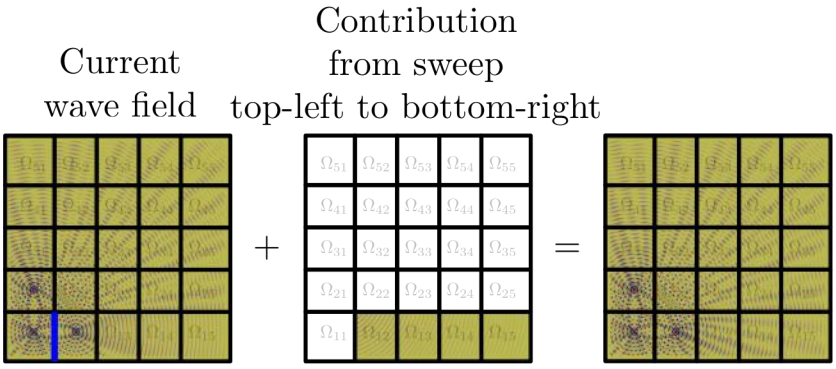


(b) The sweep from the top-right to the bottom-left corner.

Figure 15: The contribution of the sweep from the sweeps over the sweeps diagonal by diagonal in scenario 3. The extracted trace information that still needs to be propagated is shown by the colored lines.



(a) The sweep from the bottom-right to the top-left corner.



(b) The sweep from the top-left to the bottom-right corner.

Figure 16: The contribution of the sweep from the sweeps over the sweeps diagonal by diagonal in scenario 3. The extracted trace information that still needs to be propagated is shown by the colored lines.

2.1.4. Scenario 4: Arbitrary source distributions

The only remaining restriction on source distribution is that it must not cross the CDD skeleton. This restriction can be overcome by considering three more CDDs obtained from horizontal and vertical shifts in the CDD, as illustrated in Figure 17. Using the original CDD and these new shifted CDDs, we define a partition of unity of four window functions φ_i, $i = 1, 2, 3, 4$, each of which vanish on the skeleton of *one* of the four CDD (Figure 18) and we define the corresponding windowed source densities $f^{\varphi_i} := \varphi_i f$. Clearly, f^{φ_i} vanishes on the skeleton of one CDD. Using this

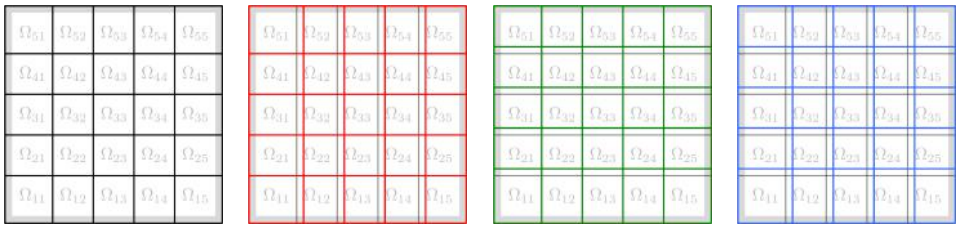


Figure 17: The original CDD (black) and the three new, shifted CDDs in red, green, and blue.

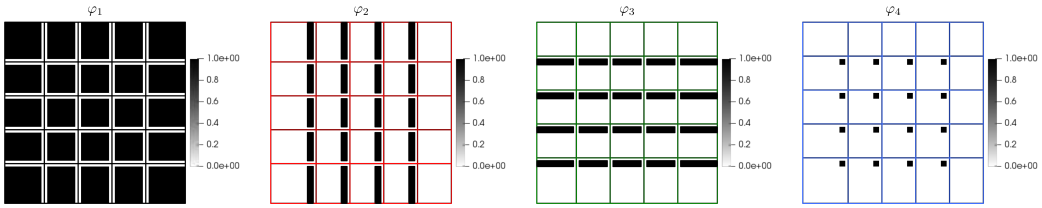


Figure 18: The window functions φ_i and the CDDs on which they vanish on the skeleton.

CDD, we can therefore apply the procedure of Section 2.1.3 to obtain the global solution u^{φ_i} corresponding to f^{φ_i}. The global solution for a general f is them simply the sum of those solutions:

$$u = \sum_{i=1}^{4} u^{\varphi_i}.$$

2.2. Discrete formulation

In this section, we show that the procedure introduced in Section 2.1 can be applied on the discrete level to approximately solve (3). We start by defining the CDD and local problems. We define the CDD so that the skeleton does not intersect with any discretization point. Then, the skeleton clearly divides the global degrees-of-freedom into sets $\boldsymbol{\Omega_{ij}}$. We then define a local problem associated with each $\boldsymbol{\Omega_{ij}}$. This problem is defined on a superset of $\boldsymbol{\Omega_{ij}}$, which we denote $\boldsymbol{\Omega_{ij}^{\delta}}$, and is the discretization of the local problem defined on $\Omega_{ij}^{\varepsilon}$. The interior boundaries are first extended by a (2δ)-layer of degrees-of-freedom and then further extended by a PML region. In the same manner as for the continuous problem, the wave speed for this problem is inherited from the global wave speed. We denote the local system matrices $\mathbf{A_{ij}}$.

Furthermore, we define interfaces on these local problems to compute discrete polarized wavefields. For the continuous problem, these interfaces are defined as straight lines along the boundaries of the subdomains Ω_{ij}. For the discrete problem, the necessary interfaces for computing discrete polarized wavefields consist of all degrees-of-freedom δ-adjacent to the polarization interface. Therefore, we define the discrete counterparts of the interfaces Γ_{ij}^{ℓ}, $\ell = L, R, T, B$, to be the δ-adjacent degrees-of-freedom to Γ_{ij}^{ℓ}. We call those sets $\boldsymbol{\Gamma_{ij}^{\ell}}$, $\ell = B, R, T, L$. Figure 19 illustrates these definitions. By construction, $\boldsymbol{\Gamma_{ij}^{\ell}}$ defines the trace information that needs to be transferred between subdomains. In the continuous problem, two neighboring local problems have to coincide in an ε-tube around the polarization interface to accurately extend the wavefield from one subdomain into the other. In the same way, the discrete problem requires two neighboring local problems to coincide in all δ-adjacent degrees-of-freedom to the polarization interface. This justifies the (2δ)-layer between $\boldsymbol{\Omega_{ij}}$ and the PML region. One δ-layer is used for $\boldsymbol{\Gamma_{ij}^{\ell}}$, and

one δ-layer is used to ensure that two neighboring local problems coincide in an appropriately sized neighborhood of $\Gamma^{\ell}_{\mathbf{ij}}$.

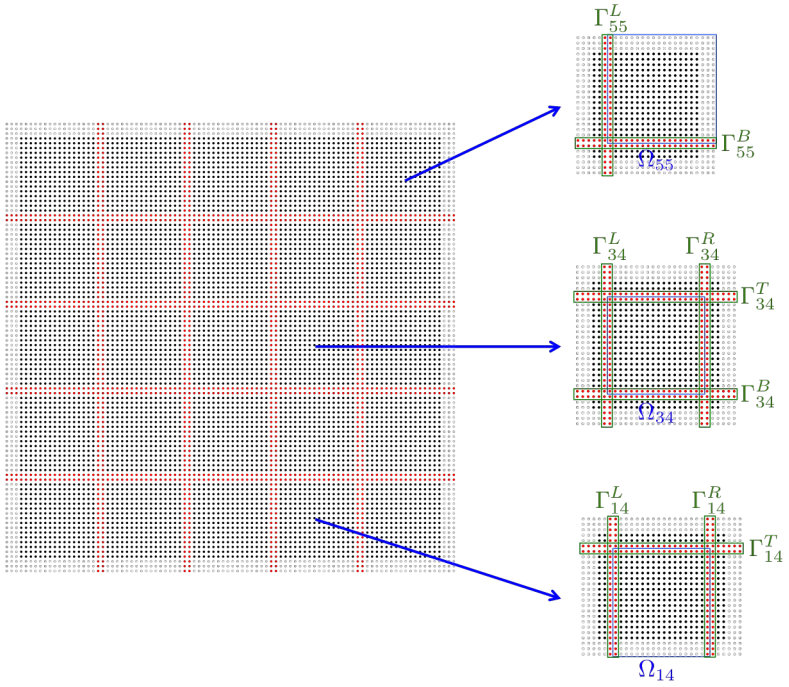


Figure 19: The definition of the discrete local problems and the discrete interfaces.

In what follows we will show how the procedures introduced in Section 2.1 can be treated on the discrete level. Since the procedure introduced in Section 2.1.3 (scenario 3) is a generalization of all previous algorithms, we start with this procedure. In the first stage, we restrict the global source vector $\mathbf{f}$ to the discretization points in $\Omega_{\mathbf{ij}}$ for each subdomain and define a local source vector $\mathbf{f_{ij}}$ on $\Omega^{\delta}_{\mathbf{ij}}$ such that $\mathbf{f_{ij}}|_{\Omega_{\mathbf{ij}}} = \mathbf{f}|_{\Omega_{\mathbf{ij}}}$ and $\mathbf{f_{ij}}$ is zero everywhere else. In the same way as in the continuous case, this local source vector can then be used to compute discrete local solutions $\mathbf{u_{ij}}$ in each subdomain:

$$\mathbf{u_{ij}} = \mathbf{A_{ij}}^{-1}\mathbf{f_{ij}}.$$

Using these local solutions, we define a global wavefield $\tilde{\mathbf{u}}$ such that $\tilde{\mathbf{u}}|_{\Omega_{\mathbf{ij}}} = \mathbf{u_{ij}}|_{\Omega_{\mathbf{ij}}}$. As with the continuous case, we extract the values of these local solutions on $\Gamma^{\ell}_{\mathbf{ij}}$ and use them in stage 2 to update $\tilde{\mathbf{u}}$.

The second and third stage of the algorithm are realized by applying sweeps over the domain in the same way as in Section 2.1.3. The only difference is that the trace information on Γ^{ℓ}_{ij} is replaced by its discrete counterparts $\Gamma^{\ell}_{\mathbf{ij}}$, and the computation of continuous polarized wavefields is replaced by the computation of discrete polarized wavefields (9) using the local system matrices $\mathbf{A_{ij}}$ and the appropriate interfaces $\Gamma^{\ell}_{\mathbf{ij}}$.

Until now, we have elided discussion of a subtle point in the discrete algorithm: construction of the L-shaped trace information. As an example, we consider the discrete counterpart of the trace Γ^{BL}_{ij}, but any other trace information can be handled analogously. Following the notation introduced in Section 2.1, let us denote the trace information on $\Gamma^{\mathbf{B}}_{\mathbf{ij}}$ by $\boldsymbol{\lambda}^{\boldsymbol{B}}_{\boldsymbol{ij}}$ and the trace information on $\Gamma^{\mathbf{L}}_{\mathbf{ij}}$ by $\boldsymbol{\lambda}^{\boldsymbol{L}}_{\boldsymbol{ij}}$. Similarly to Γ^{BL}_{ij} in Section 2.1, $\boldsymbol{\Gamma}^{\boldsymbol{BL}}_{\boldsymbol{ij}}$ is defined as the L-shaped set of degrees-of-freedom surrounding the upper right quadrant of the set $\Omega_{\mathbf{ij}}$, as illustrated in Figure 20. The trace information $\boldsymbol{\lambda}^{\boldsymbol{BL}}_{\boldsymbol{ij}}$ is then defined such that $\boldsymbol{\lambda}^{\boldsymbol{BL}}_{\boldsymbol{ij}}|_{\Gamma^{\mathbf{B}}_{\mathbf{ij}}} = \boldsymbol{\lambda}^{\boldsymbol{B}}_{\boldsymbol{ij}}$ and $\boldsymbol{\lambda}^{\boldsymbol{BL}}_{\boldsymbol{ij}}|_{\Gamma^{\mathbf{L}}_{\mathbf{ij}}\setminus\Gamma^{\mathbf{B}}_{\mathbf{ij}}} = \boldsymbol{\lambda}^{\boldsymbol{L}}_{\boldsymbol{ij}}|_{\Gamma^{\mathbf{L}}_{\mathbf{ij}}\setminus\Gamma^{\mathbf{B}}_{\mathbf{ij}}}$. Note that by convention we have defined the trace information of $\boldsymbol{\lambda}^{\boldsymbol{BL}}_{\boldsymbol{ij}}$ to coincide with $\boldsymbol{\lambda}^{\boldsymbol{B}}_{\boldsymbol{ij}}$ in $\Gamma^{\mathbf{B}}_{\mathbf{ij}} \cap \Gamma^{\mathbf{L}}_{\mathbf{ij}}$. This is an arbitrary choice, as the trace information on $\Gamma^{\mathbf{B}}_{\mathbf{ij}} \cap \Gamma^{\mathbf{L}}_{\mathbf{ij}}$ can be obtained from any of the two traces $\boldsymbol{\lambda}^{\boldsymbol{B}}_{\boldsymbol{ij}}$ or $\boldsymbol{\lambda}^{\boldsymbol{L}}_{\boldsymbol{ij}}$.

Following this procedure, we can compute approximations $\tilde{\mathbf{u}}$ of the global solution that only differ from $\mathbf{u}$ by errors induced by the discretization or PMLs, as long as the global source vector $\mathbf{f}$ is zero on the degrees-of-freedom on the

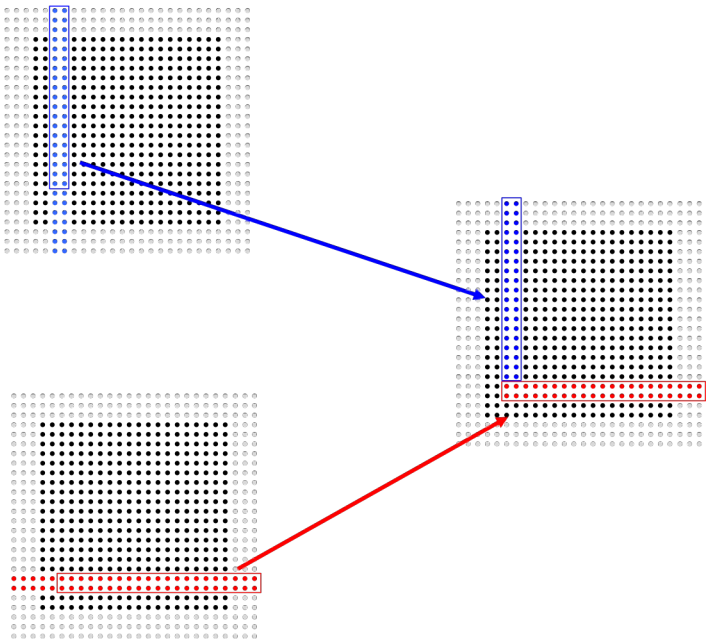


Figure 20: The definition of the discrete L-shaped trace.

skeleton of the CDD, i.e., the union of all Γ_{ij}^{ℓ}. This procedure can be extended to entirely arbitrary source vectors using window functions, as introduced in Section 2.1.4.

3. Implementation and Complexity

The procedure introduced in Section 2 produces an approximation of a global solution to (3). This approach therefore defines the approximate solution operator $\tilde{\mathbf{A}}^{-1}$, such that $\tilde{\mathbf{A}}^{-1}(\boldsymbol{f}) \approx \mathbf{A}^{-1}\boldsymbol{f}$. Thus, we use $\tilde{\mathbf{A}}^{-1}$ to precondition (3),

$$\tilde{\mathbf{A}}^{-1}\left(\mathbf{A}\mathbf{u}\right) = \tilde{\mathbf{A}}^{-1}\left(\mathbf{f}\right), \tag{15}$$

and use a Krylov subspace method, such as GMRES [65] or BiCG-stab [79], to solve (15).

The resulting preconditioned iterative solver has the following properties:

- the preconditioner $\tilde{\mathbf{A}}^{-1}$ can be applied with optimal parallel complexity, $O(N/p)$, and
- Krylov methods applied to (15) converge (empirically) in $O(\log \omega)$ iterations;

thus making it parallel scalable. In this Section we focus on the first property. In particular, we show the implementation of the preconditioner, while analyzing its complexity with respect to the computational and communication cost. For the second property, we will present extensive numerical evidence in Section 5.

We consider a standard communication model [5, 21, 61, 77]. The model assumes that each process is only able to send or receive a single message at a time, though different messages can be sent and received asynchronously. A message of size M can be communicated with $\alpha + \beta M$ complexity. The latency α represents the minimum complexity with which an arbitrary message from one process to another can be communicated, and is a constant overhead for any communication. The inverse bandwidth β is the complexity with which one unit of data can be communicated.

We implement the algorithm within a distributed memory framework using MPI. For simplicity, we assign each row of subdomains to one MPI rank[3]. By assigning multiple subdomains to one rank, the preconditioner can be applied

[3] This restriction can be relaxed to exploit asynchronous parallelism models and subdomain pipelining, for extremely large problems.

optimally in parallel, as described below. Consider a diagonal sweep, for example from the bottom-left corner to the top-right corner, for a $q \times q$ CDD. To maximize parallelism, each subdomain in a diagonal, perpendicular to the sweep direction, has to be processed in parallel. This can be realized by assigning subdomains to MPI ranks in a row-based fashion such that the i-th row of the CDD is processed by rank $i \bmod p$. Thus, each rank is assigned one or several rows of subdomains and does all of their associated computation. An illustration of the assignment of subdomains to rank is provided in Figure 21. In what follows, we assume this configuration, which allows us to also exploit the maximum parallelism in the application of the preconditioner and obtain a parallel scalable solver. The only restriction in this setup is that the number of ranks p used in this setup is bounded by the number of subdomains in one column, i.e., $p \leq q = O(n)$.

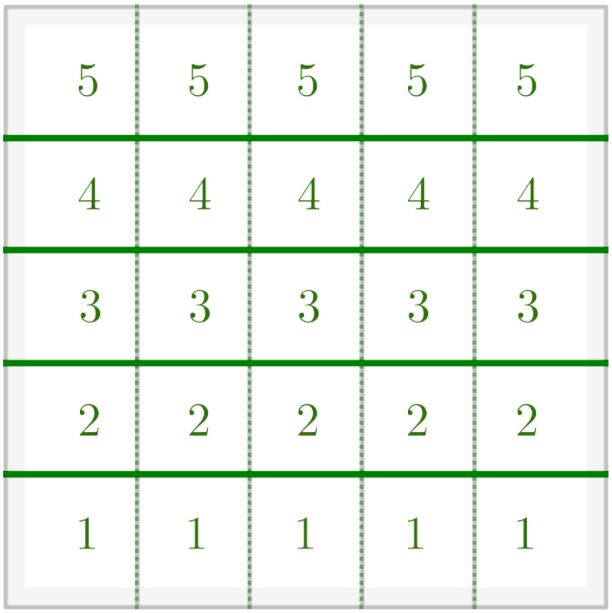


Figure 21: The assignment of subdomains to processor in a row-based fashion.

For analysis purposes, we divide the solver for (3) in three phases:

1. *Setup* the local problems, system matrices, right-hand sides, and the corresponding window functions φ_i;
2. *Factorize* the local system matrices; and
3. *Solve* the linear system using a Krylov method.

Phase 1: Setup
For two-dimensional problems, the size of the local problems associated with the subdomain Ω_{ij} is $O(1)$. A crucial requirement to ensure that these local problems are indeed of $O(1)$ size is that the thickness of the PML region, in wavelengths normal to Ω_{ij}, is held constant with mesh-refinement, i.e., the number of degrees-of-freedom in the PML does not change and the spatial PML-width scales as $O(1/\omega)$. Consequently, the local system matrices $\mathbf{A_{ij}}$, right-hand sides, and the window functions φ_{ij} can be assembled with $O(1)$ computational complexity. In a parallel computational environment, each of the $O(q^2)$ subdomains can be processed independently and no communication is required between subdomains. Thus all rows are processed with a $O(n)$ parallel computational complexity, because $q = O(n)$ and $p = O(q)$. For two-dimensional problems, the computation in the setup phase is therefore realized with optimal $O(N/p)$ parallel computational complexity. For three-dimensional problems, the size of the local problems associated with each subdomain is $O(n)$, and the same arguments also yield optimal $O(N/p)$ parallel complexity. In either case, there is no communication cost during the setup phase.

Phase 2: Factorize
For two-dimensional problems, the system matrices of the local problems have size $O(1)$ and therefore can be

factorized with $O(1)$ computational complexity. As in stage 1, each subdomain can be processed independently with zero communication, thus we can follow the same argument as in the setup phase to justify that all of the q^2 subdomains can be factorized with $O(N/p)$ parallel computational complexity. For three-dimensional problems, the system matrices of the local problems have size $O(n)$ and are quasi-one-dimensional. Thus, standard sparse direct solvers can be used to factorize the matrices with optimal computational complexity [40, 24, 44], i.e., $O(n)$. The same arguments as for the two-dimensional case can be employed to show that, in three dimensions, the q^2 subdomains can be factorized with $O(N/p)$ parallel computational complexity. In either case, there is no communication cost during the factorization phase.

Phase 3: Solve

We use the GMRES method to solve the linear system (3). Each iteration of this method consists of three main operations: the application of $\mathbf{A}$, the computation between the Ritz vectors, and the application of the preconditioner $\tilde{\boldsymbol{A}}^{-1}$. For each of the three operations, we assume that the global vectors are provided in a distributed fashion such that each subdomain Ω_{ij} holds the values of the global vector corresponding to $\boldsymbol{\Omega_{ij}}$.

The application of $\mathbf{A}$ can be realized by simply applying the local system matrices to a local vector, as long as each subdomain holds all degrees-of-freedom associated with $\boldsymbol{\Omega_{ij}}$ and $\boldsymbol{\Gamma^{\ell}_{ij}}$ for $\ell = B, R, T, L$. This requires some communication because the subdomains only store the degrees-of-freedom in $\boldsymbol{\Omega_{ij}}$. Considering the row-based assignments of subdomains to MPI ranks, the sets $\boldsymbol{\Gamma^{B}_{ij}}$ have to be communicated to the subdomain $\Omega_{(i-1)j}$, and the sets $\boldsymbol{\Gamma^{T}_{ij}}$ have to be communicated to the subdomain $\Omega_{(i+1)j}$. In practice, for two-dimensional problems, each row has to communicate $O(n)$ values to a neighboring MPI rank. Using the above communication model, this communication has complexity

$$O(\alpha + \beta n) = O(n).$$

Due to the row-based rank assignment, up to p of the q rows can simultaneously exchange information, resulting in an overall $O(qn/p) = O(N/p)$ parallel communication complexity. The same arguments can be applied for the communication of the sets $\boldsymbol{\Gamma^{T}_{ij}}$. Thus, for two dimensional problems, all necessary information can be communicated with optimal $O(N/p)$ parallel complexity. In three dimensions, each row has to communicate $O(n^2)$ values to a neighboring MPI rank. Following the same arguments as before all necessary information can be communicated with optimal $O(N/p)$ parallel complexity. Once all information is communicated, the local system matrices $\mathbf{A_{ij}}$ are applied to obtain the action of the global system matrix $\mathbf{A}$ on the degrees-of-freedom in $\boldsymbol{\Omega_{ij}}$. This application can be realized with $O(1)$ and $O(n)$ computational complexity per subdomain in two and three dimensions, respectively. Again, each of the $O(q^2)$ subdomains can be processed independently resulting in the total $O(N/p)$ parallel computational and communication complexity to apply the matrix $\mathbf{A}$ for two- and three-dimensional problems.

At iteration k, k inner products between the Ritz vectors need to be computed. Each inner product can be computed efficiently in parallel, given that each subdomain contains the local components of the Ritz vectors. Thus, each inner product are computed with $O(N/p)$ parallel computational complexity and $O(p)$ parallel communication complexity. The application of $\tilde{\boldsymbol{A}}^{-1}$ is analyzed in detail in Section 3.1, where we show that it can be applied with $O(N/p)$ parallel computational and communication complexity. The k-th GMRES iteration can therefore be realized with $O(kN/p)$ parallel computational complexity and $O(N/p)$ parallel communication complexity[4]. Due to the effectiveness of the preconditioner, the number of iterations only grows as $O(\log \omega)$. This claim is corroborated by the numerical examples in Section 5. Using restarts, the GMRES method can therefore be applied with a total parallel computational and communication complexity of $O((N/p) \log \omega)$. The computation and communication complexities are summarized in Tables 1 and 2, for two- and three-dimensional problems.

[4]To achieve the parallel communication complexity, we implicitly assume that $O(kp) = O(N/p)$.

Step	Computation per subdomain	Computation per rank	Communication	Total time
Set up	$O(1)$	$O(n) = O(N/p)$	$O(1)$	$O(N/p)$
Factorization	$O(1)$	$O(n) = O(N/p)$	$O(1)$	$O(N/p)$
Solve	$O(\log \omega)$	$O(n \log \omega) = O((N/p) \log \omega)$	$O((N/p) \log \omega)$	$O((N/p) \log \omega)$

Table 1: Parallel complexity of the different stages for two-dimensional problems, where N is the total number of discretization points, n is the number of discretization points per dimension, ω is the frequency following $\omega \sim n$, and p is the number of processors following $p = O(n)$.

Step	Computation per subdomain	Computation per rank	Communication	Total time
Set up	$O(n)$	$O(n^2) = O(N/p)$	$O(1)$	$O(N/p)$
Factorization	$O(n)$	$O(n^2) = O(N/p)$	$O(1)$	$O(N/p)$
Solve	$O(n \log \omega)$	$O(n^2 \log \omega) = O((N/p) \log \omega)$	$O((N/p) \log \omega)$	$O((N/p) \log \omega)$

Table 2: Parallel complexity of the different stages for three-dimensional problems, where $N = n^3$ are the total number of discretization points, and the number of discretization points per dimension, ω is the frequency following $\omega \sim n$, and p is the number of processors following $p = O(n)$.

3.1. Application of $\tilde{\mathbf{A}}^{-1}$

The application of $\tilde{\boldsymbol{A}}^{-1}$ can be divided in four steps:

(a) Computation of the local solutions,
(b) Extension of the local solutions into the same row,
(c) Extension of the local solutions into the same column, and
(d) Extension of the local solutions into the rest of the subdomains.

As before, we assume that at the beginning and at the end of each step, each rank holds all the values corresponding to the degrees-of-freedom in $\Omega_{\mathbf{ij}}$ for all of its associated subdomains.

The computation of the local solutions requires to solve a local problem in each subdomain. Each of the local problems can be solved in $O(1)$ and $O(n)$ computational complexity for two- and three-dimensional problems, respectively[5]. All local solutions are computed independently and p local solutions can be computed simultaneously. Therefore, for both two- and three-dimensional problems, step 1 is realized with $O(N/p)$ parallel computational complexity and there is no communication cost.

The extension of the local solutions into the subdomains contained in the same row requires one left- and one right-sweep in each row. Due to the row-based rank assignment, each of those sweeps must be realized sequentially. However, each row can be processed independently, with no inter-row communication. For example, consider the right-sweep in one row. In each subdomain, a (discrete) polarized wavefield has to be computed (i.e., the local problem has to be solved), the local solution has to be updated, and the traces to be transferred to the right have to be extracted. For two-dimensional problems, each of those stages can be realized with $O(1)$ computational complexity per subdomain because the size of the subdomains is $O(1)$. Since each subdomain in the row has to be processed sequentially and there are $q = O(n)$ subdomains in one row, this procedure can be realized with $O(n)$ computational complexity per row. Employing this procedure concurrently in each row yields $O(N/p)$ parallel computational complexity and no communication cost. For three-dimensional problems, each stage can be realized with $O(n)$ computational complexity per subdomain[6]. Because each of the $q = O(n)$ subdomains in one row is processed sequentially, this results in a $O(n^2)$ computational complexity in every row. Thus, processing each row concurrently results in the $O(N/p)$ parallel

[5] Similar to the factorization, standard sparse direct solvers are used to solve the quasi-one-dimensional problems arising from three-dimensional problems with optimal computational complexity, i.e., $O(n)$.

[6] Again, standard sparse direct solvers are used to solve the quasi-one-dimensional problems arising from three-dimensional problems with optimal complexity, i.e., $O(n)$.

computational complexity and no communication cost. Analysis of the leftward sweep follows the same logic and reaches the same conclusions.

Extending the local solutions from one subdomain into the other subdomains within the same column requires one upward- and one downward-sweep. Given that the processor assignment does not allow for each column to be processed independently, the organization of the computation is non-trivial in order to reveal parallelism. As an example, we explain the implementation of an upward sweep. We begin by considering the subdomain in the bottom-left corner, Ω_{11}. Once Ω_{11} is processed and its information sent to Ω_{21}, rank 0 can process the subdomain Ω_{12} and rank 1 can process subdomain Ω_{21}. Following this procedure, the upward sweep is applied by sweeping over the domain, essentially pipelining the rows. Parallelism is achieved by processing diagonally-adjacent sets of subdomains simultaneously. By construction, each diagonal set has at most q subdomains. For two-dimensional problems, each subdomain can be processed with $O(1)$ computational complexity, independent from all other subdomains. Thus, an entire diagonal set can be processed with $O(q/p)$ parallel computational complexity. For three-dimensional problems, each subdomain can be processed with $O(n)$ computational complexity, resulting in a $O(nq/p)$ parallel computational complexity to process each diagonal. Since there are $O(q)$ diagonals, and $q = O(n)$, this results in the total $O(N/p)$ parallel computational complexity to apply the upward sweep for both two- and three-dimensional problems.

In contrast with previous phases and steps, the upward and downward sweeps also require communication. For each subdomain Ω_{ij}, the traces $\Gamma^{\mathbf{B}}_{\mathbf{ij}}$ have to be communicated from the subdomain $\Omega_{(i-1)j}$. For two-dimensional problems, the information transfers are performed in parallel, and all $O(n)$ values, from all $\Gamma^{\mathbf{B}}_{\mathbf{ij}}$s, are communicated with $O(n/p)$ parallel communication complexity. For three-dimensional problems, the communication volume is $O(n^2)$ and, using the same arguments, the parallel communication complexity, per diagonal set, is $O(n^2/p)$. For both two- and three-dimensional problems, considering all $O(q) = O(n)$ diagonals, we find the optimal parallel $O(N/p)$ communication complexity. The analysis of the downward sweep follows the same argument and achieves the same conclusion.

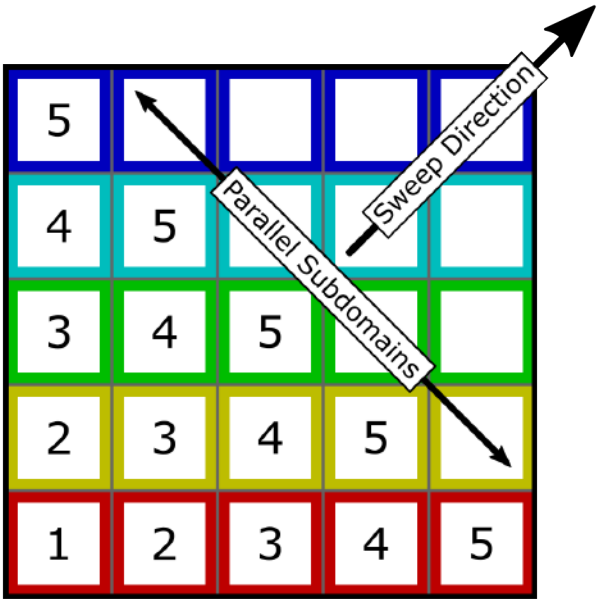


Figure 22: Illustration of parallelism in diagonal sweeps. Subdomains processed in parallel have same label. Subdomains with like color are assigned to same processor.

The extension of the local solutions into the remaining subdomains, those not in the same row or column, requires sweeps over the CDD from corner to corner, along both diagonals. In these diagonal sweeps, each step requires

processing of sets of subdomains that are diagonally-adjacent along the direction perpendicular to the sweep direction as illustrated in Fig. 22. This process has the same computation and communication patterns as the upward and downward sweeps and thus, following the same analysis, have the same complexities: they are applied with $O(N/p)$ parallel computational complexity and $O(N/p)$ parallel communcation complexity.

In summary, each step has $O(N/p)$ parallel computational complexity and at most $O(N/p)$ parallel communication complexity, thus the preconditioner can be applied with $O(N/p)$ parallel computational and communication complexity. The computational complexity and communication complexities, for applying the preconditioner, are summarized in Tables 3 and 4.

	Computation			
Step	per subdomain	per rank	Communication	Total time
Part 1	$O(1)$	$O(n) = O(N/p)$	$O(1)$	$O(N/p)$
Part 2	$O(1)$	$O(n) = O(N/p)$	$O(1)$	$O(N/p)$
Part 3	$O(1)$	$O(n) = O(N/p)$	$O(N/p)$	$O(N/p)$
Part 4	$O(1)$	$O(n) = O(N/p)$	$O(N/p)$	$O(N/p)$

Table 3: Parallel complexity of the different stages for two-dimensional problems for $p = O(n)$.

	Computation			
Step	per subdomain	per rank	Communication	Total time
Part 1	$O(n)$	$O(n^2) = O(N/p)$	$O(1)$	$O(N/p)$
Part 2	$O(n)$	$O(n^2) = O(N/p)$	$O(1)$	$O(N/p)$
Part 3	$O(n)$	$O(n^2) = O(N/p)$	$O(N/p)$	$O(N/p)$
Part 4	$O(n)$	$O(n^2) = O(N/p)$	$O(N/p)$	$O(N/p)$

Table 4: Parallel complexity of the different stages for three-dimensional problems when $p = O(n)$.

4. Heterogeneities

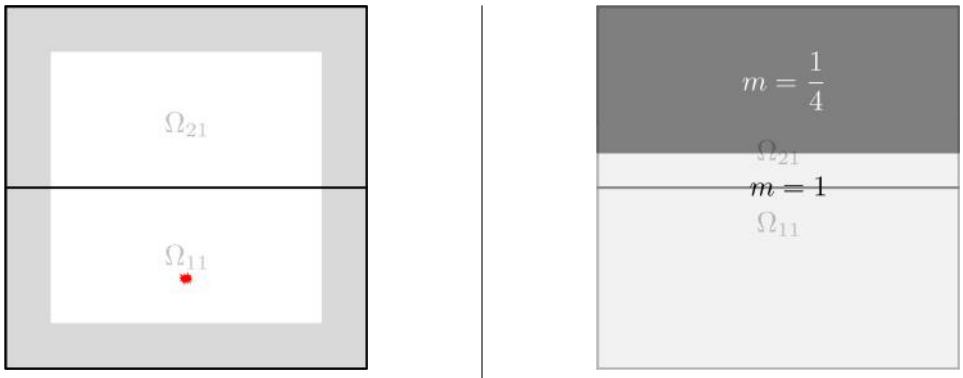
As developed in Section 2, the wavefield computed by the proposed algorithm can be interpreted as a sum of global solutions, each of which is induced by a global source density supported in a single subdomain only. We denote a source density supported in the subdomain Ω_{ij} by $\mathbf{f_{ij}}$ and the global wavefield induced by this source density as $\mathbf{u}^{\mathbf{f_{ij}}}$. Accordingly, the global wavefield $\mathbf{u}$ can be written as

$$\mathbf{u} = \sum_{i=1}^{q} \sum_{j=1}^{r} \mathbf{u}^{\mathbf{f_{ij}}}.$$

To understand the effect of heterogeneities on the effectiveness of the preconditioner, it is sufficient to consider a wavefield $\mathbf{u}^{\mathbf{f_{ij}}}$ due to a single point source in detail. We consider two examples. The first example reveals the effect of reflections on the wavefield constructed by the preconditioner. The second example shows the effects of reflections in more pathological media.

For the first example, we consider a simple domain decomposition with two subdomains, Ω_{11} and Ω_{21}, divided by a horizontal line. We consider a layered wave speed distribution with two layers where the line of discontinuity is a horizontal line in Ω_{21}. The setup is illustrated in Figure 23.

In this example, excluding local partial wavefield computation, only the upward sweep computes a non-zero partial wavefield. The solution obtained from this sweep is shown in Figure 24. The wavefield in the bottom subdomain is a solution to a problem with constant wave speed, and the reflection due to the discontinuity in the wave speed is clearly visible in the field in the top subdomain. The solutions in each subdomain can be explained in the following way: the local wavefield for Ω_{11} from stage 1 is computed by solving the local problem in Ω_{11} for the point source. Therefore



(a) The CDD.

(b) The wave speed.

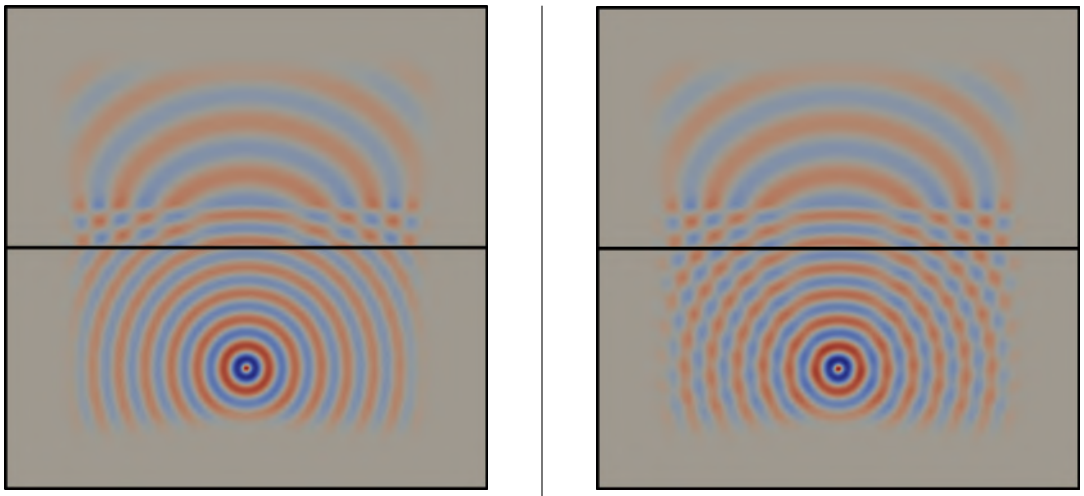
Figure 23: Illustration of a 2×1 CDD with associated wavespeed and a point source (red star).

the discontinuity in Ω_{21} is not visible and the wavefield for the constant wave speed is computed. This results in a poor approximation of the global solution in Ω_{11}. Nevertheless, the top traces $\boldsymbol{\lambda}_{11}^T$ are extracted. In the upward sweep (as part of stage 2), these traces are used in Ω_{21} to compute a polarized wavefield. This local wavefield is a good approximation of the global solution in Ω_{21} for two reasons:

- The discontinuity in the wave speed is visible to the local problem in Ω_{21}, and
- The trace $\boldsymbol{\lambda}_{11}^T$ contains all required information from Ω_{11} so that the polarized wavefield computed in Ω_{21} is a good approximation of the global wavefield.

The resulting wavefield is therefore a good approximation of the global wavefield only in Ω_{21}. As shown in Figure 24a, there is a mismatch between the two local solutions, which produces a large residual concentrated at the interface.

Following the reasoning in [71, 16], this residual contain the information necessary to propagate the wavefield locally between subdomains. The solutions obtained from applying the preconditioner to the right-hand side, and the solution after the first GMRES iteration are shown in Figure 24. It is clear that after the second iteration, the GMRES method has converged to a good approximation of the wavefield.



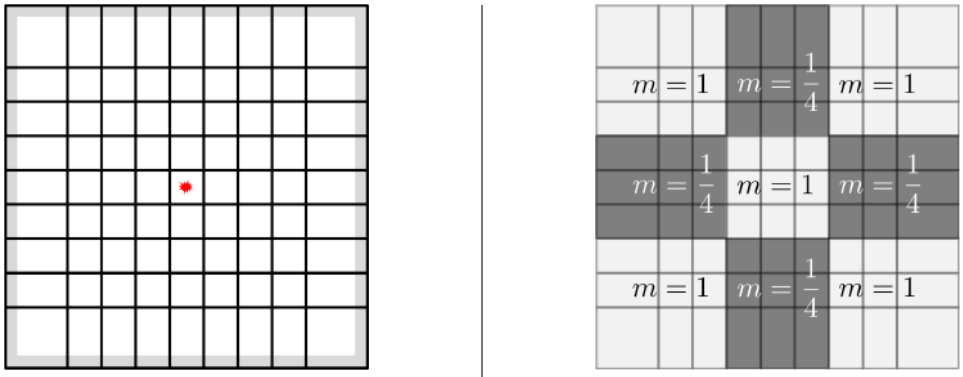
(a) Before the first GMRES iteration.

(b) After the first GMRES iteration).

Figure 24: Illustration of the computed wave fields.

This example shows that the preconditioner does not construct a good approximation of the global solution in rough media. This is because the preconditioner tracks the physical behavior of some waves propagating through the domain, including waves propagating in a straight line, intra-subdomain reflections, and some refracted waves, but does not account for any inter-subdomain reflections. Nevertheless, since the preconditioner is used as part of an iterative solver for the linear system, the inter-subdomain reflections and unresolved refractions will be treated in subsequent iterations.

To further illustrate this point, let us consider an example for a discontinuous wave speed distribution in a checkerboard pattern shown in Figure 25. Figure 26 shows the reconstructed wavefields after several GMRES iteration



(a) The CDD. (b) The wave speed.

Figure 25: Illustration of the setup for the example involving a 9×9 CDD. The point source is shown by the red star, the PML region is shown in gray.

and the corresponding residual. Note that after 2 iterations, the procedure has already computed a reasonable wavefield. After that, almost no change is visible in the plot of the wavefield, but only in the residual. It takes 27 GMRES iterations to solve this problem to an accuracy of 10^{-6}.

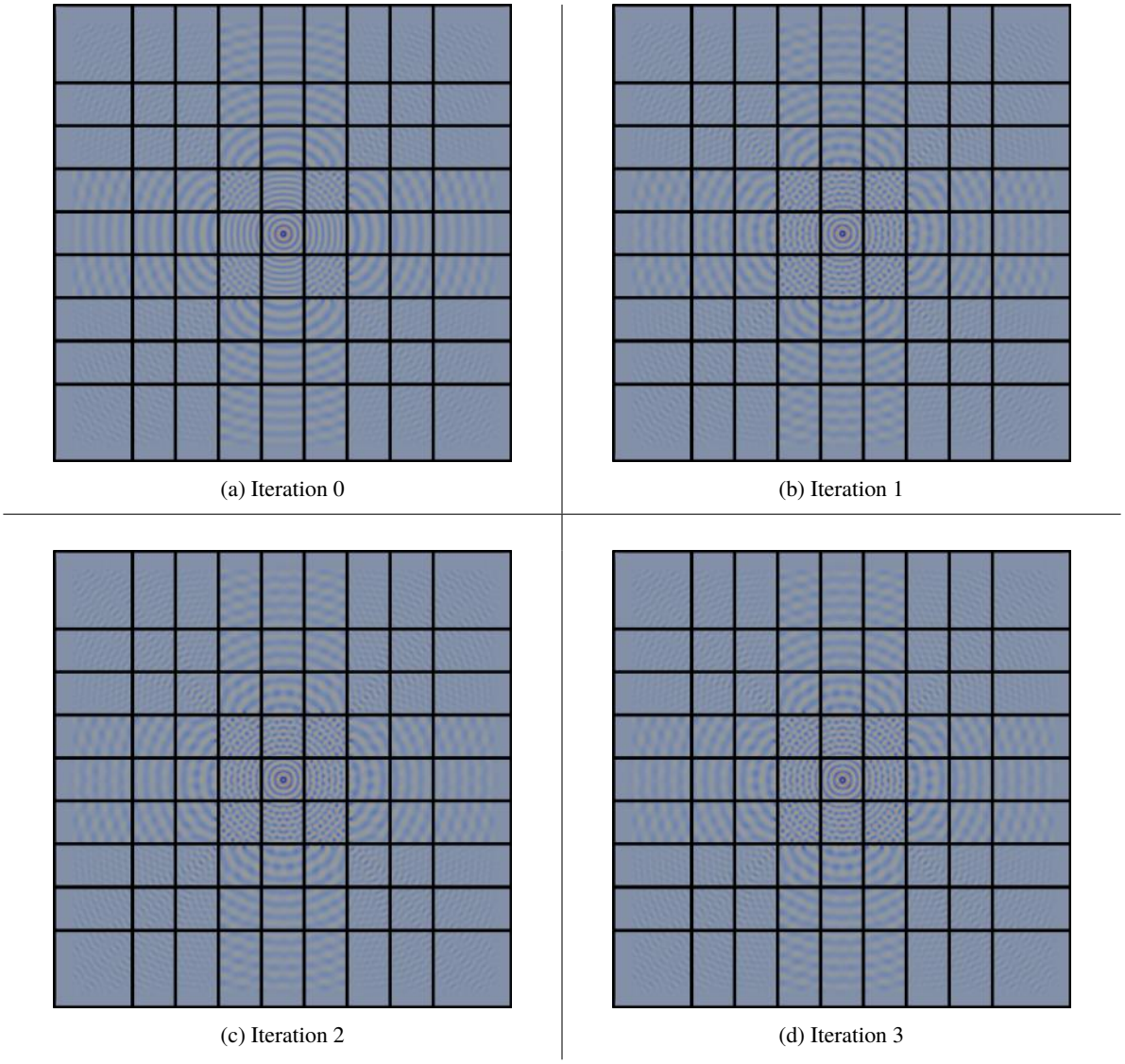
The second experiment shows that a single application of the preconditioner is insufficient to propagate all reflected waves. Thus multiple iterations are required to compute accurate global wavefields and consequently, the overall solver is sensitive to the number of reflections induced by the medium. In fact, once the dominant reflections are resolved by the preconditioner, the number of iterations only grows as $O(\log \omega)$ as the frequency (and the degrees-of-freedom) grows. This is corroborated by the numerical example for a wave-guide considered in Section 5.

The proposed solver is therefore scalable, however, the proportionality constant is strongly dependent on the number of reflections induced by the heterogeneous wavespeed. In fact, the numerical examples in Section 5 show that even in the case of wave-guides where reflections play a crucial role, the solution strategy surprisingly still only results in iteration counts lower than 40. Of course, the same arguments can be applied to refracted waves in the same way and similar effects can be seen.

5. Numerical Examples

In this section, we consider several numerical examples to corroborate the claims of this paper. All numerical examples are constructed using variations of a standard setup. The problems are posed on the unit square ($d = 2$) or the unit cube ($d = 3$) and the the wavespeed is scaled such that the the squared slowness, $m(x) \in [m_0, 1]\ \forall x \in \Omega$ for some $m_0 > 0$. For the characteristic frequency ω, the minimum wavelength is $2\pi/\omega$ and the maximum wavelength is $2\pi/(\sqrt{m_0}\omega)$. All problems are discretized using a uniform second-order finite difference approximation with N total discretization points, i.e., there are $n = N^{\frac{1}{d}}$ points in one direction. The characteristic frequency, ω, is chosen such that there are at least 10 points-per-wavelength, i.e., $\omega = 2\pi n/10$.

In each case, we decompose Ω using a CDD with $q = r$ rows and columns. Each subdomain Ω_{ij} in the CDD is chosen so that, ignoring the discretization points in the PML region, the set $\Omega_{\mathbf{ij}}$ has 101 discretization points in each direction. The skeleton of the CDD does not intersect with any discretization point, so $n = 101q$ and $N = (101q)^d$.



(a) Iteration 0

(b) Iteration 1

(c) Iteration 2

(d) Iteration 3

Figure 26: The computed wave fields after applying the preconditioner (Iteration 0) and after the first three GMRES iterations.

N (without PML)	$\omega/2\pi$	$q = r$	wavelengths in PML region 1	2	3	4	5	6	7
202×202	20.1	2	3	2	1	1	1	1	1
404×404	40.3	4	4	3	1	1	1	1	1
808×808	80.7	8	5	3	2	1	1	1	1
1616×1616	161.5	16	6	2	2	1	1	1	1
3232×3232	323.1	32	7	3	2	2	1	1	1
6464×6464	646.3	64	8	4	3	3	2	1	1
12928×12928	1292.7	128	10	5	3	3	3	2	1

Table 5: Number of iterations necessary to solve a homogeneous problem using increasingly thick PMLs and different frequencies.

Accordingly, the maximum frequency is $\omega = 2\pi(101q - 1)/10$, which is chosen so that there are 10 discretization points per wavelength. The PML region is chosen so that $\lceil 20/\sqrt{m_0} \rceil$ discretization points are in this region. This ensures that the PML region is at least 2 wavelengths thick. The source distribution of the standard setup consists of four point sources at

$$x_1 = (0.125, 0.125), \quad x_2 = (0.125, 0.875),$$
$$x_3 = (0.875, 0.125), \quad x_4 = (0.875, 0.875).$$

These point sources are modelled by an approximation with an exponential function:

$$f(x) := \frac{n^2}{\pi} \sum_{i=1}^{4} e^{-n^2|x_i - x|^2}.$$

In all examples, the global system is solved using a preconditioned GMRES method [65] with a tolerance of 10^{-6}, using the zero-vector as an initial guess. All local problems are solved using Pardiso 6.0 [47]. Source code, models, and experiment configurations for these examples are available online [75, 76].

5.1. Effect of PML-induced and discretization errrors

In this section we provide numerical evidence of the claim in Section 2 that, in a homogeneous medium, the accuracy of the preconditioner depends only on the accuracy of the discretization and the quality of the absorbing boundary conditions. We consider the standard setup for a constant squared slowness $m = 1$ and we control the accuracy of the absorbing boundary condition by increasing (or reducing) the PML thickness. Table 5 depicts the dependence of the preconditioner, measured in iterations, on the quality of the absorbing boundary condition, which is tuned by varying the number of wavelengths inside the PML region between 1 and 7. If the accuracy of the absorbing boundary condition is reduced (fewer wavelengths), the preconditioner is less effective. If the accuracy of the absorbing boundary condition is increased (more wavelengths), the preconditioner is more effective. In fact, if the accuracy of the boundary condition is sufficiently high, the preconditioner is perfectly effective resulting in iteration counts independent of ω. It can also be seen that the PML thickness has to be increased with ω in order to achieve this perfect effectiveness.

Nevertheless, if the thickness of the PML region is kept constant with the problem size, the number of required GMRES iterations to solve the global system only grows logarithmically with the wave number ω. It is surprising that this qualitative property holds independently of the thickness of the PML, even for very thin PML regions, for example one wavelength.

This shows that the thickness of the PML has a significant influence on the effectiveness of the preconditioner. On the other hand, the thickness of the PML region is also crucial for the scalability of the preconditioner. For example, if the PML region is chosen to contain $n_{\texttt{pml}}$ discretization points with $n_{\texttt{pml}} \ll n$, the sizes of the local problems associated with each subdomain grows as $O(n_{\texttt{pml}}^2)$ in 2D and as $O((n + n_{\texttt{pml}})n_{\texttt{pml}}^2)$ in 3D. Consequently, the parallel

N	$\omega/2\pi$	p	$T_{\texttt{fact}}$	$N_{\texttt{it}}$	$T_{\texttt{it}}$	$T_{\texttt{total}}$
202×202	20.1	2	1.09	2	0.66	2.63
404×404	40.3	4	1.00	3	0.58	2.56
808×808	80.7	8	1.41	3	1.26	6.02
1616×1616	161.5	16	2.80	2	3.39	14.05
3232×3232	323.1	32	4.41	3	5.47	27.47
6464×6464	646.3	64	8.34	4	11.09	67.74
12928×12928	1292.7	128	15.66	5	22.39	160.88

Table 6: Timings for the solver on a homogeneous wave speed. N is the number of degrees-of-freedom, ω is the frequency, $T_{\texttt{fact}}$ is the time spent to factorize the local problems, $N_{\texttt{it}}$ is the number of GMRES iterations needed to solve the problem, $T_{\texttt{it}}$ is the average time spent per GMRES iteration, and $T_{\texttt{total}}$ is the total time spent to solve the linear system. All timings are measured in seconds.

factorization of all subdomains scales as $O(n^2 n_{\texttt{pml}}^3/p)$ in 2D and $O(n^3 n_{\texttt{pml}}^6/p)$ in 3D and the application of the L-sweep preconditioner scales in parallel as $O(n^2 n_{\texttt{pml}}^2 \log n_{\texttt{pml}}/p)$ in 2D and $O(n^3 n_{\texttt{pml}}^4/p)$ in 3D. Therefore, it is crucial to keep the thickness of the PML region as thin as possible to preserve efficiency. This makes the trade-off between efficiency and effectiveness of the preconditioner apparent. For all subsequent numerical examples considered in this paper, we choose a PML thickness of 2 wavelengths, which is empirically sufficient to achieve satisfactory results. In particular, the logarithmic growth of the number of iterations with respect to ω with this choice of PML-thickness allows one to achieve the advocated parallel scaling of the solver.

We also performed the same experiment for finite difference discretizations with fewer than 10 discretization points per wavelength. For very low accuracy discretizations (e.g., four points per wavelengths and more than 300 wavelengths inside the domain), the preconditioner can fail, i.e., the number of iterations to solver the preconditioned linear system grows faster than $O(\log \omega)$. This is because the preconditioner relies on physical properties of wave propagation which are not captured by such inaccurate discretizations.

Considering these experiments, we conjecture that if the thickness of the PML region is kept constant with the frequency ω and the discretization is sufficiently accurate, the number of iterations required to solve the problem using a GMRES method grows as $O(\log \omega)$. In addition, if the transparent boundary conditions can be modelled perfectly, the iteration count can be reduced to $O(1)$ for reasonably smooth media.

5.2. Complexity of the solver

In this section we provide empirical run-times using the proposed methods to support our scalibility claims.

In this experiment, we consider the standard setup with constant wave speed, for $q = 2, 4, 8, \ldots, 128$ and we measure the factorization time, the time spent in one GMRES iteration, and the total time of the solver, which are summarized in Table 6. All timings are recorded on the NERSC machine 'Cori'. Cori is composed of 2388 Haswell compute nodes, each node containing two 16-core Intel Xeon Processor E5-2698 v3 running at 2.3 GHz and 128 GB of memory. The nodes communicate via a Cray Aries interconnect with Dragonfly topology. The number of MPI ranks are chosen so that each compute node is assigned one MPI rank and each MPI rank is assigned one row of subdomains in the CDD. The number of MPI ranks is therefore $p = q$.

Table 6 contains the results of this experiment, from which we observe the scalability of the preconditioner. The timings clearly reflect the $O(N/p)$ scaling as claimed in the prequel. Furthermore, the number of iterations $N_{\texttt{it}}$ grows as $O(\log \omega)$ which further proves the effectiveness of the preconditioner. Combining both leads to the total time, $T_{\texttt{total}}$, for the solver which reflects the claimed $O((N/p) \log \omega)$ scaling, as illustrated in Figure 27.

5.3. Smooth wave speeds

In this section, we demonstrate the impact of *refracted* waves, induced by smooth media, on the performance of the preconditioner and iterative solver. As established in Section 4, the effectivness of the L-sweeps preconditioner strongly depends on the presence of reflections and refractions in the solution. To evaluate the impact of refracted waves, we consider the standard setup for two different smooth wave speeds, which are shown in Figure 28. One is a random smooth wavefield, the other is a typical background wave speed used in seismic imaging. The latter is obtained by smoothing the background of the BP model [8].

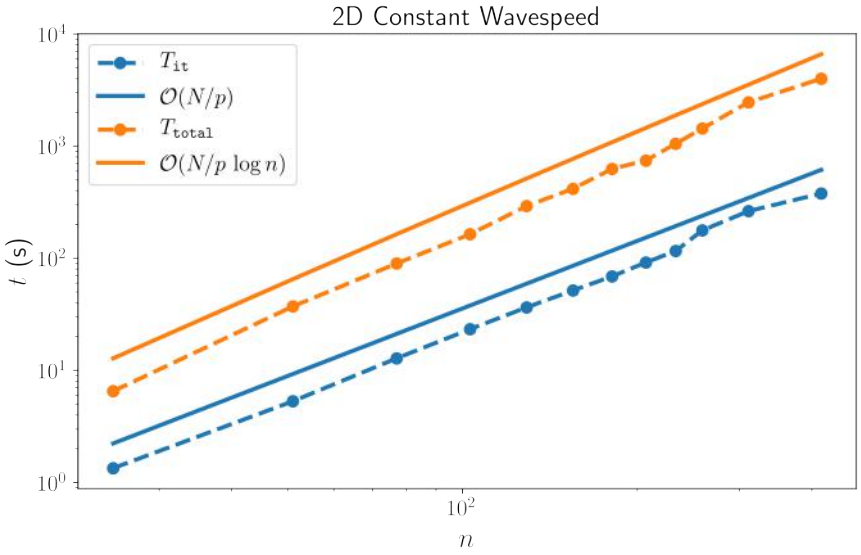


Figure 27: Timings for 2D experiment with constant wavespeed. Solid lines are empirical complexities for one application of the preconditioner (blue) and for the over-all solve (orange). Dashed lines are with measured results for one application of the preconditioner (blue) and for the over-all solve (orange).

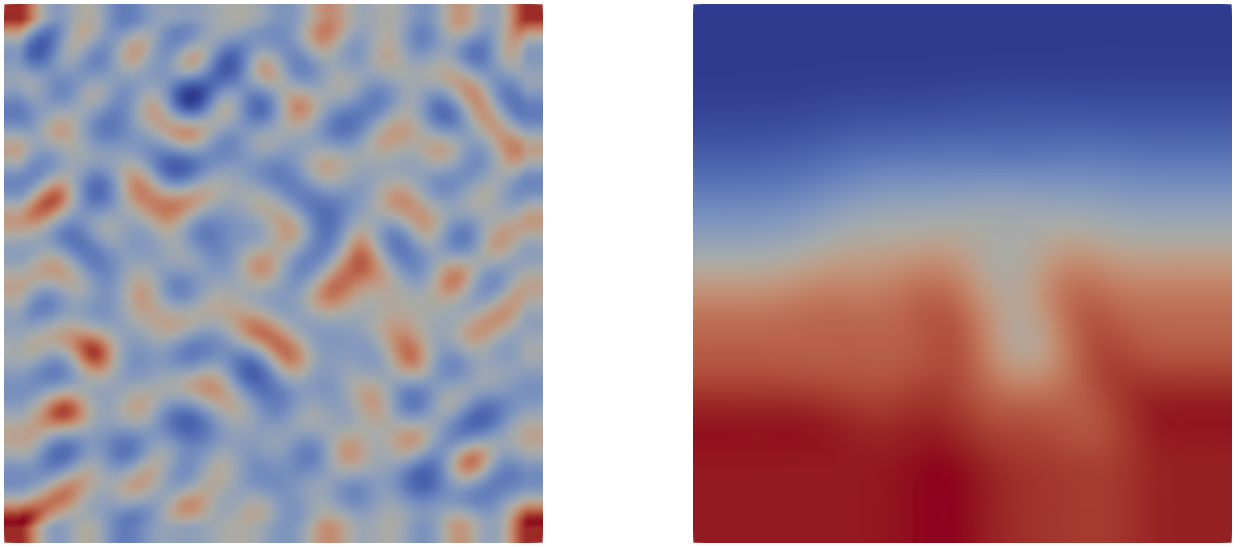


Figure 28: The smooth wave speed distributions. Left: Random smooth wave speed; Right: Smooth BP model background

N (without PML)	$\omega/2\pi$	$q = r$	Random smooth wave speed	Smooth BP model background
202×202	20.1	2	3	1
404×404	40.3	4	3	2
808×808	80.7	8	5	4
1616×1616	161.5	16	7	5
3232×3232	323.1	32	9	6
6464×6464	646.3	64	10	7
12928×12928	1292.7	128	12	8

Table 7: Iteration counts for smooth wave speeds for different frequencies.

N (without PML)	$\omega/2\pi$	$q = r$	BG1	BG2	BG1 with salt	BG2 with salt	BP model
202×202	20.1	2	1	4	7	6	7
404×404	40.3	4	2	4	9	9	9
808×808	80.7	8	4	6	12	12	12
1616×1616	161.4	16	5	6	15	15	15
3232×3232	323.1	32	6	7	17	17	16
6464×6464	646.3	64	7	7	19	19	19
12928×12928	1292.7	128	8	8	21	21	20

Table 8: Iteration counts for the BP model.

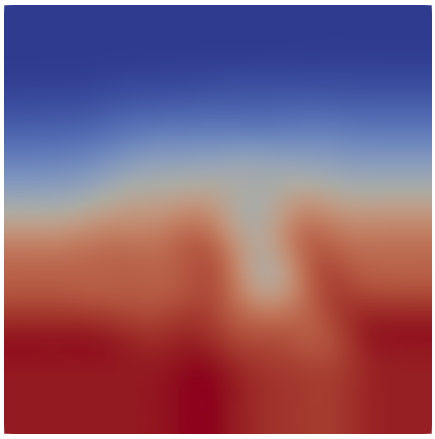
Table 7 shows the iteration counts obtained for both wave speeds. Table 7 shows the effectiveness of the preconditioner when solving problems involving a smooth wave speed. In particular, we observe that the number of iterations scales as $O(\log \omega)$, albeit with higher constants compared to the constant case.

5.4. BP model

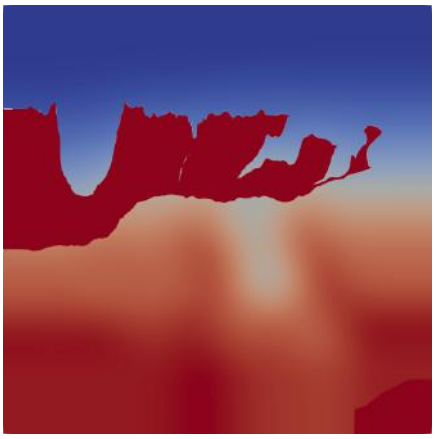
In this section, we consider the wave speed of a standard geophysical benchmark problem, a subset of the BP model [8]. It is well-known that this problem involves many reflections in the resulting wavefields. We therefore use it to study the effect of reflections on the L-sweeps preconditioner.

First, we establish that reflected waves arise primarily due to the salt body, the dark red region in Figures 29b, 29d, and 29e. To this end, we first consider two smooth background wave speeds, the one from Section 5.3, which we will call BG1 and illustrate in Figure 29a and the true background velocity provided in [8], which we call BG2 and illustrate in Figure 29c. Then, we superimpose the salt body on BG1 and BG2, building to examine the effect of the high contrasts due to the salt body on the performance of the solver. Finally, for completeness, we also run the test on the true BP model, as obtained from [8], illustrated in Figure 29e.

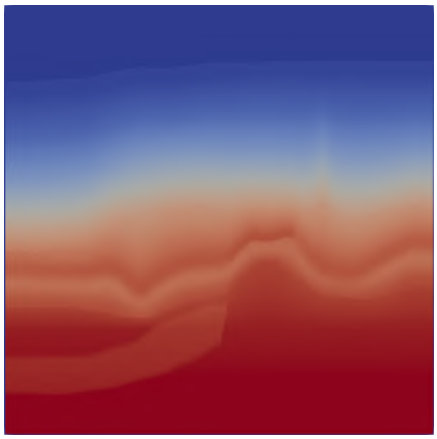
Table 8 shows the iteration counts observed for each of the 5 experiments. For both BG1 and BG2, with no salt, we observe similar iteration counts and conclude that reflections do not play a big role in the resulting wavefields. However, after superimposing salt body on top of of BG1 and BG2, we clearly see that the number of iterations increase significantly. This is explained because reflections are introduced in the wavefield due to the high contrast between the salt and the background. We also observe that the exact BP model can be solved in almost the same iteration counts as BG1 and BG2 with the salt body superimposed. Thus, even for the exact BP model, the reflections induced by the salt body dominate the performance of the preconditioner. Finally, let us note that even though reflections in the wavefield result in higher iteration counts, with increasing frequency ω, the number of iterations always grows only as $O(\log \omega)$ and they are never higher than 21 iterations, even for problems involving 1000 wavelengths inside the domain. This shows the effectiveness of the preconditioner, even in the presence of reflections as they appear in practically relevant problems. The wave field computed for the BP model and $\omega/2\pi = 323.1$ is shown in Figure 30.



(a) The smooth background model (BG1).

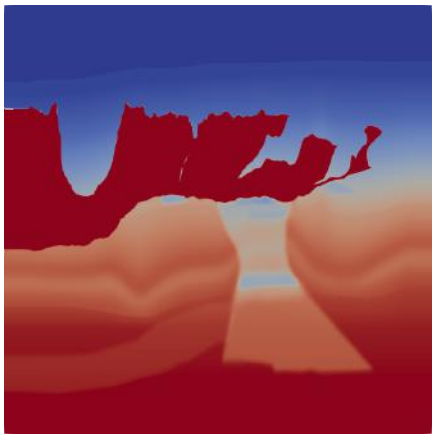


(b) BG1, with salt body.



(c) The 'true' BP model background velocity (BG2).

(d) BG2 with salt body.



(e) The 'true' BP model.

Figure 29: Wave speed distributions for the study of the BP model.

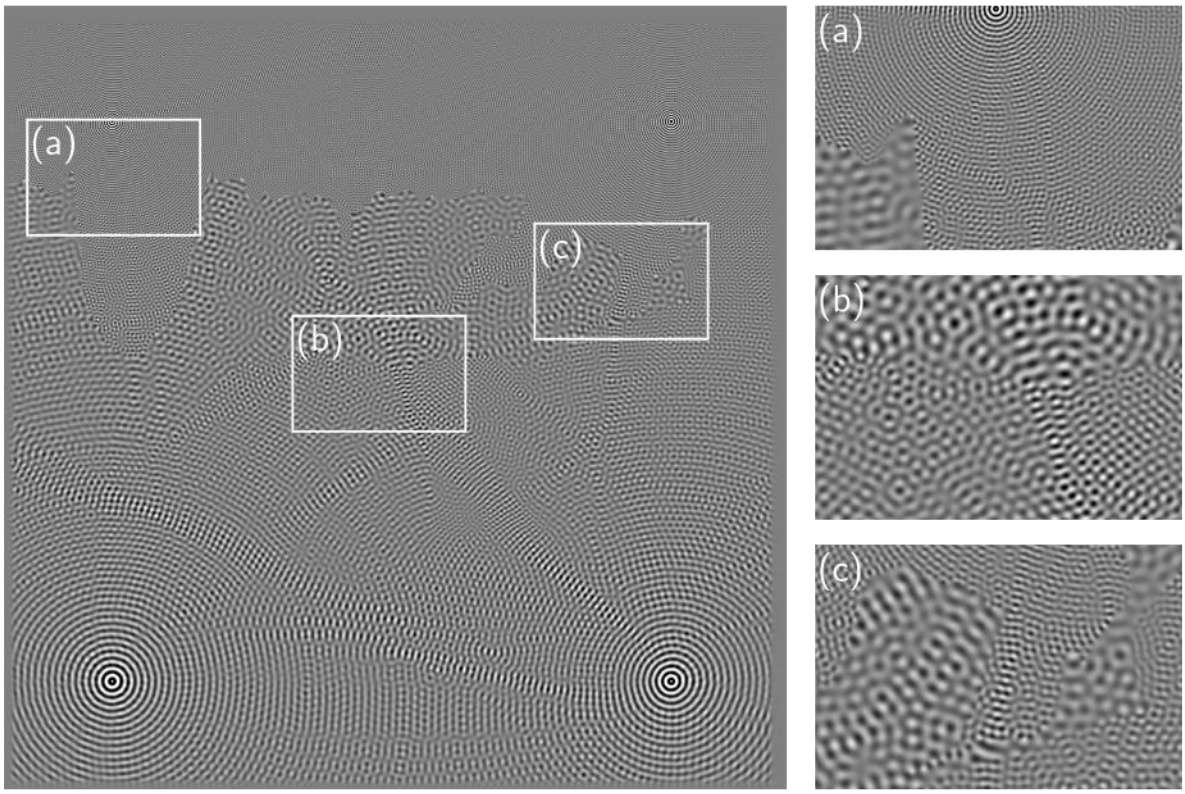


Figure 30: The computed wave field for the BP model and $\omega/2\pi = 323.1$. Zoom regions (a), (b), and (c) highlight solution quality in areas of high-contrast in the wavespeed (at the salt boundary).

N (without PML)	$\omega/2\pi$	$m = n$	Contrast ratio 2	3	4	5	6
202×202	20.1	2	18	24	24	25	26
404×404	40.3	4	28	29	29	28	30
808×808	80.7	8	30	32	34	33	33
1616×1616	161.5	16	31	33	33	34	35
3232×3232	323.1	32	32	34	36	36	37
6464×6464	646.3	64	32	34	35	36	36

Table 9: Iteration counts for wave-guide problem for different contrast ratios of the wave speed between the wave-guide and the background.

5.5. Wave-guide

To conclude the studies of two-dimensional problems, we consider a wave-guide with a point source at the entrance, a problem that is designed to stress the capabilities of the preconditioner because solutions contain many reflections. The wave-guide is illustrated in Figure 31. Other than the change in the source and velocity configuration, the experimental setup follows the standard setup.

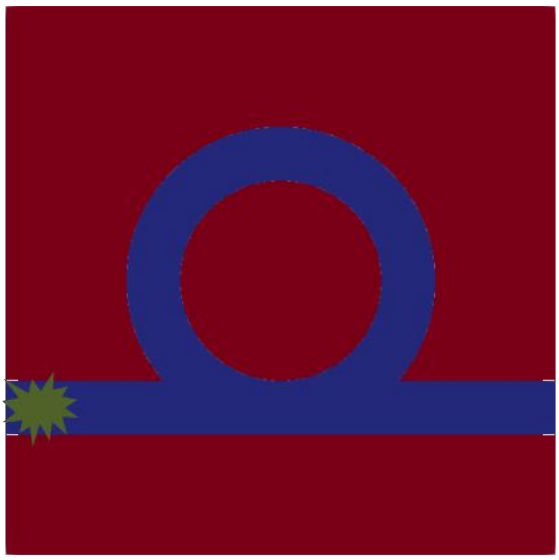


Figure 31: The wave speed distribution of the wave-guide. The point source at the entrance is shown by the green star.

To stress the preconditioner, we vary the contrast ratio between the background and wave-guide. Table 9 shows the results of the experiment. Once all reflections are resolved for each problem, the number of iterations grows logarithmically with the frequency ω. In fact, the growth is so slow that it is comparable to the problem involving constant wave speeds. In addition, while the number of iterations is clearly higher for this problem, it is interesting to observe that all problems can still be solved in under 40 iterations. It is noteworthy that the number of iterations is apparently independent the contrast ratio. The wave field computed for a contrast ratio of 6 and $\omega/2\pi = 323.1$ is shown in Figure 32.

5.6. 3D Example

As a final example, we consider a three-dimensional problem. The effectiveness of the preconditioner has been thoroughly established in the previous two-dimensional numerical examples. These results directly translate over to three-dimensional problems. Therefore, we only consider the scalability of the solver for a three-dimensional problem for a constant wave speed.

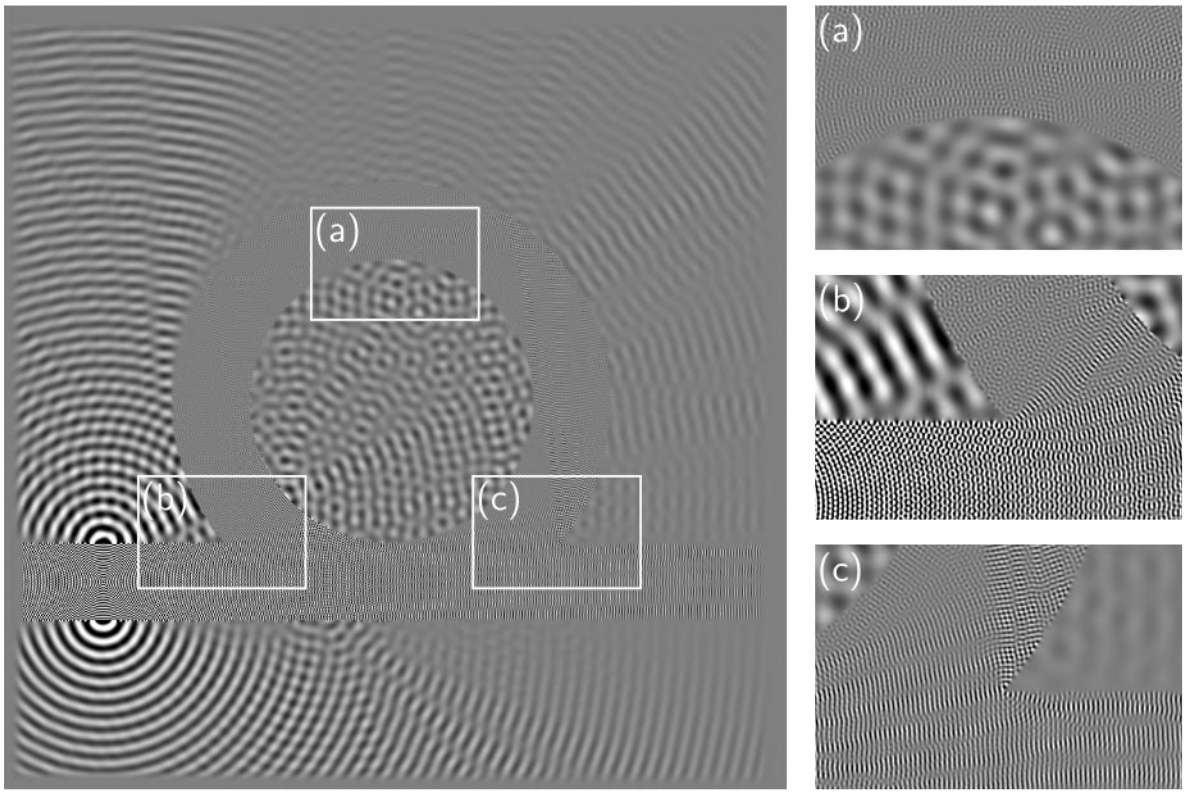


Figure 32: The computed wave field for a contrast ratio of 6 and $\omega/2\pi = 323.1$. Zoom regions (a), (b), and (c) highlight solution quality in areas of high-contrast in the wavespeed.

N (without PML)	$\omega/2\pi$	p	$T_{\texttt{fact}}$	$N_{\texttt{it}}$	$T_{\texttt{it}}$	$T_{\texttt{total}}$
$26 \times 26 \times 26$	4.17	2	.04	4	1.34	6.52
$52 \times 52 \times 52$	8.50	4	5.54	6	5.30	37.17
$78 \times 78 \times 78$	12.83	6	12.42	6	12.80	89.76
$104 \times 104 \times 104$	17.17	8	22.91	6	23.27	163.62
$13 - \times 130 \times 130$	21.50	10	37.53	7	36.47	292.33
$156 \times 156 \times 156$	25.83	12	52.47	7	51.62	417.08
$182 \times 182 \times 182$	30.17	14	71.71	8	68.92	627.23
$208 \times 208 \times 208$	34.50	16	96.14	7	91.65	743.37
$234 \times 234 \times 234$	38.83	18	124.64	8	116.08	1050.31
$260 \times 260 \times 260$	43.17	20	211.87	7	177.21	1438.12
$312 \times 312 \times 312$	51.83	24	314.93	8	263.16	2457.40
$416 \times 416 \times 416$	69.17	32	418.36	9	377.60	3992.63

Table 10: Timings for the solver on a 3D homogeneous wave speed. N is the number of degrees-of-freedom, ω is the frequency, $T_{\texttt{fact}}$ is the time spent to factorize the local problems, $N_{\texttt{it}}$ is the number of GMRES iterations needed to achieve a relative residual of 10^{-7}, $T_{\texttt{it}}$ is the average time spent per GMRES iteration, and $T_{\texttt{total}}$ is the total time spent to solve the linear system. All timings are measured in seconds.

To this end, we consider the standard setup in a 3D setting. This means that instead of constantly sized two-dimensional problems, the local problems in each subdomain are beam-shaped three-dimensional problems. These beam shaped local problems are obtained from the local 2D problems by extending them in the third dimension such that the global problem has n degrees of freedom in each direction where, as before, $q = r$ is the number of subdomains in one direction of the CDD. The only other differences in problem setup are that we consider a discretization of 6 points per wavelength, assume that each subdomain contain 2 wavelengths, and choose a PML thickness of only one wavelength. Not counting the PML region, this results in problems with $n = 13q$ degrees of freedom in one direction. All of these changes are for computational expedience.

The timings are measured in the same computational environment as in Section 5.2. As before, one row of subdomains is assigned to one node, i.e., $p = q$. Table 10 shows the resulting time $T_{\texttt{fact}}$ needed for the factorization of all the local problems, the number of iterations $N_{\texttt{it}}$ required in the GMRES method, the average time $T_{\texttt{it}}$ required in each iteration, and the total time $T_{\texttt{total}}$ required for the solver.

It can be clearly seen from Table 2 that $T_{\texttt{fact}}$ and $T_{\texttt{it}}$ both show the claimed $O(N/p)$ scaling. In addition, as in the two-dimensional cases, the number of iterations $N_{\texttt{it}}$ grows as $O(\log \omega)$ resulting in the almost optimal $O((N/p) \log \omega)$ scaling for the total time $T_{\texttt{total}}$ needed for the solver as shown in Figure 33. This corroborates the claimed complexity for three-dimensional problems.

6. Discussion

We have introduced the first solver for the high-frequency Helmholtz equation that scales as $O((N/p) \log \omega)$ a distributed-memory parallel computational environment, for a single right-hand side. The new solver constructs a preconditioner based on a CDD. This approach reveals parallelism which is exploited to obtain the optimal parallel complexity. The performance of the preconditioner is similar to the well-established method of polarized traces, i.e., the number of iterations grows as $O(\log \omega)$.

We have shown that the preconditioner is effective once all reflections and refractions in the wavefield are resolved. While this is feasible for many practical applications, the performance of the proposed solver deteriorates in the presence of excessive reflected or refracted waves. This limitation arises, for example, on problems containing resonant cavities. We do not think that sweeping strategies are the correct approach to solve these problems, and due to the relevance of these problems, we view this case as a topic for further research.

Future work of the current method includes improvements for three-dimensional problems. As it is presented in this paper, the method results in a $O(n^2) = O(N^{\frac{2}{3}})$ parallel complexity for three-dimensional problems. However, all beam-shaped local problems are treated by one MPI rank only. It is well established that these quasi-one-dimensional

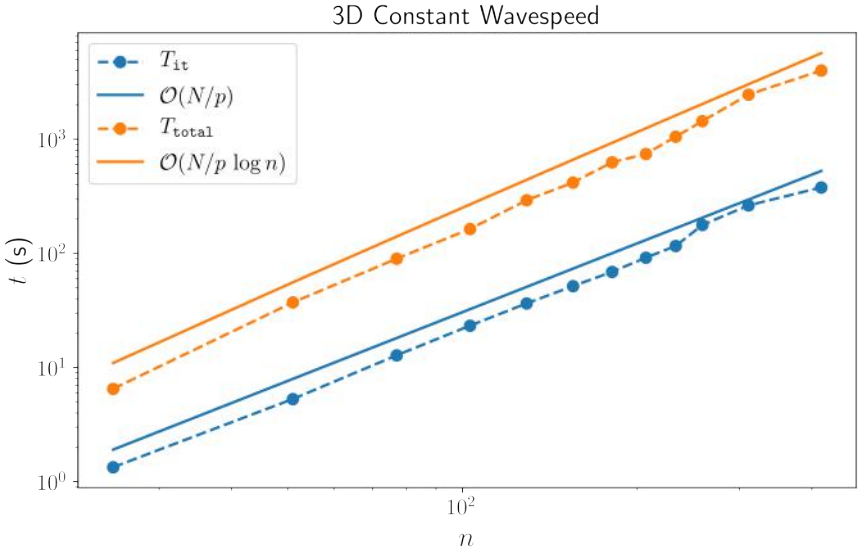


Figure 33: Timings for the 3D experiment with constant wavespeed. Solid lines are empirical complexities for one application of the preconditioner (blue) and for the over-all solve (orange). Dashed lines are with measured results for one application of the preconditioner (blue) and for the over-all solve (orange).

problems can be treated in a distributed memory parallel computational environment by the use of multi-frontal methods [64, 2, 50]. In this case, further parallelism can be exploited and an almost $O(n) = O(N^{\frac{1}{3}})$ parallel complexity can be achieved for three-dimensional problems.

Using this extension, it can be shown that the high-frequency Helmholtz equation can be optimally parallelized in all spatial directions but one. The remaining direction is due to the inherently serial nature of the sweeps. In this regard, we conjecture that this work achieves the boundary of the scalability of sweeping preconditioners for a single right-hand side and that further improvements require new ideas. Nevertheless, in the presence of $O(n)$ right-hand sides, sweeping preconditioners can be further parallelized by pipelining over the right-hand sides. In that regard, the parallel complexity can be further improved to an average $O(1)$, up to logarithmic factors, parallel complexity per right-hand side. All of these extensions are currently under investigation.

Finally, while we only consider a row-based processor assignment in this work, in highly heterogeneous parallel computational environments other processor assignments, in particular processor assignments tailored to the heterogeneous computational environment, can be applied straightforwardly with our approach. This may lead to further improvements of the efficiency of the algorithm. An investigation of these aspects would also be very interesting.

A. Perfectly Matched Layers (PMLs) and Finite Difference Discretizations

For a given squared slowness m and source density f defined on $\Omega_{\texttt{extended}}$, following the technique of PMLs, a new complex-valued squared slowness and source density can be derived so that the PMLs act as absorbing boundary conditions. The coefficients in $\Omega_{\texttt{extended}}$ can be derived from the variable transformation [46, 7]

$$\frac{\partial}{\partial x_k} \to \alpha_k(x) \frac{\partial}{\partial x_k}$$

for $k = 1, .., d$ where α_k is the complex-valued function

$$\alpha_k(x) = \frac{1}{1 + i\frac{\sigma_k(x)}{\omega}},$$

where $i = \sqrt{-1}$ is the imaginary unit and σ_k is the PML profile function.

To describe the PML profile, assume that $\Omega_{\texttt{bulk}}$ is the unit square or the unit cube. Then the extended domain $\Omega_{\texttt{extended}}$ is the square ($d = 2$) or cube ($d = 3$), $[-\delta_{\texttt{pml}}, 1 + \delta_{\texttt{pml}}]^d$, with the PML-width $\delta_{\texttt{pml}} > 0$. The PML profile σ_k is then chosen to be the cubic function

$$\sigma_k(x) := \begin{cases} \frac{C}{\delta_{\texttt{pml}}} \left(\frac{-x_k}{\delta_{\texttt{pml}}}\right)^3 & \text{for } x_k < 0 \\ 0 & \text{for } 0 \le x_k \le 1 \\ \frac{C}{\delta_{\texttt{pml}}} \left(\frac{x_k - 1}{\delta_{\texttt{pml}}}\right)^3 & \text{for } x_k > 1 \end{cases}$$

where $C > 0$ is the absorption constant chosen to be $C = \ln \omega$. Employing this variable transformation in equation (2) gives rise to the diagonal matrix Λ with diagonal entries α_k^2/β, and the the complex-valued squared slowness and source density can be written as

$$\frac{m}{\prod_{k=1}^d \alpha_k}, \quad \text{and} \quad \frac{f}{\prod_{k=1}^d \alpha_k},$$

respectively.

The resulting system is usually called the symmetric formulation of the Helmholtz equation, which in 2D takes the form:

$$-\left(\nabla \cdot \Lambda \nabla + \frac{\omega^2 m(\mathbf{x})}{\alpha_1(\mathbf{x})\alpha_2(\mathbf{x})}\right) u(\mathbf{x}) = \frac{f(\mathbf{x})}{\alpha_1(\mathbf{x})\alpha_2(\mathbf{x})}, \tag{16}$$

and

$$\Lambda(\mathbf{x}) = \begin{bmatrix} s_1(\mathbf{x}) & 0 \\ 0 & s_2(\mathbf{x}) \end{bmatrix}, \tag{17}$$

$s_1 = \alpha_1/\alpha_2$, $s_2 = \alpha_2/\alpha_1$, with homogeneous Dirichlet boundary conditions.

We discretize $\Omega_{\texttt{bulk}}$ as an equispaced regular grid of stepsize h, and of dimensions $n_x \times n_y$. For the extended domain $\Omega_{\texttt{extended}}$, we extend this grid by $n_{\texttt{pml}} = \delta_{\texttt{pml}}/h$ points in each direction, obtaining a grid of size $(2n_{\texttt{pml}} + n_x) \times (2n_{\texttt{pml}} + n_z)$. Define $\mathbf{x}_{p,q} = (x_p, y_q) = (ph, qh)$. We use the 5-point stencil Laplacian to discretize (16). For the interior points $\mathbf{x}_{i,j} \in \Omega$, we have

$$(\mathbf{Au})_{p,q} = -\frac{1}{h^2}\left(\mathbf{u}_{p-1,q} - 2\mathbf{u}_{p,q} + \mathbf{u}_{p+1,q}\right) - \frac{1}{h^2}\left(\mathbf{u}_{p,q-1} - 2\mathbf{u}_{p,q} + \mathbf{u}_{p,q+1}\right) - \omega^2 \frac{m(\mathbf{x}_{p,q})}{s_1(\mathbf{x}_{p,q}) s_2(\mathbf{x}_{p,q})}. \tag{18}$$

In the PML, we discretize $\partial_x(s_1 \partial_x u)$ as

$$\frac{s_1(\mathbf{x}_{p+1/2,q})(\mathbf{u}_{p+1,q} - \mathbf{u}_{p,q}) - s_1(\mathbf{x}_{p-1/2,q})(\mathbf{u}_{p,q} - \mathbf{u}_{p-1,q})}{h^2}, \tag{19}$$

and analogously for $\partial_y(s_2 \partial_y u)$.

B. Proof of the annihilation condition

Let $x \in \Omega_1$, then using the definition of U and the fact that u and $G(x, .)$ vanish on $\partial\Omega$, it holds

$$\begin{aligned} U(x) &= \int_\Gamma \Lambda(y)\nabla u(y) G(x,y) ds_y - \int_\Gamma u(y) \left[n(y) \cdot \left(\Lambda(y)\nabla_y G(x,y)\right)\right] ds_y \\ &= \int_{\partial\Omega_2} \Lambda(y)\nabla u(y) G(x,y) ds_y - \int_{\partial\Omega_2} u(y) \left[n(y) \cdot \left(\Lambda(y)\nabla_y G(x,y)\right)\right] ds_y. \end{aligned}$$

Then, employing the Divergence Theorem yields

$$\begin{aligned} U(x) &= \int_{\Omega_2} u(y) \left[-\operatorname{div}_y \left(\Lambda(y)\nabla_y G(x,y)\right)\right] dy - \int_{\Omega_2} \left[-\operatorname{div}\left(\Lambda(y)\nabla u(y)\right)\right] G(x,y) dy \\ &= \int_{\Omega_2} u(y) \underbrace{\left[-\operatorname{div}_y \left(\Lambda(y)\nabla_y G(x,y)\right) - m(y)G(x,y)\right]}_{=0} dy - \int_{\Omega_2} \underbrace{\left[-\operatorname{div}\left(\Lambda(y)\nabla u(y)\right) - m(y)u(y)\right]}_{=0} G(x,y) dy \end{aligned}$$

proving the Annihilation condition (6).

C. Pseudocode

C.1. Core algorithms

Algorithm 1 Extraction of traces

function EXTRACTTRACES(u, i, j)
  Define the Dirichlet traces λ^B, λ^R, λ^T, λ^L and set them to zero
  Define the Neumann traces μ^B, μ^R, μ^T, μ^L and set them to zero
  if $i > 1$ **then**
    Extract the traces λ^B and μ^B from u according to (10)
  if $i < m$ **then**
    Extract the traces λ^T and μ^T from u according to (10)
  if $j > 1$ **then**
    Extract the traces λ^L and μ^L from u according to (10)
  if $j < n$ **then**
    Extract the traces λ^R and μ^R from u according to (10)
  return λ^B, λ^R, λ^T, λ^L, μ^B, μ^R, μ^T, μ^L

Algorithm 2 Computation of a local solution

function COMPUTELOCSOL(f, i, j)
  Define f_{ij} for any $x \in \Omega_{ij}^{\varepsilon}$ such that

$$f_{ij}(x) := \begin{cases} f(x) & x \in \Omega_{ij} \\ 0 & \text{otherwise} \end{cases}$$

  Solve the local problem on $\Omega_{ij}^{\varepsilon}$ for f_{ij} and obtain the solution u_{ij}
  return u_{ij}

Algorithm 3 Computation of a local polarized wavefield

function COMPUTEPOLSOL(λ, μ, Γ, i, j)
  With λ, μ and Γ, compute u_{ij} in $\Omega_{ij}^{\varepsilon}$ using (4)
  return u_{ij}

C.2. Algorithm for the local solutions (stage 1)

Algorithm 4 Compute all local solutions

function addLocSols($\boldsymbol{u}$, f)
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^L$ of left Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^L$ of left Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^R$ of right Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^R$ of right Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^B$ of bottom Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^B$ of bottom Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^T$ of top Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^T$ of top Neumann traces and set them to zero
  for $i = 1, \ldots, m$ **do**
    for $j = 1, \ldots, n$ **do**
      $\boldsymbol{u}[i, j]$ = computeLocSol(f, i, j)
      $(\lambda^{B,\mathrm{loc}}, \lambda^{R,\mathrm{loc}}, \lambda^{T,\mathrm{loc}}, \lambda^{L,\mathrm{loc}}, \mu^{B,\mathrm{loc}}, \mu^{R,\mathrm{loc}}, \mu^{T,\mathrm{loc}}, \mu^{L,\mathrm{loc}})$ = extractTraces($\boldsymbol{u}[i, j]$, i, j)
      Set the bottom Dirichlet trace $\boldsymbol{\lambda}^B[i, j] = \lambda^{B,\mathrm{loc}}$
      Set the right Dirichlet trace $\boldsymbol{\lambda}^R[i, j] = \lambda^{R,\mathrm{loc}}$
      Set the top Dirichlet trace $\boldsymbol{\lambda}^T[i, j] = \lambda^{T,\mathrm{loc}}$
      Set the left Dirichlet trace $\boldsymbol{\lambda}^L[i, j] = \lambda^{L,\mathrm{loc}}$
      Set the bottom Neumann trace $\boldsymbol{\mu}^B[i, j] = \mu^{B,\mathrm{loc}}$
      Set the right Neumann trace $\boldsymbol{\mu}^R[i, j] = \mu^{R,\mathrm{loc}}$
      Set the top Neumann trace $\boldsymbol{\mu}^T[i, j] = \mu^{T,\mathrm{loc}}$
      Set the left Neumann trace $\boldsymbol{\mu}^L[i, j] = \mu^{L,\mathrm{loc}}$
  return $\boldsymbol{u}$, $\boldsymbol{\lambda}^B$, $\boldsymbol{\lambda}^R$, $\boldsymbol{\lambda}^T$, $\boldsymbol{\lambda}^L$, $\boldsymbol{\mu}^B$, $\boldsymbol{\mu}^R$, $\boldsymbol{\mu}^T$, $\boldsymbol{\mu}^L$

C.3. Algorithms for the horizontal and vertical sweeps (stage 2)

Algorithm 5 Global wavefield from the up sweeps

function addUpSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^T$, $\boldsymbol{\mu}^T$)
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^L$ of left Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^L$ of left Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^R$ of right Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^R$ of right Neumann traces and set them to zero
  for $d = 1 \cdots + n$ **do**
    for $i = 1 \cdots$ **do**
      $j = d - i + 1$
      if $i > 1$ **then**
        u^{loc} = computePolSol($\boldsymbol{\lambda}^T[i-1, j]$, $\boldsymbol{\mu}^T[i-1, j]$, Γ^B_{ij}, i, j)
        $(\lambda^{B,\mathrm{loc}}, \lambda^{R,\mathrm{loc}}, \lambda^{T,\mathrm{loc}}, \lambda^{L,\mathrm{loc}}, \mu^{B,\mathrm{loc}}, \mu^{R,\mathrm{loc}}, \mu^{T,\mathrm{loc}}, \mu^{L,\mathrm{loc}})$ = extractTraces(u^{loc}, i, j)
        Update wave field $\boldsymbol{u}[i, j]$ += u^{loc}
        Update top Dirichlet trace $\boldsymbol{\lambda}^T[i, j]$ += $\lambda^{T,\mathrm{loc}}$
        Update top Neumann trace $\boldsymbol{\mu}^T[i, j]$ += $\mu^{T,\mathrm{loc}}$
        Set right Dirichlet trace $\boldsymbol{\lambda}^R[i, j] = \lambda^{R,\mathrm{loc}}$
        Set right Neumann trace $\boldsymbol{\mu}^R[i, j] = \mu^{R,\mathrm{loc}}$
        Set left Dirichlet trace $\boldsymbol{\lambda}^L[i, j] = \lambda^{L,\mathrm{loc}}$
        Set left Neumann trace $\boldsymbol{\mu}^L[i, j] = \mu^{L,\mathrm{loc}}$
  return $\boldsymbol{u}$, $\boldsymbol{\lambda}^R$, $\boldsymbol{\lambda}^L$, $\boldsymbol{\mu}^R$, $\boldsymbol{\mu}^L$

Algorithm 6 Global wavefield from the down sweeps

function addDownSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^B$, $\boldsymbol{\mu}^B$)
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^L$ of left Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^L$ of left Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^R$ of right Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^R$ of right Neumann traces and set them to zero
  for $d = 1, \ldots, m + n$ **do**
    for $i = 1, \ldots, m$ **do**
      $j = d - i + 1$
      $i = m - i + 1$
      $j = n - j + 1$
      if $i < n$ **then**
        $u^{\mathtt{loc}}$ = computePolSol($\boldsymbol{\lambda}^B[i+1, j]$, $\boldsymbol{\mu}^B[i+1, j]$, Γ^T_{ij}, i, j)
        $(\lambda^{B,\mathtt{loc}}, \lambda^{R,\mathtt{loc}}, \lambda^{T,\mathtt{loc}}, \lambda^{L,\mathtt{loc}}, \mu^{B,\mathtt{loc}}, \mu^{R,\mathtt{loc}}, \mu^{T,\mathtt{loc}}, \mu^{L,\mathtt{loc}})$ = extractTraces($u^{\mathtt{loc}}$, i, j)
        Update wave field $\boldsymbol{u}[i, j]$ += $u^{\mathtt{loc}}$
        Update bottom Dirichlet trace $\boldsymbol{\lambda}^B[i, j]$ += $\lambda^{B,\mathtt{loc}}$
        Update bottom Neumann trace $\boldsymbol{\mu}^B[i, j]$ += $\mu^{B,\mathtt{loc}}$
        Set right Dirichlet trace $\boldsymbol{\lambda}^R[i, j] = \lambda^{R,\mathtt{loc}}$
        Set right Neumann trace $\boldsymbol{\mu}^R[i, j] = \mu^{R,\mathtt{loc}}$
        Set left Dirichlet trace $\boldsymbol{\lambda}^L[i, j] = \lambda^{L,\mathtt{loc}}$
        Set left Neumann trace $\boldsymbol{\mu}^L[i, j] = \mu^{L,\mathtt{loc}}$
  return $\boldsymbol{u}$, $\boldsymbol{\lambda}^R$, $\boldsymbol{\lambda}^L$, $\boldsymbol{\mu}^R$, $\boldsymbol{\mu}^L$

Algorithm 7 Global wavefield from the right sweeps

function addRightSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^R$, $\boldsymbol{\mu}^R$)
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^B$ of bottom Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^B$ of bottom Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^T$ of top Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^T$ of top Neumann traces and set them to zero
  for $i = 1, \ldots, m$ **do**
    for $j = 2, \ldots, n$ **do**
      $u^{\mathtt{loc}}$ = computePolSol($\lambda^R[i, j-1]$, $\mu^R[i, j-1]$, Γ^L_{ij}, i, j)
      $(\lambda^{B,\mathtt{loc}}, \lambda^{R,\mathtt{loc}}, \lambda^{T,\mathtt{loc}}, \lambda^{L,\mathtt{loc}}, \mu^{B,\mathtt{loc}}, \mu^{R,\mathtt{loc}}, \mu^{T,\mathtt{loc}}, \mu^{L,\mathtt{loc}})$ = extractTraces($u^{\mathtt{loc}}$, i, j)
      Update wave field $\boldsymbol{u}[i, j]$ += $u^{\mathtt{loc}}$
      Update right Dirichlet trace $\boldsymbol{\lambda}^R[i, j]$ += $\lambda^{R,\mathtt{loc}}$
      Update right Neumann trace $\boldsymbol{\mu}^R[i, j]$ += $\mu^{R,\mathtt{loc}}$
      Set bottom Dirichlet trace $\boldsymbol{\lambda}^B[i, j] = \lambda^{B,\mathtt{loc}}$
      Set bottom Neumann trace $\boldsymbol{\mu}^B[i, j] = \mu^{B,\mathtt{loc}}$
      Set top Dirichlet trace $\boldsymbol{\lambda}^T[i, j] = \lambda^{T,\mathtt{loc}}$
      Set top Neumann trace $\boldsymbol{\mu}^T[i, j] = \mu^{T,\mathtt{loc}}$
  return $\boldsymbol{u}$, $\boldsymbol{\lambda}^B$, $\boldsymbol{\lambda}^T$, $\boldsymbol{\mu}^B$, $\boldsymbol{\mu}^T$

Algorithm 8 Global wavefield from an left sweep in row i

function addLeftSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^L$, $\boldsymbol{\mu}^L$)
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^B$ of bottom Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^B$ of bottom Neumann traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\lambda}^T$ of top Dirichlet traces and set them to zero
  Define an $m \times n$ matrix $\boldsymbol{\mu}^T$ of top Neumann traces and set them to zero
  for $i = 1, \ldots, m$ **do**
    for $j = n-1, \ldots, 1$ **do**
      u^{loc} = computePolSol($\lambda^L[i, j+1]$, $\mu^L[i, j+1]$, Γ^R_{ij}, i, j)
      ($\lambda^{B,\text{loc}}$, $\lambda^{R,\text{loc}}$, $\lambda^{T,\text{loc}}$, $\lambda^{L,\text{loc}}$, $\mu^{B,\text{loc}}$, $\mu^{R,\text{loc}}$, $\mu^{T,\text{loc}}$, $\mu^{L,\text{loc}}$) = extractTraces(u^{loc}, i, j)
      Update wave field $\boldsymbol{u}[i, j]$ += u^{loc}
      Update left Dirichlet trace $\boldsymbol{\lambda}^L[i, j]$ += $\lambda^{L,\text{loc}}$
      Update left Neumann trace $\boldsymbol{\mu}^L[i, j]$ += $\mu^{L,\text{loc}}$
      Set bottom Dirichlet trace $\boldsymbol{\lambda}^B[i, j] = \lambda^{B,\text{loc}}$
      Set bottom Neumann trace $\boldsymbol{\mu}^B[i, j] = \mu^{B,\text{loc}}$
      Set top Dirichlet trace $\boldsymbol{\lambda}^T[i, j] = \lambda^{T,\text{loc}}$
      Set top Neumann trace $\boldsymbol{\mu}^T[i, j] = \mu^{T,\text{loc}}$
  return $\boldsymbol{u}$, $\boldsymbol{\lambda}^B$, $\boldsymbol{\lambda}^T$, $\boldsymbol{\mu}^B$, $\boldsymbol{\mu}^T$

C.4. Algorithms for diagonal sweeps (stage 3)

Algorithm 9 Global wavefield from a diagonal sweep from the bottom-left to the top-right corner

function addBL2TRSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^L$, $\boldsymbol{\lambda}^B$, $\boldsymbol{\mu}^L$, $\boldsymbol{\mu}^B$)
  for $d = 1, \ldots, m+n$ **do**
    for $i = 1, \ldots, m$ **do**
      if $i > 1$ and $j > 1$ **then**
        Define the L-shaped line $\Gamma^{BL}$by combining Γ^B_{ij} and Γ^L_{ij}
        Define $\lambda^B := \boldsymbol{\lambda}^T[i-1, j]$ and $\mu^B := \boldsymbol{\mu}^T[i-1, j]$
        Define $\lambda^L := \boldsymbol{\lambda}^R[i, j-1]$ and $\mu^L := \boldsymbol{\mu}^R[i, j-1]$
        Define λ^{BL} and μ^{BL} similar to (12) and (13).
        u^{loc} = computePolSol(λ^{BL}, μ^{BL}, Γ^{BL}, i, j)
        ($\lambda^{B,\text{loc}}$, $\lambda^{R,\text{loc}}$, $\lambda^{T,\text{loc}}$, $\lambda^{L,\text{loc}}$, $\mu^{B,\text{loc}}$, $\mu^{R,\text{loc}}$, $\mu^{T,\text{loc}}$, $\mu^{L,\text{loc}}$) = extractTraces(u^{loc}, i, j)
        Update wave field $\boldsymbol{u}[i, j]$ += u^{loc}
        Update right Dirichlet trace $\boldsymbol{\lambda}^R[i, j]$ += $\lambda^{R,\text{loc}}$
        Update right Neumann trace $\boldsymbol{\mu}^R[i, j]$ += $\mu^{R,\text{loc}}$
        Update top Dirichlet trace $\boldsymbol{\lambda}^T[i, j]$ += $\lambda^{T,\text{loc}}$
        Update top Neumann trace $\boldsymbol{\mu}^T[i, j]$ += $\mu^{T,\text{loc}}$
  return $\boldsymbol{u}$

Algorithm 10 Global wavefield from a diagonal sweep from the top-right to the bottom-left corner

function addTR2BLSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^R$, $\boldsymbol{\lambda}^T$, $\boldsymbol{\mu}^R$, $\boldsymbol{\mu}^T$)
  for $d = 1, \ldots, m+n$ **do**
    for $i = m, \ldots, 1$ **do**
      $j = n + m - 1 - (d + i)$
      if $i < n$ and $j < n$ **then**
        Define the L-shaped line $\Gamma^{TR}$by combining Γ^T_{ij} and Γ^R_{ij}
        Define $\lambda^T := \boldsymbol{\lambda}^B[i+1, j]$ and $\mu^T := \boldsymbol{\mu}^B[i+1, j]$
        Define $\lambda^R := \boldsymbol{\lambda}^L[i, j+1]$ and $\mu^R := \boldsymbol{\mu}^L[i, j+1]$
        Define λ^{TR} and μ^{TR} similar to (12) and (13).
        $u^{\mathtt{loc}}$ = computePolSol(λ^{TR}, μ^{TR}, Γ^{TR}, i, j)
        ($\lambda^{B,\mathtt{loc}}$, $\lambda^{R,\mathtt{loc}}$, $\lambda^{T,\mathtt{loc}}$, $\lambda^{L,\mathtt{loc}}$, $\mu^{B,\mathtt{loc}}$, $\mu^{R,\mathtt{loc}}$, $\mu^{T,\mathtt{loc}}$, $\mu^{L,\mathtt{loc}}$) = extractTraces($u^{\mathtt{loc}}$, i, j)
        Update wave field $\boldsymbol{u}[i, j]$ += $u^{\mathtt{loc}}$
        Update left Dirichlet trace $\boldsymbol{\lambda}^L[i, j]$ += $\lambda^{L,\mathtt{loc}}$
        Update left Neumann trace $\boldsymbol{\mu}^L[i, j]$ += $\mu^{L,\mathtt{loc}}$
        Update bottom Dirichlet trace $\boldsymbol{\lambda}^B[i, j]$ += $\lambda^{B,\mathtt{loc}}$
        Update bottom Neumann trace $\boldsymbol{\mu}^B[i, j]$ += $\mu^{B,\mathtt{loc}}$
  return $\boldsymbol{u}$

Algorithm 11 Global wavefield from a diagonal sweep from the bottom-right to the top-left corner

function addBR2TLSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^L$, $\boldsymbol{\lambda}^T$, $\boldsymbol{\mu}^L$, $\boldsymbol{\mu}^T$)
  for $d = 1, \ldots, m+n$ **do**
    for $i = 1, \ldots, m$ **do**
      $j = n - (d + i)$
      if $i > 1$ and $j < n$ **then**
        Define the L-shaped line $\Gamma^{BR}$by combining Γ^B_{ij} and Γ^R_{ij}
        Define $\lambda^B := \boldsymbol{\lambda}^T[i-1, j]$ and $\mu^B := \boldsymbol{\mu}^T[i-1, j]$
        Define $\lambda^R := \boldsymbol{\lambda}^L[i, j+1]$ and $\mu^R := \boldsymbol{\mu}^L[i, j+1]$
        Define λ^{BR} and μ^{BR} similar to (12) and (13).
        $u^{\mathtt{loc}}$ = computePolSol(λ^{BR}, μ^{BR}, Γ^{BR}, i, j)
        ($\lambda^{B,\mathtt{loc}}$, $\lambda^{R,\mathtt{loc}}$, $\lambda^{T,\mathtt{loc}}$, $\lambda^{L,\mathtt{loc}}$, $\mu^{B,\mathtt{loc}}$, $\mu^{R,\mathtt{loc}}$, $\mu^{T,\mathtt{loc}}$, $\mu^{L,\mathtt{loc}}$) = extractTraces($u^{\mathtt{loc}}$, i, j)
        Update wave field $\boldsymbol{u}[i, j]$ += $u^{\mathtt{loc}}$
        Update left Dirichlet trace $\boldsymbol{\lambda}^L[i, j]$ += $\lambda^{L,\mathtt{loc}}$
        Update left Neumann trace $\boldsymbol{\mu}^L[i, j]$ += $\mu^{L,\mathtt{loc}}$
        Update top Dirichlet trace $\boldsymbol{\lambda}^T[i, j]$ += $\lambda^{T,\mathtt{loc}}$
        Update top Neumann trace $\boldsymbol{\mu}^T[i, j]$ += $\mu^{T,\mathtt{loc}}$
  return $\boldsymbol{u}$

Algorithm 12 Global wavefield from a diagonal sweep from the top-left to the bottom-right corner

function addTL2BRSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^R$, $\boldsymbol{\lambda}^B$, $\boldsymbol{\mu}^R$, $\boldsymbol{\mu}^B$)
  for $d = 1, \ldots, m+n$ **do**
    for $i = m, \ldots, 1$ **do**
      $j = (d+i) - m$
      if $i < m$ and $j > 1$ **then**
        Define the L-shaped line $\Gamma^{TL}$by combining Γ^T_{ij} and Γ^L_{ij}
        Define $\lambda^T := \boldsymbol{\lambda}^B[i+1, j]$ and $\mu^T := \boldsymbol{\mu}^B[i+1, j]$
        Define $\lambda^L := \boldsymbol{\lambda}^R[i, j-1]$ and $\mu^L := \boldsymbol{\mu}^R[i, j-1]$
        Define λ^{TL} and μ^{TL} similar to (12) and (13).
        $u^{\mathtt{loc}}$ = computePolSol(λ^{TL}, μ^{TL}, Γ^{TL}, i, j)
        ($\lambda^{B,\mathtt{loc}}$, $\lambda^{R,\mathtt{loc}}$, $\lambda^{T,\mathtt{loc}}$, $\lambda^{L,\mathtt{loc}}$, $\mu^{B,\mathtt{loc}}$, $\mu^{R,\mathtt{loc}}$, $\mu^{T,\mathtt{loc}}$, $\mu^{L,\mathtt{loc}}$) = extractTraces($u^{\mathtt{loc}}$, i, j)
        Update wave field $\boldsymbol{u}[i, j]$ += $u^{\mathtt{loc}}$
        Update right Dirichlet trace $\boldsymbol{\lambda}^R[i, j]$ += $\lambda^{R,\mathtt{loc}}$
        Update right Neumann trace $\boldsymbol{\mu}^R[i, j]$ += $\mu^{R,\mathtt{loc}}$
        Update bottom Dirichlet trace $\boldsymbol{\lambda}^B[i, j]$ += $\lambda^{B,\mathtt{loc}}$
        Update bottom Neumann trace $\boldsymbol{\mu}^B[i, j]$ += $\mu^{B,\mathtt{loc}}$
  return $\boldsymbol{u}$

C.5. The algorithm of scenario 3

Algorithm 13 Computation of the wavefield for an arbitrary source density that does not intersect the skeleton of the CDD

function computeScenario3(f)
  Define an $m \times n$ matrix $\boldsymbol{u}$ of local wave fields
  ($\boldsymbol{u}$, $\boldsymbol{\lambda}^{B,\mathtt{loc}}$, $\boldsymbol{\lambda}^{R,\mathtt{loc}}$, $\boldsymbol{\lambda}^{T,\mathtt{loc}}$, $\boldsymbol{\lambda}^{L,\mathtt{loc}}$, $\boldsymbol{\mu}^{B,\mathtt{loc}}$, $\boldsymbol{\mu}^{R,\mathtt{loc}}$, $\boldsymbol{\mu}^{T,\mathtt{loc}}$, $\boldsymbol{\mu}^{L,\mathtt{loc}}$) = addLocSols($\boldsymbol{u}$, f)
  ($\boldsymbol{u}$, $\boldsymbol{\lambda}^{R,\mathtt{up}}$, $\boldsymbol{\lambda}^{L,\mathtt{up}}$, $\boldsymbol{\mu}^{R,\mathtt{up}}$, $\boldsymbol{\mu}^{L,\mathtt{up}}$) = addUpSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^{T,\mathtt{loc}}$, $\boldsymbol{\mu}^{T,\mathtt{loc}}$)
  ($\boldsymbol{u}$, $\boldsymbol{\lambda}^{R,\mathtt{down}}$, $\boldsymbol{\lambda}^{L,\mathtt{down}}$, $\boldsymbol{\mu}^{R,\mathtt{down}}$, $\boldsymbol{\mu}^{L,\mathtt{down}}$) = addDownSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^{B,\mathtt{loc}}$, $\boldsymbol{\mu}^{B,\mathtt{loc}}$)
  ($\boldsymbol{u}$, $\boldsymbol{\lambda}^{B,\mathtt{left}}$, $\boldsymbol{\lambda}^{T,\mathtt{left}}$, $\boldsymbol{\mu}^{B,\mathtt{left}}$, $\boldsymbol{\mu}^{T,\mathtt{left}}$) = addLeftSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^{L,\mathtt{loc}}$, $\boldsymbol{\mu}^{L,\mathtt{loc}}$)
  ($\boldsymbol{u}$, $\boldsymbol{\lambda}^{B,\mathtt{right}}$, $\boldsymbol{\lambda}^{T,\mathtt{right}}$, $\boldsymbol{\mu}^{B,\mathtt{right}}$, $\boldsymbol{\mu}^{T,\mathtt{right}}$) = addRightSweeps($\boldsymbol{u}$, $\boldsymbol{\lambda}^{R,\mathtt{loc}}$, $\boldsymbol{\mu}^{R,\mathtt{loc}}$)
  $\boldsymbol{u}$ = addBL2TRSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^{R,\mathtt{up}}$, $\boldsymbol{\lambda}^{T,\mathtt{right}}$, $\boldsymbol{\mu}^{R,\mathtt{up}}$, $\boldsymbol{\mu}^{T,\mathtt{right}}$)
  $\boldsymbol{u}$ = addTR2BLSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^{L,\mathtt{down}}$, $\boldsymbol{\lambda}^{B,\mathtt{left}}$, $\boldsymbol{\mu}^{L,\mathtt{down}}$, $\boldsymbol{\mu}^{B,\mathtt{left}}$)
  $\boldsymbol{u}$ = addBR2TLSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^{L,\mathtt{up}}$, $\boldsymbol{\lambda}^{T,\mathtt{left}}$, $\boldsymbol{\mu}^{L,\mathtt{up}}$, $\boldsymbol{\mu}^{T,\mathtt{left}}$)
  $\boldsymbol{u}$ = addTL2BRSweep($\boldsymbol{u}$, $\boldsymbol{\lambda}^{R,\mathtt{down}}$, $\boldsymbol{\lambda}^{B,\mathtt{right}}$, $\boldsymbol{\mu}^{R,\mathtt{down}}$, $\boldsymbol{\mu}^{B,\mathtt{right}}$)
  Define $u = 0$ in Ω
  for $i = 1, \ldots, m$ **do**
    for $j = 1, \ldots, n$ **do**
      $u|_{\Omega_{ij}} = \boldsymbol{u}[i, j]|_{\Omega_{ij}}$
  return u

[1] P. Amestoy, R. Brossier, A. Buttari, J.-Y. L'Excellent, T. Mary, L. Métivier, A. Miniussi, and S. Operto. Fast 3D frequency-domain full-waveform inversion with a parallel block low-rank multifrontal direct solver: Application to OBC data from the North Sea. *Geophysics*, 81:R363–R383, 2016.

[2] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[3] D. Aruliah and U. Ascher. Multigrid preconditioning for Krylov methods for time-harmonic Maxwell's equations in three dimensions. *SIAM Journal on Scientific Computing*, 24(2):702–718, 2002.

[4] A. V. Astaneh and M. N. Guddati. A two-level domain decomposition method with accurate interface conditions for the Helmholtz problem. *International Journal for Numerical Methods in Engineering*, 107(1):74–90, 2016. nme.5164.

[5] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz. Minimizing communication in numerical linear algebra. *SIAM Journal on Matrix Analysis and Applications*, 32(3):866–901, 2011.

[6] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering (LNCSE)*. Springer-Verlag, 2008. ISBN 978-3-540-77146-3.

[7] J.-P. Bérenger. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185–200, 1994.
[8] F. Billette and S. Brandsberg-Dahl. *The 2004 BP velocity benchmark.* EAGE, 2005.
[9] S. Boerm, L. Grasedyck, and W. Hackbusch. *Hierarchical matrices*. Max-Planck- Institute Lecture Notes, 2006.
[10] Y. Boubendir. An analysis of the BEM-FEM non-overlapping domain decomposition method for a scattering problem. *Journal of Computational and Applied Mathematics*, 204(2):282 – 291, 2007. Special Issue: The Seventh International Conference on Mathematical and Numerical Aspects of Waves (WAVES'05).
[11] Y. Boubendir, X. Antoine, and C. Geuzaine. A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation. *Journal of Computational Physics*, 231(2):262 – 280, 2012.
[12] J. Bramble and J. Pasciak. Analysis of a finite PML approximation for the three dimensional time-harmonic Maxwell and acoustic scattering problems. *Mathematics of Computation*, 76(258):597–614, 2007.
[13] A. Brandt and I. Livshits. Wave-ray multigrid method for standing wave equations. *Electronic Transactions on Numerical Analysis*, 6:162–181, 1997.
[14] H. Calandra, S. Gratton, X. Pinel, and X. Vasseur. An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media. *Numerical Linear Algebra with Applications*, 20(4):663–688, 2013.
[15] T. F. Chan and T. P. Mathew. Domain decomposition algorithms. *Acta Numerica*, 3:61–143, 1 1994.
[16] Z. Chen and X. Xiang. A source transfer domain decomposition method for Helmholtz equations in unbounded domain. *SIAM Journal on Numerical Analysis*, 51(4):2331–2356, 2013.
[17] F. Collino, S. Ghanemi, and P. Joly. Domain decomposition method for harmonic wave propagation: a general presentation. *Computer Methods in Applied Mechanics and Engineering*, 184(2–4):171 – 211, 2000.
[18] T. A. Davis. Algorithm 832: UMFPACK v4.3—an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.
[19] M. V. de Hoop, S. Wang, and J. Xia. On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver. *Geophysical Prospecting*, 59(5):857–873, 2011.
[20] A. de La Bourdonnaye, C. Farhat, A. Macedo, F. Magoules, and F.-X. Roux. A non-overlapping domain decomposition method for the exterior Helmholtz problem. *Contemporary Mathematics*, 218:42–66, 1998.
[21] J. Demmel, L. Grigori, M. Gu, and H. Xiang. Communication avoiding rank revealing qr factorization with column pivoting. Technical Report UCB/EECS-2013-46, EECS Department, University of California, Berkeley, May 2013.
[22] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM Journal Matrix Analysis and Applications*, 20(3):720–755, 1999.
[23] B. Després. Décomposition de domaine et problème de Helmholtz. *Comptes rendus de l'Académie des sciences. Série 1, Mathématique*, 311:313–316, 1990.
[24] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear. *ACM Transactions on Mathematical Software*, 9(3):302–325, September 1983.
[25] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation. *Communications on Pure and Applied Mathematics*, 64(5):697–735, 2011.
[26] B. Engquist and L. Ying. Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers. *Multiscale Modeling & Simulation*, 9(2):686–710, 2011.
[27] B. Engquist and H.-K. Zhao. Absorbing boundary conditions for domain decomposition. *Applied Numerical Mathematics*, 27(4):341 – 365, 1998. Special Issue on Absorbing Boundary Conditions.
[28] Y. A. Erlangga, C. W. Oosterlee, and C. Vuik. A novel multigrid based preconditioner for heterogeneous Helmholtz problems. *SIAM Journal on Scientific Computing*, 27(4):1471–1492, 2006.
[29] O. G. Ernst and M. J. Gander. Why it is difficult to solve Helmholtz problems with classical iterative methods. In Ivan G. Graham, Thomas Y. Hou, Omar Lakkis, and Robert Scheichl, editors, *Numerical Analysis of Multiscale Problems*, volume 83 of *Lecture Notes in Computational Science and Engineering*, pages 325–363. Springer Berlin Heidelberg, 2012.
[30] J. Fang, J. Qian, L. Zepeda-Núñez, and H. Zhao. Learning dominant wave directions for plane wave methods for high-frequency Helmholtz equations. *Research in the Mathematical Sciences*, 4(1):9, May 2017.
[31] M. Gander and F. Nataf. AILU for Helmholtz problems: a new preconditioner based on an analytic factorization. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 331(3):261–266, 2000.
[32] M. Gander and H. Zhang. A class of iterative solvers for the Helmholtz equation: Factorizations, sweeping preconditioners, source transfer, single layer potentials, polarized traces, and optimized Schwarz methods. *SIAM Review*, 61(1):3–76, 2019.
[33] M. J. Gander. Optimized Schwarz methods. *SIAM Journal on Numerical Analysis*, 44(2):699–731, 2006.
[34] M. J. Gander and F. Kwok. Optimal interface conditions for an arbitrary decomposition into subdomains. In Yunqing Huang, Ralf Kornhuber, Olof Widlund, and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering XIX*, volume 78 of *Lecture Notes in Computational Science and Engineering*, pages 101–108. Springer Berlin Heidelberg, 2011.
[35] M. J. Gander, F. Magoulès, and F. Nataf. Optimized Schwarz methods without overlap for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 24(1):38–60, 2002.
[36] M. J. Gander and Y. Xu. *Domain Decomposition Methods in Science and Engineering XXII*, chapter Optimized Schwarz Method with Two-Sided Transmission Conditions in an Unsymmetric Domain Decomposition, pages 631–639. Springer International Publishing, Cham, 2016.
[37] M. J. Gander and H. Zhang. Domain decomposition methods for the Helmholtz equation: A numerical investigation. In Randolph Bank, Michael Holst, Olof Widlund, and Jinchao Xu, editors, *Domain Decomposition Methods in Science and Engineering XX*, volume 91 of *Lecture Notes in Computational Science and Engineering*, pages 215–222. Springer Berlin Heidelberg, 2013.
[38] M. J. Gander and H. Zhang. *Domain Decomposition Methods in Science and Engineering XXI*, chapter Optimized Schwarz Methods with Overlap for the Helmholtz Equation, pages 207–215. Springer International Publishing, Cham, 2014.

[39] M.J. Gander, I.G. Graham, and E.A. Spence. Applying GMRES to the Helmholtz equation with shifted Laplacian preconditioning: what is the largest shift for which wavenumber-independent convergence is guaranteed? *Numerische Mathematik*, pages 1–48, 2015.
[40] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.
[41] S. Ghanemi. A domain decomposition method for Helmholtz scattering problems. In *Ninth International Conference on Domain Decomposition Methods*, pages 105–112, 1998.
[42] A. Gillman, A. H. Barnett, and P.-G. Martinsson. A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media. *BIT Numerical Mathematics*, 55(1):141–170, Mar 2015.
[43] Dan Gordon and Rachel Gordon. Carp-cg: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated {PDEs}. *Parallel Computing*, 36(9):495 – 515, 2010.
[44] R. W. Hockney. A fast direct solution of poisson's equation using fourier analysis. *J. ACM*, 12(1):95–113, January 1965.
[45] Q. Hu and H. Zhang. Substructuring preconditioners for the systems arising from plane wave discretization of Helmholtz equations. *SIAM Journal on Scientific Computing*, 38(4):A2232–A2261, 2016.
[46] S. Johnson. Notes on perfectly matched layers (PMLs), March 2010.
[47] D. Kourounis, A. Fuchs, and O. Schenk. Toward the next generation of multiperiod optimal power flow solvers. *IEEE Transactions on Power Systems*, 33(4):4005–4014, July 2018.
[48] A. Laird and M. Giles. Preconditioned iterative solution of the 2D Helmholtz equation. Technical Report NA 02-12, Computing Lab, Oxford University, May 2002.
[49] W. Leng and L. Ju. An additive overlapping domain decomposition method for the Helmholtz equation. *SIAM Journal on Scientific Computing*, 41(2):A1252–A1277, 2019.
[50] X. S. Li and J. W. Demmel. SuperLU DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Mathematical Software*, 29(2):110–140, June 2003.
[51] Y. Li, L. Métivier, R. Brossier, B. Han, and J. Virieux. 2D and 3D frequency-domain elastic wave modeling in complex media with a parallel iterative solver. *Geophysics*, 80:T101–T118, 2015.
[52] P.-L. Lions. On the Schwarz alternating method II. In Tony Chan, Roland Glowinski, Jacques Periaux, and Olof Widlund, editors, *Domain Decomposition Methods*, Lecture Notes in Computational Science and Engineering, pages 47–70. SIAM, 1989.
[53] F. Liu and L. Ying. Recursive sweeping preconditioner for the 3D Helmholtz equation. *ArXiv e-prints*, 2015.
[54] F. Magoules, K. Meerbergen, and J.-P. Coyette. Application of a domain decomposition with Lagrange multipliers to acoustic problems arising from the automotive industry. *Journal of Computational Acoustics*, 08(03):503–521, 2000.
[55] L. C. McInnes, R. F. Susan-Resiga, D. E. Keyes, and H. M. Atassi. Additive Schwarz methods with nonreflecting boundary conditions for the parallel computation of Helmholtz problems. *Contemporary Mathematics*, 218:325–333, 1998.
[56] A. Modave, X. Antoine, and C. Geuzaine. An efficient domain decomposition method with cross-point treatment for Helmholtz problems. 2018.
[57] A. Moiola and E. Spence. Is the Helmholtz equation really sign-indefinite? *SIAM Review*, 56(2):274–312, 2014.
[58] W.A. Mulder. A multigrid solver for 3D electromagnetic diffusion. *Geophysical Prospecting*, 54(5):633–649, 2006.
[59] R.-E. Plessix. A Helmholtz iterative solver for 3D seismic-imaging problems. *Geophysics*, 72:SM185–SM194, 2007.
[60] R.-E. Plessix and W. A. Mulder. Separation-of-variables as a preconditioner for an iterative Helmholtz solver. *Applied Numerical Mathematics*, 44(3):385–400, 2003.
[61] J. Poulson, L. Demanet, N. Maxwell, and L. Ying. A parallel butterfly algorithm. *SIAM Journal on Scientific Computing*, 36(1):C49–C65, 2014.
[62] J. Poulson, B. Engquist, S. Li, and L. Ying. A parallel sweeping preconditioner for heterogeneous 3D Helmholtz equations. *SIAM Journal on Scientific Computing*, 35(3):C194–C212, 2013.
[63] R. G. Pratt. Seismic waveform inversion in the frequency domain; part 1: Theory and verification in a physical scale model. *Geophysics*, 64(3):888–901, 1999.
[64] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov. A distributed-memory package for dense hierarchically semi-separable matrix computations using randomization. *ACM Transactions on Mathematical Software*, 42(4):27:1–27:35, June 2016.
[65] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.
[66] H. A. Schwarz. Uber einen grenzubergang durch alternierendes verfahren. *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zurich*, 15:272–286, 1870.
[67] A. H. Sheikh, D. Lahaye, and C. Vuik. On the convergence of shifted Laplace preconditioner combined with multilevel deflation. *Numerical Linear Algebra with Applications*, 20(4):645–662, 2013.
[68] F. Sourbier, A. Haiddar, L. Giraud, H. Ben-Hadj-Ali, S. Operto, and J. Virieux. Three-dimensional parallel frequency-domain visco-acoustic wave modelling based on a hybrid direct/iterative solver. *Geophysical Prospecting*, 59(5):834–856, 2011.
[69] E. A. Spence. Wavenumber-explicit bounds in time-harmonic acoustic scattering. *SIAM Journal on Mathematical Analysis*, 46(4):2987–3024, 2014.
[70] C. C. Stolk. A rapidly converging domain decomposition method for the Helmholtz equation. *Journal of Computational Physics*, 241(0):240–252, 2013.
[71] C. C. Stolk. A dispersion minimizing scheme for the 3-D Helmholtz equation based on ray theory. *Journal of Computational Physics*, 314:618 – 646, 2016.
[72] C. C. Stolk. An improved sweeping domain decomposition preconditioner for the Helmholtz equation. *Advances in Computational Mathematics*, 43(1):45–76, Feb 2017.
[73] M. Taus, L. Demanet, and L. Zepeda-Núñez. A short note on a fast and high-order hybridizable discontinuous Galerkin solver for the 2D high-frequency Helmholtz equation. In *SEG Technical Program Expanded Abstracts 2016*, pages 3835–3840, 2016.
[74] M. Taus, L. Demanet, and L. Zepeda-Núñez. A short note on a fast and high-order hybridizable discontinuous Galerkin solver for the 2D high-frequency Helmholtz equation. In *SEG Technical Program Expanded Abstracts 2016*, pages 3835–3840. Society of Exploration

Geophysicists, 2016.
[75] M. Taus, L. Zepeda-Núñez, R. J. Hewett, and L. Demanet. L-Sweeps/L-Sweeps-2D. Sep 2019. Available at `https://github.com/L-Sweeps/L-Sweeps-2D`, version 1.0.0, DOI:10.5281/zenodo.3383883.
[76] M. Taus, L. Zepeda-Núñez, R. J. Hewett, and L. Demanet. L-Sweeps/L-Sweeps-2D-examples. Sep 2019. Available at `https://github.com/L-Sweeps/L-Sweeps-2D-examples`, version 1.0.0, DOI:10.5281/zenodo.3383923.
[77] R. Thakur, R. Rabenseifner, and W. Gropp. Optimization of collective communication operations in mpich. *International Journal of High Performance Computing Applications*, 19(1):49–66, 2005.
[78] A. Toselli and O. B. Widlund. *Domain Decomposition Methods — Algorithms and Theory*, volume 34 of *Springer Series in Computational Mathematics*. Springer Berlin Heidelberg, 2005.
[79] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.
[80] A. Vion and C. Geuzaine. Double sweep preconditioner for optimized Schwarz methods applied to the Helmholtz problem. *Journal of Computational Physics*, 266(0):171–190, 2014.
[81] J. Virieux and S. Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6):WCC1–WCC26, 2009.
[82] S. Wang, M. V. de Hoop, J. Xia, and X. S. Li. Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3-D anisotropic media. *Geophysical Journal International*, 191(1):346–366, 2012.
[83] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li. Superfast multifrontal method for large structured linear systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1382–1411, 2010.
[84] L. Zepeda-Núñez and L. Demanet. A short note on the nested-sweep polarized traces method for the 2D Helmholtz equation. In *SEG Technical Program Expanded Abstracts 2015*, pages 3682–3687, 2015.
[85] L. Zepeda-Núñez and L. Demanet. The method of polarized traces for the 2D Helmholtz equation. *Journal of Computational Physics*, 308:347 – 388, 2016.
[86] L. Zepeda-Núñez and L. Demanet. Nested domain decomposition with polarized traces for the 2D Helmholtz equation. *SIAM Journal on Scientific Computing*, 40(3):B942–B981, 2018.
[87] L. Zepeda-Núñez and H. Zhao. Fast alternating bidirectional preconditioner for the 2D high-frequency Lippmann–Schwinger equation. *SIAM Journal on Scientific Computing*, 38(5):B866–B888, 2016.
[88] L. Zepeda-Núñez, R. J. Hewett, and L. Demanet. Preconditioning the 2D Helmholtz equation with polarized traces. In *SEG Technical Program Expanded Abstracts 2014*, pages 3465–3470, 2014.
[89] L. Zepeda-Núñez, A. Scheuer, R. J. Hewett, and L. Demanet. The method of polarized traces for the 3D Helmholtz equation. *Geophysics*, 84(4):1–86, 2019.