The method of polarized traces for the 3D Helmholtz equation

Leonardo Zepeda-Núñez[*§¶], Adrien Scheuer[*†], Russell J. Hewett[‡], and Laurent Demanet[*]

ABSTRACT

We present a fast solver for the 3D high-frequency Helmholtz equation in heterogeneous, constant density, acoustic media. The solver is based on the method of polarized traces, coupled with distributed linear algebra libraries and pipelining to obtain an empirical online runtime $\mathcal{O}(\max(1, R/n)N \log N)$ where $N = n^3$ is the total number of degrees of freedom and R is the number of right-hand sides. Such a favorable scaling is a prerequisite for large-scale implementations of full waveform inversion (FWI) in frequency domain.

INTRODUCTION

Efficient modeling of time-harmonic wave scattering in heterogeneous acoustic or elastic media remains a difficult problem in numerical analysis, yet it has broad application in seismic inversion techniques, as shown by Chen (1997); Pratt (1999); Virieux and Operto (2009). In the constant density acoustic approximation, time-harmonic wave propagation is modeled by the Helmholtz equation,

$$\triangle u(\mathbf{x}) + \omega^2 m(\mathbf{x}) u(\mathbf{x}) = f_s(\mathbf{x}), \qquad \text{in } \Omega \tag{1}$$

with absorbing boundary conditions, and where Ω is a 3D rectangular domain, $\triangle$ is the 3D Laplacian, $\mathbf{x} = (x, y, z)$, $m = 1/c^2(\mathbf{x})$ is the squared slowness for velocity $c(\mathbf{x})$, u is the wavefield, and f_s are the sources, indexed by $s = 1, ..., R$. There is no essential obstruction to extending the techniques presented in this paper to the case of heterogeneous density, or to the elastic, viscoelastic or poroelastic time-harmonic equations.

Throughout this paper we assume that Eq. 1 is in the high-frequency regime, i.e., when $\omega \sim n$, where n is the number of unknowns in each dimension. Alternatively, in the situation where the domain Ω grows and the frequency is fixed, the problem can be rescaled in such a way that ω grows in proportion to the domain size, which can also considered to be a "high-frequency" regime. For this paper, we focus on the former scenario.

Given the importance of solving Eq. 1 in geophysical contexts, there has been a renewed interest in developing efficient algorithms to solve the ill-conditioned linear system resulting from its discretization. Recent progress toward an efficient solver, i.e., a solver with linear complexity, has generally focused on three strategies:

- *Fast direct solvers*, such as the ones introduced by Xia et al. (2010); de Hoop et al. (2011); Gillman et al. (2014); Amestoy et al. (2015), which couple multifrontal techniques (e.g., George (1973); Duff and Reid (1983)) with compressed linear algebra (e.g., Bebendorf (2008)) to obtain efficient direct solvers with small memory footprint. However, they suffer the same sub-optimal asymptotic complexity as standard multifrontal methods (e.g., Demmel et al. (1999); Amestoy et al. (2001); Davis (2004)) in the high-frequency regime.

- *Classical preconditioners*, such as incomplete factorization preconditioners (e.g., Bollhöfer et al. (2009))

[*] Massachusetts Institute of Technology,
Department of Mathematics and Earth Resources Laboratory,
77 Massachusetts Ave,
Cambridge, MA 02139.
[†] Université Catholique de Louvain,
1, Place de l'Université,
B-1348 Louvain-la-Neuve, Belgium
[‡] Total E. & P. Research & Technology USA
1201 Louisiana St.,
Houston, TX 77002.
[§] Computational Research Division,
Lawrence Berkeley National Laboratory,
1 Cyclotron road,
Berkeley, CA 94720.
[¶] University of California Irvine,
Department of Mathematics,
540 Rowland Hall,
Irvine, CA 92963.

and multigrid-based preconditioners (e.g., Brandt and Livshits (1997); Erlangga et al. (2006); Sheikh et al. (2013); Calandra et al. (2013)), which are relatively simple to implement but suffer from super-linear asymptotic complexity and may need significant tuning to achieve effective run-times.

- *Sweeping-like preconditioners* (e.g., Gander and Nataf (2005); Engquist and Ying (2011a,b); Chen and Xiang (2013a,b); Stolk (2013); Vion and Geuzaine (2014); Liu and Ying (2015); Zepeda-Núñez and Demanet (2016)), which are a relatively recent domain decomposition based approach that has been shown to achieve linear or nearly-linear asymptotic complexity.

The method in this paper belongs to the third category. Sweeping preconditioners and their generalizations, i.e., domain decomposition techniques coupled with high-quality transmission/absorption conditions, offer the right mix of ideas to attain linear or near-linear complexity in 2D and 3D, provided that the medium does not have large resonant cavities (Zepeda-Núñez and Demanet, 2016). These methods rely on the sparsity of the linear system to decompose the domain in layers, in which classical sparse direct methods are used to compute the interactions within the layer. Interactions across layers are computed by sequentially sweeping through the sub-domains in an iterative fashion.

For current applications, empirical runtimes are a more practical measure of an algorithm's performance than asymptotic complexity. This requirement has led to a recent effort to reduce the runtimes of preconditioners with optimal asymptotic complexity by leveraging parallelism. For example, Poulson et al. (2013) introduce a new local solver, i.e., a solver for the subproblem defined on each layer, which carefully handles communication patterns between layers to obtain impressive timings. While most sweeping algorithms require visiting each subdomain in sequential fashion, Stolk (2015b) introduced a modified sweeping pattern, which changes the data dependencies during the sweeps to improve parallelism. Finally, Zepeda-Núñez and Demanet (2016) introduced the method of polarized traces, which reduces the solver's runtime by leveraging parallelism and fast summation methods. This paper builds on top of the general framework of the method of polarized traces, which we elaborate on in the sequel.

To date, most studies focus on minimizing the parallel runtime or complexity of a single solve with a single right-hand side. However, in the scope of seismic inversion, where there can be many thousands of right-hand sides, it is important to consider the overall runtime or complexity of solving *all* right-hand sides. In this context, linear complexity is $\mathcal{O}(RN)$, where N is the total number of degrees of freedom (we assume that $N = n^3$ and let n be the number of degrees of freedom in a single dimension of a 3D volume) and R is the number of right-hand sides.

In this paper, we present a solver for the 3D high-frequency Helmholtz equation with a *sublinear* online parallel runtime, given by

$$\mathcal{O}(\alpha_{\text{pml}}^2 \max(1, R/L) N \log N),$$

where $N = n^3$ is the total number of unknowns, $L \sim n$ is the number of subdomains in a layered domain decomposition, and α_{pml} is the number of points needed to implement a high-quality absorbing boundary condition between layers. We achieve this complexity by comprehensive parallelization of all aspects of the algorithm, including exploiting parallelism in local solves and by pipelining the right-hand sides. Thus, as long as $R \sim n^2$ (3D), there is a mild $R/L \sim n$ factor impacting the asymptotic complexity. The solver in this paper is based on the method of polarized traces (Zepeda-Núñez and Demanet, 2016), a layered domain decomposition method which exploits:

- local solvers, using efficient sparse direct solvers at each subdomain,
- high-quality transmission conditions between subdomains, implemented via perfectly-matched layers (PML; Bérenger (1994); Johnson (2010)), and
- an efficient preconditioner based on polarizing conditions imposed via incomplete Green's integrals.

These concepts combine to yield a global iterative method that converges in a small number of iterations. The method has two stages: an offline stage, that can be precomputed independently of the right-hand sides, and an online stage, that is computed for each right-hand side or by batch processing.

One advantage of the method of polarized traces is that only the degrees of freedom at the interfaces between layers are needed for the bulk of the computation, because the volume problem is reduced to an equivalent surface integral equation (SIE) at the interfaces between layers. Due to this efficiency, the algorithm requires a smaller memory footprint, which helps make feasible pipelining the right-hand sides. Pipelining for domain decomposition methods has been previously considered (Stolk, 2015b), albeit without complexity claims and without a fully tuned communication strategy between subdomains. Moreover, we are unaware of any complexity claims within the context of inversion algorithms, in particular, with focus on full waveform inversion (Tarantola, 1984), where there are many right-hand sides that are strongly frequency dependant.

Complexity claims

Suppose that L, the number of layers in the domain decomposition, scales as $L \sim n$, i.e., each layer has a constant thickness in number of grid points*. Each layer is

*This hypothesis is critical for obtaining a quasi-linear complexity algorithm and is related to the complexity of solving a quasi-

Stage	Polarized traces
offline	$\mathcal{O}\left(\alpha_{\text{pml}}^3 N\right)$
online	$\mathcal{O}(\alpha_{\text{pml}}^2 \max(1, R/L) N \log N)$

Table 1: Runtime of both stages of the algorithm. Note that typically $\alpha_{\text{pml}} \sim \log \omega$.

further extended by α_{pml} grid points in order to implement the PML. It has been documented in Poulson et al. (2013) that α_{pml} needs to grow with problem frequency, $\alpha_{\text{pml}} \sim \log \omega$, in order to obtain a number of Krylov solver iterations to convergence that scales as $\log \omega$. Additionally, as above, for the 3D problem it is typical that the number of sources $R \sim n^2$, because as frequency and resolution increase, the number of sources in both the in-line and cross-line direction must also increase (Li et al., 2015).

Finally, given that we solve the 3D problem in a high-performance computing (HPC) environment, we assume that the number of computing nodes in the HPC cluster is $\mathcal{O}(n^3 \log(n)/M)$, with $L \sim n$ layers, $\mathcal{O}(n^2 \log(n)/M)$ nodes inside each layer, and M is the memory of a single node. The assumptions on node growth come from the fact that computing nodes have finite memory, and thus more nodes are needed to solve larger problems. As numerical examples will show, using more nodes per layer reduces the runtime per Krylov iteration by enhancing the parallelism of the solves at each subdomain, provided that a carefully designed communication pattern is used, to keep the communication overhead low.

We summarize the asymptotic runtimes in Table 1. Like other related methods, the offline stage is linear in N and is independent of the number of right-hand sides. The online runtime is sub-linear, in the sense that linear complexity would be $\mathcal{O}(RN)$.

Related work

Modern linear algebra techniques, in particular nested dissection methods (George, 1973) coupled with $\mathcal{H}$-matrices (Boerm et al., 2006) have been applied to the Helmholtz problem, yielding, for example: the hierarchical Poincaré-Steklov solver (Gillman et al., 2014), solvers using hierarchical semi-separable (HSS) matrices (de Hoop et al., 2011; Wang et al., 2012, 2013), or block low-rank (BLR) matrices (Amestoy et al., 2015, 2016).

Multigrid methods, once thought to be inefficient for the Helmholtz problem, have been successfully applied to the Helmholtz problem by Calandra et al. (2013) and Stolk (2015a). Although these algorithms do not result in a lower computational complexity, their empirical run-times are impressive due to their highly parallelizable nature. Due to the possibility for efficient parallelization, there has been a renewed interest on multilevel preconditioners such as the one in Hu and Zhang (2016).

Within the geophysical community, the analytic incomplete LU (AILU) method was explored by Plessix and Mulder (2003) and applied in the context of 3D seismic imaging, resulting in some large computations (Plessix, 2007). A variant of Kazmarc preconditioners (Gordon and Gordon, 2010) have been studied and applied to time-harmonic wave equations by Li et al. (2015). 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 PDEs date back to Schwarz (1870), in which the Laplace equation is solved iteratively (for a more recent treatise, see Lions (1989)). The application of domain decomposition to the Helmholtz problem was first proposed by Després (1990). Cessenat and Després (1998) further refined this approach with the development of the ultra-weak variational formulation (UWVF) for the Helmholtz equation, in which the basis functions in each element, or sub-domain, are solutions to the local homogeneous equation. The UWVF approach motivated a series of related methods, such as the partition of unity method of Babuska and Melenk (1997), the least squared method of Monk and Wang (1999), the discontinuous enrichment method by Farhat et al. (2001), and Trefftz methods by Gittelson et al. (2009), and Moiola et al. (2011), among many others. A recent and thorough review of Trefftz and related methods can be found in (Hiptmair et al. (2015)).

The results in Lions (1989) and Després (1990) have inspired the development of various domain decomposition algorithms, which are now classified as Schwarz algorithms[†]. However, the convergence rate of such algorithms is strongly dependent on the boundary conditions prescribed at the interfaces between subdomains. Gander et al. (2002) introduces an optimal, non-local boundary condition for domain interfaces, which is then approximated by an optimized Robin boundary condition. This last work lead to the introduction of the framework of optimized Schwarz methods in Gander (2006) to described optimized boundary conditions that provides high convergence. The design of better interface approximations has been studied in Gander and Kwok (2011); Boubendir et al. (2012); Gander and Zhang (2013, 2014); Gander and Xu (2016) among many others.

Engquist and Zhao (1998) introduced absorbing boundary conditions for domain decomposition schemes for elliptic problems and the first application of such techniques to the Helmholtz problem traces back to the AILU factorization (Gander and Nataf (2000)). The sweeping preconditioner, introduced in Engquist and Ying (2011a,b), was

2D problem using multi-frontal methods (for further details see Engquist and Ying (2011b); Poulson et al. (2013)).

[†]For a review on classical Schwarz methods see (Chan and Mathew, 1994; Toselli and Windlund, 2005); and for other applications of domain decomposition methods for the Helmholtz equations, see (de La Bourdonnaye et al., 1998; Ghanemi, 1998; McInnes et al., 1998; Collino et al., 2000; Magoules et al., 2000; Boubendir, 2007; Astaneh and Guddati, 2016).

the first algorithm to show that those ideas could yield algorithms with quasi-linear complexity. There exists two variants of the sweeping preconditioner which involved using either $\mathcal{H}$-matrices (Engquist and Ying, 2011a) or multi-frontal solvers (Engquist and Ying, 2011b) to solve the local problem in each thin layer. These schemes are extended to different discretizations and physics by Tsuji et al. (2012, 2014); Tsuji and Ying (2012). Since the introduction of the sweeping preconditioner, several related algorithms with similar claims have been proposed, such as the source transfer preconditioner (Chen and Xiang (2013a,b)), the rapidly converging domain decomposition (Stolk (2013)) and its extensions (Stolk (2015b)), the double sweep preconditioner (Vion and Geuzaine (2014)) and the method of polarized traces (Zepeda-Núñez and Demanet (2016)).

Organization

The remainder of this paper is organized as follows: we provide the numerical formulation of the Helmholtz equation, present the reduction to a surface integral equation, and introduce the method of polarized traces for solving the SIE. Next, we elaborate on the parallelization and communication patterns and examine the empirical complexities and runtimes. Finally, we provide results from several experiments to support our claims.

FORMULATION

For this study, we discretize Eq. 1 using the standard second order finite difference method on a regular mesh of Ω, with a grid of size $n_x \times n_y \times n_z$ and a grid spacing h. Note, the method of polarized traces is not restricted to second-order finite differences, but using higher-order finite difference schemes makes the numerical implementation slightly more complicated[‡]. Absorbing boundary conditions are imposed via perfectly matched layers (PMLs) as described by Bérenger (1994) and Johnson (2010).

We describe our PML implementation in detail because the quality and structure of the PML implementation strongly impact the convergence properties of the method. Following Bérenger (1994); Johnson (2010), the PML's are implemented via a complex coordinate stretching. First, we define an extended domain $\widehat{\Omega}$ such that $\Omega \subset \widehat{\Omega}$ and we extend the Helmholtz operator from Eq. 1 to that domain as follows:

$$\mathcal{H} = \widehat{\triangle} + m\omega^2 \qquad \text{in } \widehat{\Omega}, \tag{2}$$

where m is an extension of the squared slowness to $\widehat{\Omega}$ and the extended Laplacian $\widehat{\triangle}$ is constructed by replacing the partial derivatives in the standard Laplacian $\triangle = \partial_{xx} + \partial_{yy} + \partial_{zz}$ with coordinate-stretched partial derivatives defined on $\widehat{\Omega}$:

$$\partial_x \to \beta_x(\mathbf{x})\partial_x,\ \partial_y \to \beta_y(\mathbf{x})\partial_y,\ \partial_z \to \beta_z(\mathbf{x})\partial_z. \tag{3}$$

The complex dilation function $\beta_x(\mathbf{x})$ (and similarly $\beta_y(\mathbf{x})$ and $\beta_z(\mathbf{x})$) is defined as

$$\beta_x(\mathbf{x}) = \frac{1}{1 + i\frac{\sigma_x(\mathbf{x})}{\omega}}, \tag{4}$$

where the PML profile function $\sigma_x(\mathbf{x})$ (and similarly $\sigma_y(\mathbf{x})$ and $\sigma_z(\mathbf{x})$) is,

$$\sigma_x(\mathbf{x}) = \begin{cases} \frac{C}{\delta_{\text{pml}}}\left(\frac{x}{\delta_{\text{pml}}}\right)^2, & \text{if } x \in (-\delta_{\text{pml}}, 0), \\ 0, & \text{if } x \in [0, L_x], \\ \frac{C}{\delta_{\text{pml}}}\left(\frac{x-L_x}{\delta_{\text{pml}}}\right)^2, & \text{if } x \in (L_x, L_x + \delta_{\text{pml}}), \end{cases} \tag{5}$$

where L_x is the length of Ω in the x dimension and δ_{pml} is the length of the extension. In general, δ_{pml} grows slowly with the frequency, i.e., $\delta_{\text{pml}} \propto \mathcal{O}(\log\omega)$, in order to obtain enough absorption as the frequency increases. The constant C is chosen to provide enough absorption. In practice, δ_{pml} and C can be seen as parameters to be tuned for accuracy versus efficiency.

The extended Helmholtz operator provides the definition of the global continuous problem,

$$\mathcal{H}u = f_s, \qquad \text{in } \widehat{\Omega}, \tag{6}$$

which is then discretized using finite differences to obtain the discrete global problem,

$$\mathbf{H}\mathbf{u} = \mathbf{f}_s. \tag{7}$$

In the method of polarized traces, Ω is decomposed into a set of L layers, $\{\Omega^\ell\}_{\ell=1}^L$. Without loss of generality, we assume that the decomposition is in the z dimension. Each subdomain Ω^ℓ is extended to include an absorbing region, as above, yielding the extended subdomain $\widehat{\Omega}^\ell$. For boundaries of $\widehat{\Omega}^\ell$ shared with $\widehat{\Omega}$, the absorbing layer is considered to be inherited from the global problem. For the intra-layer boundaries of Ω^ℓ, i.e., those due to the partitioning of Ω, the extension to the additional absorbing layers in $\widehat{\Omega}^\ell$ are necessary to prevent reflections at layer interfaces which are detrimental to convergence.

Then, the local Helmholtz problem is

$$\mathcal{H}^\ell v^\ell := \triangle^\ell v^\ell(\mathbf{x}) + m^\ell\omega^2 v^\ell(\mathbf{x}) = f_s^\ell(\mathbf{x}) \qquad \text{in } \widehat{\Omega}^\ell, \tag{8}$$

where m^ℓ and f^ℓ are the local restrictions of the model parameters and source functions to $\widehat{\Omega}^\ell$, generated by extending $m\chi_{\Omega^\ell}$ and $f\chi_{\Omega^\ell}$ to $\widehat{\Omega}^\ell$, where χ_{Ω^ℓ} is the characteristic function of Ω^ℓ. The local Laplacian $\triangle^\ell$ is defined using the same coordinate stretching approach as above, except on $\widehat{\Omega}^\ell$. As before, δ_{pml}, and thus α_{pml}, must scale as $\log\omega$ to obtain the convergence rate claimed in this paper.

We discretize the local problem in Eq. 8 resulting in the

[‡]Nor is the method applicable only in the context of finite differences: finite elements (Taus et al., 2016; Fang et al., 2016) and integral equations (Zepeda-Núñez and Zhao, 2016) approaches are also valid.

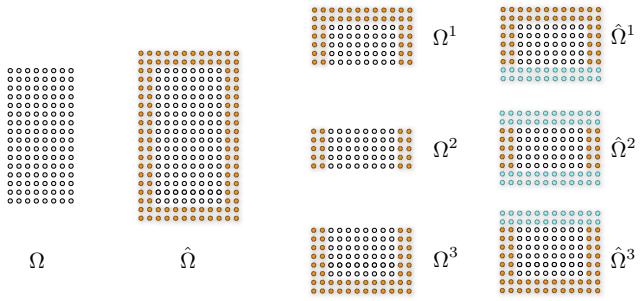

Figure 1: Sketch in 2D of the partition of the domain in layers. The domain Ω is extended to $\widehat{\Omega}$ by adding the PML nodes (orange). After decomposition into subdomains Ω^ℓ, the internal boundaries are padded with extra PML nodes (light blue) resulting in the subdomains $\widehat{\Omega}^\ell$.

discrete local Helmholtz system

$$\mathbf{H}^\ell \mathbf{v}^\ell = \mathbf{f}_s^\ell. \tag{9}$$

For the finite difference implementation in this paper, we assume a structured, equispaced Cartesian mesh with mesh points $\mathbf{x}_{i,j,k} = (x_i, y_j, z_k) = (ih, jh, kh)$. Assuming the same ordering Zepeda-Núñez and Demanet (2016), we write the global solution in terms of the depth index,

$$\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_{n_z}), \tag{10}$$

where $\mathbf{u}_k$ is a plane sampled at constant depth z_k, or in MATLAB notation,

$$\mathbf{u}_k = (u_{:,:,k}). \tag{11}$$

Let $\mathbf{u}^\ell$ be the local restriction of $\mathbf{u}$ to Ω^ℓ, i.e., $\mathbf{u}^\ell = \chi_{\Omega^\ell}\mathbf{u}$. Following the above notation, $\mathbf{u}_k^\ell$ is the local solution trace in the plane at local depth z_k^ℓ. For notational convenience, we renumber the local depth indices so that $\mathbf{u}_1^\ell$ and $\mathbf{u}_{n^\ell}^\ell$ are the top and bottom planes of the bulk domain. Points due to the PML are not considered[§]. Finally, let

$$\underline{\mathbf{u}} = \left(\mathbf{u}_{n^1}^1, \mathbf{u}_1^2, \mathbf{u}_{n^2}^2, ..., \mathbf{u}_1^{L-1}, \mathbf{u}_{n^{L-1}}^{L-1}, \mathbf{u}_1^L\right)^t, \tag{12}$$

be the vector of interface traces for all L layers.

To map solution vectors at fixed depth planes back to the discretized whole volume of Ω_ℓ, we define the Dirac delta at a fixed depth,

$$(\delta(z - z_p)\mathbf{v}_q)_{i,j,k} = \begin{cases} 0 & \text{if } k \neq p, \\ \frac{(\mathbf{v}_q)_{i,j}}{h^3} & \text{if } k = p. \end{cases} \tag{13}$$

This definition of the numerical Dirac delta is specific to a classical finite difference discretization. If the discretization changes, it is still possible to define a numerical Dirac delta using the approach developed in Zepeda-Núñez and Demanet (2015).

[§]With this renumbering the local depth index z_k^ℓ maps to the global depth index $z_{n_c^\ell + k}$ where $n_c^\ell = \sum_{j=1}^{\ell-1} n^j$.

Accuracy

Solving the Helmholtz equation in the high-frequency regime is notoriously because:

- it is difficult to efficiently discretize the PDE, and
- the resulting linear system is difficult to solve in a scalable and efficient fashion.

In this paper, we focus on the second issue. However, for completeness, we provide a brief overview of difficulties associated to the discretization.

From the Shannon-Nyquist sampling theorem, an oscillatory function at frequency ω requires $\mathcal{O}(\omega^d)$ degrees of freedom to be accurately represented, without aliasing. For example, to accurately represent the solution of Eq. 1, only $\mathcal{O}(\omega^3)$ degrees of freedom are required. Obviously, accuracy is still limited by the error in the discretization of the differential operator. Even if the medium is very smooth, standards methods based on finite differences and finite elements are subject to pollution error, i.e., the ratio between the error of the numerical approximation and the best approximation cannot be bounded independently of ω (Babuska et al., 1995; Ihlenburg and Babuska, 1995; Babuska and Banerjee, 2012).

The direct consequence of pollution error is that the approximation error, i.e., the error between the analytical and the numerical solution to the linear system, increases with the frequency, even if $n \sim \omega$. Thus, to obtain a bounded approximation error independent of the frequency, it is required to oversample the wavefield, relative to the Shannon-Nyquist criterion, i.e., n needs to grow faster than ω. Unfortunately, oversampling provides discretizations with a suboptimal number of degrees of freedom with respect to the frequency. To alleviate pollution error, several new approaches have been proposed, which can be broadly classified into two groups:

1. methods using standard polynomial bases with modified variational formulations (Goldstein, 1986; Melenk and Sauter, 2011; Melenk et al., 2013; Moiola and Spence, 2014; Graham et al., 2015);

2. methods based on well known variational formulations but using non-standard basis, such as plane waves (Hiptmair et al., 2015; Perugia et al., 2016), polynomials modulated by plane waves (Betcke and Phillips, 2012; Nguyen et al., 2015), or other specially adapted functions.

Even though the methods mentioned above have been successful in reducing pollution error, the resulting linear systems cannot, in general, be solved in quasi-linear time or better because the matrices either have a high degree of

interconnectivity or are extremely ill-conditioned. However, some new fast algorithms have recently been proposed for solving the Helmholtz equation without pollution error with quasi-linear complexity for media that are homogeneous up to smooth and compactly supported heterogeneities (Zepeda-Núñez and Zhao (2016); Fang et al. (2016); and references therein). In the case of highly heterogeneous media the accuracy of finite elements has not been extensively studied, although methods of efficient discretizations for highly heterogeneous media, coupled with fast algorithms are emerging (Taus et al., 2016).

In this paper, we assume that waves will propagate in very general and highly heterogeneous media, thus, we do not have a theoretical framework to assess the accuracy. Instead, we use numerical experiments to check the accuracy of the solution. Our numerical experiments show, for the cases considered in this paper, using 10 points per wavelength results in roughly 1 digit of accuracy at the highest frequency considered.

Reduction to a surface integral equation

The global solution is related to the local layer solutions by coupling the subdomains using the Green's representation formula (GRF) within each layer. The resulting surface integral equation (SIE), posed at the interface between layers, effectively reduces the problem from the global domain Ω to the interfaces between layers. The resulting SIE has the form

$$\underline{\mathbf{M}}\,\underline{\mathbf{u}} = \underline{\mathbf{f}}, \tag{14}$$

where $\underline{\mathbf{M}}$ is formed by interface-to-interface Green's functions, $\underline{\mathbf{u}}$ is defined in Eq. 12, and $\underline{\mathbf{f}}$ is the right-hand-side, formed as in Line 8 of Alg. 2.

The matrix $\underline{\mathbf{M}}$ is a block banded matrix (Fig. 2, left) of size $2(L-1)n^2 \times 2(L-1)n^2$. Theorem 1 of Zepeda-Núñez and Demanet (2016) gives that the solution of Eq. 14 is exactly the restriction of the solution of Eq. 7 to the interfaces between layers.

Following Zepeda-Núñez and Demanet (2016), if the traces of the exact solution are known, then it is possible to apply the GRF to locally reconstruct exactly the global solution within each layer. Equivalently, the reconstruction can be performed by modifying the local source with a measure supported on the layer interfaces and solving the local system with the local solver, as seen in lines 11-12 of Alg. 2, where a high-level sketch of the algorithm to solve the 3D high-frequency Helmholtz equation is given.

To efficiently solve the 3D problem, it is critical that the matrix $\underline{\mathbf{M}}$ is never explicitly formed. Instead, a matrix-free approach (Zepeda-Núñez and Demanet, 2015) is used to apply the blocks of $\underline{\mathbf{M}}$ via applications of the local solver, using equivalent sources supported at the interfaces between layers, as shown in Alg. 3. Moreover, as seen in Alg. 3, the application of $\underline{\mathbf{M}}$ is easily implemented in parallel, with a small communication overhead. The only non-embarrassingly parallel stage of Alg. 2 is the solution of Eq. 14, which is inherently sequential.

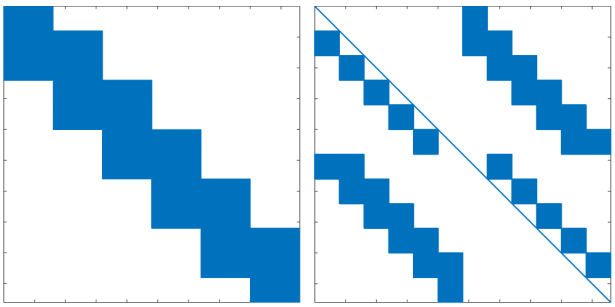

Figure 2: Sparsity pattern of the SIE matrix in Eq. 14 (left), and the polarized SIE matrix in Eq. 18 (right) .

Given that $\underline{\mathbf{M}}$ is never explicitly formed, an iterative method is the natural choice for solving Eq. 14. In practice, the condition number of $\underline{\mathbf{M}}$ is very large and it has a wide spectrum is the complex plane, which implies that a large number of iterations are required to achieve convergence. To alleviate this problem, we apply the method of polarized traces, as an efficient preconditioner for Eq. 14, which we describe below.

Algorithm 1. *Offline computation*

1: ***function*** $[L^\ell, U^\ell] =$ FACTORIZATION$(m,\, \omega)$
2: ***for*** $\ell = 1 : L$ ***do***
3: $\quad \mathbf{H}^\ell = \triangle^\ell + m^\ell \omega^2$ ▷ *Build the local system*
4: $\quad \mathbf{L}^\ell \mathbf{U}^\ell = \mathbf{H}^\ell$ ▷ *Compute the LU factorization*
5: ***end for***
6: ***end function***

Algorithm 2. *Online computation using the SIE reduction*

1: ***function*** $\mathbf{u} =$ HELMHOLTZ SOLVER$(\,\mathbf{f}\,)$
2: ***for*** $\ell = 1 : L$ ***do***
3: $\quad \mathbf{f}^\ell = \mathbf{f}\chi_{\Omega^\ell}$ ▷ *partition the source*
4: ***end for***
5: ***for*** $\ell = 1 : L$ ***do***
6: $\quad \mathbf{v}^\ell = (\mathbf{H}^\ell)^{-1}\mathbf{f}^\ell$ ▷ *solve local problems*
7: ***end for***
8: $\underline{\mathbf{f}} = \left(\mathbf{v}^1_{n^1}, \mathbf{v}^2_1, \mathbf{v}^2_{n^2}, \ldots, \mathbf{v}^L_1\right)^t$ ▷ *form r.h.s.*
9: $\underline{\mathbf{u}} = (\underline{\mathbf{M}})^{-1}\,\underline{\mathbf{f}}$ ▷ *solve Eq. 14*
10: ***for*** $\ell = 1 : L$ ***do***
11: $\quad \mathbf{g}^\ell = \mathbf{f}^\ell + \delta(z_1 - z)\mathbf{u}^{\ell-1}_{n^{\ell-1}} - \delta(z_0 - z)\mathbf{u}^\ell_1 - \delta(z_{n^\ell+1} - z)\mathbf{u}^\ell_{n^\ell} + \delta(z_{n^\ell} - z)\mathbf{u}^{\ell+1}_1$
12: $\quad \mathbf{u}^\ell = (\mathbf{H}^\ell)^{-1}\mathbf{g}^\ell$ ▷ *inner solve*
13: ***end for***
14: $\mathbf{u} = (\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^{L-1}, \mathbf{u}^L)^t$ ▷ *concatenate*
15: ***end function***

Algorithm 3. *Application of the boundary integral matrix* $\underline{\mathbf{M}}$

1: ***function*** $\underline{\mathbf{u}} =$ BOUNDARY INTEGRAL$(\,\underline{\mathbf{v}}\,)$
2: $\quad \widetilde{\mathbf{f}}^1 = -\delta(z_{n^1+1} - z)\mathbf{v}^1_{n^\ell} + \delta(z_{n^1} - z)\mathbf{v}^2_{n^1}$
3: $\quad \mathbf{w}^1 = (\mathbf{H}^1)^{-1}\widetilde{\mathbf{f}}^1$
4: $\quad \mathbf{u}^\ell_{n^\ell} = \mathbf{w}^\ell_{n^\ell} - \mathbf{v}^\ell_{n^\ell}$
5: $\quad$ ***for*** $\ell = 2 : L-1$ ***do***

$$\widetilde{\mathbf{f}}^{\ell} = \delta(z_1 - z)\mathbf{v}^{\ell-1}_{n^{\ell-1}} - \delta(z_0 - z)\mathbf{v}^{\ell}_1 - \delta(z_{n^{\ell}+1} - z)\mathbf{v}^{\ell}_{n^{\ell}} + \delta(z_{n^{\ell}} - z)\mathbf{v}^{\ell+1}_1$$

$\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\widetilde{\mathbf{f}}^{\ell}$ ▷ *inner solve*

$\mathbf{u}^{\ell}_1 = \mathbf{w}^{\ell}_1 - \mathbf{v}^{\ell}_1; \qquad \mathbf{u}^{\ell}_{n^{\ell}} = \mathbf{w}^{\ell}_{n^{\ell}} - \mathbf{v}^{\ell}_{n^{\ell}}$

end for

$\widetilde{\mathbf{f}}^{L} = \delta(z_1 - z)\mathbf{v}^{L-1}_{n^{L-1}} - \delta(z_0 - z)\mathbf{v}^{L}_1$

$\mathbf{w}^{L} = (\mathbf{H}^{L})^{-1}\widetilde{\mathbf{f}}^{L}$

$\mathbf{u}^{L}_1 = \mathbf{w}^{L}_1 - \mathbf{v}^{L}_1$

end function

METHOD OF POLARIZED TRACES

Reducing the Helmholtz problem to a SIE allows us to efficiently parallelize most of the computation required to solve Eq. 7. The only remaining sequential bottleneck is the solution of Eq. 14. Given the size and the distributed nature of $\underline{\mathbf{M}}$, iterative methods, such as GMRES (Saad and Schultz (1986)) or Bi-CGSTAB (van der Vorst (1992)), are the logical approach for solving Eq. 14. However, numerical experiments indicate that the condition number of $\underline{\mathbf{M}}$ scales as $\mathcal{O}(h^{-2})$, or as $\mathcal{O}(\omega^2)$ in the high-frequency regime (Zepeda-Núñez and Demanet, 2016). The number of iterations required for schemes like GMRES to converge is proportional to the condition number of the system, yielding poor scalability for solving the SIE at high frequencies. To alleviate this problem, we use the method of polarized traces to convert the SIE to an equivalent problem, which is easily preconditioned. This preconditioned system only requires $\mathcal{O}(\log\omega)$ GMRES iterations[¶], i.e., it is comparatively independent of the frequency. Here, we provide a high-level review of the method of polarized traces and its implementation and we direct the reader to Zepeda-Núñez and Demanet (2016) for a detailed exposition.

Preconditioner

As seen in the previous discussion, Eq. 14 is the result of decomposing the domain into a set of layers and reducing the Helmholtz problem to an equivalent SIE on the interfaces between the subdomains. To precondition the SIE with the method of polarized traces, the solution at the interfaces is decomposed in up- and down-going components such that

$$\underline{\mathbf{u}} = \underline{\mathbf{u}}^{\uparrow} + \underline{\mathbf{u}}^{\downarrow}, \tag{15}$$

which defines the polarized wavefield

$$\underline{\underline{\mathbf{u}}} = \begin{pmatrix} \underline{\mathbf{u}}^{\downarrow} \\ \underline{\mathbf{u}}^{\uparrow} \end{pmatrix}. \tag{16}$$

By introducing the polarized wavefield, we have deliberately doubled the unknowns and produced an underdetermined system. To close the system, we impose annihilation, or polarizing, conditions (see Section 3 of Zepeda-Núñez and Demanet (2016)) that are encoded in matrix form as

$$\underline{\mathbf{A}}^{\uparrow}\underline{\mathbf{u}}^{\uparrow} = 0, \quad \text{and} \quad \underline{\mathbf{A}}^{\downarrow}\underline{\mathbf{u}}^{\downarrow} = 0. \tag{17}$$

Requiring that the solution satisfies both Eq. 14 and the annihilation conditions yields another equivalent formulation,

$$\underline{\underline{\mathbf{M}}}\,\underline{\underline{\mathbf{u}}} = \underline{\underline{\mathbf{f}}}_s, \tag{18}$$

where

$$\underline{\underline{\mathbf{M}}} = \begin{bmatrix} \underline{\mathbf{M}} & \underline{\mathbf{M}} \\ \underline{\mathbf{A}}^{\downarrow} & \underline{\mathbf{A}}^{\uparrow} \end{bmatrix}, \text{ and } \underline{\underline{\mathbf{f}}}_s = \begin{pmatrix} \underline{\mathbf{f}}_s \\ 0 \end{pmatrix}. \tag{19}$$

Following a series of basic algebraic operations and permutations (see Zepeda-Núñez and Demanet (2016) for the full details), we obtain an equivalent formulation of the polarized SIE matrix in Eq. 18, given by

$$\underline{\underline{\mathbf{M}}} = \begin{bmatrix} \underline{\mathbf{D}}^{\downarrow} & \underline{\mathbf{U}} \\ \underline{\mathbf{L}} & \underline{\mathbf{D}}^{\uparrow} \end{bmatrix}. \tag{20}$$

There exist straightforward, parallel algorithms for applying the block matrices $\underline{\mathbf{D}}^{\downarrow}$, $\underline{\mathbf{D}}^{\uparrow}$, $\underline{\mathbf{L}}$, and $\underline{\mathbf{U}}$. By construction $\underline{\mathbf{D}}^{\downarrow}$ and $\underline{\mathbf{D}}^{\uparrow}$ can be easily inverted using block forward and backward substitution because they are block triangular with identity blocks on their diagonals. The blocks that appear in the sparsity pattern of $\underline{\underline{\mathbf{M}}}$ (Fig. 2; right) are a direct manifestation of interactions between the layer interfaces.

While the resulting block linear system can be solved using standard matrix-splitting iterations, such as block Jacobi iteration or block Gauss-Seidel iteration (Saad, 2003), it is natural to continue to use GMRES to solve the system due to the parallel nature of applying the constituent blocks of $\underline{\underline{\mathbf{M}}}$. However, the structure of $\underline{\underline{\mathbf{M}}}$ is convenient for using a single iteration of Gauss-Seidel as a preconditioner,

$$\mathbf{P}\underline{\underline{\mathbf{M}}}\,\underline{\underline{\mathbf{u}}} = \mathbf{P}\underline{\underline{\mathbf{f}}}_s, \tag{21}$$

where the preconditioning matrix is

$$\mathbf{P} = \begin{pmatrix} \underline{\mathbf{D}}^{\downarrow} & \mathbf{O} \\ \underline{\mathbf{L}} & \underline{\mathbf{D}}^{\uparrow} \end{pmatrix}^{-1}. \tag{22}$$

In the subsequent sections, we will elaborate on the physical and numerical meanings of the constituent blocks of $\underline{\underline{\mathbf{M}}}$ and $\mathbf{P}$.

Polarization

The main novelty of the method of polarized traces is due to the polarization conditions, which are encoded in the matrices $\underline{\mathbf{A}}^{\uparrow}$ and $\underline{\mathbf{A}}^{\downarrow}$. The polarizing conditions provide a streamlined way to define an iterative solver using standard matrix splitting techniques, and thus an efficient preconditioner for Krylov methods, such as GMRES.

The polarization conditions are constructed by projecting the solution on two orthogonal sets, physically given by waves traveling upwards and downwards. Similar constructs are well-known to the geophysics community, as methods that decompose wavefields into distinct down-

[¶]This scaling is empirically deduced , under the assumption that no large resonant cavities are present in the media.

and up-going components are the backbone of several imaging techniques (see Zhang (2006) and references therein). Commonly, the decomposition is obtained using discretizations of pseudo-differential operators, which can be interpreted as separating the wavefield into a set of wave-atoms traveling in the different directions which are then propagated accordingly. Methods for decomposing and locally extrapolating directionally decomposed wavefields are well documented (Wu (1994); Collino and Joly (1995); Ristow and Ruehl (1997); de Hoop et al. (2000)).

In our case, we rewrite the decomposition condition as an integral relation between the Neumann and Dirichlet data of the wavefield, which ultimately leads to the annihilation conditions in Eq. 17. The pair composed of the Neumann and Dirichlet traces should lie within the null space of an integral operator defined on an interface, which allows the decomposition of the total wavefield into the up- and down-going components, with each having a clear physical interpretation. In particular, an up-going wavefield is a wavefield generated by a source located beneath the interface and it satisfies a radiation condition at positive infinity and a down-going wavefield is a wavefield generated by a source located above the interface and it satisfies a radiation condition at negative infinity. As detailed in Zepeda-Núñez and Demanet (2016), defining the decomposition in this manner allows us to extrapolate each component in a stable manner using an incomplete Green's integral.

The extrapolation of up-going components is performed algorithmically by the inversion of $\underline{\mathbf{D}}^{\uparrow}$ and in the same fashion the extrapolation of down-going components is performed by the inversion of $\underline{\mathbf{D}}^{\downarrow}$. Moreover, the application of the operator $\underline{\mathbf{L}}$ isolates the up-going reflections due to down-going waves interacting with the material in each subdomain, and similarly for the operator $\underline{\mathbf{U}}$.

The application of the preconditioner to a decomposed wavefield,

$$\mathbf{P}\begin{pmatrix} \underline{\mathbf{v}}^{\downarrow} \\ \underline{\mathbf{v}}^{\uparrow} \end{pmatrix} = \begin{pmatrix} (\underline{\mathbf{D}}^{\downarrow})^{-1}\underline{\mathbf{v}}^{\downarrow} \\ (\underline{\mathbf{D}}^{\uparrow})^{-1}\mathbf{r} \end{pmatrix}, \tag{23}$$

for $\mathbf{r} = \left(\mathbf{v}^{\uparrow} - \underline{\mathbf{L}}(\underline{\mathbf{D}}^{\downarrow})^{-1}\underline{\mathbf{v}}^{\downarrow}\right)$, can be physically interpreted as follows:

1. $(\underline{\mathbf{D}}^{\downarrow})^{-1}\underline{\mathbf{v}}^{\downarrow}$: extrapolate the down-going components by propagating them downwards,
2. $\mathbf{r} = \left(\mathbf{v}^{\uparrow} - \underline{\mathbf{L}}(\underline{\mathbf{D}}^{\downarrow})^{-1}\underline{\mathbf{v}}^{\downarrow}\right)$: compute the local reflection of the extrapolated field and add them to the up-going components,
3. $(\underline{\mathbf{D}}^{\uparrow})^{-1}\mathbf{r}$: extrapolate the up-going components by propagating them upwards.

Algorithms

As with the application of $\underline{\underline{\mathbf{M}}}$ in Alg. 3, we construct matrix-free methods for solving $(\underline{\mathbf{D}}^{\downarrow})^{-1}$ and $(\underline{\mathbf{D}}^{\uparrow})^{-1}$ (Algs. 4 and 5), where local solves are applied in an inherently sequential fashion. To complete the preconditioner, a matrix-free (and embarrassingly parallel) algorithm for applying $\underline{\mathbf{L}}$ is given in Alg 6. Similar algorithms for applying $\underline{\mathbf{U}}$, $\underline{\mathbf{D}}^{\uparrow}$, and $\underline{\mathbf{D}}^{\downarrow}$, as well as the complete matrix-free algorithm for applying $\underline{\underline{\mathbf{M}}}$, are provided in the Appendix.

In solving the systems for $(\underline{\mathbf{D}}^{\downarrow})^{-1}$ and $(\underline{\mathbf{D}}^{\uparrow})^{-1}$, each application of the local solver is local to each layer, which means that some communication is required to transfer solution updates from one layer to the next. The sequential nature of the method for solving these systems implies that only one set of processors, those assigned to the current layer, are working at any given stage of the algorithm. This is illustrated in Fig. 3, where each block represents a local solve and the execution path moves from left to right. As explained in Zepeda-Núñez and Demanet (2016), it is possible to apply $\underline{\mathbf{D}}^{\downarrow}$ and $\underline{\mathbf{L}}$ simultaneously, thus decreasing the number of local solves per layer.

Algorithm 4. *Downward sweep, application of* $(\underline{\mathbf{D}}^{\downarrow})^{-1}$

function $\underline{\mathbf{u}}^{\downarrow} =$ DOWNWARD SWEEP$(\ \underline{\mathbf{v}}^{\downarrow}\ )$
  $\mathbf{u}^{\downarrow,1}_{n^1} = -\mathbf{v}^{\downarrow,1}_{n^1}$
  $\mathbf{u}^{\downarrow,1}_{n^1+1} = -\mathbf{v}^{\downarrow,1}_{n^1+1}$
  for $\ell = 2 : L-1$ ***do***
    $\widetilde{\mathbf{f}}^{\ell} = \delta(z_1 - z)\mathbf{u}^{\downarrow,\ell-1}_{n^{\ell-1}} - \delta(z_0 - z)\mathbf{u}^{\downarrow,\ell-1}_{n^{\ell-1}+1}$
    $\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\widetilde{\mathbf{f}}^{\ell}$
    $\mathbf{u}^{\downarrow,\ell}_{n^{\ell}} = \mathbf{w}_{n^{\ell}} - \mathbf{v}^{\downarrow,\ell}_{n^{\ell}}$
    $\mathbf{u}^{\downarrow,\ell}_{n^{\ell}+1} = \mathbf{w}_{n^{\ell}+1} - \mathbf{v}^{\downarrow,\ell}_{n^{\ell}+1}$
  end for
  $\underline{\mathbf{u}}^{\downarrow} = \left(\mathbf{u}^{\downarrow,1}_{n^1}, \mathbf{u}^{\downarrow,1}_{n^1+1}, \mathbf{u}^{\downarrow,2}_{n^2}, ..., \mathbf{u}^{\downarrow,L-1}_{n^{L-1}}, \mathbf{u}^{\downarrow,L-1}_{n^{L-1}+1}\right)^t$
end function

Algorithm 5. *Upward sweep, application of* $(\underline{\mathbf{D}}^{\uparrow})^{-1}$

function $\underline{\mathbf{u}}^{\uparrow} =$ UPWARD SWEEP$(\ \underline{\mathbf{v}}^{\uparrow}\ )$
  $\mathbf{u}^{\uparrow,L}_{0} = -\mathbf{v}^{\uparrow,L}_{0}$
  $\mathbf{u}^{\uparrow,L}_{1} = -\mathbf{v}^{\uparrow,L}_{1}$
  for $\ell = L-1 : 2$ ***do***
    $\widetilde{\mathbf{f}}^{\ell} = -\delta(z_{n^{\ell}+1} - z)\mathbf{u}^{\uparrow,\ell+1}_{0} + \delta(z_{n^{\ell}} - z)\mathbf{u}^{\uparrow,\ell+1}_{1}$
    $\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\widetilde{\mathbf{f}}^{\ell}$
    $\mathbf{u}^{\uparrow,\ell}_{1} = \mathbf{w}^{\ell}_{1} - \mathbf{v}^{\uparrow,\ell}_{1}$
    $\mathbf{u}^{\uparrow,\ell}_{0} = \mathbf{w}^{\ell}_{0} - \mathbf{v}^{\uparrow,\ell}_{0}$
  end for
  $\underline{\mathbf{u}}^{\uparrow} = \left(\mathbf{u}^{\uparrow,2}_{0}, \mathbf{u}^{\uparrow,2}_{1}, \mathbf{u}^{\uparrow,3}_{0}, ..., \mathbf{u}^{\uparrow,L}_{0}, \mathbf{u}^{\uparrow,L}_{1}\right)^t$
end function

Algorithm 6. *Upward reflections, application of* $\underline{\mathbf{L}}$

function $\underline{\mathbf{u}}^{\uparrow} =$ UPWARD REFLECTIONS$(\ \underline{\mathbf{v}}^{\downarrow}\ )$
  for $\ell = 2 : L-1$ ***do***
    $\mathbf{f}^{\ell} = \delta(z_1 - z)\mathbf{v}^{\downarrow,\ell}_{0} - \delta(z_0 - z)\mathbf{v}^{\downarrow,\ell}_{1}$
    $\quad -\delta(z_{n^{\ell}+1} - z)\mathbf{v}^{\downarrow,\ell+1}_{0} + \delta(z_{n^{\ell}} - z)\mathbf{v}^{\downarrow,\ell+1}_{1}$
    $\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\mathbf{f}^{\ell}$
    $\mathbf{u}^{\uparrow,\ell}_{1} = \mathbf{w}^{\ell}_{1} - \mathbf{v}^{\downarrow,\ell}_{1}$
    $\mathbf{u}^{\uparrow,\ell}_{0} = \mathbf{w}^{\ell}_{0}$
  end for
  $\mathbf{f}^{L} = \delta(z_1 - z)\mathbf{v}^{\uparrow,L}_{0} - \delta(z_0 - z)\mathbf{v}^{\uparrow,L}_{1}$
  $\mathbf{w}^{L} = (\mathbf{H}^{L})^{-1}\mathbf{f}^{L}$
  $\mathbf{u}^{\uparrow,L}_{1} = \mathbf{w}^{L}_{1} - \mathbf{v}^{\downarrow,L}_{1}$

11: $\mathbf{u}_0^{\uparrow,L} = \mathbf{w}_0^L$

12: $\underline{\mathbf{u}}^{\uparrow} = \left(\mathbf{u}_0^{\uparrow,2}, \mathbf{u}_1^{\uparrow,2}, \mathbf{u}_0^{\uparrow,3}, ..., \mathbf{u}_0^{\uparrow,L}, \mathbf{u}_1^{\uparrow,L}\right)^t$

13: ***end function***

Physical intuition

We deliberately present the preconditioner in a purely algebraic fashion, as it is instructive for implementing the method. However, there is a physical interpretation of the steps in the preconditioner, which we describe below.

As alluded to previously, the application of the preconditioner, and in particular the block back-substitution in Algs. 8 and 9, can be seen as a sequence of depth extrapolation steps. Indeed, lines 4 and 5 in Alg. 4 are the discrete counterpart of the incomplete Green's integral defined by

$$u^{\downarrow}(\mathbf{x}) = \int_{\Gamma_{\ell-1,\ell}} \left(G^{\ell}(\mathbf{x},\mathbf{y})\partial_z u^{\downarrow}(\mathbf{y}) - \partial_z G^{\ell}(\mathbf{x},\mathbf{y}) u^{\downarrow}(\mathbf{y})\right) dS_y, \tag{24}$$

which is equivalent to the Rayleigh integral used to extrapolate a wavefield measured in surface towards the interior of the Earth by Berkhout (1980). Likewise, Lines 4 and 5 of Alg. 5 are the discrete counterpart to an up-going discrete Green's integral.

The quality of the extrapolation depends directly on the quality of the approximation of the local Green's function G^{ℓ} with respect to the global Green's function. In the reductive case, if the local Green's function is precisely the global Green's function, the method will converge in two iterations, see Gander (2006). However, this is equivalent to solving the global problem, which is prohibitively expensive. Instead, we compute a local approximation of the Green's function such that the Green's representation formula is valid within the layer only, not globally. As expected with domain decomposition methods, incorrect local approximations introduce numerical artifacts, which are typically due to truncating the domain in a manner that is inconsistent with the underlying physics. In the method of polarized traces, these issues are mitigated with judicious use of high-order absorbing boundary conditions in the form of PML's. As a physical consequence, the local Green's function can only see local features within a particular layer. Far-field interactions, reflections induced by material changes in the other layers, will not be observed by the local Green's function and must be handled iteratively, by sequentially sweeping through the domains.

An important consequence of the Green's integral representation is that it completely eliminates the difficulties that most domain decomposition methods have with seamlessly connecting sub-domains together. Rather than assigning data dependent boundary conditions, the coupling is performed using potentials defined on the physical interfaces, and the absorbing boundary conditions in an extended domain effectively dampen spurious reflections. The transmission conditions given by the discrete Green's representation formula are algebraically exact, thus there is no need for tuning parameters.

PARALLELIZATION STRATEGIES

The computational effort needed to solve industrial scale 3D problems requires aggressive parallelization and optimization of the algorithm. To obtain a scalable implementation, the algorithm and code must be designed to balance the utilization and occupancy of three key resources: CPU, memory, and communication network. In this section, we describe our parallel implementation of the method of polarized traces, with a focus on maximizing the utilization of these resources.

In the previous section, we formally introduced a matrix-free approach for preconditioning the SIE system on layer interfaces. However, this approach still relies on local solves that are implemented using a direct solver. It is possible to use specially designed iterative local solvers by nesting the method of polarizes traces within each layer (Zepeda-Núñez and Demanet, 2015) or a recursive version of the sweeping factorization (Liu and Ying, 2015). However, such approach would require a complicated code with a very carefully implemented communication pattern. For simplicity, and to broaden the portability of the framework, we use a hybrid approach, where the local solves use off-the-shelf numerical linear algebra libraries and the polarization is matrix-free. We will address the parallelism on two fronts: parallelism by layer and parallelism within the layers.

Pipelining

First, we address parallelism due to the layer decomposition. Primarily, the parallelism across layers is due to the SIE and the preconditioner used to help solve it. There are five trivially parallel (by layer) applications of the local solver: four due to $\underline{\underline{\mathbf{M}}}$ and one due to the appearance of $\underline{\mathbf{L}}$ in the preconditioner. However, in the preconditioner application there are sequential bottlenecks due to the applications of $(\underline{\mathbf{D}}^{\uparrow})^{-1}$ and $(\underline{\mathbf{D}}^{\downarrow})^{-1}$ via block back-substitution. Despite the trivial parallel nature of the other local solver applications, applying the preconditioner using Algs. 4 and 5 permits work to be done on only one layer at a time, thus forcing the majority of the computer to remain idle. This is illustrated in the top half of Fig. 3, where each blue box represents a local solve and algorithm execution moves from left to right. Supposing that each local solve costs $\gamma(n)$ time, then following Fig. 3, each GMRES iteration can be performed in $5\gamma(n) + 2L\gamma(n)$, ignoring communication costs.

To alleviate the sequential bottleneck, we leverage the fact that seismic problems have thousands of right-hand sides and introduce pipelining. Pipelining allows us to process multiple right-hand sides simultaneously, each at different levels of progress through the sweeps, which helps balance the computational load on the layers, reducing the idle time and increasing the computational efficiency. The pipelining principle is demonstrated in the bottom half of Fig 3, where the boxes represent a local solve and the blue, green, and orange colors indicate distinct right-

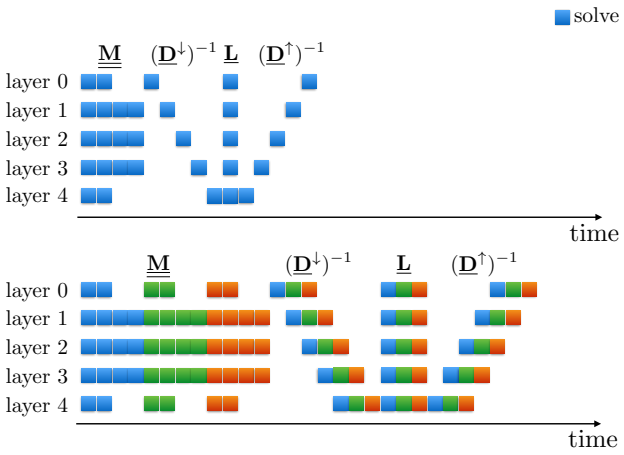

Figure 3: Sketch of the load of each node in the GMRES iteration.

hand sides. Pipelining allows the layers to perform work for different right-hand sides simultaneously. Indeed, as long as there are at least $2L$ right-hand sides, the pipeline can be completely full and all available compute resources will be occupied. For the pipelined algorithm, again disregarding communication costs, the runtime of a GMRES iteration is $5R\gamma(n)+2(L+R)\gamma(n)$. Recalling that $L \sim n$, $R \sim n$, and $\gamma(n) = \mathcal{O}(\alpha_{\text{pml}}^2 n^2 \log n)$, the cost ratio for solving R right-hand sides compared to one right-hand side is constant,

$$\frac{5R\gamma(n) + 2(L+R)\gamma(n)}{5\gamma(n) + 2L\gamma(n)} = \mathcal{O}(1). \tag{25}$$

One of the advantages of the method of polarized traces is that the memory requirement to store the intermediate representation of solution is lower than other methods, because it requires solutions for the degrees of freedom involved in the SIE only. Thus, for each right-hand side, only N/q data need to be stored, where q is the thickness of the interface. This reduction in storage, when combined with the reduced storage due to the relatively small number of GMRES iterations required for convergence, yields a smaller memory footprint for the outer GMRES iteration than methods requiring to update the full volume. It is possible to further reduce the memory footprint by using Bi-CGSTAB instead of GMRES, keeping the computational cost almost constant (Zepeda-Núñez and Demanet, 2015).

Local Solves: Parallel Multi-frontal Methods

The method polarized traces is a highly modular framework for solving the Helmholtz equation: in practice, one can use any existing algorithm or package for solving linear equations to perform the local solves, including the method of polarized traces itself (Zepeda-Núñez and Demanet, 2015). To obtain good parallel performance, we use a high-performance distributed linear algebra library to solve the local problems within each layer. Due to the sparsity pattern of the linear system at each layer, a typical recipe for the local solves is to re-order the degrees of freedom to increase stability and reduce numerical fill-in, perform a multi-frontal factorization, and solve the resulting factorized system with forward- and backward-substitution, often called triangular solves. There exists a myriad of techniques to parallelize the multi-frontal factorization and the triangular solves (see Davis et al. (2016) for a recent and extensive review of different techniques for solving sparse systems). A popular approach is to use supernodal elimination trees (see Ashcraft et al. (1987)) defined through nested dissection, which results in highly scalable factorizations (Gupta et al., 1997), albeit with less efficient triangular solves Joshi et al. (1997). To avoid the poor scalability of dense triangular solves due to this approach, Raghavan (1998) introduced a scheme called *selective inversion*, which is applied by Poulson et al. (2013) specifically for the Helmholtz problem. Although very efficient, using the techniques mentioned before would require lengthy and complex code, we use instead off-the-shelf libraries, which can be effortlessly changed if necessary.

For the results in this paper, we use STRUMPACK (Rouet et al., 2015) to perform the local solves. STRUMPACK is a state-of-the-art distributed sparse linear solver library that relies on supernodal factorizations,[‖] a 2D block-cyclic distribution of the matrix, and a static mapping technique to assign task to MPI processes based on *proportional mapping* (Pothen and Sun, 1993) to achieve good parallel performance. The implementation is competitive with other distributed linear algebra solvers with liberal licenses, such as SuperLU-DIST (cf. Li and Demmel (2003)) and MUMPS (cf. Amestoy et al. (2001)), while providing the user more freedom to arrange the distribution of the matrix and right-hand sides, within a distributed memory enviroment, in a manner that is optimal for specific applications.

A strong advantage of STRUMPACK is that the factorization and solve processes can be accelerated using compressed linear algebra, in particular HSS compression with nested bases, using randomized sampling techniques. In general, using compressed formulations allows a reduced memory footprint and in some cases faster algorithms. For the high-frequency regime, it is known that solvers based on compression techniques do not provide a lower asymptotic complexity, due to the fact that the ranks of the off-diagonal blocks are frequency dependent (Engquist and Zhao, 2014). However, these solvers still tend to provide smaller memory footprints for the cases we consider in this paper, albeit, with much bigger runtimes. Therefore, we deliberately do not considered the performance of the adaptive compression in this paper and leave such treatment for future work.

The STRUMPACK solver addresses parallelism within

[‖] It uses a ULV factorization when the compression is turned on (Chandrasekaran et al., 2006; Xia, 2013)

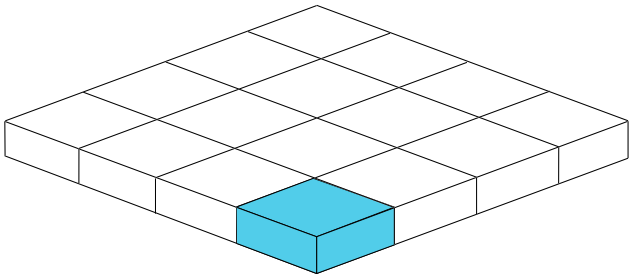

Figure 4: Sketch of the decomposition of the degrees of freedom of the slabs in cubes.

layers in two ways: classical distribution of tasks with MPI and synchronous processing within each task with OpenMP, which we exploit for the results presented in our numerical results. STRUMPACK's hybrid parallelism model allows us to maximize the utilization of computational resources. We have designed the distribution of MPI tasks to exploit highly asynchronous communication patterns, thus reducing the communication time substantially.

Finally, in most of the experiments shown in the sequel, we process at most one right-hand side per layer. It is possible to solve more than one right-hand side per layer, and take advantage of BLAS3 routines. Moreover, a quick inspection to the algorithms within the preconditioner, shows that the right-hand sides are sparse. Indeed, the sources are supported on the interfaces, and the solutions are only needed at the boundaries. In principle, it is possible to take advantage of the sparsity of the solution and the right-hand-side to reduce the constants by removing some branches from the elimination tree. These techniques would certainly reduce the constants but they would have little impact on the asymptotic scaling; hence, they were not explored in this study.

Communication

As is common in massively parallel applications, there is a bottleneck due to the communication between parallel tasks, which is strongly dependent on the distribution of the tasks on the cluster. For this implementation, we assume a very simple topology for the distribution of unknowns. We distribute the parallel tasks following the layer structure. For each slab of unknowns, we assign $\mathcal{O}(n^2)$ tasks, and using MPI directives enforce that the tasks are contiguous within physical computing nodes. Each slab is divided into $\mathcal{O}(n^2)$ cubes, as illustrated in Fig. 4, and the parallel tasks associated with that slab are divided evenly and contiguously amongst the cubes. Each cube contains a contiguous block of the solution, as shown in Fig. 4, and the associated entries of the local matrix.

Within the slab, the topology is designed so that each cube only communicates with its neighboring cubes in the same slab, generally inside the local solver. Across slabs,

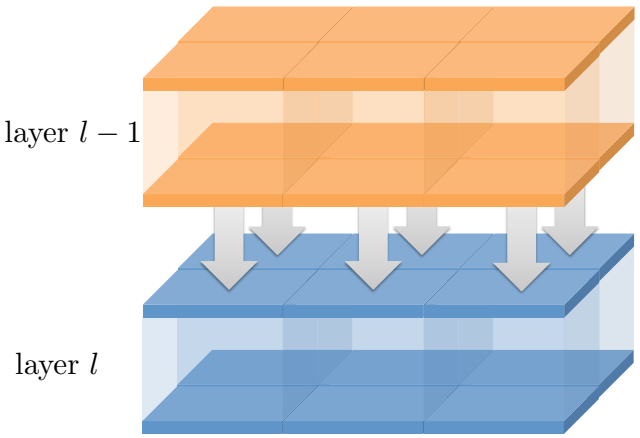

Figure 5: Sketch of the asynchronous communication between slabs.

cubes only communicate with the cube in the same position on the adjacent slabs immediately above and below it, as illustrated in Fig. 5. Under this particular topology, we can distinguish two main communications bottlenecks:

- the communication between parallel tasks within the distributed linear algebra solver, and
- the communication of the boundary data between slabs, during the application of the preconditioner.

As a consequence of using third-party distributed linear algebra solvers, we have little control over the communication pattern, particularly because STRUMPACK uses MC64 (Duff and Koster, 2001), for enhancing stability and ParMetis (Karypis and Kumar, 1998) to optimally reorder the matrix in order to reduce fill-in during the factorization. To have the desired distribution of the degrees of freedom among the cubes, we reorder the matrix with a Z ordering, so that smallest division corresponds exactly to the degrees of freedom within a cube. Then, the matrix is then assembled and passed to the linear solver in a distributed fashion.

Communication between slabs is a product of the application of the preconditioner in Eq. 22, in which back- and forward-substitution are used to apply $(\underline{\mathbf{D}}^{\uparrow})^{-1}$ and $(\underline{\mathbf{D}}^{\downarrow})^{-1}$. Algs. 8 and 9) require a local solve in each slab, followed by communication of the trace information to the next slab, in which another local solve is performed. This operation is repeated until all the slabs are visited within the sweep.

By dividing the slabs into cubes, the communication of the trace information between slabs is very efficient. Given that each cube communicates with the cube directly above and below, it is possible to perform asynchronous point-to-point communication between the cubes of two adjacent slabs as shown in Fig. 5. This allows the communication to be performed in nearly constant time, up to saturation of the network. Moreover, the trace infor-

mation is already distributed for distributed assembly of the right-hand side within the subsequent slab. As stated before, it is possible to use topologies better suited for the multi-frontal solver, such as the one due to Poulson et al. (2013). However, such an implementation requires a very precise understanding of the re-ordering mechanism within the solver, which we do not have for black-box solvers.

COMPLEXITY OF POLARIZED TRACES

The run-time complexity of the polarized traces preconditioner is driven by the costs of computation and communication. Achieving optimal performance requires delicately balancing the parallel distribution of the problem depending on the characteristics of the target HPC system. In this section we develop models for computation and communication costs, which guide problem parameter selection in HPC environments.

Computation

As before, each layer has $\mathcal{O}((n_z + \alpha_{\text{pml}}) \times n^2)$ grid points, i.e., they are n_z grid points thick with α_{pml} additional points due to the PML. We have that $n_z = \mathcal{O}(1)$ because $L \sim n$, which implies that we are solving a quasi-2D problem. The additional cost is due only to the points used to implement the absorbing boundary conditions. When applying 2D nested dissection to the quasi-2D problem we have $\mathcal{O}(\alpha_{\text{pml}} n)$ degrees of freedom in the biggest front, thus leading to a complexity of $\mathcal{O}(\alpha_{\text{pml}}^3 n^3)$ for the factorization of the systems local to each layer and $\mathcal{O}(\alpha_{\text{pml}}^2 n^2 \log n)$ for the application of the triangular solve (Duff and Reid, 1983). Sequentially, the complexity of Alg. 1 is $\mathcal{O}(\alpha_{\text{pml}}^3 N^{4/3})$, but given that the loop in line 2 of Alg. 1 is embarrassingly parallelizable, Alg. 1 can be performed in $\mathcal{O}(\alpha_{\text{pml}}^3 N)$ time**. Due to the sequential nature of Algs. 4 and 5, applying the preconditioner requires $2L$ local solves per iteration, applied sequentially. Consequently, the total complexity for the application of the preconditioner is $\mathcal{O}(\alpha_{\text{pml}}^2 N \log N)$. For $\alpha_{\text{pml}} \sim \log n$, at most $\mathcal{O}(\log n)$ iterations are empirically needed to converge, thus the complexity of the solver is linear (up to poly-logarithmic factors), provided that $L \sim n$ and that the number of iterations for convergence grows slowly.

It is possible to relax the restriction that $L \sim n$, instead allowing $L \sim n^b$ where $b < 1$. However, in this regime, maintaining the overall linear complexity requires that we exchange the multi-frontal solver for an iterative solver (Liu and Ying, 2015; Zepeda-Núñez and Demanet, 2015). The main disadvantage of this approach is that it reduces the possible parallelism due to using multi-frontal solvers, which makes an efficient implementation of the pipelining difficult and makes the communication patterns more complicated.

**As it will be shown in the numerical experiments, this scaling can be further reduced due to the parallelism at the level of the multi-frontal solver.

Pipelining

We introduced pipelining of R right-hand sides to alleviate the sequential nature of applying the preconditioner. To understand the run-time impact of pipelining multiple right-hand sides, we consider its impact on the complexity of applying $\underline{\underline{\mathbf{M}}}$ and applying the preconditioner. Given that the run-time cost of each local solve is $\mathcal{O}(\alpha_{\text{pml}}^2 n^2 \log n)$ and recalling that applying $\underline{\underline{\mathbf{M}}}$ is embarrassingly parallel, the cost of applying $\underline{\underline{\mathbf{M}}}$ to R right-hand sides is $\mathcal{O}(\alpha_{\text{pml}}^2 R n^2 \log n)$. As long as $R \sim L \sim n$, applying the preconditioner costs $\mathcal{O}(L \alpha_{\text{pml}}^2 n^2 \log n)$. However, when $R \gtrsim L$, the additional right-hand-sides are treated sequentially, resulting in a cost of $\mathcal{O}(\alpha_{\text{pml}}^2 R n^2 \log n)$. Using the fact that $L \sim n$ and that $N = n^3$, we obtain the advertised runtime of $\mathcal{O}(\alpha_{\text{pml}}^2 \max(1, R/L) N \log N))$.

Communication

We treat the distributed linear algebra solver as a black box, and therefore do not analyze the costs of communication within the local solve. Thus, we only consider the cost of communication due to the global solve, that is the costs of communicating between subdomains across layers. We assume that that each subdomain has fast access to its corresponding patch of both the R wavespeeds and the R sources. Moreover, we assume that each subdomain assembles and stores its portion of the global solution*.

The offline stage of the algorithm, assembly of the local matrices and the local factorizations, is embarrassingly parallel under the assumptions described above. The online part, has three stages:

- the preparation of the right-hand side (Lines 2-8 in Alg. 2);
- the solve of the SIE (Lines 9 in Alg. 2); and
- the assembly of the global solution (Lines 10-14 in Alg. 2).

Of these, the first and third stages require no communication under the above assumptions.

Solving the SIE has two main phases: the application of $\underline{\underline{\mathbf{M}}}$ and the application of the preconditioner. Applying $\underline{\underline{\mathbf{M}}}$, as shown in Alg. 3, is an embarassingly parallel operation with zero communication. On the other hand, applying the preconditioner, which is fully sequential, requires communication of $\mathcal{O}(n^2)$ unknowns from one layer to the next. In our implementation, these unknowns are distributed evenly between $\mathcal{O}(n^2)$ MPI tasks. Using a point-to-point communication strategy, each of the MPI tasks assigned to a layer only communicates with one corresponding MPI task in the adjacent (above and below) layers, as illustrated in Fig. 5. Thus, by exploiting asynchronous communication, the communication between layers can be performed in $\mathcal{O}(1)$ time up to saturation of the bandwidth, which is asymptotically negligible

*As a consequence, computation of imaging conditions can be performed without incurring in extra communication cost.

with respect to solving the local linear systems. This communication must be performed $\mathcal{O}(L)$ times during each sweeping operation in Algs. 4 and 5. Consequently, total communication cost of applying the preconditioner is $\mathcal{O}(n)$, up to saturation of the bandwidth.

NUMERICAL EXPERIMENTS

In this section, we present the results of several numerical experiments used to verify the complexity described above. In particular, we demonstrate the performance of the 3D preconditioner in various heterogeneous media for a single source and then illustrate the impact of pipelining on parallel performance. Our polarized traces implementation is written in C and compiled with the 2015 Intel compiler suite. The current implementation is uses IEEE double precision floating point. To perform the local solves, we use STRUMPACK v1.1.0 with Intel MKL support for fast linear algebra operations. The preconditioner is parallelized with MPI and STRUMPACK is parallelized with both MPI and OpenMP. The experiments were performed on Total's "Laure" SGI ICE-X cluster, where each computing node contains two 8-core Intel Sandy Bridge processors, 64GB of RAM, and are connected with an Infiniband interconnect.

Homogeneous media

First, we demonstrate the effectiveness of the preconditioner by solving the Helmholtz problem in homogeneous media. With no reflectors in the medium, the convergence of the algorithm is only dependent on the frequency and the quality of the absorbing boundary condition at the layer interfaces. In this experiment, as well as the subsequent experiments, we hold α_{pml} constant, with sufficient points to minimize artificial reverberations while simultaneously keeping the number of iterations low. For higher frequencies, to preserve the low iteration count we would need to scale α_{pml} as $\mathcal{O}(\log n)$.

In this experiment, we test 4 problem sizes, $n = 50$, 100, 200, and 400, which corresponds to frequencies of 8, 16, 32, and 64 Hz. Source frequency is scaled with problem size to stay in the high-frequency regime and sources are assumed to be point sources. The number of layers is also scaled with the problem, $L = 5$, 10, 20, and 40. Due to memory limitations on the computing node, in some cases the nodes were saturated before all cores could be assigned to an MPI task. For these cases, we allow the remaining cores to be used for vectorized processing with OpenMP. The outer GMRES iteration is run until the residual is reduced to 10^{-7}, which is excessive in a production, single-precision environment; however, it shows the favorable behavior of the solver under more challenging conditions. Lower tolerances can produce misleading results when frequencies are not high enough, thus only revealing a pre-asymptotic behavior. For each configuration, we report the wall-clock times for initialization, matrix assembly, matrix factorization, and total online time for $R = 1$ and $R = L$ with pipelining. Additionally, we track the number of GMRES iterations required to achieve the desired convergence.

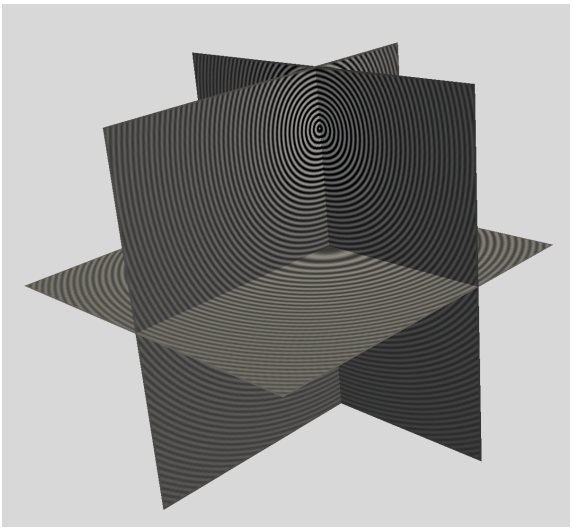

Figure 6: Solution of the Helmholtz equation, at 64 Hz, in constant wavespeed.

The solution at 64 Hz is provided in Fig. 6. The full results of the experiment are given in in Table 2 and the observed run-times, compared to the theoretical run-times for $R = 1$ and $R = L$ pipelined right-hand sides are shown in Fig. 7 and Fig. 8, respectively. Only when $R \geq L$ does the theoretical scalability break the linear threshold, however, in both cases, the method of polarized traces scales better than the theoretical scaling, which we attribute to optimizations and parallelism in the local solver. Of note in Table 2, the number of GMRES iterations grows very slowly with the frequency, even when we do not scale α_{pml} optimally. In the experiments where hybrid parallelism (MPI-OpenMP) is used, the run-times are reduced almost linearly for medium-sized problems, however the improvements fade as the size of the problem increases. This behavior is due to the linear solver, where for large problems the memory access time in the triangular solves becomes dominant, reducing the parallelism.

Smooth heterogeneous media

Using the same configurations as above we solve the Helmholtz problem in the smoothed random media shown in Fig. 9 in order to demonstrate the effectiveness of the solver in heterogeneous media. This test demonstrates that the method is particularly robust to media where the rays can bend and develop caustics. The solution for the configuration equivalent to that of Fig. 6 is given in Fig. 10. In Fig. 10, it is clear that the features of the model are comparable to the wavelength used, thus the solution presents interference, caustics, non-spherical wavefronts. Table 3 contains the complete experimental results, where we observe that the variation in the media has little real effect

N	50^3	100^3	100^3	200^3	200^3	400^3	400^3	400^3
L	5	10	10	20	20	40	40	40
MPI Tasks	5	10	10	80	80	640	640	640
OpenMP Threads per Task	1	1	2	1	2	1	2	3
Total Cores	5	10	20	80	160	640	1280	1920
Total Nodes	1	1	2	5	10	80	80	128
Single rhs								
# GMRES Iterations	4	4	4	5	5	6	6	6
Initialization [s]	0.2	1.0	0.9	6.9	4.4	18.9	18.9	18.4
Factorization [s]	4.1	41.1	21.9	153.2	78.3	320.5	200.1	148.6
Online [s]	4.0	39.2	22.6	182.0	109.7	696.6	401.4	315.5
Average GMRES [s]	0.9	8.4	4.8	32.0	19.2	103.5	59.3	46.6
Pipelined rhs								
R (number of rhs)	5	10	10	20	20	40	40	40
Online [s]	15.8	189.4	106.2	1255.5	668.5	3994.2	2654.4	1878.1
Average GMRES [s]	3.4	40.6	22.7	223.8	118.6	599.9	401.0	283.0
Online per rhs [s]	3.2	18.9	10.6	62.8	33.4	99.9	66.4	47.0
Average GMRES per rhs [s]	0.7	4.1	2.3	11.2	5.9	15.0	10.0	7.1

Table 2: Runtime (in seconds) for one and several right-hand sides for the solution of the Helmholtz equation using an homogeneous model.

on the run-time or convergence properties. In this case, even if the rays bend, the preconditioner does a remarkable job at tracking the rays in the correct direction and propagating them accordingly. The timings are given in in Figs. 7 and Fig 8.

Fault model

In general, iterative methods are very sensitive to discontinuous media. At high frequency, interaction with short-wavelength structures, such as discontinuities, increases the number of reflections. Each additional reflection requires additional iterations to convergence, hindering the efficiency of iterative methods.

Using the same configuration as for the homogeneous model, with the discontinuous velocity given in Fig. 11, we demonstrate that the method of polarized traces deteriorates only marginally as a function of the frequency and number of subdomains. A solution at 64 Hz is given in Fig. 12 and the run-time scalability is again given in Figs. 7 and 8. As shown in Table 4, we observe the same behavior as in the previous cases and that the strong reflection is handed efficiently by the transmission and polarizing conditions.

SEAM model

Beyond mere sensitivity to discontinuities of the medium, iterative solvers are highly sensitive to the roughness and heterogeneity of the velocity model, due to the great amount of interactions, reflections, and drastic changes of direction of waves due to high gradients in the wavespeed. However, the performance method of polarized traces degrades only marginally for highly heterogeneous media, excepting resonant cavities. We demonstrate this desirable performance on the SEAM Phase I velocity model (Fig. 13). In this experiment, we test 3 problem sizes, $N = 0.65$M, 5.16M, and 41.2M degrees of freedom, which use $L = 12, 24$, and 48 layers, respectively. The remainder of the experimental setup is unchanged.

As seen in the data in Table 5 and plotted in Figs. 15 and 16, the run-times are sub-linear with respect to the total number of unknowns. Interestingly, we also observe a sub-linear run-time in the offline stages of the algorithm, which we attribute to the parallelism in the multi-frontal factorization. This is expected, as the factorization is more computationally intensive than memory intensive. In the more memory-intensive, and thus less parallel, solve phase, we still see an improvement over the theoretical curve, but the improvement is less pronounced.

Finally, for the largest test case, we demonstrate the impact of pipelining by comparing the scalability of our method with the theoretical scalability, as a function of R. As shown in Fig. 17, experimental results indicate that we obtain the expected scalability. Slight divergence from the theoretical curve is expected once the pipeline is fully saturated because the theoretical curve does not take into account the cost of filling and flushing the pipeline.

Finally, Fig. 17 depics the behavior of the pipelining as we increase the number of right-hand sides. As expected, as we add more and more righ-hand sides to be solved simultaneously the runtime per right-hand side decreases, until the pipeline is full, when the average runtime remains almost constant.

N	50^3	100^3	100^3	200^3	200^3	400^3	400^3	400^3
L	5	10	10	20	20	40	40	40
MPI Tasks	5	10	10	80	80	640	640	640
OpenMP Threads per Task	1	1	2	1	2	1	2	3
Total Cores	5	10	20	80	160	640	1280	1920
Total Nodes	1	1	2	5	10	80	80	128
Single rhs								
# GMRES Iterations	5	5	5	5	5	6	6	6
Initialization [s]	0.2	1.1	1.0	7.3	4.6	21.3	21.2	20.8
Factorization [s]	3.8	41.1	21.8	156.0	79.4	323.7	204.5	151.5
Online [s]	4.6	45.9	26.1	202.2	106.9	717.0	400.1	314.5
Average GMRES [s]	0.8	8.1	4.6	35.5	18.7	106.4	59.2	46.5
Pipelined rhs								
R (number of rhs)	5	10	10	20	20	40	40	40
Online [s]	17.1	225.1	118.8	1260.9	650.2	4085.0	2714.8	1872.1
Average GMRES [s]	3.0	39.8	20.9	223.6	115.6	613.3	409.2	281.9
Online per rhs [s]	3.4	22.5	11.9	63.0	32.5	102.1	67.9	46.8
Average GMRES per rhs [s]	0.6	4.0	2.1	11.2	5.8	15.3	10.2	7.0

Table 3: Runtime (in seconds) for one and several right-hand sides for the solution of the Helmholtz equation for the smooth model in Fig. 9.

N	50^3	100^3	100^3	200^3	200^3	400^3	400^3	400^3
L	5	10	10	20	20	40	40	40
MPI Tasks	5	10	10	80	80	640	640	640
OpenMP Threads per Task	1	1	2	1	2	1	2	3
Total Cores	5	10	20	80	160	640	1280	1920
Total Nodes	1	1	2	5	10	80	80	128
Single rhs								
# GMRES Iterations	4	5	5	5	5	6	6	6
Initialization [s]	0.4	1.1	1.0	7.3	4.7	20.4	20.3	21.0
Factorization [s]	3.8	40.4	22.1	152.2	79.9	317.6	199.5	152.5
Online [s]	3.7	46.2	26.2	188.5	109.8	713.2	395.8	315.6
Average GMRES [s]	0.8	8.1	4.6	33.0	19.2	106.2	58.7	46.5
Pipelined rhs								
R (number of rhs)	5	10	10	20	20	40	40	40
Online [s]	13.7	226.7	122.4	1222.7	647.1	4031.6	2710.6	1838.9
Average GMRES [s]	2.9	40.1	21.6	216.5	114.7	605.0	409.9	276.3
Online per rhs [s]	2.7	22.7	12.2	61.1	32.4	100.8	67.7	46.0
Average GMRES per rhs [s]	0.6	4.0	2.2	10.8	5.7	15.1	10.2	6.9

Table 4: Runtime (in seconds) for one and several right-hand sides for the solution of the Helmholtz equation for the fault model.

N	$6.51 \cdot 10^5$	$5.16 \cdot 10^6$	$4.12 \cdot 10^7$	$4.12 \cdot 10^7$
L	12	24	48	48
MPI Tasks	12	48	384	384
OpenMP Threads per Task	1	2	2	3
Total Cores	12	96	768	1152
Total Nodes	1	6	77	77
Single rhs				
# GMRES Iterations	4	5	6	6
Initialization [s]	0.6	2.3	10.4	10.7
Factorization [s]	15.2	46.5	111.4	97.9
Online [s]	21.4	85.6	269.8	228.4
Average GMRES [s]	4.6	14.9	40.0	33.7
Pipelined rhs				
R (number of rhs)	12	24	48	48
Online [s]	106.3	474.8	1527.1	1415.4
Average GMRES [s]	22.8	83.9	229.4	212.9
Online per rhs [s]	8.8	19.8	31.8	29.5
Average GMRES per rhs [s]	1.9	3.5	4.8	4.4

Table 5: Runtime (in seconds) for one and several right-hand sides for the solution of the Helmholtz equation for the SEAM model.

CONCLUSION

We have presented a new and efficient solver for the 3D high-frequency Helmholtz equation in heterogeneous media. The solver achieves a sub-linear runtimes by leveraging the solution of batches of right-hand sides properly pipelined and parallelism. The method presented in this paper broadens the applicability of parallel direct methods by embedding them in a domain decomposition method, whose rate of convergence is independent of the frequency.

Finally, the main limitations of the present method are large resonant cavities presenting high contrasts, for which the number of reflections can be large implying that the number of iterations for convergence can still be high.

ACKNOWLEDGMENTS

This worked was sponsored by Total SA. LD is also sponsored by AFOSR grant FA9550-17-1-0316, ONR grant N00014-16-1-2122, and NSF grant DMS-1255203. The authors are grateful to Mike Fehler and Total SA for their permission to use the 3D SEAM model.

REFERENCES

Amestoy, P., C. Ashcraft, O. Boiteau, A. Buttari, J.-Y. L'Excellent, and C. Weisbecker, 2015, Improving multifrontal methods by means of block low-rank representations: SIAM Journal on Scientific Computing, **37**, A1451–A1474.

Amestoy, P., R. Brossier, A. Buttari, J.-Y. L'Excellent, T. Mary, L. Métivier, A. Miniussi, and S. Operto, 2016, 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.

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

Ashcraft, C., R. Grimes, J. Lewis, B. Peyton, H. Simon, and P. Bjrstad, 1987, Progress in sparse matrix methods for large linear systems on vector supercomputers: The International Journal of Supercomputing Applications, **1**, 10–30.

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

Babuska, I., and U. Banerjee, 2012, Stable generalized finite element method (SGFEM): Computer Methods in Applied Mechanics and Engineering, **201204**, 91 – 111.

Babuska, I., F. Ihlenburg, E. T. Paik, and S. A. Sauter, 1995, A generalized finite element method for solving the H elmholtz equation in two dimensions with minimal pollution: Computer Methods in Applied Mechanics and Engineering, **128**, 325–359.

Babuska, I., and J. M. Melenk, 1997, The partition of unity method: International Journal for Numerical Methods in Engineering, **40**, 727–758.

Bebendorf, M., 2008, Hierarchical matrices: A means to efficiently solve elliptic boundary value problems: Springer-Verlag, volume **63** *of* Lecture Notes in Computational Science and Engineering (LNCSE). (ISBN 978-3-540-77146-3).

Bérenger, J.-P., 1994, A perfectly matched layer for the absorption of electromagnetic waves: Journal of Computational Physics, **114**, 185–200.

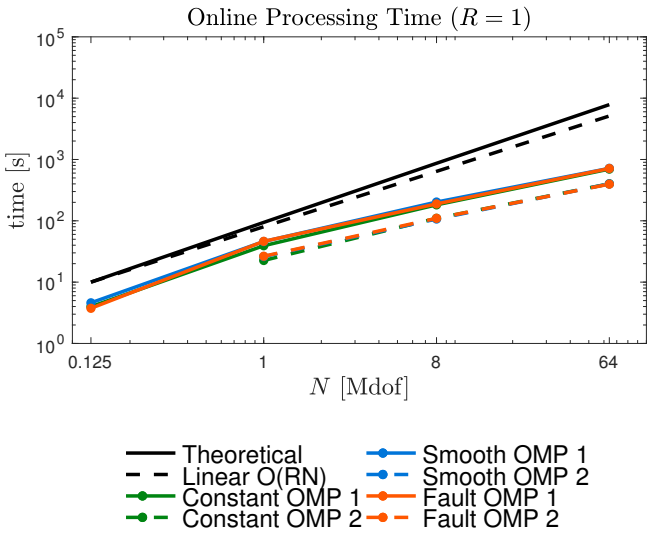

Figure 7: Observed run-time as function of N for homogeneous media (green), smooth heterogeneous media (blue) and "fault" model (orange), with pure MPI (solid) and hybrid MPI-OpenMP (dashed) for $R = 1$ right-hand sides. For comparison, theoretical scaling of polarized traces algorithm is given (solid black) as well as linear scaling (dashed black).

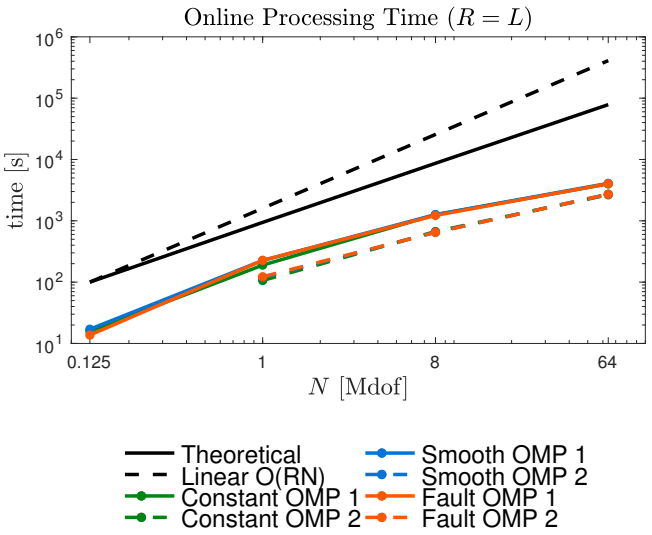

Figure 8: Observed run-time as function of N for homogeneous media (green), smooth heterogeneous media (blue) and "fault" model (orange), with pure MPI (solid) and hybrid MPI-OpenMP (dashed) for $R = L$ right-hand sides. For comparison, theoretical scaling of polarized traces algorithm is given (solid black) as well as linear scaling (dashed black).

Berkhout, A. J., 1980, Seismic migration: Imaging of acoustic energy by wave field extrapolation: Elsevier.

Betcke, T., and J. Phillips, 2012, Approximation by dominant wave directions in plane wave methods: Technical report.

Boerm, S., L. Grasedyck, and W. Hackbusch, 2006, Hierarchical matrices: Max-Planck- Institute Lecture Notes.

Bollhöfer, M., M. Grote, and O. Schenk, 2009, Algebraic multilevel preconditioner for the Helmholtz equation in heterogeneous media: SIAM Journal on Scientific Computing, **31**, 3781–3805.

Boubendir, Y., 2007, An analysis of the BEM-FEM non-overlapping domain decomposition method for a scattering problem: Journal of Computational and Applied Mathematics, **204**, 282 – 291. (Special Issue: The Seventh International Conference on Mathematical and Numerical Aspects of Waves (WAVES05)).

Boubendir, Y., X. Antoine, and C. Geuzaine, 2012, A quasi-optimal non-overlapping domain decomposition algorithm for the Helmholtz equation: Journal of Computational Physics, **231**, 262 – 280.

Brandt, A., and I. Livshits, 1997, Wave-ray multigrid method for standing wave equations: Electronic Transactions on Numerical Analysis, **6**, 162–181.

Calandra, H., S. Gratton, X. Pinel, and X. Vasseur, 2013, An improved two-grid preconditioner for the solution of three-dimensional Helmholtz problems in heterogeneous media: Numerical Linear Algebra with Applications, **20**, 663–688.

Cessenat, O., and B. Després, 1998, Application of an ultra weak variational formulation of elliptic pdes to the two-dimensional Helmholtz problem: SIAM Journal on Numerical Analysis, **35**, 255–299.

Chan, T. F., and T. P. Mathew, 1994, Domain decomposition algorithms: Acta Numerica, **3**, 61–143.

Chandrasekaran, S., M. Gu, and T. Pals, 2006, A fast ULV decomposition solver for hierarchically semiseparable representations: SIAM Journal on Matrix Analysis and Applications, **28**, 603–622.

Chen, Y., 1997, Inverse scattering via Heisenberg's uncertainty principle: Inverse Problems, **13**, 253.

Chen, Z., and X. Xiang, 2013a, A source transfer domain decomposition method for Helmholtz equations in unbounded domain: SIAM Journal on Numerical Analysis, **51**, 2331–2356.

———, 2013b, A source transfer domain decomposition method for Helmholtz equations in unbounded domain part II: Extensions: Numerical Mathematics: Theory, Methods and Applications, **6**, 538–555.

Collino, F., S. Ghanemi, and P. Joly, 2000, Domain decomposition method for harmonic wave propagation: a general presentation: Computer Methods in Applied Mechanics and Engineering, **184**, 171 – 211.

Collino, F., and P. Joly, 1995, Splitting of operators, alternate directions, and paraxial approximations for the three-dimensional wave equation: SIAM Journal on Scientific Computing, **16**, 1019–1048.

Davis, T. A., 2004, Algorithm 832: UMFPACK v4.3—an unsymmetric-pattern multifrontal method: ACM Transactions on Mathematical Software, **30**, 196–199.

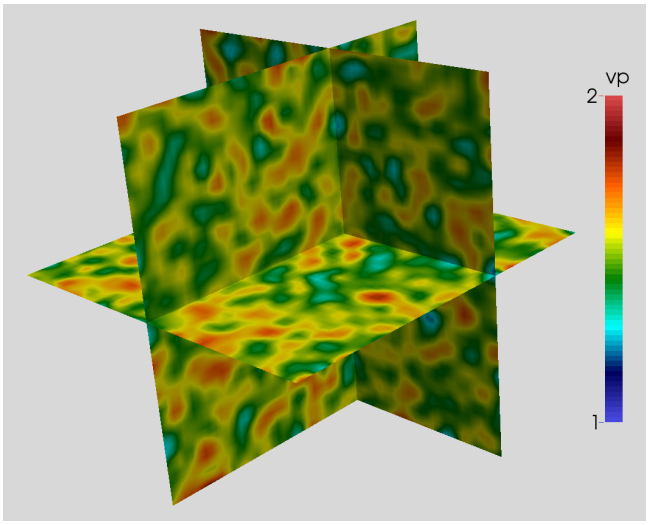

Figure 9: Randomly generated smooth heterogeneous medium.

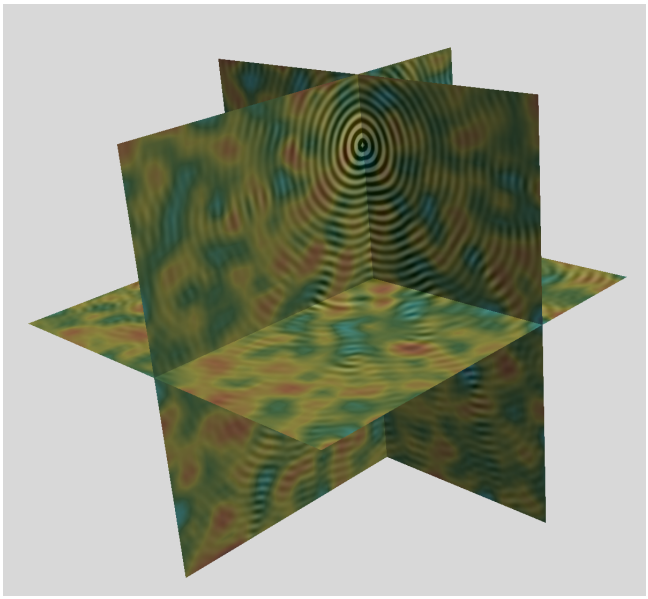

Figure 10: Real part of the solution to the Helmholtz equation using the random smooth medium.

Davis, T. A., S. Rajamanickam, and W. M. Sid-Lakhdar, 2016, A survey of direct methods for sparse linear systems: Acta Numerica, **25**, 383–566.

de Hoop, M., J. H. Le Rousseau, and R.-S. Wu, 2000, Generalization of the phase-screen approximation for the scattering of acoustic waves: Wave Motion, **31**, 43–70.

de Hoop, M. V., S. Wang, and J. Xia., 2011, On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver: Geophysical Prospecting, **59**, 857–873.

de La Bourdonnaye, A., C. Farhat, A. Macedo, F. Magoules, and F.-X. Roux, 1998, A non-overlapping domain decomposition method for the exterior Helmholtz problem: Contemporary Mathematics, **218**, 42–66.

Demmel, J. W., S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, 1999, A supernodal approach to sparse partial pivoting: SIAM J. Matrix Analysis and Applications, **20**, 720–755.

Després, B., 1990, Décomposition de domaine et problème de Helmholtz: Comptes rendus de l'Académie des sciences. Série 1, Mathématique, **311**, 313–316.

Duff, I. S., and J. Koster, 2001, On algorithms for permuting large entries to the diagonal of a sparse matrix: SIAM Journal on Matrix Analysis and Applications, **22**, 973–996.

Duff, I. S., and J. K. Reid, 1983, The multifrontal solution of indefinite sparse symmetric linear: ACM Trans. Math. Softw., **9**, 302–325.

Engquist, B., and L. Ying, 2011a, Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation: Communications on Pure and Applied Mathematics, **64**, 697–735.

———, 2011b, Sweeping preconditioner for the Helmholtz equation: moving perfectly matched layers: Multiscale Modeling & Simulation, **9**, 686–710.

Engquist, B., and H.-K. Zhao, 1998, Absorbing boundary conditions for domain decomposition: Applied Numerical Mathematics, **27**, 341 – 365. (Special Issue on Absorbing Boundary Conditions).

———, 2014, Approximate separability of Green's function for high frequency Helmholtz equations.

Erlangga, Y. A., C. W. Oosterlee, and C. Vuik, 2006, A novel multigrid based preconditioner for heterogeneous Helmholtz problems: SIAM Journal on Scientific Computing, **27**, 1471–1492.

Fang, J., J. Qian, L. Zepeda-Núñez, and H.-K. Zhao, 2016, Learning dominant wave directions for plane wave methods for high-frequency Helmholtz equations: ArXiv e-prints, [**math.NA**] **arXiv:1608.08871**.

Farhat, C., I. Harari, and L. P. Franca, 2001, The discontinuous enrichment method: Computer Methods in Applied Mechanics and Engineering, **190**, 6455–6479.

Gander, M., and F. Nataf, 2000, AILU for Helmholtz problems: a new preconditioner based on an analytic factorization: Comptes Rendus de l'Académie des Sciences-Series I-Mathematics, **331**, 261–266.

Gander, M. J., 2006, Optimized schwarz methods: SIAM Journal on Numerical Analysis, **44**, 699–731.

Gander, M. J., and F. Kwok, 2011, Optimal interface conditions for an arbitrary decomposition into subdomains, *in* Domain Decomposition Methods in Science and Engineering XIX: Springer Berlin Heidelberg, volume **78** *of* Lecture Notes in Computational Science and Engineering, 101–108.

Gander, M. J., F. Magoulès, and F. Nataf, 2002, Optimized Schwarz methods without overlap for the Helmholtz equation: SIAM Journal on Scientific Computing, **24**, 38–60.

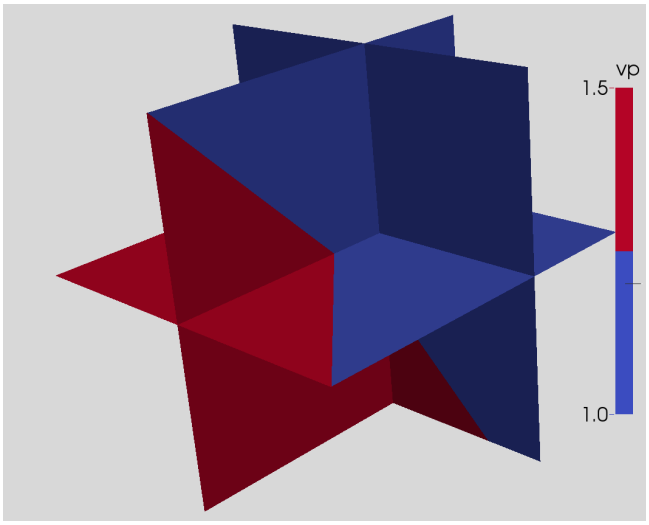

Figure 11: Simple fault model.

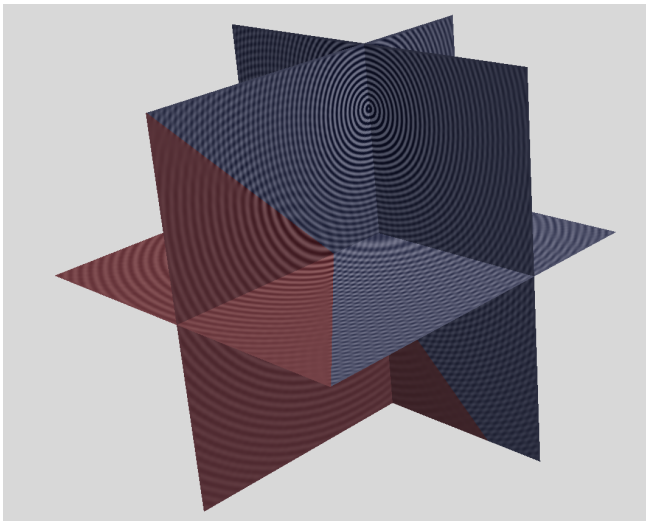

Figure 12: Real part of the solution to the Helmholtz equation using the simple fault model.

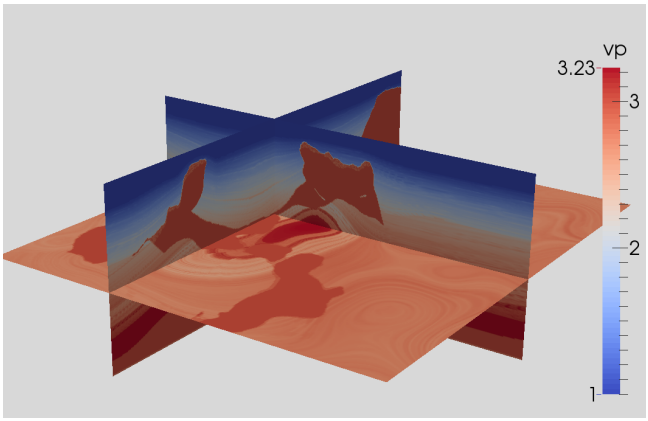

Figure 13: Plot of the SEAM model.

Gander, M. J., and F. Nataf, 2005, An incomplete LU preconditioner for problems in acoustics: Journal of Computational Acoustics, **13**, 455–476.

Gander, M. J., and Y. Xu, 2016, Optimized Schwarz method with two-sided transmission conditions in an unsymmetric domain decomposition, *in* Domain Decomposition Methods in Science and Engineering XXII: Springer International Publishing, 631–639.

Gander, M. J., and H. Zhang, 2013, Domain decomposition methods for the Helmholtz equation: A numerical investigation, *in* Domain Decomposition Methods in Science and Engineering XX: Springer Berlin Heidelberg, volume **91** *of* Lecture Notes in Computational Science and Engineering, 215–222.

———, 2014, Optimized Schwarz methods with overlap for the Helmholtz equation, *in* Domain Decomposition Methods in Science and Engineering XXI: Springer International Publishing, 207–215.

George, A., 1973, Nested dissection of a regular finite element mesh: SIAM Journal on Numerical Analysis, **10**, 345–363.

Ghanemi, S., 1998, A domain decomposition method for Helmholtz scattering problems: Ninth International Conference on Domain Decomposition Methods, 105–112.

Gillman, A., A. Barnett, and P. Martinsson, 2014, A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media: BIT Numerical Mathematics, 1–30.

Gittelson, C. J., R. Hiptmair, and I. Perugia, 2009, Plane wave discontinuous Galerkin methods: Analysis of the h-version: ESAIM: Mathematical Modelling and Numerical Analysis, **43**, no. 02, 297–331.

Goldstein, C. I., 1986, The weak element method applied to Helmholtz type equations: Applied Numerical Mathematics, **2**, 409 – 426.

Gordon, D., and R. Gordon, 2010, Carp-cg: A robust and efficient parallel solver for linear systems, applied to strongly convection dominated {PDEs}: Parallel Computing, **36**, 495 – 515.

Graham, I., M. Löhndorf, J. Melenk, and E. Spence, 2015, When is the error in the h-BEM for solving the Helmholtz equation bounded independently of k?: BIT Numerical Mathematics, **55**, 171–214.

Gupta, A., G. Karypis, and V. Kumar, 1997, Highly scalable parallel algorithms for sparse matrix factorization: IEEE Trans. Parallel Distrib. Syst., **8**, 502–520.

Hiptmair, R., A. Moiola, and I. Perugia, 2015, A survey of Trefftz methods for the Helmholtz equation: ArXiv e-prints.

Hu, Q., and H. Zhang, 2016, Substructuring preconditioners for the systems arising from plane wave discretization of helmholtz equations: SIAM Journal on Scientific Computing, **38**, A2232–A2261.

Ihlenburg, F., and I. Babuska, 1995, Finite element solution of the Helmholtz equation with high wave number

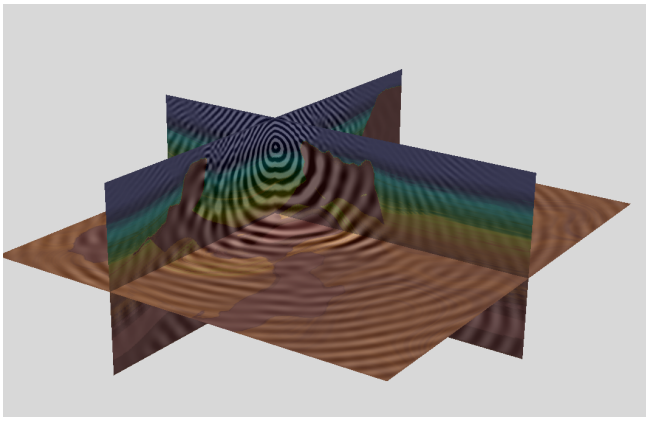

Figure 14: Real part of the solution of the Helmholtz equation using the SEAM model.

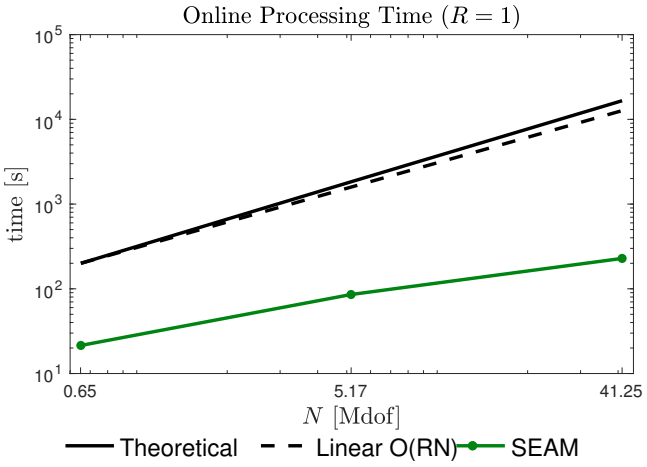

Figure 15: Observed runtime as function of N for the SEAM model for $R = 1$ right-hand side. For comparison, we show the theoretical scaling of the polarized traces algorithm (solid black), as well as a linear scaling (dashed black).

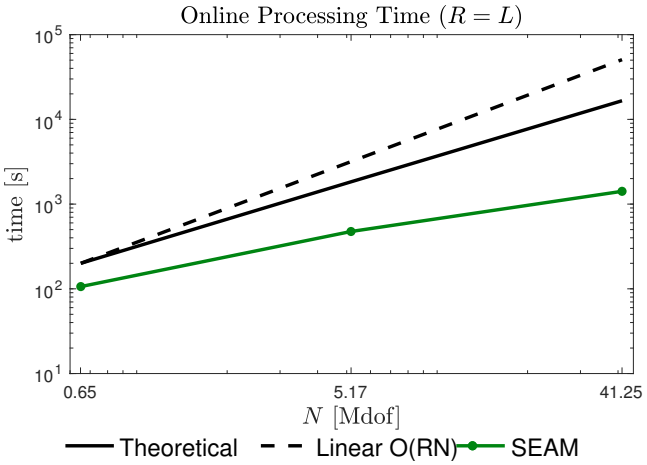

Figure 16: Observed runtime as function of N for the SEAM model for $R = L$ right-hand sides. For comparison, we show the theoretical scaling of the polarized traces algorithm (solid black), as well as a linear scaling (dashed black).

part I: The h-version of the FEM: Computers & Mathematics with Applications, **30**, 9 – 37.

Johnson, S., 2010, Notes on perfectly matched layers (PMLs).

Joshi, M. V., A. Gupta, G. Karypis, and V. Kumar, 1997, A high performance two dimensional scalable parallel algorithm for solving sparse triangular systems: Proceedings Fourth International Conference on High-Performance Computing, 137–143.

Karypis, G., and V. Kumar, 1998, A parallel algorithm for multilevel graph partitioning and sparse matrix ordering: Journal of Parallel and Distributed Computing, **48**, 71 – 95.

Li, X. S., and J. W. Demmel, 2003, SuperLU DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems: ACM Trans. Mathematical Software, **29**, 110–140.

Li, Y., L. Mtivier, R. Brossier, B. Han, and J. Virieux, 2015, 2D and 3D frequency-domain elastic wave modeling in complex media with a parallel iterative solver: GEOPHYSICS, **80**, T101–T118.

Lions, P.-L., 1989, On the Schwarz alternating method II, *in* Domain Decomposition Methods: SIAM, Lecture Notes in Computational Science and Engineering, 47–70.

Liu, F., and L. Ying, 2015, Recursive sweeping preconditioner for the 3D Helmholtz equation: ArXiv e-prints.

Magoules, F., K. Meerbergen, and J.-P. Coyette, 2000, Application of a domain decomposition with Lagrange multipliers to acoustic problems arising from the automotive industry: Journal of Computational Acoustics, **08**, 503–521.

McInnes, L. C., R. F. Susan-Resiga, D. E. Keyes, and H. M. Atassi, 1998, Additive Schwarz methods with nonreflecting boundary conditions for the parallel computation of Helmholtz problems: Contemporary Mathematics, **218**, 325–333.

Melenk, J. M., A. Parsania, and S. Sauter, 2013, General DG-methods for highly indefinite Helmholtz problems: J Sci Comput, **57**, 536–581.

Melenk, J. M., and S. Sauter, 2011, Wavenumber explicit convergence analysis for Galerkin discretizations of the Helmholtz equation: SIAM Journal on Numerical Analysis, **49**, 1210–1243.

Moiola, A., R. Hiptmair, and I. Perugia, 2011, Plane wave approximation of homogeneous Helmholtz solutions: Zeitschrift für angewandte Mathematik und Physik, **62**, 809–837.

Moiola, A., and E. Spence, 2014, Is the Helmholtz equation really sign-indefinite?: SIAM Review, **56**, 274–312.

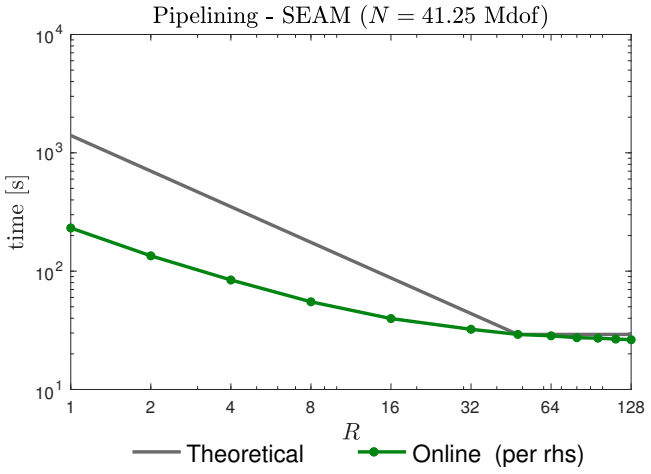

Figure 17: Impact of scalability on pipelining for the SEAM model. N is held constant and R is increased. The observed runtime as function of R is in green, and the dashed black line illustrates theoretical scalability.

Monk, P., and D.-Q. Wang, 1999, A least-squares method for the Helmholtz equation: Computer Methods in Applied Mechanics and Engineering, **175**, 121–136.

Nguyen, N. C., J. Peraire, F. Reitich, and B. Cockburn, 2015, A phase-based hybridizable discontinuous Galerkin method for the numerical solution of the Helmholtz equation: J. Comput. Physics, **290**, 318–335.

Perugia, I., P. Pietra, and A. Russo, 2016, A plane wave virtual element method for the Helmholtz problem: ESAIM: Mathematical Modelling and Numerical Analysis, **50**, 783–808.

Plessix, R.-E., 2007, A Helmholtz iterative solver for 3D seismic-imaging problems: Geophysics, **72**, SM185–SM194.

Plessix, R.-E., and W. A. Mulder, 2003, Separation-of-variables as a preconditioner for an iterative Helmholtz solver: Applied Numerical Mathematics, **44**, 385–400.

Pothen, A., and C. Sun, 1993, A mapping algorithm for parallel sparse cholesky factorization: SIAM Journal on Scientific Computing, **14**, 1253–1257.

Poulson, J., B. Engquist, S. Li, and L. Ying, 2013, A parallel sweeping preconditioner for heterogeneous 3D Helmholtz equations: SIAM Journal on Scientific Computing, **35**, C194–C212.

Pratt, R. G., 1999, Seismic waveform inversion in the frequency domain; part 1: Theory and verification in a physical scale model: Geophysics, **64**, 888–901.

Raghavan, P., 1998, Efficient parallel sparse triangular solution using selective inversion: Parallel Processing Letters, **08**, 29–40.

Ristow, D., and T. Ruehl, 1997, 3-D implicit finite-difference migration by multiway splitting: Geophysics, **62**, 554–567.

Rouet, F.-H., X. S. Li, P. Ghysels, and A. Napov, 2015, A distributed-memory package for dense Hierarchically Semi-Separable matrix computations using randomization: ArXiv e-prints.

Saad, Y., 2003, Iterative methods for sparse linear systems, second ed.: Society for Industrial and Applied Mathematics.

Saad, Y., and M. H. Schultz, 1986, Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems: SIAM J. Sci. Stat. Comput., **7**, 856–869.

Schwarz, H. A., 1870, Uber einen grenzubergang durch alternierendes verfahren: Vierteljahrsschrift der Naturforschenden Gesellschaft in Zurich, **15**, 272–286.

Sheikh, A. H., D. Lahaye, and C. Vuik, 2013, On the convergence of shifted Laplace preconditioner combined with multilevel deflation: Numerical Linear Algebra with Applications, **20**, 645–662.

Stolk, C., 2013, A rapidly converging domain decomposition method for the Helmholtz equation: Journal of Computational Physics, **241**, 240–252.

Stolk, C. C., 2015a, A dispersion minimizing scheme for the 3-D Helmholtz equation with applications in multigrid based solvers: ArXiv e-prints.

———, 2015b, An improved sweeping domain decomposition preconditioner for the Helmholtz equation: ArXiv e-prints.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

Taus, M., L. Demanet, and L. Zepeda-Núñez, 2016, A short note on a fast and high-order hybridizable discontinuous Galerkin solver for the 2D high-frequency Helmholtz equation: SEG Technical Program Expanded Abstracts 2016, 3835–3840.

Toselli, A., and O. B. Windlund, 2005, Domain decomposition methods algorithms and theory: Springer Berlin Heidelberg, volume **34** *of* Springer Series in Computational Mathematics.

Tsuji, P., B. Engquist, and L. Ying, 2012, A sweeping preconditioner for time-harmonic Maxwell's equations with finite elements: Journal of Computational Physics, **231**, 3770 – 3783.

Tsuji, P., J. Poulson, B. Engquist, and L. Ying, 2014, Sweeping preconditioners for elastic wave propagation with spectral element methods: ESAIM: Mathematical Modelling and Numerical Analysis, **48**, no. 02, 433–447.

Tsuji, P., and L. Ying, 2012, A sweeping preconditioner for Yee's finite difference approximation of time-harmonic Maxwell's equations: Frontiers of Mathematics in China, **7**, 347–363.

van der Vorst, H. A., 1992, 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**, 631–644.

Vion, A., and C. Geuzaine, 2014, Double sweep preconditioner for optimized Schwarz methods applied to the Helmholtz problem: Journal of Computational Physics, **266**, 171–190.

Virieux, J., and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: GEOPHYSICS, **74**, WCC1–WCC26.

Wang, S., M. V. de Hoop, J. Xia, and X. S. Li, 2012, Massively parallel structured multifrontal solver for time-harmonic elastic waves in 3-D anisotropic media: Geophysical Journal International, **191**, 346–366.

Wang, S., X. S. Li, J. Xia, Y. Situ, and M. V. de Hoop, 2013, Efficient scalable algorithms for solving dense linear systems with hierarchically semiseparable structures: SIAM Journal on Scientific Computing, **35**, C519–C544.

Wu, R.-S., 1994, Wide-angle elastic wave one-way propagation in heterogeneous media and an elastic wave complex-screen method: Journal of Geophysical Research: Solid Earth, **99**, 751–766.

Xia, J., 2013, Randomized sparse direct solvers: SIAM Journal on Matrix Analysis and Applications, **34**, 197–227.

Xia, J., S. Chandrasekaran, M. Gu, and X. S. Li, 2010, Superfast multifrontal method for large structured linear systems of equations: SIAM Journal on Matrix Analysis and Applications, **31**, 1382–1411.

Zepeda-Núñez, L., and L. Demanet, 2015, Nested domain decomposition with polarized traces for the 2D Helmholtz equation: ArXiv e-prints.

———, 2016, The method of polarized traces for the 2D Helmholtz equation: Journal of Computational Physics, **308**, 347 – 388.

Zepeda-Núñez, L., and H. Zhao, 2016, Fast alternating bidirectional preconditioner for the 2D high-frequency Lippmann–Schwinger equation: SIAM Journal on Scientific Computing, **38**, B866–B888.

Zhang, Y., 2006, The theory of true amplitude one-way wave equation migrations: Chinese Journal of Geophysics, **49**, 1267–1289.

APPENDIX: MATRIX-FREE ALGORITHMS

The application of the polarized system defined in Eq. 18 is achieved by applying each block as shown in Alg. 7.

Algorithm 7. *Application of* $\underline{\underline{\mathbf{M}}}$

function $\underline{\underline{\mathbf{u}}}$ = Application Polarized($\underline{\underline{\mathbf{v}}}$)
  $(\underline{\mathbf{v}}^{\downarrow}, \underline{\mathbf{v}}^{\uparrow}) = \underline{\underline{\mathbf{v}}}$
  $\underline{\mathbf{u}}^{\downarrow} = \underline{\mathbf{D}}^{\downarrow}\mathbf{v}^{\downarrow} + \underline{\mathbf{U}}\mathbf{v}^{\uparrow}$
  $\underline{\mathbf{u}}^{\uparrow} = \underline{\mathbf{D}}^{\uparrow}\mathbf{v}^{\uparrow} + \underline{\mathbf{L}}\mathbf{v}^{\downarrow}$
  $\underline{\underline{\mathbf{u}}} = (\underline{\mathbf{u}}^{\downarrow}, \underline{\mathbf{u}}^{\uparrow})$
end function

To apply the blocks in a matrix-free fashion, we use Algs. 8 and 6 in line 4 of Alg. 7, and we use Algs. 9 and 10 in line 5 of Alg. 7. All algorithms in this section are embarrasingly parallel at the level of the layers as depicted in Fig. 3.

Algorithm 8. *Application of* $\underline{\mathbf{D}}^{\downarrow}$

function $\underline{\mathbf{u}}^{\downarrow}$ = Downward Sweep($\underline{\mathbf{v}}^{\downarrow}$)
  $\mathbf{u}^{\downarrow,1}_{n^1} = -\mathbf{v}^{\downarrow,1}_{n^1}$
  $\mathbf{u}^{\downarrow,1}_{n^1+1} = -\mathbf{v}^{\downarrow,1}_{n^1+1}$
  for $\ell = 2 : L-1$ ***do***
    $\widetilde{\mathbf{f}}^{\ell} = \delta(z_1 - z)\mathbf{v}^{\downarrow,\ell-1}_{n^{\ell-1}} - \delta(z_0 - z)\mathbf{v}^{\downarrow,\ell-1}_{n^{\ell-1}+1}$
    $\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\widetilde{\mathbf{f}}^{\ell}$
    $\mathbf{u}^{\downarrow,\ell}_{n^{\ell}} = \mathbf{w}_{n^{\ell}} - \mathbf{v}^{\downarrow,\ell}_{n^{\ell}}$
    $\mathbf{u}^{\downarrow,\ell}_{n^{\ell}+1} = \mathbf{w}_{n^{\ell}+1} - \mathbf{v}^{\downarrow,\ell}_{n^{\ell}+1}$
  end for
  $\underline{\mathbf{u}}^{\downarrow} = \left(\mathbf{u}^{\downarrow,1}_{n^1}, \mathbf{u}^{\downarrow,1}_{n^1+1}, \mathbf{u}^{\downarrow,2}_{n^2}, ..., \mathbf{u}^{\downarrow,L-1}_{n^{L-1}}, \mathbf{u}^{\downarrow,L-1}_{n^{L-1}+1}\right)^t$
end function

Algorithm 9. *Upward sweep, application of* $(\underline{\mathbf{D}}^{\uparrow})^{-1}$

function $\underline{\mathbf{u}}^{\uparrow}$ = Upward sweep($\underline{\mathbf{v}}^{\uparrow}$)
  $\mathbf{u}^{\uparrow,L}_{0} = -\mathbf{v}^{\uparrow,L}_{0}$
  $\mathbf{u}^{\uparrow,L}_{1} = -\mathbf{v}^{\uparrow,L}_{1}$
  for $\ell = L-1 : 2$ ***do***
    $\widetilde{\mathbf{f}}^{\ell} = -\delta(z_{n^{\ell}+1} - z)\mathbf{v}^{\uparrow,\ell+1}_{0} + \delta(z_{n^{\ell}} - z)\mathbf{v}^{\uparrow,\ell+1}_{1}$
    $\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\widetilde{\mathbf{f}}^{\ell}$
    $\mathbf{u}^{\uparrow,\ell}_{1} = \mathbf{w}^{\ell}_{1} - \mathbf{v}^{\uparrow,\ell}_{1}$
    $\mathbf{u}^{\uparrow,\ell}_{0} = \mathbf{w}^{\ell}_{0} - \mathbf{v}^{\uparrow,\ell}_{0}$
  end for
  $\underline{\mathbf{u}}^{\uparrow} = \left(\mathbf{u}^{\uparrow,2}_{0}, \mathbf{u}^{\uparrow,2}_{1}, \mathbf{u}^{\uparrow,3}_{0}, ..., \mathbf{u}^{\uparrow,L}_{0}, \mathbf{u}^{\uparrow,L}_{1}\right)^t$
end function

Algorithm 10. *Downwards reflections, application of* $\underline{\mathbf{U}}$

function $\underline{\mathbf{u}}^{\uparrow}$ = Upward Reflections($\underline{\mathbf{v}}^{\downarrow}$)
  for $\ell = 2 : L-1$ ***do***
    $\mathbf{f}^{\ell} = \delta(z_1 - z)\mathbf{v}^{\downarrow,\ell}_{0} - \delta(z_0 - z)\mathbf{v}^{\downarrow,\ell}_{1}$
      $-\delta(z_{n^{\ell}+1} - z)\mathbf{v}^{\downarrow,\ell+1}_{1} + \delta(z_{n^{\ell}} - z)\mathbf{v}^{\downarrow,\ell+1}_{0}$
    $\mathbf{w}^{\ell} = (\mathbf{H}^{\ell})^{-1}\mathbf{f}^{\ell}$
    $\mathbf{u}^{\uparrow,\ell}_{1} = \mathbf{w}^{\ell}_{1} - \mathbf{v}^{\downarrow,\ell}_{1}$
    $\mathbf{u}^{\uparrow,\ell}_{0} = \mathbf{w}^{\ell}_{0}$
  end for
  $\mathbf{f}^{L} = \delta(z_1 - z)\mathbf{v}^{\uparrow,L}_{0} - \delta(z_0 - z)\mathbf{v}^{\uparrow,L}_{1}$
  $\mathbf{w}^{L} = (\mathbf{H}^{L})^{-1}\mathbf{f}^{L}$
  $\mathbf{u}^{\uparrow,L}_{1} = \mathbf{w}^{L}_{1} - \mathbf{v}^{\downarrow,L}_{1}$
  $\mathbf{u}^{\uparrow,L}_{0} = \mathbf{w}^{L}_{0}$
  $\underline{\mathbf{u}}^{\uparrow} = \left(\mathbf{u}^{\uparrow,2}_{0}, \mathbf{u}^{\uparrow,2}_{1}, \mathbf{u}^{\uparrow,3}_{0}, ..., \mathbf{u}^{\uparrow,L}_{0}, \mathbf{u}^{\uparrow,L}_{1}\right)^t$
end function